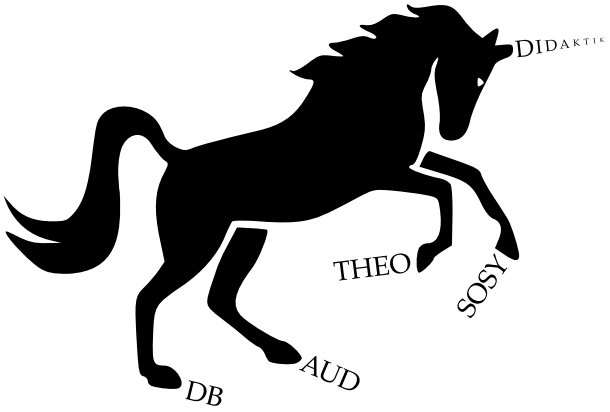
Die komplette Sammlung

Alle Übungs- und Examensaufgaben



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Inhaltsverzeichnis

I Datenbanken (DB) 19

1 Datenbank-Übersicht 20

Examensaufgabe „Theoriefragen Datenbank“ (46116-2015-H.T1-TA2-A1) 20
Examensaufgabe „Tupelidentifikator“ (46116-2017-F.T1-TA2-A4) 21
Examensaufgabe „Physische Datenorganisation“ (66116-2016-H.T1-TA1-A5) 22
Examensaufgabe „Allgemeine Fragen“ (66116-2019-F.T2-TA1-A1) 25
Examensaufgabe „Vermischte Datenbank-Fragen“ (66116-2019-H.T2-TA2-A6) 28
Examensaufgabe „Vermischte Fragen“ (66116-2021-F.T1-TA2-A1) 31
Übungsaufgabe „Drei-Schichten-Modell“ (Drei-Schichten-Modell) . . . 35
Vorgänge 35
Hinweis 35
Übungsaufgabe „Speicherung-Dateisystem“ (Datenbank-Übersicht) . . 37
Übungsaufgabe „Terminologie“ (Datenbanksystem, Datenbank, Datenbankmanagementsystem) 39

2 Datenbankentwurf 40

Examensaufgabe „Rennstall“ (46116-2013-F.T1-TA2-A1) 40
Examensaufgabe „Konsulat“ (46116-2015-F.T1-TA2-A3) 42
Examensaufgabe „Freizeitparks“ (46116-2018-H.T1-TA2-A2) 44
Examensaufgabe „Schulverwaltung“ (46116-2018-H.T2-TA2-A5) 45
Examensaufgabe „Universitätsdatenbank“ (66111-1996-F.A2) 47
Examensaufgabe „Fertigung“ (66111-1997-H.A3) 49
Entity-Typen: 49
Relationship-Typen: 49
Examensaufgabe „Handelsunternehmen“ (66116-2012-F.T1-TA1-A1) . . 50
Examensaufgabe „Musik-Datenbank“ (66116-2015-F.T1-TA1-A1) 51
Examensaufgabe „Online-Auktionshaus“ (66116-2015-H.T1-TA1-A1) . . 52
Examensaufgabe „Forstverwaltung“ (66116-2016-F.T1-TA1-A1) 53
Examensaufgabe „Polizei“ (66116-2016-F.T1-TA1-A2) 55
Examensaufgabe „Zirkus“ (66116-2018-F.T2-TA1-A2) 61
Examensaufgabe „Schule Hogwarts aus Harry Potter“ (66116-2019-H.T1-TA2-A2) 63
Examensaufgabe „Sportverein“ (66116-2019-H.T2-TA2-A1) 64
Examensaufgabe „Einwohnermeldeamt“ (66116-2020-F.T1-TA2-A1) . . . 65
Examensaufgabe „Wetterdienst“ (66116-2020-F.T2-TA2-A1) 67
Examensaufgabe „Automobilproduktion“ (66116-2021-F.T1-TA2-A2) . . 69
Examensaufgabe „Online-Marktplatze“ (66116-2021-F.T2-TA2-A2) . . . 71
Übungsaufgabe „DBTec“ (Entity-Relation-Modell) 72

Übungsaufgabe „Fahrzeugverwaltung“ (Entity-Relation-Modell) 73
Übungsaufgabe „Flughafen“ (Entity-Relation-Modell) 75
Übungsaufgabe „Bundesliga-Datenbank“ (Relationale Algebra, SQL, Entity-Relation-Modell) 79

3 Relationales Modell **82**
Examensaufgabe „Browser-Online-Spiele“ (46116-2013-F.T1-TA2-A2) . . 82
2. Anfragen 82
Examensaufgabe „Mitfahrgelegenheiten“ (46116-2014-F.T2-TA2-A2) . . 85
Aufgabe 2: Relationale Algebra 85
Examensaufgabe „Computer „Chiemsee““ (46116-2015-F.T1-TA2-A1) . . 88
Examensaufgabe „Relationen R, S und T“ (46116-2018-H.T2-TA2-A2) . . 91
Examensaufgabe „Harry Potter“ (46116-2019-H.T2-A4) 94
Examensaufgabe „Gebrauchtwagen“ (66111-1996-H.A4) 95
Examensaufgabe „Tupelkalkül bei Dozenten-Datenbank“ (66116-2018-F.T2-TA1-A4) 97
Examensaufgabe „Medikamente“ (66116-2019-F.T1-TA1-A2) 99
Examensaufgabe „Relation X und Y“ (66116-2019-F.T2-TA1-A3) 103
Examensaufgabe „Relationen „Professor“ und „Vorlesung““ (66116-2019-H.T2-TA2-A5) 105
Examensaufgabe „Universitätsschema“ (66116-2020-F.T1-TA2-A3) 106
Examensaufgabe „Relationale Algebra und Optimierung“ (66116-2020-H.T2-TA2-A3) 107
Examensaufgabe „Relationen R1 und R2“ (66116-2021-F.T1-TA2-A3) . . 110
Examensaufgabe „Autoverleih“ (66116-2021-F.T2-TA2-A3) 113
Examensaufgabe „Mitarbeiter einer Abteilung“ (66116-2021-F.T2-TA2-A5) 114
Übungsaufgabe „Bus-Unternehmen“ (Relationenmodell) 115
Übungsaufgabe „Krankenhaus“ (Verfeinertes Relationenmodell) 116
Übungsaufgabe „Mitarbeiter-Projekte einer Abteilung“ (Relationenmodell, Verfeinertes Relationenmodell) 117
Übungsaufgabe „Süße Produktion“ (Relationenmodell, Kartesisches Produkt) 119
Übungsaufgabe „Tutor“ (Relationenmodell) 120
Übungsaufgabe „Division“ (Relationale Algebra, Division) 122
Übungsaufgabe „Freizeitcenter“ (Relationale Algebra) 124
Übungsaufgabe „Universität“ (Relationale Algebra) 128
Übungsaufgabe „Vater-Muter-Kind“ (Division) 129
Übungsaufgabe „Wassned“ (Relationale Algebra) 130
Übungsaufgabe „Xenokrates“ (Tupelkalkül) 131
Übungsaufgabe „Kaufhaus“ (SQL, SQL mit Übungsdatenbank, Relationale Algebra) 132
Teilaufgabe 2 134
Teilaufgabe 2 138
Teilaufgabe 4 141
Teilaufgabe 5 142

4 Relationale Anfragesprachen **143**
Examensaufgabe „Personalverwaltung“ (46116-2012-F.T1-TA1-A3) . . . 143
Aufgabe 3 . . . 143
Examensaufgabe „Mitfahrgelegenheiten“ (46116-2014-F.T2-TA2-A3) . . 145
Examensaufgabe „Turmspringen“ (46116-2017-H.T2-TA2-A4) . . . 148
Examensaufgabe „Kundenverwaltungssystem“ (46116-2018-H.T1-TA1-A4) . . . 150
Examensaufgabe „Kundenverwaltungssystem“ (46116-2018-H.T1-TA2-A4) . . . 156
Examensaufgabe „Schuldatenbank“ (46116-2018-H.T2-TA2-A3) . . . 157
Examensaufgabe „Mitarbeiterverwaltung“ (66113-2003-F.T1-A5) . . . 164
Examensaufgabe „Universitätsverwaltung“ (66113-2003-F.T2-A3) . . . 168
Examensaufgabe „Gebrauchtwagen“ (66116-2012-F.T1-TA1-A3) . . . 170
Examensaufgabe „Musik-CDs“ (66116-2015-F.T1-TA1-A2) . . . 172
Examensaufgabe „Vater und Mutter“ (66116-2015-H.T1-TA1-A3) . . . 174
Examensaufgabe „Personalverwaltung“ (66116-2016-H.T1-TA1-A4) . . . 176
Examensaufgabe „Schulverwaltung“ (66116-2016-H.T2-TA1-A2) . . . 181
Examensaufgabe „SQL-Syntax-Überprüfung“ (66116-2017-H.T1-TA1-A5) 187
Examensaufgabe „Fluginformationssystem“ (66116-2017-H.T1-TA1-A6) 189
Examensaufgabe „Triathlon“ (66116-2018-H.T1-TA2-A4) . . . 192
Examensaufgabe „App für konfizierte Schüler-Smartphones“ (66116-2019-F.T2-TA1-A5) . . . 196
Examensaufgabe „Game of Thrones“ (66116-2019-H.T1-TA2-A3) . . . 198
Examensaufgabe „Formel-1-Rennen“ (66116-2019-H.T2-TA2-A7) . . . 203
Examensaufgabe „Zehnkampf“ (66116-2020-F.T1-TA2-A7) . . . 207
Examensaufgabe „Universitätssschema“ (66116-2020-F.T1-TA2-A8) . . . 209
Examensaufgabe „Mode-Kollektionen“ (66116-2020-F.T2-TA2-A2) . . . 214
Examensaufgabe „Restaurant“ (66116-2020-H.T2-TA2-A2) . . . 220
Examensaufgabe „Fußballweltmeisterschaft“ (66116-2021-F.T1-TA2-A6) 222
Examensaufgabe „Mitarbeiter einer Abteilung“ (66116-2021-F.T2-TA2-A4) 225
Übungsaufgabe „Bands“ (SQL, SQL mit Übungsdatenbank) . . . 228
Übungsaufgabe „SQL abstrakt“ (SQL) . . . 232

5 Relationale Entwurfstheorie **235**
Examensaufgabe „Studentenbibliothek“ (66111-1994-F.A7) . . . 235
Examensaufgabe „Funktionale Abhängigkeiten, Normalisierung“ (66113-2002-H.T2-A2) . . . 236
Examensaufgabe „Nachteile vollständige Normalisierung“ (66113-2003-H.T2-A1) . . . 239
Examensaufgabe „Wareneingänge“ (66116-2012-F.T1-TA1-A2) . . . 241
Examensaufgabe „Relation A-F“ (66116-2015-F.T1-TA1-A3) . . . 243
Examensaufgabe „Relation A-H“ (66116-2015-H.T1-TA1-A2) . . . 244
Examensaufgabe „Entwurfstheorie“ (66116-2017-F.T2-TA1-A5) . . . 245
Examensaufgabe „Relation A-F“ (66116-2019-F.T1-TA1-A3) . . . 248
Examensaufgabe „Normalisierung“ (66116-2019-F.T2-TA1-A6) . . . 251
Examensaufgabe „Sekretäre“ (66116-2020-F.T1-TA2-A2) . . . 254

Examensaufgabe „Relation A-F“ (66116-2020-F.T1-TA2-A4) 255
Examensaufgabe „Schlüssel“ (66116-2020-F.T1-TA2-A5) 258
Examensaufgabe „Relation A-F“ (66116-2020-F.T2-TA2-A3) 259
Examensaufgabe „Entwurfstheorie“ (66116-2020-H.T2-TA2-A4) 262
Examensaufgabe „Normalisierung“ (66116-2021-F.T1-TA2-A4) 264
Examensaufgabe „Relation Prüfung“ (66116-2021-F.T2-TA2-A6) 267
Übungsaufgabe „Schlüsselkandidat von R“ (Schlüsselkandidat) 268
Übungsaufgabe „Arbeitsvermittler“ (Normalformen, Funktionale Abhängigkeiten, Schlüsselkandidat, Synthese-Algorithmus) 269
Übungsaufgabe „Drei-Schemata“ (Boyce-Codd-Normalform, Dritte Normalform, Zweite Normalform, Synthese-Algorithmus) 271
Übungsaufgabe „Relation A-H“ (Synthese-Algorithmus) 273
Übungsaufgabe „Relation-MNVTPPN“ (Synthese-Algorithmus, Kanonische Überdeckung) 276
Übungsaufgabe „Supermarkt“ (Zweite Normalform, Schlüsselkandidat, Update-Anomalie, Delete-Anomalie, Synthese-Algorithmus) . . 279
Übungsaufgabe „Abstraktes R“ (Schlüsselkandidat, Zweite Normalform, Kanonische Überdeckung) 284
Übungsaufgabe „Anomalien Abhängigkeiten“ (Update-Anomalie, Delete-Anomalie, Insert-Anomalie, Funktionale Abhängigkeiten, Attributhüllen-Algorithmus, Attributhülle, Superschlüssel) 287
Übungsaufgabe „Kanonische Überdeckung (Kemper)“ (Kanonische Überdeckung) 289
Übungsaufgabe „Mietwagenfirma“ (Zweite Normalform, Delete-Anomalie, Update-Anomalie, Insert-Anomalie, Dritte Normalform) 291
Übungsaufgabe „Minimale Überdeckung“ (Kanonische Überdeckung) . 293

6 Transaktionsverwaltung **294**
Examensaufgabe „Transaktionen“ (46116-2016-F.T1-TA1-A5) 294
Examensaufgabe „Schedule S“ (66116-2020-F.T2-TA2-A4) 296
Examensaufgabe „Transaktionen T1 und T2“ (66116-2021-F.T1-TA2-A5) 300
Übungsaufgabe „ACID“ (Transaktionen, ACID) 303
Übungsaufgabe „PKW“ (Transaktionen, Deadlock) 304
Übungsaufgabe „Tabelle TAB“ (Transaktionsverwaltung, Lost-Update, Dirty-Read) 306

7 Sonstige **308**
Examensaufgabe „Wissensfragen“ (66116-2019-H.T1-TA2-A1) 308
Examensaufgabe „Vermischte Fragen“ (66116-2021-F.T2-TA2-A1) 310
Examensaufgabe „Optimierung“ (66116-2021-F.T2-TA2-A7) 312

II Algorithmen und Datenstrukturen (AUD) **314**

8 Rekursion **315**
Examensaufgabe „Binomialkoeffizient“ (46115-2014-F.T2-A4) 315

Aufgabe 4 . . . 315
Examensaufgabe „Klasse „LeftFactorial“ und Methode „lfBig()““ (66115-2014-F.T1-A1) . . . 319
Examensaufgabe „Dateisystem: Implementierung durch Kompositum“ (66116-2019-H.T2-TA1-A1) . . . 323
Übungsaufgabe „Feld-Invertierer“ (Rekursion, Implementierung in Java) 328
Übungsaufgabe „Fibonacci Fakultät“ (Rekursion) . . . 331
Übungsaufgabe „Potenz“ (Rekursion) . . . 332
Übungsaufgabe „Rater“ (Rekursion) . . . 333
Übungsaufgabe „iterativ-rekursiv“ (Iterative Realisation, Rekursion, Selectionsort) . . . 334
Übungsaufgabe „Methode „fill()““ (Backtracking, Rekursion) . . . 335
Übungsaufgabe „Playlist“ (Einfach-verkettete Liste, Implementierung in Java, Doppelt-verkettete Liste, Rekursion) . . . 338
Aufgabe 2 . . . 343
Rekursion . . . 347

9 Suche **348**
Examensaufgabe „Unimodale Zahlenfolge“ (46115-2015-H.T1-A3) . . . 348
Aufgabe 3 . . . 348
Examensaufgabe „Bruchsicherheit von Smartphones“ (46115-2016-F.T2-A3) . . . 354
Examensaufgabe „Minimum und Maximum“ (46115-2021-F.T1-TA2-A2) 356
Examensaufgabe „Lineare und Binäre Suchverfahren“ (46115-2021-F.T2-TA2-A3) . . . 360
Examensaufgabe „Schnelle Suche von Schlüsseln: odd-ascending-even-descending-Folge“ (66115-2020-H.T2-TA2-A5) . . . 362
Examensaufgabe „Minimum und Maximum“ (66115-2021-F.T1-TA2-A2) 365
Examensaufgabe „Code-Inspection bei Binärer Suche“ (66116-2017-H.T1-TA2-A4) . . . 369
Übungsaufgabe „Methode „sucheBinaer()““ (Binäre Suche) . . . 373

10 Sortieralgorithmen **375**
Examensaufgabe „Schreibtischlauf Haldensortierung“ (46115-2013-F.T2-A6) . . . 375
Aufgabe 6 . . . 375
Examensaufgabe „Bubble- und Quicksort bei 25,1,12,27,30,9,33,34,18,16“ (46115-2016-F.T1-A8) . . . 377
Examensaufgabe „händisch sortieren, implementieren, Komplexität“ (46115-2017-F.T2-A4) . . . 378
Examensaufgabe „(Sortierverfahren)“ (46115-2019-H.T1-A4) . . . 380
Examensaufgabe „Pseudo-Code Insertionsort, Bubblesort, Quicksort“ (46115-2021-F.T1-TA2-A1) . . . 385
Examensaufgabe „Qualitätssicherung, Testen bei Bubblesort“ (46116-2017-H.T1-TA1-A4) . . . 386

Examensaufgabe „AVL-Baum, Dijkstra, Tiefensuche“ (66115-2006-H.T1-A4) . . . 388
Examensaufgabe „Selectionsort“ (66115-2014-H.T2-A6) . . . 391
Examensaufgabe „Haldensortierung“ (66115-2015-H.T2-A2) . . . 393
Examensaufgabe „1 45 8 53 9 2 17 10“ (66115-2016-F.T1-A6) . . . 395
Examensaufgabe „Sortieren mit Quicksort“ (66115-2016-H.T2-A7) . . . 397
Examensaufgabe „Top-Level-Domains (TLD)“ (66115-2017-F.T1-A2) . . . 402
Examensaufgabe „Quicksort“ (66115-2018-F.T2-A7) . . . 405
Examensaufgabe „Sortieren von 15,4,10,7,1,8,10 mit Bubble- und Selectionsort“ (66115-2018-H.T2-A8) . . . 406
Examensaufgabe „Notation des Informatik-Duden“ (66115-2019-H.T1-A5) 409
Examensaufgabe „Sortieren“ (66115-2021-F.T1-TA2-A1) . . . 414
Examensaufgabe „Sort-Methode und datenflussorientierte Überdeckungskritierien“ (66116-2016-H.T1-TA2-A3) . . . 421
Übungsaufgabe „Händisch Quick- und Mergesort“ (Mergesort, Quicksort) 424
Übungsaufgabe „Händisches Sortieren“ (Bubblesort, Mergesort, Quicksort) . . . 425
Übungsaufgabe „Sortier-Vorlage“ (Selectionsort, Bubblesort) . . . 430

11 Algorithmische Komplexität (O-Notation) 433
Examensaufgabe „Methoden „matrixSumme()“ und „find()““ (46115-2016-H.T2-A2) . . . 433
Examensaufgabe „Nächstes rot-blaues Paar auf der x-Achse“ (46115-2020-F.T2-A6) . . . 435
Examensaufgabe „O-Notation a(), b(), c(), d(), e(n)“ (46115-2021-F.T2-TA2-A1) . . . 437
Examensaufgabe „4 Rekursionsgleichungen“ (66115-2011-F.T1-A1) . . . 438
Examensaufgabe „limes“ (66115-2012-H.T2-A6) . . . 442
Examensaufgabe „Klasse „Stapel“ mit Methode „merge()““ (66115-2014-H.T2-A5) . . . 443
Examensaufgabe „Sortieren mit Stapel“ (66115-2015-F.T2-A5) . . . 449
Examensaufgabe „Methode „m()““ (66115-2018-F.T2-A6) . . . 455
Examensaufgabe „Sortieren von O-Klassen“ (66115-2019-H.T1-A6) . . . 457
Examensaufgabe „Mastertheorem“ (66115-2019-H.T2-A6) . . . 459
Examensaufgabe „Nächstes rot-blaues Paar auf der x-Achse“ (66115-2020-F.T2-A8) . . . 461
Examensaufgabe „O-Notation“ (66115-2020-H.T1-TA2-A4) . . . 463
Übungsaufgabe „Algorithmen-Vergleich“ (Algorithmische Komplexität (O-Notation)) . . . 466
Übungsaufgabe „Klasse-QueueElement“ (Algorithmische Komplexität (O-Notation)) . . . 468
Übungsaufgabe „Methode „magicStaff()““ (Algorithmische Komplexität (O-Notation)) . . . 469
Übungsaufgabe „Polynome f(n) in g(n)“ (Algorithmische Komplexität (O-Notation)) . . . 470

Übungsaufgabe „mehrere Funktionen“ (Algorithmische Komplexität (O-Notation)) 471

12 Master-Theorem **472**

13 Algorithmenmuster **473**
Examensaufgabe „Quicksort“ (46114-2008-H.T2-A3) 473
Examensaufgabe „Methode „a()““ (46115-2016-H.T2-A4) 474
Examensaufgabe „Primzahl“ (46115-2017-H.T2-A3) 478
Examensaufgabe „Springerproblem beim Schach“ (46115-2018-H.T2-A5) 482
Examensaufgabe „maximale Teilsumme“ (66115-2012-H.T1-A4) 485
Examensaufgabe „Zahl der Inversionen von A“ (66115-2016-H.T1-A4) . 488
Examensaufgabe „Rucksackproblem“ (66115-2018-H.T2-A6) 490
Examensaufgabe „Muffinsorten“ (66115-2019-F.T1-A6) 493
Examensaufgabe „Gutschein“ (66115-2020-H.T2-TA2-A4) 495
Übungsaufgabe „Münzwechsler“ (Greedy-Algorithmus) 499
Übungsaufgabe „Wechselgeld“ (Greedy-Algorithmus) 500
Formalisierung 500
Übungsaufgabe „Hanoi“ (Teile-und-Herrsche (Divide-and-Conquer)) . 503
Übungsaufgabe „Wegberechnung im Gitter“ (Dynamische Programmierung) 504
Übungsaufgabe „Damenproblem“ (Implementierung in Java, Backtracking) 506
Übungsaufgabe „Nikolaus“ (Backtracking) 508
Realisierung des Programms 509

14 Listen **513**
Examensaufgabe „dfs-number Graph s,a-h“ (46115-2014-H.T1-A8) . . . 513
Aufgabe 8 513
Examensaufgabe „Mystery-Stacks“ (46115-2019-H.T1-A6) 515
Examensaufgabe „Java-Klasse Stack“ (46115-2021-F.T2-TA2-A2) 518
Examensaufgabe „Reiseunternehmen“ (46116-2010-F.T1-A1) 523
Examensaufgabe „Firmenstruktur“ (46116-2011-F.T1-TA2-A1) 526
Firmenstruktur 526
Examensaufgabe „Klassen „QueueElement“ und „Queue““ (66115-2007-F.T1-A7) 529
Examensaufgabe „DoubleLinkedList“ (66115-2021-F.T2-TA2-A2) 533
Examensaufgabe „Getränkelieferservice“ (66116-2012-H.T2-TA2-A1) . . 536
Examensaufgabe „MyList Kompositium“ (66116-2021-F.T1-TA1-A5) . . 542
Übungsaufgabe „Maut“ (Einfach-verkettete Liste, Klassendiagramm, Kompositum (Composite)) 544
Übungsaufgabe „Wörterbuch“ (Einfach-verkettete Liste, Kompositum (Composite)) 545
Übungsaufgabe „Tellerstapel-Biberschlagen“ (Warteschlange (Queue), Stapel (Stack)) 550
Übungsaufgabe „Informatik-Biber“ (Stapel (Stack)) 551

Übungsaufgabe „Sackbahnhof“ (Stapel (Stack)) 552

15 Bäume **553**
Examensaufgabe „Binäre Suchbäume, AVL-Bäume, Implementierung“ (46115-2010-F.T1-A5) 553
5. Datenstrukturen und Algorithmen: Binäre Suchbäume und AVL-Bäume . 553
Examensaufgabe „2-3-4-Baum“ (46115-2011-F.T1-A3) 558
Examensaufgabe „Graph A-F“ (46115-2012-F.T1-A6) 564
Aufgabe 6 564
Examensaufgabe „Hashing mit Modulo 11“ (46115-2013-F.T2-A4) 566
„Streuspeicherung“ 566
Examensaufgabe „Zahlenfolge in binärer Suchbaum, Min-Heap und AVL-Baum“ (46115-2014-F.T1-A7) 568
Examensaufgabe „Binärer Suchbaum 17, 7, 21, 3, 10, 13, 1, 5“ (46115-2014-F.T2-A3) 571
Examensaufgabe „Hashing mit Modulo 8“ (46115-2015-H.T2-A1) 578
Examensaufgabe „Halden - Heaps“ (46115-2017-H.T2-A6) 581
Examensaufgabe „Aussage widerlegen“ (46115-2019-F.T2-A3) 582
Examensaufgabe „Heapify“ (46115-2019-H.T2-A7) 584
Examensaufgabe „Sondierfolgen für Hashing mit offener Adressierung“ (46115-2021-F.T1-TA2-A4) 586
Examensaufgabe „Klasse „BinBaum““ (66112-2003-H.T2-A8) 587
Examensaufgabe „Hashing mit Modulo 7“ (66112-2005-F.T2-A8) 596
Examensaufgabe „B-Baum m=2“ (66112-2005-H.T2-A6) 597
Examensaufgabe „3,5,1,2,4 in leerer Suchbaum und Heap“ (66115-2012-H.T2-A7) 600
Examensaufgabe „AVL 15,9,25,4,10,23,33,2,27; Einfüge 1,28; Löschen 15“ (66115-2012-H.T2-A8) 605
Examensaufgabe „IP und ULR mit Hashes“ (66115-2013-F.T1-A6) 608
Examensaufgabe „Heap und binärer Suchbaum“ (66115-2013-H.T2-A7) 610
Examensaufgabe „AVL-Baum 12,5,20,2,9,16,25,3,21“ (66115-2013-H.T2-A8) 614
Examensaufgabe „Binäre Bäume“ (66115-2014-F.T1-A2) 616
Examensaufgabe „Einfügen und dreimal einen Knoten löschen“ (66115-2014-F.T1-A3) 619
Examensaufgabe „Hashing mit verketteten Listen und offener Adressierung“ (66115-2016-F.T2-A4) 624
Examensaufgabe „Vergleich Suchbäume“ (66115-2016-F.T2-A7) 626
Examensaufgabe „Binärbaum, Halde, AVL“ (66115-2017-H.T2-A8) . . . 628
Examensaufgabe „AVL-Baum 5,14,28,10,3,12,13“ (66115-2018-F.T2-A8) . 633
Examensaufgabe „Binärer Suchbaum“ (66115-2018-H.T1-A5) 637
Examensaufgabe „k-kleinste Elemente“ (66115-2019-F.T2-A1) 640
Examensaufgabe „Binärer Baum mit Methode foo“ (66115-2019-H.T1-A7) 641
Examensaufgabe „AVL-Baum 10,5,25,14 erweitern und Knoten entfernen“ (66115-2019-H.T2-A7) 643
Examensaufgabe „Hashing mit mod 11 und 13“ (66115-2019-H.T2-A9) . 646

Examensaufgabe „Niedrigster gemeinsame Vorfahre“ (66115-2020-F.T2-A10) . . . 647
Examensaufgabe „Bäume“ (66115-2020-H.T1-TA2-A2) . . . 650
Examensaufgabe „Streuspeicherung“ (66115-2020-H.T1-TA2-A5) . . . 655
Examensaufgabe „Vornamen“ (66115-2020-H.T2-TA2-A1) . . . 659
Examensaufgabe „DeleteMin“ (66115-2020-H.T2-TA2-A3) . . . 662
Examensaufgabe „Binärbäume“ (66115-2021-F.T2-TA2-A3) . . . 665
Examensaufgabe „Hashing“ (66115-2021-F.T2-TA2-A5) . . . 669
Examensaufgabe „B-Baum k=2“ (66116-2013-F.T2-TA1-A3) . . . 671
Examensaufgabe „Indexstrukturen“ (66116-2015-F.T2-TA1-A3) . . . 673
Examensaufgabe „B-Baum der Ordnung 3“ (66116-2015-H.T2-TA1-A3) . 674
Examensaufgabe „Aufbau eines B-Baums“ (66116-2017-F.T1-TA1-A2) . . 675
Examensaufgabe „Physische Datenstrukturen)“ (66116-2020-F.T1-TA2-A6) . . . 676
Übungsaufgabe „Binärbaum: Klassendiagramm und Implementierung“ (Binärbaum, Klassendiagramm, Implementierung in Java) . . . 677
Übungsaufgabe „AVL-Baum 2, 8, 10, 1, 4, 5, 11“ (AVL-Baum) . . . 683
Übungsaufgabe „Einfügen und Löschen in B-Bäumen“ (B-Baum) . . . 685
Übungsaufgabe „Löschen in B-Bäumen“ (B-Baum) . . . 687
Übungsaufgabe „Modulo 11 und 17“ (Hashfunktion, Streutabellen (Hashing), Separate Verkettung, Lineares Sondieren, Quadratisches Sondieren) . . . 689
Übungsaufgabe „“ (Streutabellen (Hashing)) . . . 694
Übungsaufgabe „plus und minus i hoch 2“ (Streutabellen (Hashing)) . 695

16 Graphen **696**
Examensaufgabe „Adjazenzmatrix und Adjazenzliste“ (46114-2006-F.T2-A6) . . . 696
Aufgabe 6 (Graphrepräsentation) . . . 696
Examensaufgabe „Dijkstra“ (46114-2008-H.T1-A2) . . . 697
Examensaufgabe „Prim nach Adjazenzmatrix, Tripelnotation“ (46115-2018-F.T1-A8) . . . 699
Examensaufgabe „Graph a-f“ (46115-2018-F.T2-A4) . . . 702
Examensaufgabe „Schwach zusammenhängend gerichteter Graph“ (46115-2021-F.T1-TA2-A3) . . . 704
Examensaufgabe „Graph a-i“ (46115-2021-F.T2-TA2-A4) . . . 705
Examensaufgabe „Städte gemischt gerichtet / ungerichtet“ (66112-2004-F.T1-A5) . . . 707
Examensaufgabe „Graph a-g“ (66115-2013-H.T2-A9) . . . 709
Examensaufgabe „Karlsruhe nach Kassel“ (66115-2016-F.T2-A6) . . . 711
Examensaufgabe „Bayerischee Autobahnen“ (66115-2017-F.T1-A1) . . . 713
Examensaufgabe „Graph a-h“ (66115-2018-F.T2-A10) . . . 716
Examensaufgabe „Graph a-g, Startknoten s“ (66115-2018-F.T2-A11) . . . 719
Examensaufgabe „Negative Kantengewichte“ (66115-2018-F.T2-A9) . . . 720
Examensaufgabe „Graph A-E“ (66115-2020-H.T1-TA2-A3) . . . 723

Examensaufgabe „Schwach zusammenhängend gerichteter Graph“ (66115-2021-F.T1-TA2-A3) . . . 724
Examensaufgabe „Kürzeste-Wege-Bäume und minimale Spannbäume“ (66115-2021-F.T1-TA2-A4) . . . 725
Übungsaufgabe „Graph A-I“ (Algorithmus von Dijkstra) . . . 727
Übungsaufgabe „Städte A-F“ (Algorithmus von Dijkstra) . . . 729
Übungsaufgabe „Minimaler Spannbaum A-H“ (Minimaler Spannbaum, Algorithmus von Prim) . . . 731
Übungsaufgabe „Knoten-1-20“ (Breitensuche, Tiefensuche) . . . 734

17 Sonstige **738**
Examensaufgabe „Algorithmenanalyse“ (66115-2020-H.T1-TA2-A1) . . . 738

III Softwaresysteme (SOSY) 743

18 Projektmanagement **744**
Examensaufgabe „modernen Softwaretechnologie. 3 Begriffe in 3 Sätzen“ (46116-2013-F.T1-TA1-A2) . . . 744
Examensaufgabe „Multiple-Choice: Allgemeine SWT, Vorgehensmodelle und Requirements“ (46116-2014-H.T2-TA1-A1) . . . 745
Aufgabe 1: Allgemeine SWT, Vorgehensmodelle und Requirements Engineering . . . 745
Examensaufgabe „Vorgehensmodelle“ (46116-2014-H.T2-TA1-A2) . . . 747
Examensaufgabe „Vermischte Softwaresysteme-Fragen“ (66116-2013-H.T1-TA2-A3) . . . 751
Examensaufgabe „Softwaresysteme: Begriffe und Konzepte“ (66116-2016-H.T1-TA2-A1) . . . 754
Examensaufgabe „Wahrheitsgehalt-Tabelle Software Engineering“ (66116-2016-H.T2-TA2-A1) . . . 758
Examensaufgabe „Lebenszyklus“ (66116-2020-H.T1-TA1-A5) . . . 760
Examensaufgabe „Wissensfragen“ (66116-2020-H.T2-TA1-A1) . . . 762
Examensaufgabe „Entwicklungsprozesse“ (66116-2021-F.T2-TA1-A2) . . . 765
Übungsaufgabe „Multiple-Choice Allgemeine Software-Technologie“ (EXtreme Programming, V-Modell, Wasserfallmodell, SCRUM, Prototyping, Unit-Test, Anforderungsanalyse) . . . 768
Übungsaufgabe „Teacher-Data“ (Nicht-funktionale Anforderungen, Funktionale Anforderungen, Wasserfallmodell, Spiralmodell, V-Modell, Evolutionäre Softwaremodelle, Inkrementelle Prozessmodelle, SCRUM) 770
Übungsaufgabe „Grundwissen“ (White-Box-Testing, Black-Box-Testing, Funktionalorienteres Testen, V-Modell) . . . 779

19 Modellierung **780**
Examensaufgabe „Hotel-Verwaltung“ (46116-2012-F.T2-TA2-A1) . . . 780
Hotel-Verwaltung . . . 780
Examensaufgabe „CreditCard, Order“ (46116-2013-F.T2-TA1-A1) . . . 782

Examensaufgabe „Bestellsystem“ (46116-2014-H.T2-TA1-A3) 785
Examensaufgabe „Geldautomat“ (46116-2017-F.T1-TA1-A2) 787
Examensaufgabe „Korrektheit von UML-Diagrammen“ (46116-2017-F.T1-TA1-A3) 788
Examensaufgabe „Entwurfsmuster bei Bankkonten, Hockeyspiel, Dateisystem“ (46116-2017-H.T2-TA1-A3) 790
Examensaufgabe „Fußballweltmeisterschaft“ (46116-2018-F.T1-TA1-A3) 791
Examensaufgabe „Verhaltens-Modellierung mit Zustandsdiagrammen. Digitaluhr“ (46116-2018-H.T1-TA1-A3) 795
Examensaufgabe „Banksystem“ (66112-2002-H.T1-A4) 797
Examensaufgabe „Klasse „DoublyLinkedList““ (66112-2005-F.T1-A1) . . 803
Examensaufgabe „Wahlsystem“ (66116-2014-H.T2-TA2-A2) 807
Examensaufgabe „Radiotuner“ (66116-2015-F.T1-TA2-A2) 808
Examensaufgabe „Kunden und Angestellte einer Firma“ (66116-2015-F.T1-TA2-A3) 809
Examensaufgabe „OOP/OOD - Reverse Engineering“ (66116-2015-F.T2-TA2-A2) 813
Examensaufgabe „Entwurfsmuster in UML-Diagramm erkennen“ (66116-2016-F.T1-TA2-A2) 816
Examensaufgabe „PKI-System Lehrer Schüler“ (66116-2016-H.T1-TA2-A2) 820
Examensaufgabe „UML-Diagramme entspreched Java-Code zeichnen“ (66116-2018-F.T2-TA2-A1) 823
Examensaufgabe „Countdown und Observer“ (66116-2018-F.T2-TA2-A2) 826
Examensaufgabe „Beatles“ (66116-2018-H.T1-TA1-A2) 830
Examensaufgabe „Grafik: Kreis, Quadrat, Dreieck“ (66116-2019-F.T1-TA2-A1) 834
Examensaufgabe „Roboter in einer Montagehalle“ (66116-2019-F.T1-TA2-A2) 835
Examensaufgabe „Critical Path Method“ (66116-2019-H.T1-TA1-A1) . . 837
Examensaufgabe „Zustand-Entwurfsmuster bei Verwaltung von Prozessen“ (66116-2019-H.T1-TA1-A4) 838
Examensaufgabe „Bankautomat“ (66116-2020-F.T1-TA1-A1) 844
Examensaufgabe „Ticket-Handel“ (66116-2020-H.T1-TA1-A3) 846
Examensaufgabe „Objektorientierte Analyse“ (66116-2020-H.T2-TA1-A4) 851
Examensaufgabe „Terme über die Rechenarten“ (66116-2020-H.T2-TA1-A5) 853
Examensaufgabe „Elementtypen UML-Klassendiagramm“ (66116-2021-F.T1-TA1-A3) 857
Examensaufgabe „Wissen Erbauer“ (66116-2021-F.T1-TA1-A6) 858
Examensaufgabe „MyParser“ (66116-2021-F.T1-TA1-A7) 860
Übungsaufgabe „Fußballmeisterschaft“ (Klassendiagramm) 861
Übungsaufgabe „Gasthausen“ (Klassendiagramm) 862
Übungsaufgabe „Kleintierpraxis“ (Klassendiagramm, Vererbung) . . . 863
Übungsaufgabe „Universitätsverwaltung“ (Klassendiagramm) 867

Übungsaufgabe „Alle UML-Diagramme“ (UML-Diagramme, Klassendiagramm, Objektdiagramm, Zustandsdiagramm Wissen, Sequenzdiagramm, Aktivitätsdiagramm, Anwendungsfalldiagramm, Kommunikationsdiagramm) 868
Übungsaufgabe „Bankkonten“ (Vererbung, Generalisierung, Spezialisierung, Klassendiagramm, Implementierung in Java) 869
Übungsaufgabe „Kleintierpraxis“ (Vererbung, Klassendiagramm, Implementierung in Java) 873
Übungsaufgabe „Grundwissen“ (Entwurfsmuster) 877

20 Projektplanung **878**
Examensaufgabe „Gantt und CPM“ (46116-2015-F.T1-TA1-A3) 878
Examensaufgabe „Gantt und PERT“ (46116-2015-H.T1-TA1-A3) 880
Examensaufgabe „Petri-Netz“ (46116-2016-F.T2-TA1-A2) 882
Examensaufgabe „Gantt und PERT“ (46116-2017-F.T1-TA1-A5) 884
Examensaufgabe „Gantt und CPM“ (66116-2012-H.T2-TA2-A2) 886
Examensaufgabe „Automatisierungsanlage mit zwei Robotern“ (66116-2015-F.T2-TA2-A3) 889
Examensaufgabe „Gantt und zwei Softwareentwickler“ (66116-2018-H.T2-TA1-A1) 891
Examensaufgabe „Projektplanung“ (66116-2020-H.T1-TA1-A2) 893
Examensaufgabe „Projektmanagement“ (66116-2020-H.T2-TA1-A2) . . . 897
Examensaufgabe „Projektmanagement“ (66116-2021-F.T2-TA1-A1) . . . 899
Übungsaufgabe „Alles“ (Petri-Netz, Erreichbarkeitsgraph) 904
Übungsaufgabe „Erreichbarkeitsgraph“ (Petri-Netz, Erreichbarkeitsgraph) 906
Übungsaufgabe „Modellierung“ (Petri-Netz) 908
Übungsaufgabe „Rechnen“ (Petri-Netz) 909
Übungsaufgabe „CPM und Gantt“ (CPM-Netzplantechnik) 910
Übungsaufgabe „CPM-Netzwerk“ (CPM-Netzplantechnik) 914
Übungsaufgabe „CPM mit Scheinvorgang“ (CPM-Netzplantechnik) . . 916
Übungsaufgabe „Anordnungsbeziehungen“ (Gantt-Diagramm) 918

21 Softwarearchitektur **920**
Examensaufgabe „Softwarearchitektur und Agilität“ (66116-2019-H.T2-TA1-A3) 920
Examensaufgabe „AJAX“ (66116-2021-F.T1-TA1-A10) 922
Examensaufgabe „HTTP“ (66116-2021-F.T1-TA1-A11) 923
Examensaufgabe „Richtig-Falsch“ (66116-2021-F.T1-TA1-A12) 924
Examensaufgabe „Client-Server-Modell“ (66116-2021-F.T1-TA1-A8) . . . 926
Examensaufgabe „Client-Server-Technologien“ (66116-2021-F.T1-TA1-A9) 927

22 Testen **928**
Examensaufgabe „Methode function: Formale Verifikation - Induktionsbeweis“ (46115-2015-H.T2-A4) 928
Examensaufgabe „Hanoi“ (46116-2014-F.T2-TA1-A1) 930
Examensaufgabe „Methode „isPalindrom()““ (46116-2015-H.T2-TA1-A2) 932

Examensaufgabe „ASCII“ (46116-2015-H.T2-TA1-A3) 936
Examensaufgabe „Catalan-Zahl“ (46116-2016-H.T2-TA1-A4) 942
Examensaufgabe „drei hoch“ (66112-2003-H.T2-A5) 946
Examensaufgabe „Methode „sumOfSquares()““ (66115-2017-F.T1-A4) . 947
Examensaufgabe „Methode „specialSums()““ (66116-2014-H.T2-TA2-A3) 949
Examensaufgabe „Methode „doubleFac()“: wp-Kalkül und Schleifeninvariante“ (66116-2015-H.T2-TA2-A3) 952
Examensaufgabe „Methode „binToInt()“ und Kontrollflussgraph“ (66116-2017-F.T2-TA2-A1) 957
Examensaufgabe „wp-Kalkül mit Invariante bei Methode „mul()““ (66116-2017-F.T2-TA2-A4) 962
Examensaufgabe „Roboter in einer Montagehalle“ (66116-2019-F.T1-TA2-A3) 966
Examensaufgabe „Test-getriebene Entwicklung“ (66116-2019-F.T2-TA2-A1) 967
Examensaufgabe „White-Box-Tests“ (66116-2019-H.T1-TA1-A3) 969
Examensaufgabe „Verifikation“ (66116-2020-H.T1-TA1-A1) 972
Examensaufgabe „White-Box-Testverfahren“ (66116-2020-H.T1-TA1-A4) 975
Übungsaufgabe „Gaußsche Summenformel“ (Vollständige Induktion) . 980
Übungsaufgabe „Geometrische Summenformel geoSum()“ (Vollständige Induktion) 983
Übungsaufgabe „Summe ungerader Zahlen (Maurolicus 1575)“ (Vollständige Induktion) 985
Übungsaufgabe „Grundwissen“ (Formale Verifikation, wp-Kalkül, Hoare-Kalkül, Partielle Korrektheit, Totale Korrektheit, Invariante, Terminierungsfunktion) 987
Übungsaufgabe „Methode „f()““ (wp-Kalkül) 989
Übungsaufgabe „Vorlesungsaufgabe WP-Kalkül“ (wp-Kalkül) 992
Übungsaufgabe „Größter gemeinsamer Teiler“ (Datenfluss-annotierter Kontrollflussgraph, Zyklomatische Komplexität nach Mc-Cabe, C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)) 993
Übungsaufgabe „Methode „log()““ (Kontrollflussgraph, Überdeckbarkeit, C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)) 996

IV Theoretische Informatik (THEO) 998

23 Reguläre Sprache 999
Examensaufgabe „Alphabet ab“ (46115-2010-F.T2-A1) 999
Aufgabe 1 999
Examensaufgabe „Sprache abc“ (46115-2015-F.T1-A1) 1001
Aufgabe 1 1001
Examensaufgabe „Alphabet ab, vorvorletztes Zeichen a“ (46115-2016-H.T1-A1) 1002

Examensaufgabe „Reguläre Sprachen)“ (46115-2019-H.T1-A1) 1004
Examensaufgabe „Komplemetieren eines NEA“ (46115-2019-H.T2-A1) . 1005
Examensaufgabe „Rechtslineare Grammatik“ (46115-2019-H.T2-A2) . . 1007
Examensaufgabe „Reguläre Sprache“ (46115-2020-F.T1-A1) 1009
Examensaufgabe „Minimierung von Endlichen Automaten“ (46115-2021-F.T1-TA1-A3) 1012
Examensaufgabe „Alphabet abc“ (46115-2021-F.T2-TA1-A1) 1013
Examensaufgabe „NEA ab“ (46115-2021-F.T2-TA1-A2) 1015
Examensaufgabe „L1, L2, L3 regulär oder kontextfrei“ (46115-2021-F.T2-TA1-A3) 1017
Examensaufgabe „Reguläre Sprache“ (66115-2007-H.T2-A1) 1018
Examensaufgabe „NEA und Minimalisierung“ (66115-2012-H.T1-A1) . 1020
Examensaufgabe „Minimierung DFA“ (66115-2013-H.T2-A3) 1023
Examensaufgabe „Alphabet „01“ Anzahl Unterschied höchstes 3“ (66115-2015-F.T1-A1) 1025
Examensaufgabe „Reguläre Sprachen“ (66115-2016-F.T1-A1) 1029
Examensaufgabe „Exponentieller Blow-Up“ (66115-2018-F.T2-A3) 1032
Examensaufgabe „NEA nach DEA“ (66115-2019-F.T1-A2) 1035
Examensaufgabe „Automaten mit Zuständen q, r, s, t“ (66115-2020-F.T1-A2) 1038
Examensaufgabe „Vermische Fragen“ (66115-2020-H.T1-TA1-A1) 1040
Examensaufgabe „Reguläre Sprache xyz“ (66115-2020-H.T1-TA1-A2) . . 1042
Examensaufgabe „Palindrom über Alphabet „abc““ (66115-2020-H.T1-TA1-A3) 1045
Examensaufgabe „Minimierungsalgorithmus“ (66115-2020-H.T2-TA1-A1) 1046
Examensaufgabe „Reguläre Sprachen Automaten zuordnen“ (66115-2021-F.T1-TA1-A1) 1050
Examensaufgabe „Reguläre Sprachen“ (66115-2021-F.T2-TA1-A1) 1053
Übungsaufgabe „Grammatik aus Automat“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA), Reguläre Grammatik) . . . 1054
Übungsaufgabe „NEA-DEA-Aequivalenzklassen“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA), Minimierungsalgorithmus, Reguläre Ausdrücke, Äquivalenzklassen) 1056
Übungsaufgabe „Noten“ (Reguläre Sprache) 1059
Übungsaufgabe „Vorlesungsaufgaben“ (Reguläre Grammatik) 1060
Übungsaufgabe „Deterministischer endlicher Automat“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA)) 1062
Übungsaufgabe „Vorlesungsaufgaben“ (Deterministisch endlicher Automat (DEA)) 1063
Übungsaufgabe „Vorlesungsaufgaben“ (Nichtdeterministisch endlicher Automat (NEA)) 1064
Übungsaufgabe „Studiflix-Minimierung“ (Minimierungsalgorithmus) . 1065
Übungsaufgabe „Minimalisierung“ (Minimierungsalgorithmus) 1067
Übungsaufgabe „NEA: z012, Alphabet: abc“ (Erweiteter Potenzmengenalgorithmus) 1069

Übungsaufgabe „NEA: z01234, Alphabet: ab“ (Erweiteter Potenzmengenalgorithmus) 1071
Übungsaufgabe „Vorlesungsaufgaben“ (Potenzmengenalgorithmus) . . 1073
Übungsaufgabe „Vorlesungsaufgaben“ (Potenzmengenalgorithmus) . . 1074
Übungsaufgabe „„w w““ (Pumping-Lemma (Reguläre Sprache)) 1075
Übungsaufgabe „„wn2“ „an bm cn““ (Pumping-Lemma (Reguläre Sprache)) 1076
Übungsaufgabe „w c wR“ (Pumping-Lemma (Reguläre Sprache)) . . . 1078
Übungsaufgabe „a n b m“ (Pumping-Lemma (Reguläre Sprache)) . . . 1080
Übungsaufgabe „Arztpraxis und Autohauskette“ (Reguläre Ausdrücke) 1081
Übungsaufgabe „Reguläre Grammatik, reguläre Ausdrücke und DEA“ (Reguläre Sprache, Reguläre Grammatik, Ableitung (Reguläre Sprache), Reguläre Ausdrücke, Deterministisch endlicher Automat (DEA))1083
Übungsaufgabe „Vorlesungsaufgaben“ (Reguläre Ausdrücke) 1087
Übungsaufgabe „Reguläre Sprache in kontextfreier Sprache“ (Reguläre Sprache, Kontextfreie Sprache) 1092

24 Kontextfreie Sprache **1093**
Examensaufgabe „Sprachen L1 und L2“ (46115-2019-H.T1-A2) 1093
Examensaufgabe „Nonterminale SABCD, Terminale ab“ (46115-2021-F.T1-TA1-A2) 1096
Examensaufgabe „Kontextfrei aber nicht regulär“ (66115-2012-F.T1-A3) 1097
Examensaufgabe „Nonterminale: SAB, Terminale: ab“ (66115-2012-F.T1-A4) 1099
Examensaufgabe „Kontextfreie Grammatiken“ (66115-2013-F.T1-A2) . . 1101
Examensaufgabe „Nonterminale: SA, Terminale: 012“ (66115-2016-F.T1-A2) 1103
Examensaufgabe „Nonterminale: STU, Terminale: abcde“ (66115-2017-F.T2-A2) 1105
Examensaufgabe „Kontextfreie Sprachen“ (66115-2017-H.T1-A2) 1108
Examensaufgabe „CYK mit fehlenden Zellen (T: SABC N: ab)“ (66115-2017-H.T2-A5) 1110
Examensaufgabe „Kontextfreie Sprachen“ (66115-2018-H.T1-A3) 1111
Examensaufgabe „Nonterminale: STU, Terminale: ab“ (66115-2019-F.T1-A3) 1112
Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-F.T1-A3) 1114
Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-F.T2-A3) 1116
Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-H.T2-TA1-A2) . 1118
Examensaufgabe „CYK mit Wort „aaaccbbb““ (66115-2021-F.T1-TA1-A2) 1119
Examensaufgabe „w w1 w w2“ (66115-2021-F.T2-TA1-A2) 1120
Übungsaufgabe „Ableitungen“ (Ableitung (Kontextfreie Sprache), Ableitungsbaum) 1122
Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache, Ableitung (Kontextfreie Sprache), Ableitungsbaum) 1124
Übungsaufgabe „Klammerausdrücke“ (Kontextfreie Sprache) 1127
Übungsaufgabe „Kontextfreie-Grammatik“ (Kontextfreie Grammatik) . 1131

Übungsaufgabe „(an bn)m“ (Kontextfreie Sprache, Kellerautomat) . . . 1132
Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache) 1135
Übungsaufgabe „AB5“ (CYK-Algorithmus) 1136
Übungsaufgabe „CYK-Algorithmus“ (CYK-Algorithmus) 1137
Übungsaufgabe „Foliensatz“ (CYK-Algorithmus) 1139
Übungsaufgabe „Youtube-Video Karsten-Morisse“ (CYK-Algorithmus) . 1140
Übungsaufgabe „Drei Grammatiken (SABCX, ST, SAB)“ (Chomsky-Normalform)1141
Übungsaufgabe „Vorlesungsaufgabe (S, SAB, SABCD)“ (Chomsky-Normalform)1143
Übungsaufgabe „0-1“ (Kontextfreie Grammatik) 1148
Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache, Ableitung (Kontextfreie Sprache), Kontextfreie Grammatik) 1149
Übungsaufgabe „Balancierte Klammern“ (Kellerautomat) 1151
Übungsaufgabe „an bn“ (Kellerautomat) 1152
Übungsaufgabe „zu drei Grammatiken“ (Kellerautomat) 1153
Übungsaufgabe „Konfigurationsfolge doppelt so viele b’s wie a’s“ (Kellerautomat) 1156
Übungsaufgabe „0n 1n, gleich viele ab, kein Präfix mehr Einsen“ (Kellerautomat) 1158
Übungsaufgabe „a hoch n c hoch i b hoch n“ (Kontextfreie Sprache, Kellerautomat, Kontextfreie Grammatik, Konfigurationsfolge) 1160
Übungsaufgabe „Nonterminal: P, Terminale: 01“ (Kellerautomat) 1163
Übungsaufgabe „an bn cn“ (Pumping-Lemma (Kontextfreie Sprache)) . 1165
Übungsaufgabe „„w w“ und „ak bl cm““ (Pumping-Lemma (Kontextfreie Sprache)) 1166
Übungsaufgabe „„an bn“ „c2n“ und „an bn2““ (Pumping-Lemma (Kontextfreie Sprache)) 1167

25 Kontextsensitive Sprache **1171**
Übungsaufgabe „Kontextsensitive Grammatik“ (Kontextsensitive Grammatik) 1171
Übungsaufgabe „Vorlesungsaufgaben kontextsensitive Grammatiken“ (Kontextsensitive Grammatik) 1172
Übungsaufgabe „Vorlesungsaufgaben Komplement der Binärzahl“ (Kontextsensitive Sprache) 1173

26 Unbeschränkte Sprache **1174**
Examensaufgabe „Turingmaschinen“ (46115-2013-F.T1-A4) 1174
Aufgabe 4 1174
Examensaufgabe „Berechen- und Entscheidbarkeit“ (66115-2017-F.T2-A3)1175
Examensaufgabe „Turingmaschine Konfigurationsfolge“ (66115-2019-F.T1-A4) 1177
Examensaufgabe „Multiplikation mit 3“ (66115-2019-H.T2-A1) 1179
Übungsaufgabe „Binärzahl dekrementieren“ (Turing-Maschine) 1182
Übungsaufgabe „Turing-Maschine Multiplikation“ (Turing-Maschine) . 1184
Übungsaufgabe „Übergangsfunktion“ (Turing-Maschine) 1188

Übungsaufgabe „Vorlesungsaufgaben ab-Wörter umkehren 2-Band-Turingmaschine“ (Turing-Maschine) 1191
Übungsaufgabe „Vorlesungsaufgaben ab-Wörter umkehren“ (Turing-Maschine) 1192
Übungsaufgabe „unäre Kodierung von n und m“ (Turing-Maschine) . . 1194
Übungsaufgabe „a-2-hoch-n“ (Unbeschränkte Sprache) 1195

27 Berechenbarkeit **1196**
Examensaufgabe „Gödelisierung aller Registermaschinen (RAMs)“ (66115-2015-F.T2-A4) 1196
Examensaufgabe „Verständnis Berechenbarkeitstheorie“ (66115-2016-F.T2-A2) 1197
Examensaufgabe „Registermaschinen (RAMs)“ (66115-2016-H.T2-A3) . 1199
Examensaufgabe „Berechenbarkeitstheorie“ (66115-2020-F.T2-A4) 1200
Übungsaufgabe „GOTO-Programme“ (GOTO-berechenbar) 1201
Übungsaufgabe „LOOP-Fakultät“ (LOOP-berechenbar) 1203
Übungsaufgabe „Vorlesungsaufgaben“ (Berechenbarkeit) 1204
Übungsaufgabe „Primitiv-rekursiv“ (Berechenbarkeit) 1206
Übungsaufgabe „Vorlesungsaufgaben“ (Berechenbarkeit) 1207

28 Entscheidbarkeit **1208**
Examensaufgabe „Halteproblem H m“ (66115-2014-F.T1-A5) 1208

29 Komplexitätstheorie **1209**
Examensaufgabe „NP“ (46115-2016-F.T1-A5) 1209
Examensaufgabe „SUBSET SUM, Raumausstattungsunternehmen“ (46115-2016-F.T2-A2) 1210
Examensaufgabe „VertexCover“ (46115-2016-H.T1-A4) 1212
Examensaufgabe „k-COL“ (66115-2016-F.T1-A5) 1214
Examensaufgabe „Verständnis“ (66115-2016-F.T2-A3) 1215
Examensaufgabe „SAT DOPPELSAT“ (66115-2019-H.T1-A4) 1217
Examensaufgabe „CLIQUE - ALMOST CLIQUE“ (66115-2021-F.T2-TA1-A4) 1220
Übungsaufgabe „Reduktion-Turingmaschine“ (Polynomialzeitreduktion) 1222
Übungsaufgabe „SAT-3SAT“ (Polynomialzeitreduktion) 1223

Teil I

Datenbanken (DB)

Datenbank-Übersicht

Examensaufgabe „Theoriefragen Datenbank“ (46116-2015-H.T1-TA2-A1)

Erläutern Sie die folgenden Begriffe in knappen Worten:

(a) Datenunabhängigkeit

Lösungsvorschlag

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

(b) Superschlüssel

Lösungsvorschlag

Ein Superschlüssel ist ein Attribut oder Attributkombination, von der alle Attribute einer Relation funktional abhängen.

(c) Referentielle Integrität

Lösungsvorschlag

Unter Referentieller Integrität verstehen wir Bedingungen, die zur Sicherung der Datenintegrität bei Nutzung relationaler Datenbanken beitragen können. Demnach dürfen Datensätze über ihre Fremdschlüssel nur auf existierende Datensätze verweisen.

Danach besteht die Referentieller Integrität grundsätzlich aus zwei Teilen:

(i) Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel oder einem eindeutigen Alternativschlüssel existiert.

(ii) Eine Datensatzlöschung oder Änderung des Schlüssels in einem Primär-Datensatz ist nur möglich, wenn zu diesem Datensatz keine abhängigen Datensätze in Beziehung stehen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/09/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Tupelidentifikator“ (46116-2017-F.T1-TA2-A4)

(a) Erläutern Sie in ein bis zwei Sätzen, aus welchen zwei Teilen sich ein TID (Tupelidentifikator) zusammensetzt.

Lösungsvorschlag

> Seitennummer (Seiten bzw. Blöcke sind größere Speichereinheiten auf der Platte) Relative Indexposition innerhalb der Seite

(b) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz innerhalb einer Seite verschoben werden muss.

Lösungsvorschlag

> Satzverschiebung innerhalb einer Seite bleibt ohne Auswirkungen auf TID,

(c) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz erstmalig in eine andere Seite verschoben werden muss.

Lösungsvorschlag

> wird ein Satz auf eine andere Seite migriert, wird eine „Stellvertreter-TID“ zum Verweis auf den neuen Speicherort verwendet. Die eigentliche TID-Adresse bleibt stabil

(d) Erläutern Sie in zwei bis drei Sätzen das Vorgehen, wenn ein durch einen TID adressierter und bereits einmal über Seitengrenzen hinweg verschobener Satz erneut in eine andere Seite verschoben werden muss.

Lösungsvorschlag

> Es wird eine neue stellvertreter TID aktualisiert.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/03/Thema-1/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Physische Datenorganisation“ (66116-2016-H.T1-TA1-A5)

Tupel-Identifikator
B-Baum

(a) Erläutern Sie die wesentliche Eigenschaft eines Tupel-Identifikators (TID) in ein bis zwei Sätzen.

Lösungsvorschlag

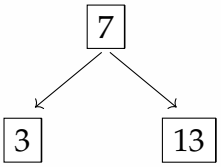
- Daten werden in Form von *Sätzen* auf der Festplatte abgelegt, um auf Sätze zugreifen zu können, verfügt jeder Satz über eine *eindeutige, unveränderliche Satzadresse*
- TID = Tupel Identifier: dient zur Adressierung von Sätzen in einem Segment und besteht aus zwei Komponenten:
  - Seitennummer (Seiten bzw. Blöcke sind größere Speichereinheiten auf der Platte)
  - Relative Indexposition innerhalb der Seite
- Satzverschiebung innerhalb einer Seite bleibt ohne Auswirkungen auf den TID. Wird ein Satz auf eine andere Seite migriert, wird eine „Stellvertreter-TID“ zum Verweis auf den neuen Speicherort verwendet. Die eigentliche TID-Adresse bleibt stabil.

(b) Fügen Sie in einen anfangs leeren B-Baum mit k = 1 (maximal 2 Schlüsselwerte pro Knoten) die im Folgenden gegebenen Schlüsselwerte der Reihe nach ein. Zeichnen Sie den Endzustand des Baums nach jedem Einfügevorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie die sieben Endzustände deutlich.

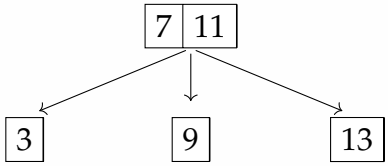
3, 7, 13, 11, 9, 10, 8

Lösungsvorschlag

- 3 (einfaches Einfügen)
- 7 (einfaches Einfügen)
- 13 (Split)

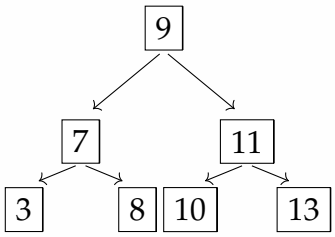


- 11 (einfaches Einfügen)
- 9 (Split)



- 10 (einfaches Einfügen)

- 8 (Doppel-Split)



(c) Gegeben ist der folgende B-Baum:

Die folgenden Teilaufgaben sind voneinander unabhängig.

(i) Löschen Sie aus dem gegebenen B-Baum den Schlüssel 3 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

(ii) Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 17 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

(iii) Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 43 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-1/Teilaufgabe-1/Aufgabe-5.tex`

Examensaufgabe „Allgemeine Fragen“ (66116-2019-F.T2-TA1-A1)

Datenbank

(a) Das ACID-Prinzip fordert unter anderem die Atomaritat und die Isolation einer Transaktion. Beschreiben Sie kurz die Bedeutung dieser Begriffe.

Lösungsvorschlag

Atomicity Eine Transaktion ist atomar, ðvon den vorgesehenen Änderungsoperationen auf die Datenbank haben entweder alle oder keine eine Wirkung auf die Datenbank.

Isolation Eine Transaktion bemerkt das Vorhandensein anderer (parallel ablaufender) Transaktionen nicht und beeinflusst auch andere Transaktionen nicht.

(b) Was versteht man unter physischer Datenunabhängigkeit?

Lösungsvorschlag

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

(c) In welche drei Ebenen unterteilt die 3-Ebenen-Architektur Datenbanksysteme?

Lösungsvorschlag

Externe Ebene / (Benutzer)sichten (Views) Die externe Ebene stellt den Benutzern und Anwendungen individuelle Benutzersichten bereit, wie beispielsweise Formulare, Masken-Layouts, Schnittstellen.

Konzeptionelle / logische Ebene Die konzeptionelle Ebene beschriebt, welche Daten in der Datenbank gespeichert sind, sowie deren Beziehungen. Ziel des Datenbankdesigns ist eine vollständige und redundanzfreie Darstellung aller zu speichernden Informationen. Hier findet die Normalisierung des relationalen Datenbankschemas statt.

Interne / physische Ebene Die interne Ebene stellt die physische Sicht der Datenbank im Computer dar. Sie beschreibt, wie und wo die Daten in der Datenbank gespeichert werden. Oberstes Designziel ist ein effizienter Zugriff auf die gespeicherten Informationen, der meist durch bewusst in Kauf genommene Redundanz (z. B. Index) erreicht wird.

(d) Definieren Sie, was ein Schlüssel ist.

Lösungsvorschlag

Ein Schlüssel ist ein Attribut oder eine Attributkombination, von der alle Attribute einer Relation funktional abhängen.

(e) Gegeben ist eine Relation $R(A, B, C)$, deren einziger Schlüsselkandidat A ist. Nennen Sie zwei Superschlüssel.

Lösungsvorschlag

{ A, B } oder { A, C } oder { A, B, C }

(f) Gegeben seien 2 Relationen $R(A, B, C)$ und $S(C, D, E)$. Die Relation R enthalte 9 Tupel und die Relation S enthalte 7 Tupel. Sei p ein beliebiges Selektionsprädikat. Gegeben seien außerdem folgende Anfragen:

Q_1:

```
SELECT *
FROM R NATURAL JOIN S
WHERE p;
```

Q_2:

```
SELECT DISTINCT A
FROM R LEFT OUTER JOIN S ON R.C=S.C;
```

(i) Wieviele Ergebnistupel liefert die Anfrage Q_1 mindestens?

Lösungsvorschlag

0

(ii) Wieviele Ergebnistupel liefert die Anfrage Q_1 höchstens?

Lösungsvorschlag

7

(iii) Wieviele Ergebnistupel liefert die Anfrage Q_2 mindestens?

Lösungsvorschlag

9

(iv) Wieviele Ergebnistupel liefert die Anfrage Q_2 höchstens?

Lösungsvorschlag

9

(g) Erläutern Sie, was es bedeutet, wenn ein Attribut prim ist.

Lösungsvorschlag

Sei R ein Relationenschema. Ein Attribut $A \in R$ heißt *prim*, falls A Teil eines Schlüsselkandidaten von R ist. Andernfalls heißt A *nichtprim*.

(h) Nennen Sie die drei Armstrong-Axiome, die dazu dienen aus einer Menge von funktionalen Abhängigkeiten, die auf einer Relation gelten, weitere funktionale Abhängigkeiten abzuleiten.

Lösungsvorschlag

Reflexivität: Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt, $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$.

Verstärkung: Gilt $\alpha \rightarrow \beta$, so gilt auch $\alpha\gamma \rightarrow \beta\gamma$ für jede Menge von Attributen γ der Relation.

Transitivität: Gilt $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, so gilt auch $\alpha \rightarrow \gamma$.

(i) Nennen Sie die höchste Normalform, der die Relation $R(A, B, C, D)$ mit den Abhängigkeiten $\{A\} \rightarrow \{C\}$ und $\{A, B\} \rightarrow \{D\}$ entspricht. Begründen Sie Ihre Antwort.

Lösungsvorschlag

1NF. $\{A\} \rightarrow \{C\}$ verletzt die 2NF, da das Nichtprimärattribut C funktional von einer echten Teilmenge (A) des Schlüsselkandidaten $(\{A, B\} \rightarrow \{D\})$ abhängt.

(j) Gegeben seien die Transaktionen T_1, T_2 und T_3. Wie viele mögliche verschiedene serialisierbare Schedules gibt es mindestens? (Geben Sie Ihren Rechenweg an.)

Lösungsvorschlag

Vergleiche Kemper Seite 341 Kapitel „Theorie der Serialisierbarkeit“.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Vermischte Datenbank-Fragen“ (66116-2019-H.T2-TA2-A6)

Datenbank-Übersicht
Natural-Join
Equi-Join
Theta-Join

Begründen oder erläutern Sie Ihre Antworten.

(a) Erklären Sie kurz den Unterschied zwischen einem Natural-Join und einem Equi-Join.

Lösungsvorschlag

Ein Natural Join ist eine Kombination von zwei Tabellen, in denen Spalten gleichen Namens existieren. Die Werte in diesen Spalten werden sodann auf Übereinstimmungen geprüft (analog Equi-Join). Einige Datenbanksysteme erkennen das Schlüsselwort NATURAL und eliminieren entsprechend automatisch doppelte Spalten.

Während beim Kreuzprodukt keinerlei Anforderungen an die Kombination der Datensätze gestellt werden, führt der Equi-Join eine solche ein: Die Gleichheit von zwei Spalten.

a

[a] wiki.selfhtml.org

(b) Erläutern Sie kurz was man unter einem Theta-Join versteht.

Lösungsvorschlag

Ein Theta-Join ist eine Verbindung von Relationen bezüglich beliebiger Attribute und mit einem Selektionsprädikat. [a]

[a] `https://www.datenbank-grundlagen.de/theta-join.html`

(c) Was versteht man unter Unionkompatibilität? Nennen Sie drei SQL-Operatoren welche Unionkompatibilität voraussetzen.

Lösungsvorschlag

Bestimmte Operationen der relationalen Algebra wie Vereinigung, Schnitt und Differenz verlangen Unionkompatibilität. Unionkompatibilität ist eine Eigenschaft des Schemas einer Relation. Zwei Relationen R und S sind genau dann union-kompatibel, wenn folgende Bedingungen erfüllt sind:

(i) Die Relationen R und S besitzen dieselbe Stelligkeit n, ðsie haben die selbe Anzahl von Spalten.

(ii) Für alle Spalten der Relationen gilt, dass die Domäne der i-ten Spalte der Relation R mit dem Typ der i-ten Spalte der Relation S übereinstimmt $(0 < i < n)$.

Die Namen der Attribute spielen dabei keine Rolle. [a]

SQL-Operatoren mit Unionkompatibilität

- UNION
- INTERSECT
- EXCEPT

[a] https://studylibde.com/doc/1441274/übungstool-für-relationale-algebra

(d) Erläutern Sie Backward und Forward Recovery und grenzen Sie diese voneinander ab.

(e) Erklären Sie das Zwei-Phasen-Freigabe-Protokoll.

(f) Erläutern Sie Partial Undo / Redo und Global Undo / Redo und deren Bedeutung für die Umsetzung des ACID-Prinzips. Geben Sie zu jeder dieser Konzepte an, ob System-, Programm- oder Gerätefehler damit korrigiert werden können.

(g) Erklären Sie das WAL-Prinzip (Write ahead logging)!

Lösungsvorschlag

Das sogenannte write ahead logging (WAL) ist ein Verfahren der Datenbanktechnologie, das zur Gewährleistung der Atomarität und Dauerhaftigkeit von Transaktionen beiträgt. Es besagt, dass Modifikationen vor dem eigentlichen Schreiben (dem Einbringen in die Datenbank) protokolliert werden müssen.

Durch das WAL-Prinzip wird ein sogenanntes „update-in-place“ ermöglicht, ðdie alte Version eines Datensatzes wird durch die neue Version an gleicher Stelle überschrieben. Das hat vor allem den Vorteil, dass Indexstrukturen bei Änderungsoperationen nicht mit aktualisiert werden müssen, weil die geänderten Datensätze immer noch an der gleichen Stelle zu finden sind. Die vorherige Protokollierung einer Änderung ist erforderlich, um im Fehlerfall die Wiederholbarkeit der Änderung sicherstellen zu können.

(h) Erklären Sie den Begriff „Datenbankindex“ und nennen Sie zwei häufige Arten.

Lösungsvorschlag

Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.

Gruppierte Indizes (Clustered Index)

Bei der Verwendung eines gruppierten Index werden die Datensätze entsprechend der Sortierreihenfolge ihres Index-Schlüssels gespeichert. Wird für eine Tabelle beispielsweise eine Primärschlüssel-Spalte „NR“ angelegt, so stellt diese den Index-Schlüssel dar. Pro Tabelle kann nur ein gruppierter Index erstellt werden. Dieser kann jedoch aus mehreren Spalten zusammengesetzt sein.

Nicht-gruppierte Indizes (Nonclustered Index)

> Besitzt eine Tabelle einen gruppierten Index, so können weitere nicht-gruppierte Indizes angelegt werden. Dabei zeigen die Einträge des Index auf den Speicherbereich des gesamten Datensatzes. Die Verwendung eines nicht-gruppierten Index bietet sich an, wenn regelmäßig nach bestimmten Werten in einer Spalte gesucht wird z.B. dem Namen eines Kunden. [a]
>
> [a] `https://www.datenbanken-verstehen.de/datenmodellierung/datenbank-index`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-2/Aufgabe-6.tex`

Examensaufgabe „Vermischte Fragen“ (66116-2021-F.T1-TA2-A1)

Datenbank

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

(a) Erläutern Sie die Begriffe Kardinalität und Partizipität. Welche Arten von Partizipität gibt es in der ER-Modellierung? Nennen und erklären Sie diese kurz.

Lösungsvorschlag

Kardinalitäten Für die noch genauere Darstellung der Beziehungen im ER-Modell verwendet man Kardinalitäten (auch Grad der Beziehungen genannt). Diese geben an wie viele Entitätsinstanzen mit wie vielen Entitätsinstanzen einer anderen Entitätsinstanz in Beziehung stehen. [a]

Partizipation Die Partizipation eines Beziehungstyps (in einem Entity-Relationship-Modell) bestimmt, ob alle Entities eines beteiligten Entitätstyps an einer bestimmten Beziehung teilnehmen müssen. [b]

totale Partizipation: Wenn eine Beziehung Entität A und Entität B in Beziehung setzt, dann muss ein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existiert und umgekehrt. Beide Entitäten müssen also an der Relation teilnehmen. Eine Entitätsinstanz aus A kann also nicht ohne eine in-Beziehung-stehende Entitätsinstanz aus B existieren und umgekehrt.

partielle Partizipation: Wenn eine Beziehung Entität A mit Entität B in Beziehung setzt, dann muss kein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existieren kann und umgekehrt. Die beiden Entitäten müssen also nicht an der Relation teilnehmen (enthalten sein). [c]

Die Kardinalität definiert, wie viele Entities eines Typs mit wie vielen Entities eines anderen Typs in Beziehung stehen können (siehe Schneider et al., S. 446)

Partizipität – ein anderer Begriff dafür ist Totalität – beschreibt den Grad der Teilnahmeverpflichtung zweier Entitäten an einer Beziehung. Sie kann partiell, total oder einseitig total sein.

a. Totale P.: Jede Entity A und Entity B besteht nur dann, wenn sie an dieser Beziehung teilnehmen.

b. Einseitige totale P.: Eine Entity A besteht nur dann, wenn sie an der Beziehung zu Entity B teilnimmt. Entitäten von B dagegen können, müssen aber nicht teilnehmen.

c. Partielle P.: Die Existenz der Entitäten ist unabhängig von der Teilnahme an dieser Beziehung. Die Teilnahme ist nicht verpflichtend.

[a] https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda

[b] https://lehrbuch-wirtschaftsinformatik.org/glossar/kapitel03/Partizipation

[c] https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda

(b) Mit welchen beiden Befehlen kann eine Transaktion beendet werden? Nennen Sie diese und erklären Sie den Unterschied.

Lösungsvorschlag

Für den Abschluss einer Transaktion gibt es 2 Möglichkeiten:

- Den erfolgreichen Abschluss mit `commit`.
- Den erfolglosen Abschluss mit `abort`

(c) Erläutern Sie den Unterschied zwischen einer kurzen und einer langen Sperre.

Lösungsvorschlag

lange Sperren: LOCKs werden erst nach dem commit zurückgegeben ($\rightarrow$ striktes 2PL)

kurze Sperren: LOCKs werden direkt nach dem schreiben/lesen zurückzugeben

[a]

[a] `https://www.dbs.ifi.lmu.de/Lehre/DBSII/SS2015/vorlesung/DBS2-03-Synchronisation.pdf`

(d) Stellen Sie außerdem die Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips mit den richtigen Werten dar. S stehe dabei für eine Lese- und X für eine Schreibsperre.

Lösungsvorschlag

Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips (Atomicity, Consistency, Isolation, Durability)

	NL (no lock)	S (Lesesperre)	X (Schreibsperre)
S (Lesesperre)	ja	ja	nein
X (Schreibsperre)	ja	nein	nein

(e) Nennen und erklären Sie kurz die Armstrong-Axiome. Sind diese vollständig und korrekt?

Lösungsvorschlag

Reflexivität: Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt, $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$.

Verstärkung: Gilt $\alpha \rightarrow \beta$, so gilt auch $\alpha\gamma \rightarrow \beta\gamma$ für jede Menge von Attributen γ der Relation.

Transitivität: Gilt $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, so gilt auch $\alpha \rightarrow \gamma$.

Die Armstrong-Axiome sind korrekt und vollständig: Diese Regeln sind gültig (korrekt) und alle anderen gültigen Regeln können von diesen Regeln ab-

geleitet werden (vollständig). [a]

[a] https://dbresearch.uni-salzburg.at/teaching/2019ss/db1/db1_06-handout-1x1.pdf

(f) Was versteht man unter einem (Daten-)Katalog (Data Dictionary) und was enthält dieser (es genügt eine Auswahl zu nennen)?

Lösungsvorschlag

Bei einer relationalen Datenbank ist ein Datenkatalog eine Menge von Tabellen und Ansichten, die bei Abfragen nur gelesen werden. Das Data-Dictionary ist wie eine Datenbank aufgebaut, enthält aber nicht Anwendungsdaten, sondern Metadaten, das heißt Daten, welche die Struktur der Anwendungsdaten beschreiben (und nicht den Inhalt selbst).

Zu einem Data-Dictionary zur physischen Datenmodellierung gehören genaue Angaben zu:

- Tabellen und Datenfeldern
- Primär- und Fremdschlüsselbeziehungen
- Integritätsbedingungen, z. B. Prüfinformationen

a

[a] https://de.wikipedia.org/wiki/Data-Dictionary

(g) Erklären Sie das konservative und das strikte Zwei-Phasen-Sperrprotokoll.

Lösungsvorschlag

Konservatives 2-Phasen-Sperrprotokoll Das konservative 2-Phasen-Sperrprotokoll (Preclaiming), bei welchem zu Beginn der Transaktion alle benötigten Sperren auf einmal gesetzt werden. Dies verhindert in jedem Fall Deadlocks, führt aber auch zu einem hohen Verlust an Parallelität, da eine Transaktion ihre erste Operation erst dann ausführen kann, wenn sie alle Sperren erhalten hat.

Striktes 2-Phasen-Sperrprotokoll Das strikte 2-Phasen-Sperrprotokoll, bei welchem alle gesetzten Write-Locks erst am Ende der Transaktion (nach der letzten Operation) freigegeben werden. Dieses Vorgehen verhindert den Schneeballeffekt, also das kaskadierende Zurücksetzen von sich gegenseitig beeinflussenden Transaktionen. Der Nachteil ist, dass Sperren häufig viel länger gehalten werden als nötig und sich somit die Wartezeit von blockierten Transaktionen verlängert. Die Read-Locks werden entsprechend dem Standard-2PL-Verfahren entfernt.

(h) Erklären Sie die Begriffe „Steal/NoSteal“ und „Force/NoForce“ im Kontext der Systempufferverwaltung eines DBS.

Lösungsvorschlag

No-Steal Schmutzige Seiten dürfen nicht aus dem Puffer entfernt und in die Datenbank übertragen werden, solange die Transaktion noch aktiv ist. Die Datenbank enthält keine Änderungen nicht-erfolgreicher Transaktionen. Eine UNDO-Recovery ist nicht erforderlich. langen Änderungs-Transaktionen können zu Problemen führen, da große Teile des Puffers blockiert werden

Steal Schmutzige Seiten dürfen jederzeit ersetzt und in die Datenbank eingebracht werden. Die Datenbank kann unbestätigte Änderungen enthalten. Eine UNDO-Recovery ist erforderlich. Es handelt sich um eine effektivere Puffernutzung bei langen Transaktionen mit vielen Änderungen.

Force Alle geänderten Seiten werden spätestens bei EOT (vor COMMIT) in die Datenbank geschrieben. Bei einem Systemfehler ist keine REDO-Recovery erforderlich. Die Force-Strategie benötigt einen hohen I/O-Aufwand, da Änderungen jeder Transaktion einzeln geschrieben werden. Die Vielzahl an Schreibvorgängen führt zu schlechteren Antwortzeiten, länger gehaltenen Sperren und damit zu mehr Sperrkonflikten. Große Datenbank-Puffer werden schlecht genutzt.

No-Force Änderungen können auch erst nach dem COMMIT in die Datenbank geschrieben werden. Die Änderungen durch mehrere Transaktionen werden „gesammelt“. Beim COMMIT werden lediglich REDO-Informationen in die Log-Datei geschrieben. Bei einem Systemfehler ist eine REDO-Recovery erforderlich. Die Änderungen auf einer Seite über mehrere Transaktionen hinweg können gesammelt werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Übungsaufgabe „Drei-Schichten-Modell“ (Drei-Schichten-Modell)

Drei-Schichten-Modell

Das Drei-Schichten-Modell trägt den verschiedenen Sichten auf eine Datenbank Rechnung. Geben Sie zu den unten (unter a bis e) genannten Vorgängen jeweils an, welche der folgenden Aussagen zutreffen:

(a) Änderungen in bestehenden Anwendungsprogrammen notwendig

(b) Änderungen im externen Schema notwendig

(c) Änderungen im konzeptionellen Schema notwendig

(d) Änderungen im internen Schema notwendig

Vorgänge

(a) Ein neues Anwendungsprogramm wird geschrieben, das bestehende Daten nutzt.

(b) Der Datentyp eines Attributs wird geändert, z. B. wird statt VARCHAR(20) VARCHAR(30) verwendet.

(c) Ein neues Anwendungsprogramm wird entwickelt, das neue (zusätzliche) Datenstrukturen benötigt.

(d) Es werden neue Daten eingespeichert bzw. bestehende gelöscht.

(e) Der Zugriff auf die Daten wird optimiert.

Sie könnten dazu folgende Tabelle ausfüllen: Kreuzen Sie das entsprechende Feld an, wenn die Aussage diesbezüglich wahr ist, oder lassen Sie es andernfalls leer. Ist die Aussage nicht eindeutig und situationsbedingt wahr, setzen sie ein eingeklammertes Kreuz (x) ein!

Lösungsvorschlag

	1	2	3	4
a		(x)		
b	(x)	(x)	(x)	x
c	(x)	(x)	x	x
d				
e				x

Hinweis

- Zu b) Bei der vorgegebenen Änderung des Datentyps ist sicher das interne Schema betroffen, da sich die Speicherstruktur ändert. Wird der Datentyp „stark“ geändert, ðbeispielsweise char durch integer ersetzt, kann das auch Änderungen bei (1), (2) und (3) nach sich ziehen.

- Zu d) Schemata beschreiben Strukturen. Das Speichern bzw. Löschen von Daten kann damit keinen Einfluss auf ein Schema haben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/10_Uebersicht/Aufgabe_Drei-Schichten-Modell.tex

Übungsaufgabe „Speicherung-Dateisystem“ (Datenbank-Übersicht)

Datenbank-Übersicht

Zur Speicherung von Daten kann ein Dateisystem verwendet werden. Betrachten Sie die Informationsseiten der Didaktik der Informatik an der FAU im Internet (https://ddi.cs.fau.de/). Gehen Sie davon aus, dass für jede Seite, die Sie betrachten, eine Datei existiert, in der die auf der Seite sichtbaren Informationen gespeichert sind.

(a) Warum ist diese Art der Datenabspeicherung nicht besonders günstig?

Lösungsvorschlag

Redundanz: In dieser Art zu speichern sind viele Redundanzen enthalten. So muss beispielsweise das Menü mit den HTML-Links auf jeder Seite gespeichert sein, um durch die Seite navigieren zu können. Die gleiche Information kommt auf sehr vielen verschiedenen Seiten vor. Damit muss dieselbe Information in unterschiedlichen Dateien abgespeichert werden, ðein Großteil der Information ist redundant gespeichert.

Beschränkte Zugriffsmöglichkeit: Die Daten können nur schlecht maschinell abgefragt werden. Sie können nur einzeln im Browser aufgerufen werden.

Beschränkte Zugriffskontrolle: Die HTML-Seiten können entweder als ganze Seite unter einen Passwortschutz gestellt werden oder vollkommen öffentlich ins Netz gestellt werden. So können einzelne Informationen auf den Seite, wie zum Beispiel die Raumnummer nicht einzeln in ihrer Sichtbarkeit beeinflusst werden.

(b) Es wird angenommen, dass folgende Aktualisierungen durchgeführt werden müssen:

- Die Mitarbeiter haben sich geändert.
- Das Projekt „AMI – Agile Methoden im Informatikunterricht“ ist abgeschlossen. Alle diesbezüglichen Informationen sollen deshalb gelöscht werden.

(c) Zu welchem Problemen kann es dabei (aufgrund der Datenspeicherung) kommen?

Lösungsvorschlag

Nicht aus alle Seiten werden die Links zur der Unterseite Projekt „AMI“ entfernt. Es kann zur Inkonsistenz kommen.

Mitarbeiterinformationen wie auch die Daten des Kurses sind redundant gespeichert. Änderungen oder Löschungen müssen deshalb in allen Dateien erfolgen, in denen die entsprechenden Informationen gespeichert sind. Bei vielen Dateien hat man aber in der Regel keinen Überblick, wo eine bestimmte Information überall abgespeichert ist. Dies kann sehr leicht zu einem inkonsistenten, d. h. nicht stimmigen, Datenbestand führen. Konkret können beispielsweise folgende Probleme auftreten:

- Die Änderung der Mitarbeiter wird an einer Stelle vergessen, d. h. dass

z. B. Dr. X noch Anfragen und E-Mails an die Adresse X@fau.de bekommt, obwohl er schon längst ausgeschieden ist und die o. g. E-Mail-Adresse nicht mehr existiert.

- Zum Löschen der AMI-Seite müssen die zentrale Projektseite und alle Links gelöscht werden. Die Löschung eines Link auf die zentrale Projektseite wird vergessen, d. h. man bekommt einen „toten“ Link.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/10_Uebersicht/Aufgabe_Speicherung-Dateisystem.tex`

Übungsaufgabe „Terminologie“ (Datenbanksystem, Datenbank, Datenbankmanagementsystem)

Datenbanksystem
Datenbank
Datenbankmanagementsystem

Beschreiben die Begriffe *„Datenbank“* und *„Datenbanksystem“* das Gleiche? Kurze Begründung!

Lösungsvorschlag

Nein. *Datenbanksystem* ist der Oberbegriff. Zu *Datenbankensystem* (DBS) gehört die *Datenbank* (DB) und das *Datenbankmanagementsystem* (DBMS). Unter dem Begriff *Datenbank* versteht man die Menge der gespeicherten Daten. (Die zweite Komponente eines Datenbanksystems ist das Datenbankmanagementsystems, mit Hilfe dessen die Daten in der Datenbank verwaltet werden können.)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/10_Uebersicht/Aufgabe_Terminologie.tex`

Datenbankentwurf

Examensaufgabe „Rennstall“ (46116-2013-F.T1-TA2-A1)

Sie sollen ein System zur Verwaltung von Pferderennen entwerfen. Gehen Sie dabei von folgendem Szenario aus:

- **Unternehmen** werden ihre eindeutige *Unternehmens-ID* identifiziert. Sie haben eine *Adresse* und besitzen Rennställe.
- Der *Name* eines **Rennstalls** ist nur innerhalb eines Unternehmens eindeutig. Für jeden Rennstall wird das *Gründungsdatum* gespeichert.
- **Pferde** gehören immer zu einem Rennstall. *Pferdenamen* werden in einem Rennstall nur jeweils maximal einmal vergeben.
- **Jockeys** sind in einem Rennstall beschäftigt. Jeder Rennstall vergibt seine eigenen *Personalnummern*. Für jeden Jockey werden *Vorname* und *Name* gespeichert.
- **Rennen** haben ein *Datum*, ein *Preisgeld* und einen *Namen*, über den sie identifiziert werden.
- Unternehmen unterstützen Rennen finanziell mit einem bestimmten Betrag.
- Jockeys nehmen mit Pferden an Rennen teil. Im Rennen erreichen sie einen bestimmten Platz. Die Kombination aus Jockey und Pferd ist nicht fest, bei unterschiedlichen Rennen können Jockeys verschiedene Pferde reiten. Jockeys können auch mit Rennpferden von fremden Rennställen, die anderen Unternehmen gehören können, an Rennen teilnehmen.

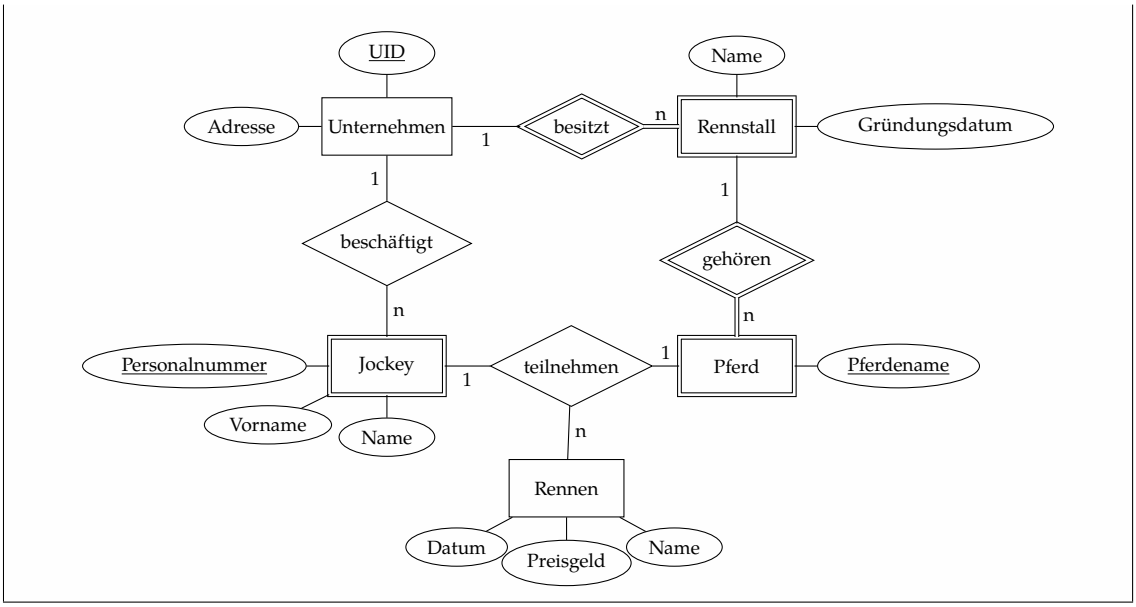
- ▢ **E: Unternehmen**
- ○ *A: Unternehmens-ID*
- ○ *A: Adresse*
- ◇ R: besitzen
- ○ *A: Name*
- ▢ **E: Rennstalls**
- ○ *A: Gründungsdatum*
- ▢ **E: Pferde**
- ◇ R: gehören
- ○ *A: Pferdenamen*
- ▢ **E: Jockeys**
- ◇ R: beschäftigt
- ○ *A: Personalnummern*
- ○ *A: Vorname*
- ○ *A: Name*
- ▢ **E: Rennen**
- ○ *A: Datum*
- ○ *A: Preisgeld*
- ○ *A: Namen*
- ◇ R: unterstützen
- ◇ R: nehmen

(a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

Lösungsvorschlag

Relationenmodell



(b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

```
Unternehmen (UnternehmensID, Addresse)
Rennstall (S_Name, UnternehmensID[Unternehmen], Gründungsdatum)
Jockey (PersNr, S_Name[Rennstall], ID[Unternehmen], Vorname, Name)
Rennen (R_Name, Datum, Preisgeld)
Pferd (P_Name, S_Namen, UnternehmensID[Unternehmen])

teilnehmen (R_Name[Rennen], PersNr[Jockey], S_Name1, ID1, P_Namen, S_Namen2,
↪  ID2, Platz)
unterstuetzen (UnternehmensID[Unternehmen], R_Name[Rennen], Betrag)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2013/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex

Examensaufgabe „Konsulat“ (46116-2015-F.T1-TA2-A3)

Entity-Relation-Modell
Relationenmodell

Es sind folgende Informationen zu einer Datenbank für Konsulate gegeben:

- Jedes Konsulat hat einen Sitz in einer Stadt
- Zu einem **Konsulat** soll ein eindeutiger *Name* (`KonsulatName`) (z. B. Konsulat Bayern), die *Adresse* und der *Vor-* (`KVorname`) bzw. *Nachname* (`KNachname`) des Konsuls gespeichert werden.
- Für jede **Stadt** sollen der *Name* (`StadtName`), die *Anzahl der Einwohner* (`EinwohnerAnzahl`), sowie das Land in dem es liegt, festgehalten werden. Gehen Sie davon aus, dass eine Stadt nur in Zusammenhang mit dem zugehörigen Land identifizierbar ist.
- Für ein **Land** soll der Name in *Landessprache*, der *Name des Staatspräsidenten* (`Staatspräsident`) und eine eindeutige *ID* (`LandesID`) gespeichert werden.

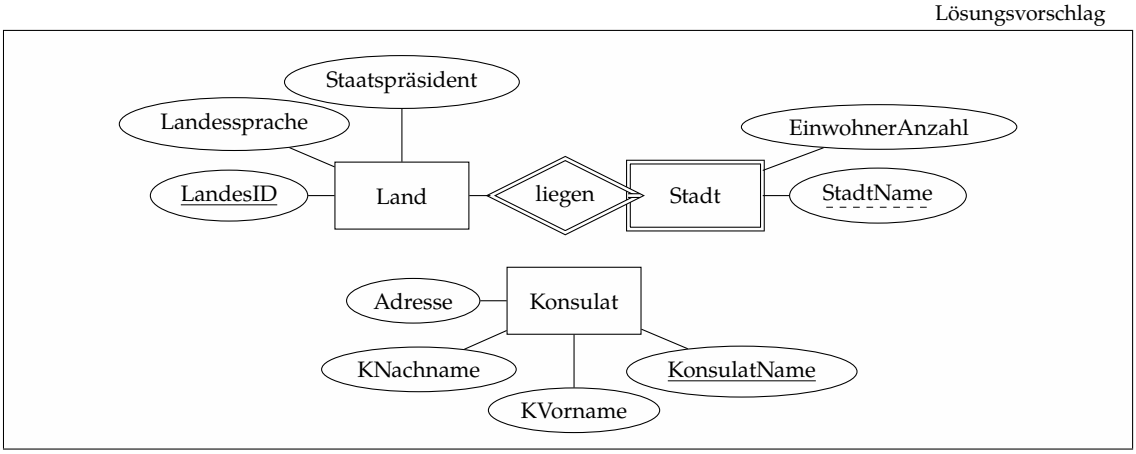
□ **E: Konsulat**
○ *A: Name*
○ *A: Adresse*
○ *A: Vor-*
○ *A: Nachname*
□ **E: Stadt**
○ *A: Name*
○ *A: Anzahl der Einwohner*
◇ R: liegt
□ **E: Land**
○ *A: Landessprache*
○ *A: Name des Staatspräsidenten*
○ *A: ID*

(a) Entwerfen Sie für das obige Szenario ein ER-Diagramm in Chen-Notation. Bestimmen Sie hierzu:

- Die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- Die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- Die Funktionalitäten der Relationship-Typen.

Hinweis: Achten Sie darauf, alle Totalitäten einzutragen.

Lösungsvorschlag



(b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Lösungsvorschlag

Konsulat(KonsulatName, KVorname, KNachname, Adresse, StadtName, LandesID)

Stadt(<u>LandesID, StadtName</u>, EinwohnerAnzahl)

Land(<u>LandesID</u>, Landessprache, Staatspraesident)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

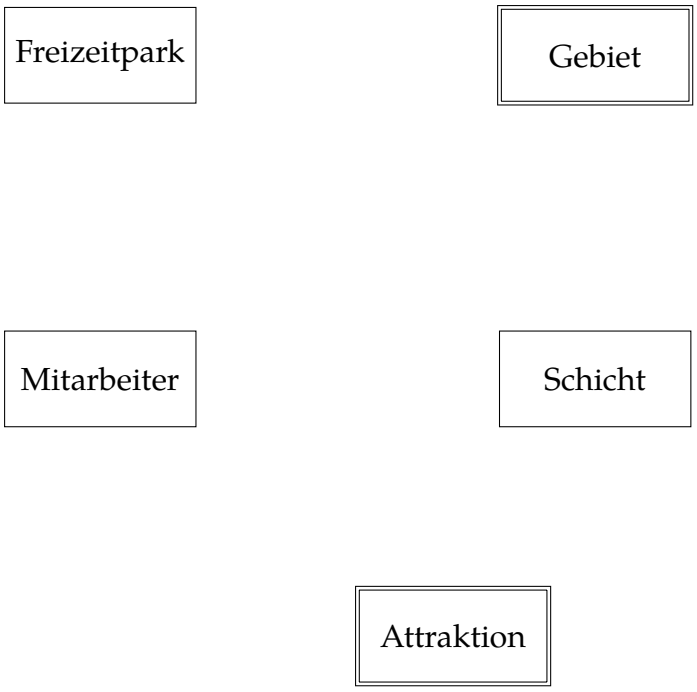
Examensaufgabe „Freizeitparks“ (46116-2018-H.T1-TA2-A2)

Entity-Relation-Modell

Im Folgenden finden Sie die Beschreibung eines Systems zur Verwaltung von Freizeitparks. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

- Der **Freizeitpark** ist in mehrere Gebiete eingeteilt.
- Ein **Gebiet** hat einen eindeutigen *Namen* und eine *Beschreibung*.
- In jedem Gebiet gibt es eine oder mehrere **Attraktionen**. Diese verfügen über eine innerhalb ihres Gebiets eindeutige *Nummer*. Außerdem gibt es zu jeder Attraktion einen *Namen*, eine *Beschreibung* und ein oder mehrere Fotos.
- Der Freizeitpark hat **Mitarbeiter**. Zu diesen werden jeweils eine eindeutige *ID*, der *Vorname* und der *Nachname* gespeichert. Weiterhin hat jeder Mitarbeiter ein *Geburtsdatum*, das sich aus *Tag*, *Monat* und *Jahr* zusammensetzt.
- Die Arbeit im Freizeitpark ist in **Schichten** organisiert. Eine Schicht kann eindeutig durch das *Datum* und die *Startzeit* identifiziert werden. Jede Schicht hat weiterhin eine *Dauer*.
- Mitarbeiter können in Schichten an Attraktionen arbeiten. Dabei wird die *Aufgabe* gespeichert, die der Mitarbeiter übernimmt. Pro Schicht kann der selbe Mitarbeiter nur an maximal einer Attraktion arbeiten.

- ☐ **E: Freizeitpark**
- ◇ R: eingeteilt
- ☐ **E: Gebiet**
- ○ *A: Namen*
- ○ *A: Beschreibung*
- ◇ R: gibt
- ☐ **E: Attraktionen**
- ○ *A: Nummer*
- ○ *A: Namen*
- ○ *A: Beschreibung*
- ◇ R: hat
- ☐ **E: Mitarbeiter**
- ○ *A: ID*
- ○ *A: Vorname*
- ○ *A: Nachname*
- ○ *A: Geburtsdatum*
- ○ *A: Tag*
- ○ *A: Monat*
- ○ *A: Jahr*
- ☐ **E: Schichten**
- ○ *A: Datum*
- ○ *A: Startzeit*
- ○ *A: Dauer*
- ◇ R: arbeiten
- ○ *A: Aufgabe*



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

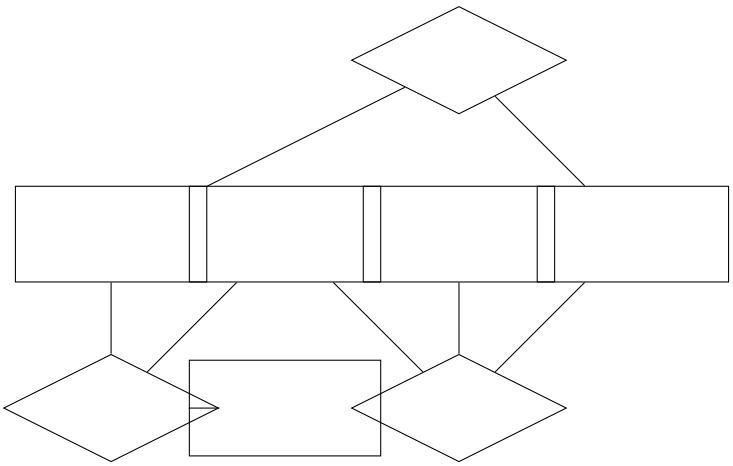
Examensaufgabe „Schulverwaltung“ (46116-2018-H.T2-TA2-A5)

Entity-Relation-Modell
Relationenmodell

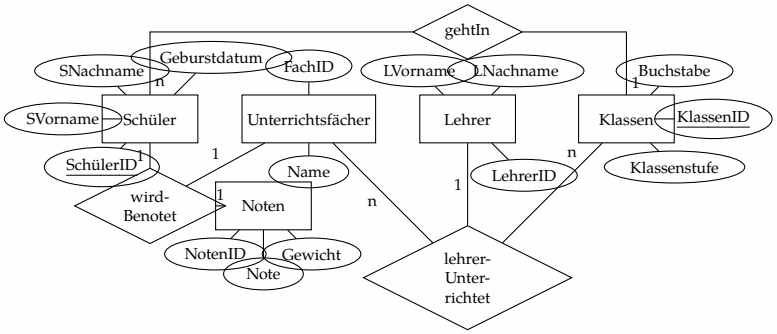
Überführen Sie das Datenbankschema in ein ER-Diagramm. Verwenden Sie hierfür die bereits eingezeichneten Entity-Typen und Relationship-Typen. Weisen Sie die Relationen zu und schreiben Sie deren Namen in die dazugehörigen Felder. Fügen Sie, falls erforderlich, Attribute hinzu und beschriften Sie die Beziehungen. Markieren Sie Schlüsselattribute durch unterstreichen.

Gegeben sei das folgende Datenbankschema, wobei Primärschlüssel unterstrichen und Fremdschlüssel überstrichen sind. Die von einem Fremdschlüssel referenzierte Relation ist in eckigen Klammem nach dem Fremdschlüsselattribut angegeben.

```
Schüler (SchülerID, SVorname, SNachname, KlassenID[Klassen], Geburtsdatum)

Lehrer (LehrerID, LVorname, LNachname)

Klassen (KlassenID, Klassenstufe, Buchstabe)

Unterrichtsfächer (FachID, Name)

Noten (NotenID, SchülerID[Schüler], FachID[Unterrichtsfächer], Note, Gewicht)

LehrerUnterrichtet (LehrerID[Lehrer], KlassenID[Klassen], FachID[Unterrichtsfächer])
```



Lösungsvorschlag



Funktionalitäten

gehtIn n:1 Eine Klasse hat n Schüler, einE SchülerIn geht in eine Klasse

wirdBenotet 1:1:1: Das ist anderes nicht möglich, da nur `NotenID` Primärschlüssel ist, dann muss aber die Kombination aus `Schüler` und `Unterrichtsfach` auch einmalig sein, ðUNIQUE und es gilt `Note` und `Unterrichtsfach` bestimmt `Schüler`, `Note` und `Schüler` bestimmt `Unterrichtsfach` und `Schüler` und `Unterrichtsfach` bestimmt `Noten`.

LehrerUnterrichtet 1(Lehrer):n:n

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-2/Teilaufgabe-2/Aufgabe-5.tex`

Examensaufgabe „Universitätsdatenbank“ (66111-1996-F.A2)

Eine Universitätsdatenbank soll folgende Daten verwalten.

- Studenten (Name, Matrikelnummer, Geburtsdatum, Adressen, Semesteranzahl, Studiengang, Fakultät, belegte Lehrveranstaltungen und deren Art)
- Lehrveranstaltungen (Anfangszeit, Semester, Art (Vorlesung, Seminar, Übung, Praktikum) Name, Anzahl der Hörer, Nummer im Vorlesungsverzeichnis, Anzahl Semesterwochenstunden, Dozent, Raum)

(a) Entwerfen Sie ein ER-schema für diese Applikation! Berücksichtigen Sie dabei, dass eine Vorlesung in 2 Unterrichtseinheiten aufgeteilt werden kann (z. B. Mo 10.00 und Do 14.00). Begründen Sie Ihren Entwurf!

(b) Spezifizieren Sie für die Entity-Typen Attribut und zeichnen Sie die Schlüsselattribute aus!

(c) Geben Sie ein relationales Datenbankschema an!

Lösungsvorschlag

```
Dozent(PersNr:INT, Name:VARCHAR(20), FakName:VARCHAR(40)[Fakultät])

Fakultät(Name:VARCHAR(40))

Lehrveranstaltung(Nr:INT, Name:VARCHAR(20), SWS:INT, Semester:INT, Art:VARCHAR(20),
PersNr:INT[Dozemtn])

Student(MatrNr:INT, Name:VARCHAR(40), Geburtsdatum:DATE, Semesteranzahl:INT, Fak
Name:VARCHAR(40)[Fakultät], Studiengang:VARCHAR(40))

Adresse(Adresse:VARCHAR(100))

belegt(Nr:INT[Lehrveranstaltung], MatrNr:INT[Adresse])

besitzt(Adresse:VARCHAR(100)[Adresse], MatrNr:INT[Student])

Durchführung(Zeit:DATE, RaumNr:INT, Nr:INT[Student])
```

(d) Welche Fremdschlüssel gibt es in diesem Schema?

Es soll nun mit SQL eine entsprechende relationale Datenbank angelegt werden. Geben Sie für folgende Aufgaben die jeweiligen SQL-Befehle an.

(c) Die Tabellenschemata von `Student` und `besitzt` sollen erzeugt werden.

Lösungsvorschlag

```sql
CREATE TABLE Fakultaet (
  Name VARCHAR(20) PRIMARY KEY
);

CREATE TABLE Adresse (
  Adresse VARCHAR(30) PRIMARY KEY
);

CREATE TABLE Student (
  MatrNr INTEGER PRIMARY KEY,
  Name VARCHAR(20) NOT NULL,
  Geburtsdatum DATE,
  Semesteranzahl INTEGER,
  Fakultaetsname VARCHAR(20),
  Studiengang VARCHAR(20),
  FOREIGN KEY (Fakultaetsname) REFERENCES Fakultaet(Name)
);

CREATE TABLE besitzt (
  Adresse VARCHAR(30) NOT NULL,
  MatrNr INTEGER NOT NULL,
  PRIMARY KEY (Adresse, MatrNr),
  FOREIGN KEY (Adresse) REFERENCES Adresse(Adresse),
  FOREIGN KEY (MatrNr) REFERENCES Student(MatrNr)
);

INSERT INTO Adresse VALUES ('Kaulbacherstraße 3');
```

(d) Am Tabellenschema von `Student` werden zwei Änderungen vorgenommen:

- Es soll ein weiteres Attribut Vorname hinzugefügt werden.

Lösungsvorschlag

```sql
ALTER TABLE Student ADD COLUMN Vorname VARCHAR(20);
```

- Als Integritätsbedingung wird festgelegt, dass die Semesterzahl kleiner als 15 sein muss.

Lösungsvorschlag

```sql
ALTER TABLE Student ADD CHECK (Semesteranzahl < 15);
```

oder

```sql
ALTER TABLE Student
ADD CONSTRAINT begrenzung_Semester
CHECK (Semesteranzahl < 15);
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66111/1996/03/Aufgabe-2.tex

Examensaufgabe „Fertigung“ (66111-1997-H.A3)

Entity-Relation-Modell
Relationenmodell

Für ein Unternehmen soll eine Fertigungsdatenbank aufgebaut werden. Der Erhebungsprozess liefert folgenden Informationsbedarf:

Entity-Typen:

- ABTEILUNG mit den Attributen ANR, ANAME, AORT, MNR
- PERSONAL mit den Attributen PNR, NAME, BERUF
- MASCHINE mit den Attributen MANR, FABRIKAT, TYP, BEZ, LEISTUNG
- TEILE mit den Attributen LNR, BEZ, GEWICHT, FARBE, PREIS

Relationship-Typen:

- ABT-PERS zwischen ABTEILUNG und PERSONAL
- SETZT-EIN zwischen ABTEILUNG und MASCHINEN
- KANN-BEDIENEN zwischen PERSONAL und MASCHINEN
- GEEIGNET-FÜR-DIE-HERSTELLUNG-VON zwischen MASCHINEN und TEILE
- PRODUKTION zwischen PERSONAL, TEILE und MASCHINEN mit den Attributen DATUM und MENGE

Dabei sollen folgende grundlegenden Bedingungen gelten:

- Zu einer Abteilung gehört mindestens ein Beschäftigter
- Eine Person ist immer nur genau einer Abteilung zugeordnet
- Eine Maschine kann, wenn überhaupt, nur von einer Abteilung eingesetzt werden
- Alle anderen (Teil-)Beziehungen sind nicht weiter eingeschränkt.

(a) Zeichnen Sie zu dem obigen Szenario das zugehörige ER-Diagramm.

(b) Legen Sie die Schlüsselkandidaten fest und zeichnen Sie diese in das ER-Diagramm ein. Tragen Sie die oben genannten Bedingungen mit Hilfe der (min, max) – Notation in das ER-Diagramm. Formulieren Sie weitere sinnvolle Bedingungen und tragen Sie diese ebenfalls in das Diagramm ein.

Konvertieren Sie das folgende ER-Modell in ein (vereinfachtes) relationales Schema! Geben Sie dabei geeignete Domänenattribute an!

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66111/1997/09/Aufgabe-3.tex`

Examensaufgabe „Handelsunternehmen“ (66116-2012-F.T1-TA1-A1)

Entity-Relation-Modell

Ein Handelsunternehmen möchte seine Struktur verbessern und ein Datenbanksystem zur Verwaltung seiner Filialen, angebotenen Waren und Kunden erstellen.

Die Basis dieses Systems bilden die Filialen des Unternehmens. Jede Filiale ist eindeutig durch ihre Filialnummer gekennzeichnet und befindet sich in einer Stadt. Außerdem hat jede Filiale einen Filialleiter.

Zu jeder Filiale gehört genau ein Lager mit einer eindeutigen Lagernummer und ebenfalls einem Leiter. Jedes Lager verfügt über eine bestimmte Menge an verschiedenen Waren. Jede Ware kann in mehreren Lagern vorrätig sein und ist über eine Nummer, einen Namen und einen Preis gekennzeichnet.

Ein Kunde kann in einer Filiale des Unternehmens Bestellungen aufgeben. Der Kunde hat eine Kundennummer, einen Namen und eine Adresse, Eine Bestellung enthält dabei jeweils einen Warenartikel, dessen gewünschte Menge und das Datum, an dem die Bestellung abgeholt wird.

(a) Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank.

(b) Setzen Sie das in Teilaufgabe a) erstellte Entity-Relationship-Diagramm in ein Relationenschema um. Relationships sollen mit einer möglichst geringen Anzahl von Relationen realisiert werden. Dabei sind unnötige Redundanzen zu vermeiden. Ein Relationenschema ist in folgender Form anzugeben: Relation (Attributl, Attribut2, ...). Schlüsselattribute sind dabei zu unterstreichen. Achten Sie bei der Wahl des Schlüssels auf Eindeutigkeit und Minimalität.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2012/03/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Musik-Datenbank“ (66116-2015-F.T1-TA1-A1)

Entity-Relation-Modell

In einer Musik-Datenbank sollen folgende Informationen zu Interpreten und deren CDs modelliert werden:

- Zu einem Interpreten soll eine eindeutige ID, der Name, das Jahr seines Bühnenstarts, seine Geschäftsadresse sowie sein Musikgenre angegeben sein. Das Musikgenre kann mehrere Werte umfassen (mehrwertiges Attribut).
- Eine CD hat eine eindeutige ID, einen Namen (Titel), einen Interpreten, ein Erscheinungsdatum und bis zu 20 Positionen (Musikstücke). An jeder Position steht ein Musikstück. Für dieses ist der Titel und die Länge in Sekunden angegeben.
- Eine CD kann Auszeichnungen - z. B. vom Typ goldene Schallplatte oder Emmy bekommen. Ebenso kann auch ein einzelnes Musikstück Auszeichnungen bekommen.

(a) Modellieren Sie das oben dargestellte Szenario möglichst vollständig in einem ER-Modell. Verwenden Sie, wann immer möglich, (binäre oder auch höherstellige) Relationships. Modellieren Sie Musikstücke in einem schwachen Entity-Typen.

(b) Übertragen Sie Ihr ER-Modell - bis auf die Typen zu den Auszeichnungen - ins relationale Datenmodell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE-Statements in Sov. Berücksichtigen Sie die Fremdschlüsselbeziehungen.

(c) Es soll die Integritätsbedingung eingehalten werden, so dass die Anzahl der Positionen auf einer CD höchstens 20 ist. Schreiben Sie ein SELECT-Statement, das diese Integritätsbedin. gung überprüft, indem es die verletzenden CDs ausgibt.

(d) Geben Sie geeignete INSERT-Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügen, so dass alle Integritätsbedingungen erfüllt sind, nachdem alle Einfü- gungen ausgeführt wurden. Lediglich zu den Auszeichnungen müssen keine Tupel eingefügt werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Online-Auktionshaus“ (66116-2015-H.T1-TA1-A1)

Entity-Relation-Modell

Wir wollen eine relationale Datenbankstruktur für ein Online-Auktionshaus modellieren.

Das Auktionshaus hat Mitglieder. Diese Mitglieder haben Kundennummern, Namen und Adressen. Sie können Verkäufer und/oder Käufer sein. Verkäufer können eine Laden-URL (eine Subdomain) erhalten, die zu einer Seite mit ihren aktuellen Auktionen führt. Verkäufer können neue Auktionen starten. Die Auktionen eines Verkäufers werden durchnummeriert. Diese Auktionsnummer ist nur je Verkäufer eindeutig. Jede Auktion hat ein Mindestgebot und eine Ablaufzeit.

Auf Auktionen können Gebote abgegeben werden. Die Gebote auf eine Auktion werden nach ihrem Eintreffen nummeriert. Diese Nummer ist nur innerhalb einer Auktion eindeutig. Zu einem Gebot werden noch die Zeit des Gebots sowie der gebotene Geldbetrag angegeben. Die Gebote werden von Käufern abgegeben. Jedes Gebot muss einem Käufer zugeordnet sein.

Jede Auktion besteht aus einer Menge von Artikeln, aber aus mindestens einem. Artikel haben eine Beschreibung und können und über ihre Artikel-ID identifiziert werden. Zur Katalogisierung der Artikel gibt es Kategorien. Kategorien haben eine eindeutige ID und einen Namen. Jede Kategorie kann Subkategorien besitzen. Auch Subkategorien sind Kategorien (und können damit weitere Subkategorien besitzen). Jeder Artikel kann beliebig vielen Kategorien zugeordnet werden.

Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen,
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/09/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Forstverwaltung“ (66116-2016-F.T1-TA1-A1)

Entity-Relation-Modell
Relationenmodell

Für die bayerische Forstverwaltung wird eine Datenbank zur Erschließung einer Jagd-Statistik benötigt. Gehen Sie dabei von folgendem Szenario aus:

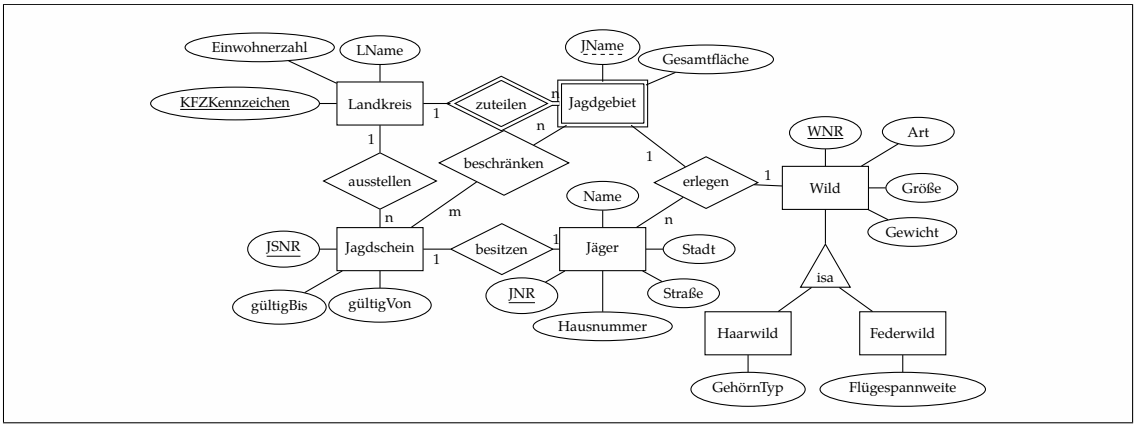
- Die Administration von Jagdgebieten obliegt den Landkreisen. Jeder **Landkreis** besitzt, neben seinem *Namen* (LName) und der *Einwohnerzahl*, ein eindeutiges *KFZ-Kennzeichen* (`KFZKennzeichen`).
- Die Jagd findet in Jagdgebieten statt. Ein **Jagdgebiet** soll dem Landkreis zugeteilt werden, indem es liegt. Gehen Sie davon aus, dass Jagdgebiete nicht in mehreren Landkreisen liegen können. Zusätzlich ist für jedes Jagdgebiet der *Name* (`JName`) und die *Gesamtfläche* zu speichern. Dabei ist zu beachten, dass die Namen nur innerhalb eines einzelnen Landkreises eindeutig sind.
- Die Erlaubnis zum Jagen wird durch einen **Jagdschein** erteilt. Dieser kann nur von einem Landkreis ausgestellt werden und beschränkt sich auf ein oder mehrere Jagdgebiete. Er wird durch eine *Jagdschein-Nummer* (`JSNR`) identifiziert und ist in einem bestimmtem Zeitintervall gültig. Dieses soll über zwei Zeitpunkte festgelegt werden (*gültig von* (`gültigVon`), *gültig bis* (`gültigBis`)).
- Ein **Jäger** besitzt genau einen Jagdschein. Zu einem Jäger sollen *Name*, *Stadt*, *Straße* und *Hausnummer*, gespeichert werden. Da die Jagdtradition innerhalb einer Familie häufig von einer zur nächsten Generation weitergegeben wird, kann es vorkommen, dass Name und Adresse von zwei unterschiedlichen Jägern gleich ist (z. B. Vater und Sohn). Aus diesem Grund ist eine eindeutige *Identifikationsnummer* (`JNR`) notwendig.
- Um Statistiken erheben zu können, muss berücksichtigt werden, welches **Wild** von welchen Jägern zu welchem Zeitpunkt in welchem Jagdgebiet erlegt worden ist. Gehen Sie davon aus, dass es mehrere Jäger geben kann, die gemeinsam ein Wild erlegen (z. B. in einer Jagdgesellschaft). Zu einem Wild gehört die *Art* (z. B. Reh), die *Größe*, das *Gewicht*, sowie eine eindeutige *Identifikationsnummer* (`WNR`). Zusätzlich unterscheidet man zwischen **Haarwild** und **Federwild**, wobei beim Haarwild der *Typ des Gehörns* (`GehörnTyp`) (z. B. Hirschgeweih) und beim Federwild die *Flügelspannweite* betrachtet werden soll.

(a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen.

□ **E: Landkreis**
○ *A: Namen*
○ *A: Einwohnerzahl*
○ *A: KFZ-Kennzeichen*
□ **E: Jagdgebiet**
◇ R: zugeteilt
○ *A: Name*
○ *A: Gesamtfläche*
□ **E: Jagdschein**
◇ R: ausgestellt
◇ R: beschränkt
○ *A: Jagdschein-Nummer*
○ *A: gültig von*
○ *A: gültig bis*
□ **E: Jäger**
◇ R: besitzt
○ *A: Name*
○ *A: Stadt*
○ *A: Straße*
○ *A: Hausnummer*
○ *A: Identifikationsnummer*
□ **E: Wild**
◇ R: erlegt
○ *A: Art*
○ *A: Größe*
○ *A: Gewicht*
○ *A: Identifikationsnummer*
□ **E: Haarwild**
□ **E: Federwild**
○ *A: Typ des Gehörns*
○ *A: Flügelspannweite*

Lösungsvorschlag



(b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Lösungsvorschlag

```
Landkreis(KFZKennzeichen, LName, Einwohnerzahl)

Jagdgebiet(JName, KFZKennzeichen[Landkreis], Gesamtfläche)

Jagdschein(JSNR, KFZKennzeichen[Landkreis], gültigVon, gültigBis)

Jäger(JNR, JSNR, Name, Stadt, Straße, Hausnummer)

Wild(WNR, Art, Größe, Gewicht)

Haarwild(WNR, GehörnTyp)

Federwild(WNR, Flügelspannweite)

erlegen(JNR[Jäger], WNR[Wild], JName[Jagdgebiet], KFZKennzeichen[Landkreis])
beschränken(JSNR[Jagdschein], JName[Jagdgebiet], KFZKennzeichen[Landkreis])
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/03/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Polizei“ (66116-2016-F.T1-TA1-A2)

Entity-Relation-Modell
Relationale Algebra
SQL

Gehen Sie dabei von dem dazugehörigen relationalen Schema aus:

```
Polizist : {[ PersNr, DSID, Vorname, Nachname, Dienstgrad, Gehalt ]}

Dienststelle : {[ DSID, Name, Strasse, HausNr, Stadt ]}

Fall : {[ AkZ, Titel, Beschreibung, Status ]}

Arbeitet_An : {[ PersNr, AkZ, Von, Bis ]}

Vorgesetzte : {[ PersNr, PersNr, Vorgesetzter ]}
```

Gegeben sei folgendes ER-Modell, welches Polizisten, deren Dienststelle und Fälle, an denen sie arbeiten, speichert:

(a) Formulieren Sie eine Anfrage in relationaler Algebra, welche den *Vornamen* und *Nachnamen* von Polizisten zurückgibt, deren Dienstgrad *„Polizeikommissar“* ist und die mehr als 1500 Euro verdienen.

Lösungsvorschlag

$$\pi_{\text{Vorname,Nachname}}(\sigma_{\text{Dienstgrad}=\text{'Polizeikommissar'}\wedge\text{Gehalt}>1500}(\text{Polizist}))$$

(b) Formulieren Sie eine Anfrage in relationaler Algebra, welche die *Titel* der *Fälle* ausgibt, die von *Polizisten* mit dem *Nachnamen „Mayer“* bearbeitet wurden.

Lösungsvorschlag

$$\pi_{\text{Titel}}(\sigma_{\text{Nachname}=\text{'Mayer'}}(\text{Polizist}) \bowtie_{\text{PersNr}} \text{Arbeitet\_An} \bowtie_{\text{AkZ}} \text{Fall})$$

(c) Formulieren Sie eine SQL-Anfrage, welche die Anzahl der Polizisten ausgibt, die in der Stadt *„München“* arbeiten und mit Nachnamen *„Schmidt“* heißen.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Polizisten
FROM Polizist p, Dienststelle d
WHERE
  p.DSID = d.DSID AND
  d.Stadt = 'München' AND
  p.Nachname = 'Schmidt';


 anzahl_polizisten
-------------------
                   1
(1 row)
```

(d) Formulieren Sie eine SQL-Anfrage, welche die *Namen* der *Dienststellen* ausgibt, die am *14.02.2012* an dem Fall mit dem *XZ1508* beteiligt waren. Ordnen Sie die Ergebnismenge alphabetisch (aufsteigend) und achten Sie darauf, dass keine Duplikate enthalten sind.

Lösungsvorschlag VIEW WITH UNION

```
SELECT DISTINCT d.Name
FROM Dienststelle d, Polizist p, Arbeitet_An a
WHERE
  a.AkZ = 'XZ1508' AND
  p.PersNr = a.PersNr AND
  p.DSID = d.DSID AND
  a.Von <= '2012-02-14' AND
  a.Bis >= '2012-02-14'
ORDER BY d.Name ASC;


             name
-------------------------------
 Dienststelle Nürnberg (Mitte)
(1 row)
```

(e) Definieren Sie die View *„Erstrebenswerte Dienstgrade“*, welche Dienstgrade enthalten soll, die in *München* mit durchschnittlich mehr als *2500* Euro besoldet werden.

Lösungsvorschlag

```
CREATE VIEW ErstrebenswerteDienstgrade AS (
  SELECT DISTINCT p.Dienstgrad
  FROM Polizist p, Dienststelle d
  WHERE
    p.DSID = d.DSID AND
    d.Stadt = 'München'
  GROUP BY Dienstgrad
  HAVING (AVG(Gehalt) > 2500)
);

SELECT * FROM ErstrebenswerteDienstgrade;


   dienstgrad
-----------------
 Polizeikommisar
 Polizeimeister
(2 rows)
```

(f) Formulieren Sie eine SQL-Anfrage, welche *Vorname*, *Nachname* und *Dienstgrad* von *Polizisten* mit *Vorname*, *Nachname* und *Dienstgrad* ihrer *Vorgesetzten* als ein Ergebnis-Tupel ausgibt (siehe Beispiel-Tabelle). Dabei sind nur *Polizisten* zu selektieren, die an Fällen gearbeitet haben, deren Titel den Ausdruck „Fussball“ beinhalten. An *Vorgesetzte* sind keine Bedingungen gebunden. Achten Sie darauf, dass Sie nicht nur direkte Vorgesetzte, sondern alle Vorgesetzte innerhalb der Vorgesetzten-Hierarchie betrachten. Ordnen Sie ihre Ergebnismenge alphabetisch (absteigend) nach Nachnamen des Polizisten.

Hinweis: Sie dürfen Views verwenden, um Teilergebnisse auszudrücken.

Lösungsvorschlag

Vorarbeiten:

```
  SELECT p.Vorname, p.Nachname
  FROM Polizist p, Arbeitet_An a, Fall f
  WHERE
    p.PersNr = a.PersNr AND
    a.AkZ = f.Akz AND
    f.Titel LIKE '%Fussball%';
```

```
 vorname | nachname
---------+----------
 Hans    | Müller
 Josef   | Fischer
(2 rows)
```

Lösungsansatz 1

```
WITH RECURSIVE Fussball_Vorgesetzte (PersNr, VN, NN, DG, PN_VG, VN_VG, NN_VG,
↪  DG_VG) AS
(
  SELECT
    p1.PersNr,
    p1.Vorname AS VN,
    p1.Nachname AS NN,
    p1.Dienstgrad AS DG,
    p2.PersNr AS PN_VG,
    p2.Vorname AS VN_VG,
    p2.Nachname AS NN_VG,
    p2.Dienstgrad AS DG_VG
  FROM Polizist p1, Fall f, Arbeitet_An a, Vorgesetzte v
  LEFT JOIN Polizist p2 ON v.PersNr_Vorgesetzter = p2.PersNr
  WHERE
    p1.PersNr = a.PersNr AND
    a.AkZ = f.Akz AND
    f.Titel LIKE '%Fussball%' AND
    p1.PersNr = v.PersNr

  UNION ALL

  SELECT
    m.PersNr,
    m.VN AS VN,
    m.NN AS NN,
    m.DG AS DG,
    p.PersNr AS PN_VG,
    p.Vorname AS VN_VG,
    p.Nachname AS NN_VG,
    p.Dienstgrad AS DG_VG
  FROM Fussball_Vorgesetzte m, Vorgesetzte v
  LEFT JOIN Polizist p ON v.PersNr_Vorgesetzter = p.PersNr
```

```
  WHERE m.PN_VG = v.PersNr
)

SELECT VN, NN, DG, VN_VG, NN_VG, DG_VG
FROM Fussball_Vorgesetzte
ORDER BY NN DESC;
```

```
  vn   |   nn    |         dg          |   vn_vg   |  nn_vg   |        dg_vg
-------+---------+---------------------+-----------+----------+--------------------
 Hans  | Müller  | Polizeimeister      | Andreas   | Schmidt  | Polizeikommisar
 Hans  | Müller  | Polizeimeister      | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Sebastian | Wagner   | Polizeioberkommis
(4 rows)
```

Lösungsansatz 2

```
CREATE VIEW naechste_Vorgesetzte AS
  SELECT
    p.PersNR,
    p.Vorname,
    p.Nachname,
    p.Dienstgrad,
    v.PersNr_Vorgesetzter AS Vorgesetzter
  FROM Polizist p LEFT JOIN Vorgesetzte v
  ON p.PersNr = v.PersNr;

WITH RECURSIVE Fussball_Vorgesetzte (VN, NN, DG, VN_VG, NN_VG, DG_VG) AS (
  SELECT
    x.Vorname AS VN,
    x.Nachname AS NN,
    x.Dienstgrad AS DG,
    y.Vorname AS VN_VG,
    y.Nachname AS NN_VG,
    y.Dienstgrad AS DG_VG
  FROM naechste_Vorgesetzte x, Fall f, Arbeitet_An a,
  naechste_Vorgesetzte y
  WHERE
    f.Titel LIKE '%Fussball%' AND
    f.AkZ = a.AkZ AND
    x.PersNr = a.PersNr AND
    x.Vorgesetzter = y.PersNr
  UNION ALL
  SELECT
    a.Vorname AS VN,
    a.Nachname AS NN,
    a.Dienstgrad AS DB,
    Vorname AS VN_VG,
    Nachname AS NN_VG,
    Dienstgrad AS DG_VG
```

```
  FROM naechste_Vorgesetzte a INNER JOIN Fussball_Vorgesetzte
  ON a.Vorgesetzter = PersNr
)

SELECT *
FROM Fussball_Vorgesetzte;
```

```
  vn   |   nn    |         dg          |   vn_vg   |  nn_vg   |        dg_vg
-------+---------+---------------------+-----------+----------+---------------------
 Hans  | Müller  | Polizeimeister      | Andreas   | Schmidt  | Polizeikommisar
 Hans  | Müller  | Polizeimeister      | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Sebastian | Wagner   | Polizeioberkommis
(4 rows)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Zirkus“ (66116-2018-F.T2-TA1-A2)

Entity-Relation-Modell
Relationenmodell

Das Fremdenverkehrsamt will sich einen besseren Überblick über Zirkusse verschaffen. In einer Datenbank sollen dazu die Zirkusse, die angebotenen Vorstellungen, die einzelnen Darbietungen in einer Vorstellung sowie die zugehörigen Dompteure und Tiere verwaltet werden.

Ein Zirkus wird durch seinen Namen gekennzeichnet und hat einen Besitzer. Vorstellungen haben eine `VorstellungsID` und ein Datum. Darbietungen haben neben der eindeutigen `ProgrammNr` eine Uhrzeit. Ein Dompteur hat eine eindeutige `AngestelltenNr` sowie einen Künstlernamen. Tiere sind eindeutig durch eine TierNr bestimmt und haben außerdem eine Bezeichnung der Tierart.

Ein Zirkus bietet Vorstellungen an und stellt Dompteure an. Eine Darbietung findet in einer Vorstellung statt. Des weiteren trainiert ein Dompteur Tiere. In einer Darbietung tritt ein Dompteur mit Tieren auf.

(a) 1.1

(i) Listen Sie die Entity-Typen und die zugehörigen Attribute auf.

```
* Zirkusse (Zirkus-Nummer, Namen)
  * Besitzer
  * Namen

* Vorstellungen (VorstellungsID)
  * VorstellungsID
  * Datum

* Darbietungen (ProgrammNr)
  * ProgrammNr
  * Datum

* Dompteuere (AngestelltenNr)
  * AngestelltenNr
  * Künstlernamen

* Tiere (TierNr)
  * TierNr
  * Tierart
```

(ii) Bestimmen Sie zu jedem Entity-Typen einen Schlüssel. Fügen Sie, wenn nötig einen künstlichen Schlüssel hinzu.

```
Zirkus (ZID, Besitzer, Name)
Vorstellung (VorstellungsID, Datum, ZID[Zirkus])
Darbietung (ProgrammNr, VorstellungsID[Vorstellung], Uhrzeit)
Dompteur (AngestelltenNr, Kuenstlername, ZID[Zirkus])
Tier (TierNr, Tierart)

trainiert (AngestelltenNr[Dompteur], TierNr[Tier])
trittAuf (AngestelltenNr[Dompteur], TierNr[Tier], ProgrammNr[Darbietung],
↪  VorstellungsID[Vorstellung])
```

(iii) Erstellen Sie das ER-Diagramm!

Vorstellungen werden von genau einem Zirkus angeboten. Ein Zirkus bietet mehrere Vorstellungen an und stellt mehrere Dompteure an. Ein Dompteur ist genau bei einem Zirkus angestellt. Eine Darbietung findet in einer bestimmten Vorstellung statt. Des weiteren trainiert ein Dompteur mehrere Tiere, ein Tier kann allerdings auch von mehreren Dompteuren trainiert werden. In einer Darbietung tritt genau ein Dompteur mit mindestens einem Tier auf.

(b) 1.2 Ergänzen Sie die Funktionalitäten im ER-Diagramm.

Vorstellungen werden von genau einem Zirkus angeboten. Ein Zirkus bietet mehrere Vorstellungen an und stellt mehrere Dompteure an. Ein Dompteur ist genau bei einem Zirkus angestellt. Eine Darbietung findet in einer bestimmten Vorstellung statt. Des weiteren trainiert ein Dompteur mehrere Tiere, ein Tier kann allerdings auch von mehreren Dompteuren trainiert werden. In einer Darbietung tritt genau ein Dompteur mit mindestens einem Tieren auf.

(c) 1.3

(i) Was bedeutet „mehrere“?

(ii) Ergänzen Sie die Kardinalitäten in min-max Notation im ER-Diagramm.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Schule Hogwarts aus Harry Potter“ (66116-2019-H.T1-TA2-A2)

Entity-Relation-Modell

Entwerfen Sie ein ER-Diagramm für eine Schule aus einer imaginären Film-Reihe. Geben Sie alle Attribute an und unterstreichen Sie Schlüsselattribute. Für die Angabe der Kardinalitäten von Beziehungen soll die Min-Max-Notation verwendet werden. Führen Sie wenn nötig einen Surrogatschlüssel ein.

An der Schule werden Schüler ausgebildet. Sie haben einen Namen, ein Geschlecht und ein Alter. Da die Schule klein ist, ist der Name eindeutig. Jeder Schüler ist Teil seines Jahrgangs, bestimmt durch Jahr und Anzahl an Schüler (Jahrgänge ohne Schüler sind erlaubt), und besucht mit diesem Kurse. Dabei wird jeder Kurs von min. einem Jahrgang besucht und jeder Jahrgang hat zwischen 2 und 5 Kurse. Kurse haben einen Veranstaltungsort und einen Namen.

Außerdem wird jeder Schüler einem von vier Häusern zugeordnet. Diese Häuser sind Gryffindor, Slytherin, Hufflepuff und Ravenclaw. Jedes Haus hat eine Anzahl an Mitgliedern.

Um die Organisation an der Schule zu erleichtern, gibt es pro Haus einen Vertrauensschiiler und pro Jahrgang einen Jahrgangssprecher. AuBerdem können Schüler Quidditch spielen. Dabei können sie die Rollen Sucher, Treiber, Jäger und Hüter spielen. Jedes Haus der Schule hat eine Mannschaft. Diese besteht aus genau einem Sucher, einem Hüter, drei Jäger und zwei Treiber. Jedes Jahr gibt es an der Schule eine Trophäe zu gewinnen. Diese ist abhängig von der Mannschaft und weiterhin durch das Jahr identifiziert.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Sportverein“ (66116-2019-H.T2-TA2-A1)

Entity-Relation-Modell

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-MaxNotation. Führen Sie Surrogatschlüssel nur dann ein, wenn es nötig ist und modellieren Sie nur die im Text vorkommenden Elemente.

Ein örtlicher Sportverein möchte seine Vereinsangelegenheiten mittels einer Datenbank verwalten. Der Verein besteht aus verschiedenen Abteilungen, welche eine eindeutige Nummer und einen aussagekräftigen Namen besitzen. Für jede Abteilung soll zudem automatisch die Anzahl der Mitglieder gespeichert werden, wobei ein Mitglied zu mehreren Abteilungen gehören kann. Die Mitglieder des Vereins können keine, eine oder mehrere Rollen (auch Ämter genannt) einnehmen. So gibt es die Ämter: 1. Vorstand, 2. Vorstand, Kassier, Jugendleiter, Trainer sowie einen Abteilungsleiter für jede Abteilung. Es ist dabei auch möglich, dass ein Abteilungsleiter mehrere Abteilungen leitet oder ein Mitglied mehrere Aufgaben übernimmt, mit der Einschränkung, dass die Vorstandsposten und Kassier nicht von der gleichen Person ausgeübt werden dürfen. Zu jedem Trainer wird eine Liste von Lizenzen gespeichert. Jeder Trainer ist zudem in mindestens einer Abteilung eine bestimmte Anzahl von Stunden tätig. Zu allen Mitgliedern werden Mitgliedsnummer, Name (bestehend aus Vor- und Nachname), Geburtsdatum, E-Mail, Eintrittsdatum, Adresse (bestehend aus PLZ, Ort, Straße, Hausnummer), IBAN und die Vereinszugehörigkeit in Jahren gespeichert.

Im Verein fallen Finanztransaktionen an. Zu jeder Transaktion wird ein Zeitstempel, der Betrag und eine eindeutige Transaktionsnummer gespeichert. Die Mitglieder leisten Zahlungen an den Verein. Umgekehrt erstattet der Verein auch bestimmte Kosten. Im Verein existieren drei verschiedene Mitgliedsbeiträge. So gibt es einen Kinder-und-Jugendlichen-Tarif, einen Erwachsenentarif und einen Familientarif.

Mitgliedern entstehen des Öfteren Fahrtkosten. Für jede Fahrtkostenabrechnung werden das Datum, die gefahrenen Kilometer und Start und Ziel, der Zweck sowie das Mitglied gespeichert, welches den Antrag gestellt hat. Zu jeder Fahrtkostenabrechnung existiert genau eine Erstattung.

Durch die Teilnahme an verschiedenen Wettbewerben besteht die Notwendigkeit die von einem Team (also mehreren Mitgliedern zusammen) oder Mitgliedern erzielten sportlichen Erfolge, d.h. Platzierungen, zu verwalten. Jeder Wettkampf besitzt eine eindeutige ID, ein Datum und eine Kurzbeschreibung.

Das Vereinsleben besteht aus zahlreichen Terminen, die durch Datum und Uhrzeit innerhalb einer

Abteilung eindeutig identifiziert werden können. Zu jedem Termin wird zusätzlich eine Kurzbeschreibung gespeichert.

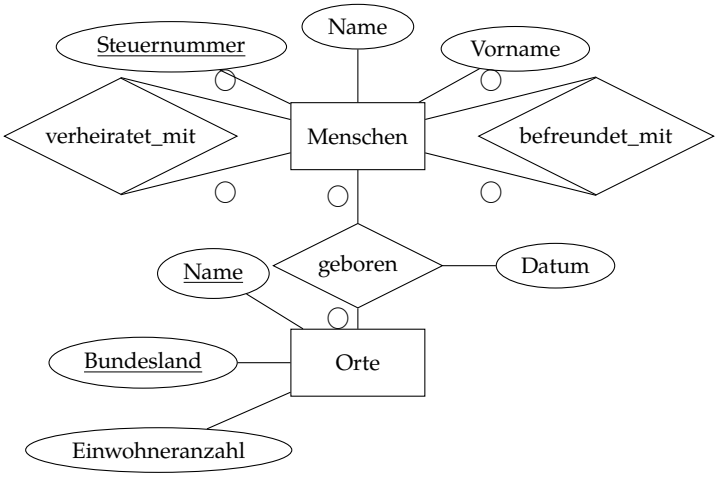
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Einwohnermeldeamt“ (66116-2020-F.T1-TA2-A1)

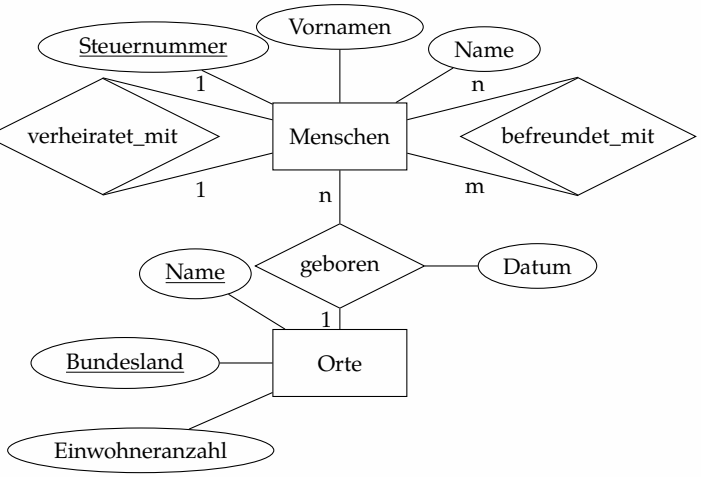
Entity-Relation-Modell

Gegeben sei folgendes ER-Diagramm:

(a) Übernehmen Sie das ER-Diagramm auf Ihre Bearbeitung und ergänzen Sie die Funktionalitätsangaben im Diagramm.



Lösungsvorschlag



(b) Übersetzen Sie das ER-Diagramm in ein relationales Schema. - Datentypen müssen nicht angegeben werden.

Lösungsvorschlag

```
Menschen(Steuernummer, Name, Vorname)

Orte(Name, Bundesland, Einwohneranzahl)

verheiratet_mit(Mensch[Menschen], Ehepartner[Menschen])

befreundet_mit(Mensch[Menschen], Freund[Menschen])
```

```
geboren(Datum, Steuernummer, Geburtsort[Orte], Geburtsbundesland[Orte])
```

(c) Verfeinern Sie das Schema aus Teilaufgabe b) indem Sie die Relationen zusammenfassen.

Lösungsvorschlag

```
Menschen(Steuernummer, Name, Vorname, Ehepartner[Menschen], Geburtsdatum, Geburts
ort[Orte], Geburtsbundesland[Orte])

Orte(Name, Bundesland, Einwohneranzahl)

befreundet_mit(Mensch[Menschen], Freund[Menschen])
```

(d) Geben Sie sinnvolle SQL Datentypen für Ihr verfeinertes Schema an.

Lösungsvorschlag

```
CREATE TABLE Menschen (
  Steuernummer BIGINT PRIMARY KEY,
  Name VARCHAR(30),
  Vorname VARCHAR(30),
  Ehepartner BIGINT REFERENCES Steuernummer,
  Geburtsdatum DATE,
  Geburtsort VARCHAR(30) REFERENCES Orte(Name),
  Geburtsbundesland VARCHAR(30) REFERENCES Orte(Bundesland)
);

CREATE TABLE Orte (
  Name VARCHAR(30),
  Bundesland VARCHAR(30),
  Einwohneranzahl INTEGER,
  PRIMARY KEY (Name, Bundesland)
);

CREATE TABLE befreundet_mit (
  Mensch BIGINT REFERENCES Mensch(Steuernummer),
  Freund BIGINT REFERENCES Mensch(Steuernummer),
  PRIMARY KEY (Mensch, Freund)
);
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex

Examensaufgabe „Wetterdienst“ (66116-2020-F.T2-TA2-A1)

Entity-Relation-Modell

Hinweis: Bei Wahl dieser Aufgabe wird Wissen über das erweiterte Entity- Relationship-Modell (beispielsweise schwache Entity-Typen, Vererbung) sowie die Verfeinerung eines relationalen Sche- mas vorausgesetzt.

Gegeben seien folgende Informationen:

- Ein Wetterdienst - identifiziert durch einen eindeutigen Namen - betreibt mehrere Wetterstationen, die mit einer eindeutigen Nummer je Wetterdienst identifiziert werden können. Jede Wetterstation hat zudem mehrere Messgeräte, die wiederum pro Wetterstation einen eindeutigen Code besitzen und zudem eine Betriebsdauer.
- Ein Wetterdienst hat eine Adresse.
- Bei den Messgeräten wird unter anderem zwischen manuellen und digitalen Messgeräten unterschieden. Dabei kann ein Messgerät immer nur zu einer Kategorie gehören.
- Meteorologen sind Mitarbeiter eines Wetterdienstes, haben einen Namen und werden über eine Personalnummer identifiziert. Zudem soll gespeichert werden, in welcher Wetterstation welcher Mitarbeiter zu welchem Zeitpunkt arbeitet.
- Meteorologen können für eine Menge an Messgeräten verantwortlich sein. Manuelle Messgeräte werden von Meteorologen abgelesen.
- Ein Wettermoderator präsentiert das vorhergesagte Wetter eines Wetterdienstes für einen Fernsehsender.
- Für einen Wettermoderator wird der eindeutige Name und die Größe gespeichert. Ein Fernsehsender wird ebenfalls über den eindeutigen Namen identifiziert.

(a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.

Verwenden Sie dabei - wenn angebracht - das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen (existenzabhängige Beziehungen) und schwache Entity-Typen.

Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

(b) Überführen Sie das in Teilaufgabe a) erstellte ER-Modell in ein verfeinertes relationales Schema. Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden. Die einzelnen Schritte müssen angegeben werden.

Lösungsvorschlag

1. Schritt: Starke Entity-Typen einfügen Wetterdienst Name, Adresse Meteorologe Personalnummer, Name Wettermoderator Name, Größe Fernsehsender Name 2. Schritt: Schwache Entity-Typen einfügen Wetterstation Name [Wetterdienst], Nummer Messgeraet Name [Wetterdienst], Nummer [Wetterstation], Code, Betriebsdauer 3. Schritt: Is-A-Beziehung einfügen ManuellesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Mess-

geraet] DigitalesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] 4. Schritt: Beziehungen einfügen Betreibt Name [Wetterdienst], Nummer Hat Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] ArbeitetIn Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Zeitpunkt IstMitarbeiterVon Personalnummer [Meteorologe], Name [Wetterdienst] IstVerantwortlichFür Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] LiestAb Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] PraesentiertWetter ModeratorenName [Wettermoderator], FernsehsenderName [Fernsehsender], WetterdienstName [Wetterdienst] (Bei PraesentiertWetter sind nur zwei Felder Teil des Primärschlüssels, da es sich um eine 1 : 1 : n- Beziehung handelt.) 5. Schritt: Verfeinerung Wetterdienst Name, Adresse Wetterstation Name [Wetterdienst], Nummer [Wetterstation] Messgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet], Betriebsdauer, VerantwortlicherMitarbeiterPersonalnummer [Meteorologe] ManuellesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] DigitalesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] Meteorologe Personalnummer, Name, WetterdienstName [Wetterdienst] ArbeitetIn Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Zeitpunkt LiestAb Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] Wettermoderator Name, Größe Fernsehsender Name PraesentiertWetter ModeratorenName [Wettermoderator], FernsehsenderName [Fernsehsender], WetterdienstName [Wetterdienst]

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Automobilproduktion“ (66116-2021-F.T1-TA2-A2)

Entity-Relation-Modell

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-Max-Notation. Führen Sie Surrogatschlüssel (künstlich definierte Schlüssel) nur dann ein, wenn es nötig ist, und modellieren Sie nur die im Text vorkommenden Elemente.

Zunächst gibt es **Autos**, die einen eindeutigen *Namen* (`AName`), einen *Typ* sowie eine Liste an *Ausstattungen* besitzen.

- ☐ **E: Autos**
- ○ *A: Namen*
- ○ *A: Typ*
- ○ *A: Ausstattungen*

Autos werden aus **Bauteilen** zusammengesetzt. Diese besitzen eine *ID* sowie eine *Beschreibung*.

- ☐ **E: Bauteilen**
- ◇ R: zusammengesetzt
- ○ *A: ID*
- ○ *A: Beschreibung*

Jedes Bauteil wird von genau einem **Zulieferer** geliefert. Zu jedem Zulieferer werden sein *Name* (`ZName`) sowie seine *E-Mail-Adresse* (`EMailAdresse`) gespeichert.

- ☐ **E: Zulieferer**
- ○ *A: Name*
- ○ *A: E-Mail-Adresse*

Weiter gibt es **Werke**, die einen eindeutigen *Namen* sowie einen *Standort* besitzen.

- ☐ **E: Werke**
- ○ *A: Namen*
- ○ *A: Standort*

Jedes Werk besteht aus **Hallen**, welche werksintern eindeutig *nummeriert* sind. Zudem besitzt eine Halle noch eine gewisse *Größe* (in m^2).

- ◇ R: besteht
- ☐ **E: Hallen**
- ○ *A: nummeriert*
- ○ *A: Größe*

Es gibt genau zwei Typen von Hallen: **Produktionshallen** und **Ersatzteillager**.

- ☐ **E: Produktionshallen**
- ☐ **E: Ersatzteillager**

In jeder Produktionshalle wird mindestens ein Auto hergestellt.

- ◇ R: hergestellt

Zu den Ersatzteillagern wird festgehalten, welche Bauteile und wie viele davon sich dort befinden.

- ◇ R: festgehalten

Zu jedem **Mitarbeiter** werden eine eindeutige *ID*, sein *Vor-* und *Nachname*, die *Adresse* (*Straße, PLZ, Ort*), das *Gehalt* sowie das *Geschlecht* gespeichert.

- ☐ **E: Mitarbeiter**
- ○ *A: ID*
- ○ *A: Vor-*
- ○ *A: Nachname*
- ○ *A: Adresse (Straße, PLZ, Ort)*
- ○ *A: Gehalt*
- ○ *A: Geschlecht*

Mitarbeiter werden unter anderem in **Reinigungskräfte**, **Werksarbeiter** und **Ingenieure** unterteilt.

- ☐ **E: Reinigungskräfte**
- ☐ **E: Werksarbeiter**
- ☐ **E: Ingenieure**

Zu den Ingenieuren wird zusätzlich der *Hochschulabschluss* gespeichert. Ingenieure sind genau einem Werk zugeordnet, Werksarbeiter einer Halle.

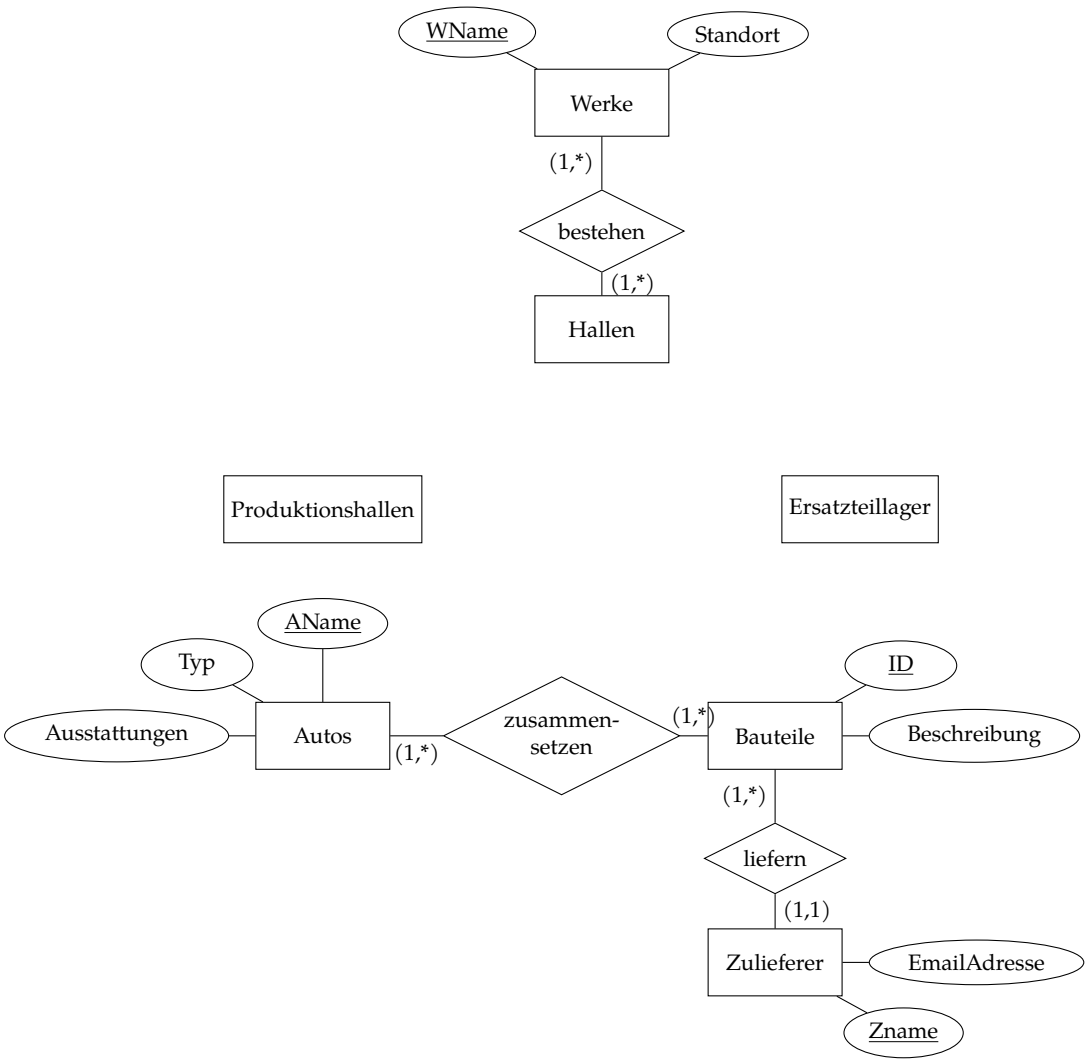
- ○ *A: Hochschulabschluss*
- ◇ R: zugeordnet
- ◇ R: Halle

Eine Reinigungskraft reinigt mindestens eine Halle. Jede Halle muss regelmäßig gereinigt werden.

- ◇ R: reinigt

Weiter sind Ingenieure Projekten zugeteilt. Zudem wird zu jedem Projekt genau ein Ingenieur als *Projektleiter* festgehalten.

- ◇ R: zugeteilt
- ○ *A: Projektleiter*



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Online-Marktplatze“ (66116-2021-F.T2-TA2-A2)

Entity-Relation-Modell

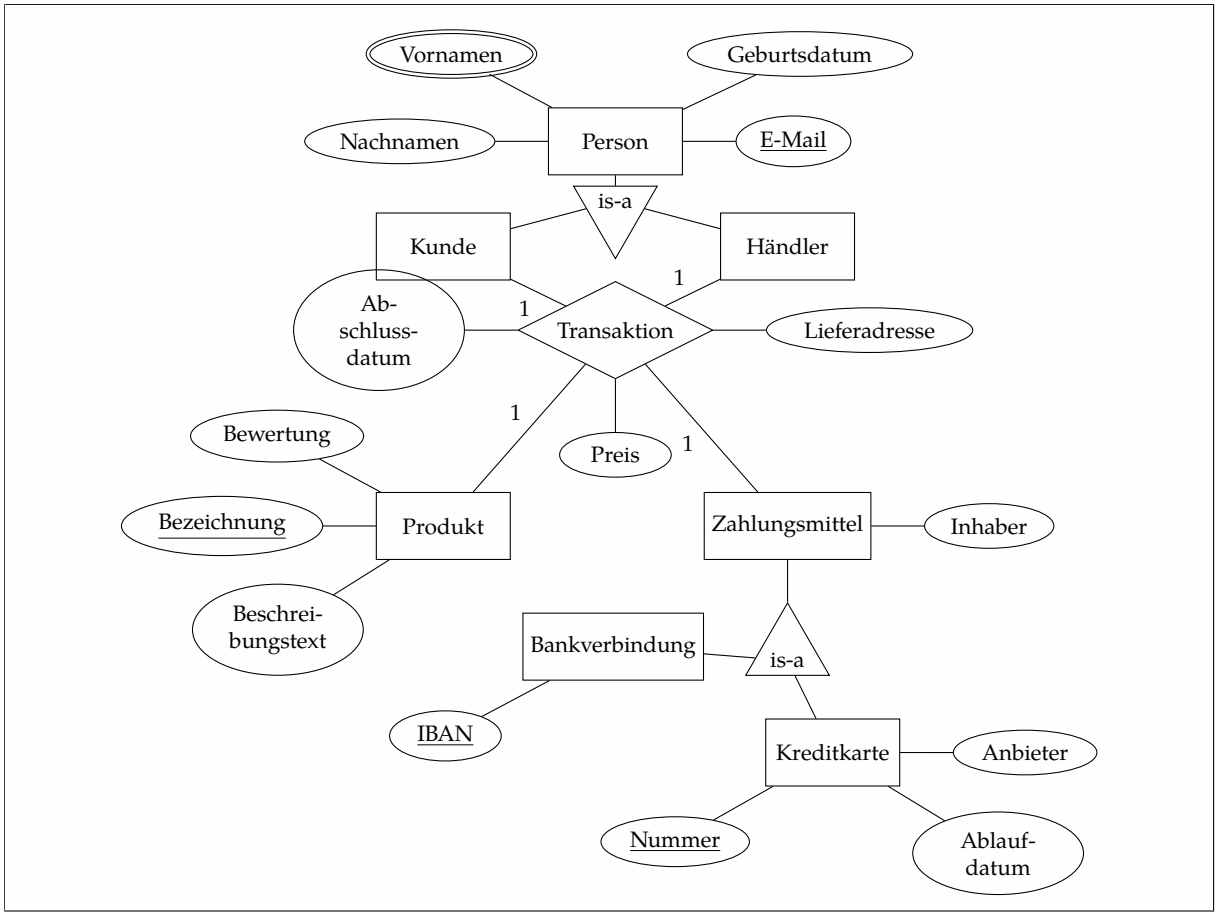
Im Folgenden finden Sie die Beschreibung eines Online-Marktplatzes. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme von Entity-Typen an Beziehungstypen.

Es gibt **Produkte**. Diese haben eine eindeutige *Bezeichnung*, einen *Beschreibungstext* und eine *Bewertung*. Außerdem gibt es **Personen**, die entweder **Kunde**, **Händler** oder beides sind. Jede Person hat einen *Nachnamen*, einen oder mehrere *Vornamen*, ein *Geburtsdatum* und eine *E-Mail-Adresse*, mit der diese eindeutig identifiziert werden kann.

Das System verwaltet außerdem **Zahlungsmittel**. Jedes Zahlungsmittel ist entweder eine **Kreditkarte** oder eine **Bankverbindung** für Lastschriften. Für das Lastschriftverfahren wird die international eindeutige *IBAN* und der Name des *Kontoinhabers* erfasst, bei Zahlung mit Kreditkarte der Name des *Karteninhabers*, die eindeutige *Kartennummer*, das *Ablaufdatum* sowie der *Kartenanbieter*. Es gibt Transaktionen. Jede Transaktion bezieht sich stets auf ein Produkt, einen Kunden, einen Händler und auf ein Zahlungsmittel, das für die Transaktion verwendet wird. Jede Transaktion enthält außerdem den *Preis*, auf den sich Kunde und Händler geeinigt haben, das *Abschlussdatum* sowie eine *Lieferadresse*, an die das Produkt versandt wird.

- ☐ **E: Produkte**
- ○ *A: Bezeichnung*
- ○ *A: Beschreibungstext*
- ○ *A: Bewertung*
- ☐ **E: Personen**
- ☐ **E: Kunde**
- ☐ **E: Händler**
- ○ *A: Nachnamen*
- ○ *A: Vornamen*
- ○ *A: Geburtsdatum*
- ○ *A: E-Mail-Adresse*
- ☐ **E: Zahlungsmittel**
- ☐ **E: Kreditkarte**
- ☐ **E: Bankverbindung**
- ○ *A: IBAN*
- ○ *A: Kontoinhabers*
- ○ *A: Karteninhabers*
- ○ *A: Kartennummer*
- ○ *A: Ablaufdatum*
- ○ *A: Kartenanbieter*
- ◇ R: Transaktionen
- ○ *A: Preis*
- ○ *A: Abschlussdatum*
- ○ *A: Lieferadresse*

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Übungsaufgabe „DBTec“ (Entity-Relation-Modell)

Entity-Relation-Modell

Die Firma *DBTec* fertigt verschiedene Geräte. Für die betriebliche Organisation dieser Firma soll eine relationale Datenbank eingesetzt werden. Dabei gilt folgendes: Jedes **Bauteil**, das verwendet wird, hat eine eindeutige *Nummer* und eine *Bezeichnung*, die allerdings für mehrere verschiedene Bauteile gleich sein kann. Von jedem Teil werden außerdem der *Name des Herstellers*, der *Einkaufspreis* pro Stück und der am Lager vorhandene *Vorrat* gespeichert.

Jedes herzustellende **Gerät** hat eine eindeutige *Bezeichnung*. Auch von jedem schon gefertigten Gerätetyp soll der aktuelle *Lagerbestand* gespeichert werden, ebenso wie der *Verkaufspreis* des Gerätes. In unserem fiktiven Betrieb gilt die Regelung, dass Maschinen, die mehr als 1000,- EUR kosten, unentgeltlich an die Kunden ausgeliefert werden; für Geräte, die weniger kosten, ist zusätzlich zum Preis eine gerätespezifische *Anliefergebühr* zu entrichten. In der Datenbank ist ebenfalls zu speichern, welche Bauteile für welche Geräte benötigt werden. Es gibt Bauteile, die für mehrere Geräte verwendet werden.

Von jedem **Kunden** werden der *Name*, die *Adresse* und die *Branche* gespeichert. Es kann verschiedene Kunden mit demselben Namen oder derselben Adresse geben. Außerdem ist zu jedem Kunden vermerkt, wer aus unserer Firma für die entsprechende Kundenbetreuung zuständig ist. Natürlich ist auch zu speichern, welche Kunden mit welchen Geräten beliefert werden. Es kann sein, dass gewissen Kunden für bestimmte Geräte *Sonderkonditionen* eingeräumt worden sind, dies soll ggf. ebenfalls in der Datenbank vermerkt werden.

- **E: Bauteil**
  - *A: Nummer*
  - *A: Bezeichnung*
  - *A: Name des Herstellers*
  - *A: Einkaufspreis*
  - *A: Vorrat*
- **E: Gerät**
  - *A: Bezeichnung*
  - *A: Lagerbestand*
  - *A: Verkaufspreis*
  - *A: Anliefergebühr*
- R: benötigt
- **E: Kunden**
  - *A: Name*
  - *A: Adresse*
  - *A: Branche*
- R: Kundenbetreuung zuständig
- R: beliefert
  - *A: Sonderkonditionen*

(a) Bestimmen Sie die Entity- und die Relationship-Typen mit ihren Attributen und zeichnen Sie ein mögliches Entity-Relationship-Diagramm!

(b) Bestimmen Sie zu allen Entity-Typen einen Primärschlüssel und tragen Sie diese in das Modell ein.

(c) Bestimmen Sie die Funktionalitäten (1:1, 1:n, n:m) der Relationship-Typen und tragen Sie diese in das Modell ein.

(d) In der Firma wird ein neues Betreuungssystem eingeführt. Jeder **Kundenbetreuer** ist für die Kunden eines festgelegten Bezirks zuständig. Die **Bezirke** sind *durchnummeriert*. Für jeden Bezirk existiert eine *Beschreibung*, die nicht näher festgelegt ist. Erweitern Sie Ihr ER-Modell aus Teilaufgabe a) entsprechend. Bezirke werden nur festgelegt, wenn es dazu auch Kunden gibt.

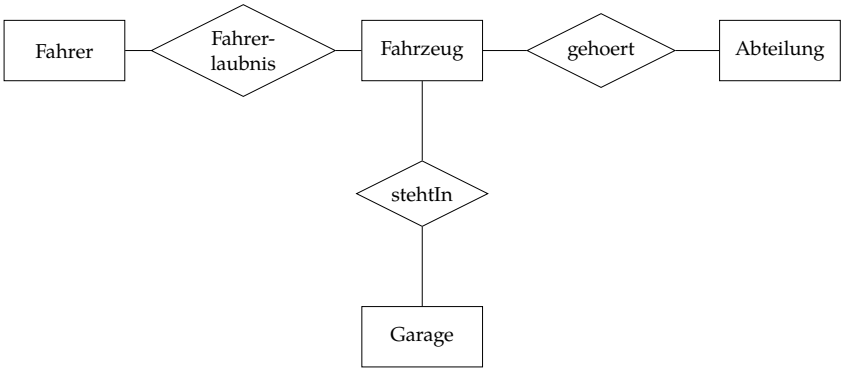
- **E: Kundenbetreuer**
- R: zuständig
- **E: Bezirke**
  - *A: durchnummeriert*
  - *A: Beschreibung*

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/20_Datenbankentwurf/01_Entity-Relation-Modell/Aufgabe_DBTec.tex`

Übungsaufgabe „Fahrzeugverwaltung“ (Entity-Relation-Modell)

Entity-Relation-Modell

Gegeben ist das folgende ER-Modell der Fahrzeugverwaltung einer Firma:

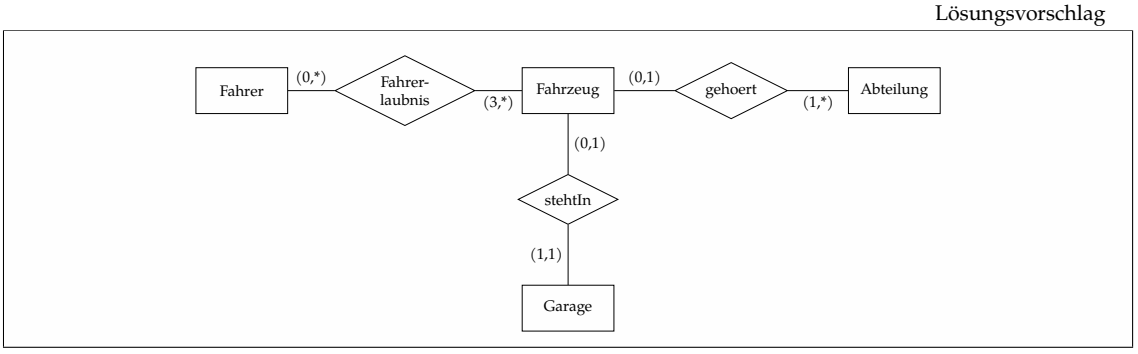


Die Attribute wurden aus Einfachheitsgründen weggelassen. Es gelten folgende Bedingungen:

- Jedes Fahrzeug gehört zu höchstens einer Abteilung, wobei aber jede Abteilung mindestens ein Fahrzeug hat.
- Für fast alle Fahrzeuge gibt es eine (fest zugeordnete) Einzelgarage. Jede dieser Garagen ist belegt.
- Für jedes Fahrzeug muss es mindestens drei Personen mit einer entsprechenden Fahrerlaubnis geben. Ansonsten gibt es keine Einschränkung.

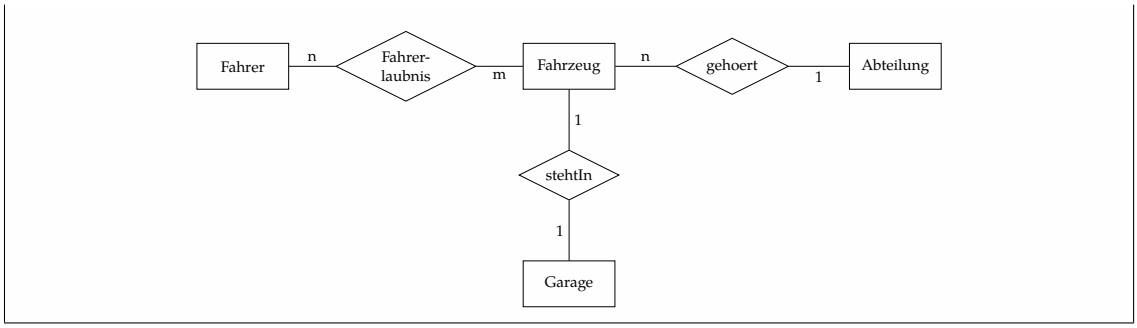
(a) Geben Sie gemäß obiger Bedingungen geeignete Funktionalitäten in der (min, max)-Notation an.

Lösungsvorschlag



(b) Wie lauten die entsprechenden Funktionalitätsangaben (z. B. 1:1, n:m etc.)?

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/20_Datenbankentwurf/01_Entity-Relation-Modell/Aufgabe_Fahrzeugverwaltung.tex`

Übungsaufgabe „Flughafen“ (Entity-Relation-Modell)

Für einen Flughafen soll eine Datenbank für folgendes Szenario entwickelt werden: Von den **Fluggesellschaften** sollen *Name* und *Hauptsitz* abgespeichert werden. Der Gesellschaftsname ist dabei eindeutig. Die Gesellschaften sind Eigentümer von Flugzeugen. Wichtig ist, seit wann das Flugzeug für die Gesellschaft im Einsatz ist und wie viele Flugzeuge die Gesellschaft insgesamt besitzt.

Die **Flugzeuge** tragen eine eindeutige *FZ-Nr*. Weiterhin soll auch der **Typ** des Flugzeuges mit *Sitzplatzanzahl* und *Größe der Besatzung* abrufbar sein.

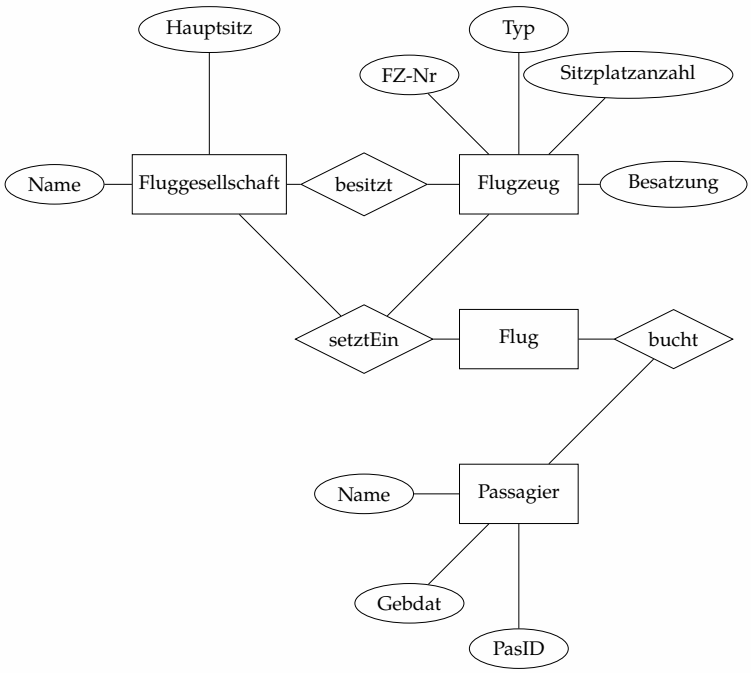
Für einen **Flug** setzt eine Fluggesellschaft ein Flugzeug ein. Dabei muss dieses Flugzeug aber nicht Eigentum der Fluggesellschaft sein. Der Flug hat eine *Flug-Nr*. Bezüglich eines Fluges sind *Flug-Nr*, *Abflugzeit* und *Zielflughafen* abzuspeichern.

Ein **Passagier** kann Flüge buchen. Von den Passagieren müssen *Name* und *Gebdat* bekannt sein. Dabei ist aber davon auszugehen, dass es Passagiere mit gleichem Namen und gleichem Gebdat geben kann.

Bei der **Buchung** wird dem Passagier eine *Sitzplatz-Nr* zugeteilt. Für jeden Flug muss die Anzahl der gebuchten Plätze feststellbar sein.

(a) Erstellen Sie ein ER-Diagramm! Verarbeiten Sie dabei nur die unbedingt notwendigen Informationen. Geben Sie an, wie die nicht im ER-Modell „auftauchenden“ Informationen bestimmt werden können.

Lösungsvorschlag



Wie viele Flugzeuge eine Fluggesellschaft besitzt ergibt sich nicht direkt aus dem ER-Modell, sondern kann später durch eine einfache SQL-Abfrage ermittelt werden:

```
SELECT FluggesellschaftName, COUNT (*) AS Anzahl
FROM Eigentuemer_von
GROUP BY FluggesellschaftName;
```

Auch die Anzahl der gebuchten Sitze pro Flug lässt sich durch eine SQL-Abfrage ermit- teln:

```
SELECT Flug-Nr, COUNT (*) AS Anzahl
FROM bucht
GROUP BY Flug-Nr;
```

(b) Legen Sie die Primärschlüssel fest. Begründen Sie Ihre Entscheidung, falls Sie zusätzliche künstliche Schlüssel einfügen.

Lösungsvorschlag

Von einem Passagier werden nur Name und Gebdat gespeichert. Da diese beiden nicht eindeutig sind, können sie nicht als Primärschlüssel verwendet werden. Folglich muss ein künstlicher Schlüssel eingeführt werden, die `Passagier-Nr`. `Typ-Nr` des Flugzeugtyps sollte auch gespeichert werden, da Personalzahl und Sitzplatzzahl nicht eindeutig sind.

(c) Geben Sie die Funktionalitäten an!

(d) Erstellen Sie nun ein zu dieser Modellierung passendes Relationenschema! Markieren Sie Schlüssel und Fremdschlüssel.

Lösungsvorschlag

`angehoeren` wird aufgelöst: `Typ-Nr[Flugzeugtyp]` nach `Flugzeug`

`besitzen` wird aufgelöst: `FGName[Fluggesellschaft]` nach `Flugzeug`

`fliegen` wird aufgelöst: `FGName[Fluggesellschaft] FZ-Nr[Flugzeug]` nach `Flug`

`Fluggesellschaft(`<u>`FGName`</u>`, Hauptsitz)`

`Flug(`<u>`Flug-Nr`</u>`, Abflugzeit, Zielflughafen,` <u>`FGName[Fluggesellschaft]`</u>`,` <u>`FZ-Nr[Flugzeug]`</u>`)`

`Flugzeug(`<u>`FZ-Nr`</u>`,` <u>`FGName[Fluggesellschaft]`</u>`, DatumErsterEinsatz,` <u>`Typ-Nr[Flugzeugtyp]`</u>`)`

`Flugzeugtyp(`<u>`Typ-Nr`</u>`, SitzplatzAnzahl, GroesseBesatzung)`

`Passagier(`<u>`Pass-Nr`</u>`, Name, Gebdat)`

`buchen(`<u>`Pass-Nr[Passagier]`</u>`,` <u>`Flug-Nr[Flug]`</u>`, Sitzplatz-Nr)`

(e) Finden Sie nun jeweils eine Datenbank-Anfrage (in SQL und in relationaler Algebra) zur Lösung der folgenden Problemstellungen:

a. Die Fluggesellschaft „Never-Come-Back-Airlines“ (NCA) will wissen, ob (und wenn ja bei welchem Flug) heute Abend Passagiere (Name, Sitzplatz-Nr) mit einem ihrer Flugzeuge unterwegs sind, die heute Geburtstag haben (GebDat = TODAY).

Lösungsvorschlag

$$\pi_{\text{Name,Sitzplatz-Nr}}(\sigma_{\text{GebDat=TODAY}}(\text{Passagier}) \bowtie \text{buchen} \bowtie \sigma_{\text{Name='Never-Come-Back-Airlines'} \wedge \text{Abflugzeit}>18.00}(\text{Flug}))$$

```
SELECT p.Name, p.Sitzplatz-Nr
FROM Passagier p, Flug f, buchen b
WHERE
  f.Name = 'Never-Come-Back-Airlines' AND
  f.Abflugzeit > 18.00 AND
  p.Gebdat = TODAY AND
  f.Flug-Nr = b.Flug-Nr AND
  b.Pass-Nr = p.Pass-Nr;
```

b. Ein Passagier möchte erfahren, welcher Flug (Flug-Nr, FZ-Nr, Abflugzeit, FGName) derjenige mit dem „modernsten“ Flugzeug ist, der nach „London“ geht.

Lösungsvorschlag

```
SELECT f.Flug-Nr, f.FZ-Nr, f.Abflugzeit, f.FGName
FROM Flug f, Flugzeug fz
WHERE
  f.FZ-Nr = fz.FZ-Nr AND
  f.Zielflughafen = 'London' AND
  fz.DatumErsterEinsatz = (
    SELECT MAX(Flugzeug.DatumErsterEinsatz)
    FROM Flug, Flugzeug
    WHERE
      Flug.Zielflughafen = 'London' AND
      Flug.FZ-Nr = Flugzeug.FZ-Nr
  )
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/20_Datenbankentwurf/01_Entity-Relation-Modell/Aufgabe_Flughafen.tex

Übungsaufgabe „Bundesliga-Datenbank“ (Relationale Algebra, SQL, Entity-Relation-Modell)

Relationale Algebra
SQL
Entity-Relation-Modell

Gegeben sei die folgende Bundesliga-Datenbank, in der die Vereine, Spiele, Trainer und Spieler mit ihren Einsätzen für die laufende Saison verwaltet werden:

- VEREIN (VNAME, ORT, PRÄSIDENT)
- SPIELE (HEIM, GAST, RESULTAT, ZUSCHAUER, TERMIN, SPIELTAG, H-TRAINER, G-TRAINER)
- SPIELER (SPNR, NAME, VORNAME, VEREIN, ALTER, GEHALT, GEB-ORT)
- TRAINER (TRNR, NAME, VORNAME, VEREIN, ALTER, GEHALT, GEB-ORT)
- EINSATZ (HEIM, GAST, SPNR, VON, BIS, TORE, KARTE)

(a) Zeichnen Sie das „zugehörige“ ER-Modell!

(b) Formulieren Sie folgende Anfragen in relationaler Algebra und in SQL:

- Welche Spieler haben beim Spiel TSV 1860 München – FC Bayern München mitgewirkt?

Lösungsvorschlag

$\pi_{\text{NAME,VORNAME}}(\text{Spieler} \bowtie (\sigma_{\text{HEIM}=\text{'TSV 1860 München'} \wedge \text{GAST}=\text{'FC Bayern München'}}(\text{Einsatz})))$

```
SELECT NAME, VORNAME FROM Spieler, Einsatz,
WHERE
  HEIM = 'TSV 1860 München' AND GAST = 'FC Bayern München' AND
  Einsatz.SPNR = Spieler.SPNR;
```

- Welche Spiele sind 2 : 0 ausgegangen?

Lösungsvorschlag

$\pi_{\text{HEIM,GAST,SPIELTAG}}(\sigma_{\text{RESULTAT}=\text{'2:0'}}(\text{SPIELE}))$

```
SELECT HEIM, GAST, SPIELTAG FROM SPIELE
WHERE RESULTAT = '2 : 0';
```

- Welche Spieler spielen in einem Verein ihres Geburtsortes?

Lösungsvorschlag

$\pi_{\text{NAME,VORNAME}}(\text{VEREIN} \bowtie_{\text{VEREIN.ORT}=\text{SPIELER.GEB-ORT} \wedge \text{SPIELER.VEREIN}=\text{VEREIN.VNAME}} \text{SPIELER})$

```
SELECT NAME, VORNAME
FROM SPIELER, VEREIN
WHERE VEREIN.ORT = SPIELER.GEB-ORT AND SPIELER.VEREIN = VEREIN.VNAME;
```

- Welche Spieler vom 1. FC Köln haben alle Spiele mitgemacht?

Lösungsvorschlag

$$\pi_{\text{NAME,VORNAME}}($$
$$\sigma_{\text{VEREIN='1. FC Köln'}}(\text{SPIELER})$$
$$\bowtie$$
$$(\pi_{\text{HEIM,GAST,SPNR}}(\sigma_{\text{HEIM='1. FC Köln'} \vee \text{GAST='1. FC Köln'}}(\text{EINSATZ}))$$
$$\div$$
$$\pi_{\text{HEIM,GAST}}(\sigma_{\text{HEIM='1. FC Köln'} \vee \text{GAST='1. FC Köln'}}(\text{SPIELE}))))$$

```sql
SELECT NAME, VORNAME
FROM SPIELER
WHERE VEREIN = '1. FC Koeln' AND SPNR IN (
  SELECT SPNR
  FROM EINSATZ
  WHERE HEIM = '1. FC Koeln' OR GAST = '1. FC Koeln'
  GROUP BY SPNR
  HAVING COUNT(*) = (
    SELECT COUNT(*)
    FROM SPIELE
    WHERE HEIM = '1. FC Koeln' OR GAST = '1. FC Koeln'
  )
);
```

- Wie heißen die Präsidenten der Vereine, die zur Zeit einen Trainer beschäftigen, der jünger ist als der älteste Spieler, der beim Verein beschäftigt ist?

Lösungsvorschlag

$$\pi_{\text{PRÄSIDENT}}($$
$$\text{VEREIN}$$
$$\bowtie$$
$$\pi_{\text{VEREIN}}($$
$$\text{TRAINER}$$
$$\bowtie_{\text{TRAINER.ALTER} < \text{SPIELTER.ALTER} \wedge \text{TRAINER.VEREIN} = \text{SPIELER.VEREIN}}$$
$$\text{SPIELER}$$
$$)$$
$$)$$

```sql
SELECT PRÄSIDENT
FROM VEREIN, SPIELER, TRAINER
WHERE
  VEREIN.VNAME = SPIELER.VEREIN AND
  VEREIN.VNAME = VEREIN.TRAINER AND
  TRAINER.ALTER < SPIELER.ALTER;
```

- Welche Spieler haben bisher noch nie gespielt?

Lösungsvorschlag

$\pi_{\text{SPNR}}(\text{SPIELER}) - \pi_{\text{SPNR}}(\text{EINSATZ})$

```
SELECT SPNR FROM SPIELER
EXCEPT
SELECT SPNR FROM EINSATZ;
```

- Welche Spieler haben bisher noch kein Tor geschossen?

Lösungsvorschlag

$\pi_{\text{SPNR}}(\text{SPIELER}) - \pi_{\text{SPNR}}(\sigma_{\text{TORE}>0}(\text{EINSATZ}))$

```
SELECT SPNR FROM SPIELER
EXCEPT
SELECT SPNR FROM EINSATZ WHERE TORE > 0;
```

- Welcher Trainer hat schon mehr als einen Verein trainiert? Welche Vereine haben schon mehrere Trainer gehabt?

Lösungsvorschlag

Vereine:

```
SELECT HEIM FROM SPIELE l, SPIELE r
WHERE l.HEIM = r.HEIM AND NOT (l.H-TRAINER = r.H-TRAINER)
UNION
SELECT GAST FROM SPIELE l, SPIELE r
WHERE l.GAST = r.GAST AND NOT (l.G-TRAINER = r.G-TRAINER)
UNION
SELECT HEIM FROM SPIELE l, SPIELE r
WHERE l.HEIM = r.GAST AND NOT (l.H-TRAINER = r.G-TRAINER)
```

Trainer:

```
SELECT H-TRAINER FROM SPIELE l, SPIELE r
WHERE l.H-TRAINER = r.HEIM AND NOT (l.HEIM = r.HEIM)
UNION
SELECT G-TRAINER FROM SPIELE l, SPIELE r
WHERE l.G-TRAINER = r.G-TRAINER AND NOT (l.GAST = r.GAST)
UNION
SELECT H-TRAINER FROM SPIELE l, SPIELE r
WHERE l.H-TRAINER = r.G-TRAINER AND NOT (l.H-TRAINER = r.G-TRAINER)
```

- Welche Spiele am 10. Spieltag hatten mehr als 30.000 Zuschauer?

Lösungsvorschlag

$\pi_{\text{HEIM,GAST}}(\sigma_{\text{ZUSCHAUER}>30000 \wedge \text{SPIELTAG}=10}(\text{SPIELE}))$

```
SELECT HEIM, GAST
  FROM SPIELE
  WHERE ZUSCHAUER > 30000 AND SPIELTAG = 10;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Bundesliga-Datenbank.tex

Relationales Modell

Examensaufgabe „Browser-Online-Spiele“ (46116-2013-F.T1-TA2-A2)

2. Anfragen

Zu einer Website, auf der Besucher im Browser Online-Spiele spielen können, liegt das folgende relationale Schema einer Datenbank vor:

```
Team : {[TNr, Name, Teamfarbel]}
Spieler : {[SNr, Name, Icon, TNr, EMail]}
Minispiel : {[MNr, Name, Kategorie, Schwierigkeit]}
Wettkampf : {[WNr, Sieger, Geschlagener, MNr, Dauer]}
```

Auf der Website treten jeweils zwei Spieler gegeneinander in Minispielen an. In diesen ist es das Ziel, den gegnerischen Spieler in möglichst kurzer Zeit zu besiegen. Minispiele gibt es dabei in verschiedenen Schwierigkeitsstufen („leicht“, „mittel“, „schwer“, „sehr schwer“) und verschiedenen Kategorien („Denkspiel“, „Geschicklichkeitsspiel“, usw.). Die Attribute *Sieger* und *Geschlagener* sind jeweils Fremdschlüsselattribute, die auf das Attribut *SNr* der Relation *Spieler* verweisen. Beachten Sie, dass das Dauer-Attribut der Wettkampf-Relation die Dauer eines Wettkampfes in der Einheit Sekunden speichert.

(a) Formulieren Sie geeignete Anfragen in relationaler Algebra für die folgenden Teilaufgaben:

(i) Geben Sie die *Namen* der *Minispiele* zurück, die zur *Kategorie „Denkspiele“* zählen.

Lösungsvorschlag

$$\pi_{\text{Name}}(\sigma_{\text{Kategorie}=\text{'Denkspiele'}}(\text{Minispiele}))$$

(ii) Geben Sie die *Namen* und *E-Mail-Adressen* aller Spieler zurück, die in einem Minispiel des Typs *„Geschicklichkeitsspiel“* gewonnen haben.

Lösungsvorschlag

$$\pi_{\text{Spieler.Name,Spieler.Email}}(\sigma_{\text{Kategorie}=\text{'Geschicklichkeitsspiel'}}(\text{Minispiele}) \bowtie_{\text{Minispiel.MNr}=\text{Wettkampf.MNr}} \text{Wettkampf} \bowtie_{\text{Wettkampf.Sieger}=\text{Spieler.SNr}} \text{Spieler})$$

(b) Formulieren Sie geeignete SQL-Anfragen für die folgenden Teilaufgaben. Beachten Sie dabei, dass Ihre Ergebnisrelationen keine Duplikate enthalten sollen.

(i) Geben Sie jede Spielekategorie aus, für die ein Minispiel der Schwierigkeitsstufe sehr schwer vorhanden ist.

Lösungsvorschlag

```
SELECT DISTINCT Kategorie
FROM Minispiel
WHERE Schwierigkeit = 'sehr schwer';
```

(ii) Geben Sie die Wettkämpfe aus, deren Dauer unter der durchschnittlichen Dauer der Wettkämpfe liegt.

Lösungsvorschlag

```
SELECT WNr
FROM Wettkampf
WHERE Dauer < (
  SELECT AVG(Dauer) FROM Wettkampf
);
```

(iii) Geben Sie für jeden *Spieler* seine *SNr*, seinen *Namen*, die Anzahl seiner Siege, die durchschnittliche Dauer seiner siegreichen Wettkämpfe und die Anzahl der Teams, aus denen er bereits mindestens einen Spieler besiegt hat, zurück.

```
SELECT
  SNr,
  Name,
  (
    SELECT COUNT(*)
    FROM Wettkampf
    WHERE Wettkampf.Sieger = SNr
  ) AS Anzahl_Siege,
  (
    SELECT AVG(Dauer)
    FROM Wettkampf
    WHERE Wettkampf.Sieger = SNr
  ),
  (
    SELECT COUNT(DISTINCT Team.TNr)
    FROM Team, Spieler, Wettkampf
    WHERE
      Team.TNr = Spieler.TNr AND
      Wettkampf.Geschlagener = Spieler.SNr AND
      Wettkampf.Sieger = SNr
  )
FROM Spieler;
```

Lösungsvorschlag

```
SELECT
  s.SNR,
  s.Name,
  COUNT(*) AS AnzahlSiege,
  AVG(w.Dauer) AS DurchschnittlicheWettkampfdauer,
  COUNT(DISTINCT g.TNr) AS TeamsBesiegt
FROM Spieler s, Wettkampf w, Spieler g
WHERE
  s.SNr = w.Sieger AND
```

```
  g.SNr = w.Geschlagener
GROUP BY s.SNR, s.Name;
```

Tupelkalkül

(c) Verwenden Sie den relationalen Tupelkalkül, um die folgenden Anfragen zu formulieren:

(i) Finden Sie die Namen der Spieler des Teams Dream Team.

(ii) Geben Sie die Namen der Minispiele zurück, bei denen bereits Spieler gegeneinander angetreten sind, deren Teams dieselbe Teamfarbe haben.

Hinweis: Die Anfragen im relationalen Tupelkalkül dürfen auch nicht sicher sein.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2013/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex

Examensaufgabe „Mitfahrgelegenheiten“ (46116-2014-F.T2-TA2-A2)

Aufgabe 2: Relationale Algebra

Gegeben sei das folgende relationale Schema mitsamt Beispieldaten für eine Datenbank von Mitfahrgelegenheiten. Die Primärschlüssel-Attribute sind jeweils unterstrichen, Fremdschlüssel sind überstrichen.

„Kunde”:

KID	Name	Vorname	Stadt
K1	Meier	Stefan	S3
K2	Müller	Peta	S3
K3	Schmidt	Christine	S2
K4	Schulz	Michael	S4

„Stadt”

SID	SName	Bundesland
S1	Berlin	Berlin
S2	Nürn	Bayern
S3	Köln	Nordrhein-Wesffalen
S4	Stuttgart	Baden-Württemberg
S5	München	Bayer

„Angebot”:

KID	Start	Ziel	Datum	Plätze
K4	S4	S5	08.07.2011	3
K4	S5	S4	10.07.2011	3
K1	S1	S5	08.07.2011	3
K3	S2	S3	15.07.2011	1
K4	S4	S1	15.07.2011	3
K1	S5	S5	09.07.2011	2

„Anfrage”:

KID	Start	Ziel	Datum
K2	S4	S5	08.07.2011
K2	S5	S4	10.07.2011
K3	S2	S3	08.07.2011
K3	S3	S2	10.07.2011
K2	S4	S5	05.07.2011
K2	S5	S4	17.07.2011

(a) Formulieren Sie die folgenden Anfragen auf das gegebene Schema in relationaler Algebra:

- Finden Sie die Namen aller Städte in Bayern!

Lösungsvorschlag

$$\pi_{\text{SName}}(\sigma_{\text{Bundesland}=\text{Bayern}}(\text{Stadt}))$$

- Finden Sie die SIDs aller Städte, für die weder als Start noch als Ziel eine Anfrage vorliegt!

Lösungsvorschlag

$$\pi_{\text{SID}}(\text{Stadt}) - \pi_{\text{Start}}(\text{Anfage}) - \pi_{\text{Ziel}}(\text{Anfrage})$$

- Finden Sie alle IDs von Kunden, welche eine Fahrt in ihrer Heimatstadt starten.

Lösungsvorschlag

$$\pi_{\text{KID}}(\text{Kunde} \bowtie_{\text{Kunde.KID}=\text{Anfrage.KID} \wedge \text{Kunde.Stadt}=\text{Anfrage.Stadt}} \text{Anfrage}) \wedge \pi_{\text{KID}}(\text{Kunde} \bowtie_{\text{Kunde.KID}=\text{Angebot.KID} \wedge \text{Kunde.Stadt}=\text{Angebot.Stadt}} \text{Angebot})$$

- Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus!

Lösungsvorschlag

$$\pi_{\text{Datum}}((\text{Angebot} \bowtie_{\text{Start}=\text{SID} \wedge \text{SName}=\text{'München'}} \text{Stadt}) \bowtie_{\text{Ziel}=\text{SID} \wedge \text{SName}=\text{'Stuttgart'}} \text{Stadt})$$

Variante 2:

Lösungsvorschlag

$$\pi_{\text{Datum}}(\sigma_{\text{Sname}=\text{'München'} \wedge \text{Zname}=\text{'Stuttgart'}}(\rho_{\text{Zname} \leftarrow \text{Sname}, \text{SID1} \leftarrow \text{SID}}(\text{Stadt}) \bowtie_{\text{Ziel}=\text{SID1}} \text{Angebot} \bowtie_{\text{Start}=\text{SID}} \text{Stadt}))$$

(b) Geben Sie das Ergebnis (bezüglich der Beispieldaten) der folgenden Ausdrücke der relationalen Algebra als Tabellen an:

- $\pi_{\text{KID}}(\text{Angebot}) \bowtie \text{Kunde}$

Lösungsvorschlag

Zeile mit der Petra Müller fällt weg.

$\underline{\text{KID}}$	Name	Vorname	$\overline{\text{Stadt}}$
K1	Meier	Stefan	S3
K3	Schmidt	Christine	S2
K4	Schulz	Michael	S4

- $\pi_{(\text{KID,Stadt})}(\text{Kunde}) \bowtie_{\text{Kunde.Stadt}=\text{Angebot.Ziel}} \pi_{\text{Plaetze}}(\text{Angebot})$

Lösungsvorschlag

KID	Stadt	Plätze
K1	S3	1
K2	S3	1
K4	S4	1
K4	S4	2

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Computer „Chiemsee““ (46116-2015-F.T1-TA2-A1)

Relationale Algebra
SQL
GROUP BY
HAVING

Gegeben sei folgendes relationales Schema, dessen Attribute nur atomare Attributwerte besitzen.

Computer: {IP, Name, Hersteller, Modell, Standort}

(a) Geben Sie für die folgenden Anfragen einen relationalen Ausdruck an:

(i) Geben Sie die IP-Adresse des Computers mit Namen „Chiemsee“ aus.

Lösungsvorschlag

$$\pi_{\text{IP}}(\sigma_{\text{Name}=\text{Chiemsee}}(\text{Computer}))$$

(ii) Geben Sie 2er-Tupel von IP-Adressen der Computer am selben Standort aus.

Lösungsvorschlag

$$\pi_{\text{c1.IP,c2.IP}}(\sigma_{\text{c1.Standort}=\text{c2.Standort}}(\rho_{\text{c1}}(\text{Computer}) \times \rho_{\text{c2}}(\text{Computer})))$$

(b) Formulieren Sie die folgenden Anfragen in SQL:

(i) Geben Sie die IP-Adressen der Rechner am Standort „Büro2“ aus.

Lösungsvorschlag

```sql
SELECT IP FROM Computer WHERE Standort = 'Büro2';
```

(ii) Geben Sie alle Computer-Namen in aufsteigender Ordnung mit ihren IP-Adressen aus.

Lösungsvorschlag

```sql
SELECT Name, IP FROM Computer ORDER BY Name ASC;
```

(iii) Geben Sie für jeden Hersteller die Anzahl der unterschiedlichen Modelle aus.

Lösungsvorschlag

```sql
SELECT COUNT(DISTINCT Modell), Hersteller
FROM Computer
GROUP BY Hersteller;
```

(iv) Geben Sie für jeden Hersteller, welcher mindestens 2 unterschiedliche Modelle hat, die Anzahl der unterschiedlichen Modelle aus.

Lösungsvorschlag

```sql
SELECT Hersteller, COUNT(*) FROM Modelle GROUP BY Hersteller HAVING
↪  COUNT(*) > 1;
```

oder

Lösungsvorschlag

```sql
SELECT COUNT(DISTINCT Modell), Hersteller
FROM Computer
GROUP BY Hersteller
HAVING COUNT(DISTINCT Modell) >= 2;
```

```sql
-- AB 2 Einstieg Sql

-- Aufgabe 3: SQL-Anfragen auf einer Tabelle & Relationale Algebra

-- sudo mysql  < Computer.sql
-- DROP DATABASE IF EXISTS Computer;
-- CREATE DATABASE Computer;
-- USE Computer;

CREATE TABLE Computer (
    IP VARCHAR(15) PRIMARY KEY NOT NULL,
    Name VARCHAR(30),
    Hersteller VARCHAR(30),
    Modell VARCHAR(30),
    Standort VARCHAR(30)
);

INSERT INTO Computer VALUES ('10.11.12.1', 'Chiemsee', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.2', 'Computer2', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.3', 'Computer3', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.4', 'Computer4', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.5', 'Computer5', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.6', 'Computer6', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.7', 'Computer7', 'HP', 'Envy', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.8', 'Computer8', 'DELL', 'G3', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.9', 'Computer9', 'DELL', 'G7', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.10', 'Computer10', 'DELL', 'Latitude',
↪  'Büro2');
INSERT INTO Computer VALUES ('10.11.12.11', 'Computer11', 'DELL', 'Alienware',
↪  'Büro2');
INSERT INTO Computer VALUES ('10.11.12.12', 'Computer12', 'DELL', 'Inspirion',
↪  'Büro2');
INSERT INTO Computer VALUES ('10.11.12.13', 'Computer13', 'DELL', 'XPS', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.14', 'Computer14', 'Apple', 'MacBook Air',
↪  'Büro2');
INSERT INTO Computer VALUES ('10.11.12.15', 'Computer15', 'Apple', 'MacBook Air',
↪  'Büro2');
INSERT INTO Computer VALUES ('10.11.12.16', 'Computer16', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.17', 'Computer17', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.18', 'Computer18', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.19', 'Computer19', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.20', 'Computer20', 'Apple', 'MacBook Air',
↪  'Büro3');
```

```
INSERT INTO Computer VALUES ('10.11.12.21', 'Computer21', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.22', 'Computer22', 'Apple', 'MacBook Air',
↪  'Büro3');
INSERT INTO Computer VALUES ('10.11.12.23', 'Computer23', 'Apple', 'MacBook Air',
↪  'Büro3');
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Relationen R, S und T“ (46116-2018-H.T2-TA2-A2)

Relationale Algebra

Geben Sie die Ergebnisrelation folgender Ausdrücke der relationalen Algebra als Tabellen an. Begründen Sie Ihr Ergebnis, gegebenenfalls durch Zwischenschritte. Gegeben seien folgende Relationen:

R

A	B	C	D	E	F
6	8	1	7	3	7
5	3	4	4	5	7
0	6	3	0	1	7

S

A	C	X	Z
7	8	6	1
0	3	0	0
2	3	0	5
0	6	1	6
6	7	1	7
7	1	2	2
1	8	8	0
5	1	5	5
7	3	0	2
4	8	2	7

T

X	Y
5	3
0	5
8	6
3	6
5	7
2	8

(a) $\sigma_{A>6}(S) \bowtie_{S.X=T.Y} \pi_Y(T)$

Lösungsvorschlag

A	C	X	Z	Y
7	8	6	1	6

(b) $\pi_{A,C}(S) - (\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$

Lösungsvorschlag

$\sigma_{x=1}(S)$:

A	C	X	Z
0	6	1	6
6	7	1	7

$\pi_C(\sigma_{x=1}(S))$:

C
6
7

$\pi_A(R)$:

A
6
5
0

$(\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$ $\pi_{A,C}(S)$

A	C
6	6
5	6
0	6
6	7
5	7
0	7

A	C
7	8
0	3
2	3
0	6
6	7
7	1
1	8
5	1
7	3
4	8

A	C
7	8
0	3
2	3
7	1
1	8
5	1
7	3
4	8

(c) $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$

Lösungsvorschlag

$\pi_D(R) \times \pi_E(R)$ $\pi_E(R)$ $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$

A	E
7	3
4	3
0	3
7	5
4	5
0	5
7	1
4	1
0	1

E
3
5
1

D
7
4
0

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Harry Potter“ (46116-2019-H.T2-A4)

Relationale Algebra
Tupelkalkül

Gegeben ist das Datenbankenschema aus Aufgabe 3.

Übertragen Sie die folgenden Ausdrücke in die relationale Algebra. Beschreiben Sie diese Ausdrücke umgangssprachlich, bevor Sie die Ausdrücke umformen.

(a) $\{s | s \in \text{Schüler} \wedge \neg \exists t \in \text{teil\_von}(t.\text{Id} = s.\text{Id})\}$

Lösungsvorschlag

$$\text{Schüler} - (\text{Schüler} \bowtie (\pi_{Id}(\text{teil\_von})))$$

(b) $\{s | s \in \text{Schüler} \wedge \exists t \in \text{teil\_von}(r.\text{Id} = s.\text{Id}) \wedge \exists h \in \text{Haus}(f.\text{Name} = h.\text{Name}) \wedge \exists q \in \text{Quidditch}(h.\text{Name} = q.\text{Haus} \wedge q.\text{Captain} = \text{'Harry Potter'})\}$

Lösungsvorschlag

$$\sigma_{\text{Id,SName,Patronus,Haarfarbe,Aktiv,Gesamtnote}}\Bigg(\Big(\rho_{\text{Sname} \leftarrow \text{Name}}(\text{Schüler}) \bowtie \text{teil\_von}\Big) \bowtie \Big(\text{Haus} \bowtie_{\text{Haus.Name=Quidditch.Haus}} \big(\sigma_{\text{Captain='Harry Potter'}}(\text{Quidditch})\big)\Big) \Bigg)$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2019/09/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Gebrauchtwagen“ (66111-1996-H.A4)

Dritte Normalform
Relationale Algebra
SQL

Gegeben sei die folgende, in dritter Normalform vorliegende relationale Datenbank zur Modellierung des Gebrauchtwagenparks eines Autohändlers.

Die Relation Modelle beinhaltet alle Fahrzeugtypen, die der Händler im Gebrauchtwagenprogramm führt. Die Modelle sind über das Attribut „mnr“ über alle Hersteller hinweg eindeutig nummeriert. „mnr“ ist daher Primärschlüssel in der Relation Modelle. Über das Attribut „hnr“ wird von Modelle auf die Relation Hersteller verwiesen. In der Relation Fahrzeuge werden alle tatsächlich beim Händler am Lager befindlichen Fahrzeuge geführt. Über „mnr“ wird von Fahrzeuge auf Modelle verwiesen. Bei gegebener Modellnummer ist die vergebene Fahrgestellnummer („fgnr“) eindeutig. Darum bilden „mnr“ und „fgnr“ zusammen den Primärschlüssel der Relation Fahrzeuge.

- Formulieren Sie folgende Anfragen jeweils in relationaler Algebra und SQL!

(a) Bestimmen Sie alle Modelle mit mehr als 60 PS

Lösungsvorschlag

Relationale Algebra

$\pi_{mnr}(\sigma_{ps>60}(Modelle))$

SQL

```
SELECT mnr FROM Modelle WHERE ps > 60;
```

(b) Bestimmen Sie die Typen aller Modelle des Herstellers VW.

Lösungsvorschlag

Relationale Algebra

$\pi_{typ}(Modelle \bowtie \sigma_{hersteller='vw'}(Hersteller))$
oder
$\pi_{typ}(\sigma_{hersteller='vw'}(Modelle \bowtie Hersteller))$

SQL

```
SELECT m.typ
FROM Modelle m, Hersteller h
WHERE h.hnr = m.hnr AND
h.hersteller = 'VV';
```

(c) Bestimmen Sie die Nummern aller Modelle des Herstellers Opel, von denen tatsächlich Fahrzeuge auf Lager sind

Lösungsvorschlag

Relationale Algebra

$$\pi_{mnr}(Fahrzeuge \bowtie (Modelle \bowtie \sigma_{hersteller='Opel'}(Hersteller)))$$

GROUP BY

SQL

```
SELECT DISTINCT m.mnr
FROM Modelle m, Hersteller h, Fahrzeuge f
WHERE f.mnr = m.mnr AND m.hnr = h.hnr AND h.hersteller = 'Opel';
```

- Formulieren Sie folgende Anfrage nur in SQL! Bestimmen Sie die Namen der Hersteller, für deren sämtliche Modelle mindestens ein Fahrzeug im aktuellern Bestand vorhanden ist.

- Formulieren Sie folgende SQL-Anfrage umgangssprachlich, aber exakt!

```
SELECT AVG(neupreis), hnr
FROM Modelle
GROUP BY hnr;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66111/1996/09/Aufgabe-4.tex

Examensaufgabe „Tupelkalkül bei Dozenten-Datenbank“ (66116-2018-F.T2-TA1-A4) Tupelkalkül

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Universität entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen. Die Relation *Dozent* enthält allgemeine Daten zu den Dozentinnen und Dozenten. Dozentinnen und Dozenten halten Vorlesungen, die in der Relation *Vorlesung* abgespeichert sind. Wir gehen davon aus, dass es zu jeder Vorlesung genau einen Dozenten (und nicht mehrere) gibt. Zusätzlich wird in der Relation *Vorlesung* das *Datum* gespeichert, an dem die Klausur stattfindet. In der Relation *Student* werden die Daten der teilnehmenden Studierenden verwaltet, während die Relation *besucht* Auskunft darüber gibt, welche Vorlesung von welchen Studierenden besucht wird.

```
Dozent (DNR, DVorname, DNachname, DTitel)
Vorlesung (VNR, VTitel, Klausurtermin, Dozent)
Student (Matrikelnummer, SVorname, SNachname, Semesterzahl)
besucht (Student, Vorlesung)
```

Formulieren Sie die folgenden Anfragen im Tupelkalkül. Datumsvergleiche können Sie mit $>$, $\geq$, $<$, $\leq$ oder $=$ angeben:

(a) Geben Sie die Vornamen aller Studierenden aus, die die Vorlesung *„Datenbanksysteme“* besuchen oder besucht haben.

Lösungsvorschlag

$$\{s.SVorname | s \in Student \wedge \forall v \in Vorlesung(v.VTitel = \text{'Datenbanksysteme'} \Rightarrow \exists b \in besucht(b.Vorlesung = v.VNR \wedge b.Student = s.Matrikelnummer))\}$$

oder

Lösungsvorschlag

$$\{\text{s.SVorname} | \text{s} \in \text{Student} \wedge \text{s.Matrikelnummer} = \text{b.Student} \wedge \text{b} \in \text{besucht} \wedge \text{b.Vorlesung} = \text{v.NVR} \wedge \text{v.VNR} \in \text{Vorlesung} \wedge \text{v.VTitel} = \text{'Datenbanksysteme'}\}$$

(b) Geben Sie die Matrikelnummern der Studierenden an, die keine Vorlesung mit einem Klausurtermin nach dem 31.12.2017 besuchen oder besucht haben.

Lösungsvorschlag

$$\{\text{s.Matrikelnummer} \mid \text{s} \in \text{Student} \wedge \forall \text{v} \in \text{Vorlesung}(\text{v.Klausurtermin} > \text{‘31.12.2017’} \Rightarrow \text{b} \in \text{besucht}(\text{b.Vorlesung} = \text{v.VNR} \wedge \text{b.Student} = \text{s.Matrikelnummer}))\}$$

(c) Geben Sie die Matrikelnummern der Studierenden aus, die alle Vorlesungen von Prof. Dr. Schulz hören oder gehört haben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/03/Thema-2/Teilaufgabe-1/Aufgabe-4.tex`

Examensaufgabe „Medikamente“ (66116-2019-F.T1-TA1-A2)

SQL

Gegeben sei der folgende Ausschnitt des Schemas für die Verwaltung der Einnahme von Medikamenten:

```
Person : {[  ID : INTEGER, Name : VARCHAR(255), Wohnort : VARCHAR(255)  ]}

Hersteller : {[  ID : INTEGER, Name : VARCHAR(255), Standort : VARCHAR(255), AnzahlMitarbeiter

: INTEGER  ]}

Medikament : {[  ID : INTEGER Name : VARCHAR(255), Kosten : INTEGER, Wirkstoff : VARCHAR(255),

produziert_von : INTEGER  ]}

nimmt : {[  Person : INTEGER, Medikament : INTEGER, von : DATE, bis : DATE  ]}

hat_Unverträglichkeit_gegen : {[  Person : INTEGER, Medikament : INTEGER  ]}
```

```sql
CREATE TABLE Person (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255)
);

CREATE TABLE Hersteller (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Standort VARCHAR(255),
  AnzahlMitarbeiter INTEGER
);

CREATE TABLE Medikament (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Kosten INTEGER,
  Wirkstoff VARCHAR(255),
  produziert_von INTEGER REFERENCES Hersteller(ID)
);

CREATE TABLE nimmt (
  Person INTEGER,
  Medikament INTEGER,
  von DATE,
  bis DATE,
  PRIMARY KEY (Person, Medikament, von, bis)
);

CREATE TABLE hat_Unverträglichkeit_gegen (
  Person INTEGER REFERENCES Person(ID),
  Medikament INTEGER REFERENCES Medikament(ID)
);

INSERT INTO Person VALUES
```

```
  (1, 'Walter Müller', 'Holzapfelkreuth'),
  (2, 'Dilara Yildiz', 'Fürth');

INSERT INTO Hersteller VALUES
  (1, 'Hexal', 'Holzkirchen', 3700),
  (2, 'Ratiopharm', 'Ulm', 563);

INSERT INTO Medikament VALUES
  (1, 'IbuHexal', 3, 'Ibuprofen', 1),
  (2, 'Ratio-Paracetamol', 2, 'Paracetamol', 2),
  (3, 'BudeHexal', 4, 'Budesonid', 1),
  (4, 'Ratio-Budesonid', 5, 'Budesonid', 2);

INSERT INTO nimmt VALUES
  (1, 1, '2021-07-12', '2021-07-23'),
  (1, 3, '2021-07-12', '2021-07-23'),
  (2, 4, '2021-02-13', '2021-03-24');

INSERT INTO hat_Unverträglichkeit_gegen VALUES
  (1, 1),
  (1, 3),
  (2, 2);
```

Die Tabelle `Person` beschreibt Personen über eine eindeutige ID, deren Namen und Wohnort. Die Tabelle `Medikament` enthält Informationen über Medikamente, unter anderem deren Namen, Kosten, Wirkstoffe und einen Verweis auf den Hersteller dieses Medikaments. Die Tabelle `Hersteller` verwaltet verschiedene Hersteller von Medikamenten. Die Tabelle `hat_Unverträglichkeit_gegen` speichert die IDs von Personen zusammen mit den IDs von Medikamenten, gegen die diese Person eine Unverträglichkeit hat. Die Tabelle `nimmt` hingegen verwaltet die Einnahme der verschiedenen Medikamente und speichert zudem in welchem Zeitraum eine Person welches Medikament genommen hat bzw. nimmt.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

(a) Schreiben Sie SQL-Anweisungen, die für die bereits existierende Tabelle `nimmt` alle benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalten `von` und `bis` Teil des Primärschlüssels sind.

Lösungsvorschlag

```
ALTER TABLE nimmt
ADD CONSTRAINT FK_Person
  FOREIGN KEY (Person) REFERENCES Person(ID),
ADD CONSTRAINT FK_Medikament
  FOREIGN KEY (Medikament) REFERENCES Medikament(ID);
```

(b) Schreiben Sie eine SQL-Anweisung, welche sowohl den `Namen` als auch die `ID` von `Personen` und `Medikamenten` ausgibt, bei denen die Person das jeweilige Medikament nimmt.

Lösungsvorschlag

```
SELECT DISTINCT
  p.ID as Personen_ID,
  p.Name as Personen_Name,
  m.ID as Medikamenten_ID,
  m.Name as Medikamenten_Name
FROM Person p, Medikament m, nimmt n
WHERE
  n.Person = p.ID AND
  n.Medikament = m.ID;
```

(c) Schreiben Sie eine SQL-Anweisung, welche die ID und den Namen der Medikamente mit den niedrigsten Kosten je Hersteller bestimmt.

Lösungsvorschlag

```
SELECT m.ID, m.Name
FROM Medikament m, Medikament n
WHERE
  m.produziert_von = n.produziert_von AND
  m.Kosten >= n.Kosten
GROUP BY m.ID, m.Name
HAVING COUNT(*) <= 1;
```

(d) Schreiben Sie eine SQL-Anweisung, die die Anzahl aller Personen ermittelt, die ein Medikament genommen haben, gegen welches sie eine Unverträglichkeit entwickelt haben.

Lösungsvorschlag

```
SELECT COUNT(DISTINCT p.ID)
FROM
  Person p,
  nimmt n,
  hat_Unverträglichkeit_gegen u
WHERE
  p.ID = n.Person AND
  u.Medikament = n.Medikament;
```

(e) Schreiben Sie eine SQL-Anweisung, die die ID und den Namen derjenigen Personen ermittelt, die weder ein Medikament mit dem Wirkstoff Paracetamol noch ein Medikament mit dem Wirkstoff Ibuprofen genommen haben.

Lösungsvorschlag

```
SELECT ID, Name FROM Person
EXCEPT
SELECT p.ID, p.Name
FROM Person p, nimmt n, Medikament m
WHERE
  p.ID = n.Person AND
  n.Medikament = m.ID AND
  m.Wirkstoff IN ('Ibuprofen', 'Paracetamol');
```

(f) Schreiben Sie eine SQL-Anweisung, welche die Herstellernamen und die Anzahl der bekannten Unverträglichkeiten gegen Medikamente dieses Herstellers ermittelt. Das Ergebnis soll aufsteigend nach der Anzahl der bekannten Unverträglichkeiten sortiert werden. Relationale Algebra

Lösungsvorschlag

```sql
SELECT p.Name, (
SELECT COUNT(h.ID)
  FROM Hersteller h, Medikament m, hat_Unverträglichkeit_gegen u
  WHERE
    m.produziert_von = h.ID AND
    u.Medikament = m.ID AND
    h.ID = p.ID
) AS Unverträglichkeiten
FROM Hersteller p
ORDER BY Unverträglichkeiten ASC;
```

(g) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Wohnorte aller Personen bestimmt, welche ein Medikament mit dem Wirkstoff Paracetamol nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

Lösungsvorschlag

$$\pi_{\text{Person.Wohnort}}\left(\sigma_{\text{Medikament.Wirkstoff}='\text{Paracetamol}'}(\text{Person} \bowtie_{\text{Person.ID}=\text{nimmt.Person}} \text{nimmt} \bowtie_{\text{nimmt.Medi}}\right.$$

(h) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Namen aller Personen bestimmt, die von allen bekannten Herstellern, deren Standort München ist, Medikamente nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

Lösungsvorschlag

Lösungsvorschlag

$$\pi_{\text{Person.Name}}\left(\sigma_{\text{Hersteller.Standort}='\text{München}'}((\text{Person} \bowtie_{\text{Person.ID}=\text{nimmt.Person}} \text{nimmt}) \bowtie_{\text{nimmt.Medikam}}\right.$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Relation X und Y“ (66116-2019-F.T2-TA1-A3)

Relationale Algebra
RelaX - relational algebra calculator

Gegeben seien folgende Relationen:

X

A	B	C
1	2	3
2	3	4
4	3	3
1	2	2
2	3	2
3	3	2
3	1	2
2	2	1
1	1	1

Y

B	C	D
2	3	1
1	1	3
2	1	3
3	2	3
2	2	3
1	3	2

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an; machen Sie Ihren Rechenweg kenntlich.

(a) $\sigma_{A=2}(X) \bowtie Y$

Lösungsvorschlag

A	B	C	D	E
2	3	2	3	1
2	3	2	1	3
2	3	2	2	3
2	2	1	1	3
2	2	1	3	2

(b) $(\pi_{B,C}(X) - \pi_{B,C}(Y)) \bowtie X$

Lösungsvorschlag

(c)

A
1
2
3

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Relationen „Professor“ und „Vorlesung““ (66116-2019-H.T2-TA2-A5)

Gegeben seien die beiden Relationen Professor und Vorlesung mit folgendem Umfang: Relation Professor: 5 Spalten, 164 Datensätze Relation Vorlesung: 10 Spalten, 333 Datensätze.

Optimieren Sie auf geeignete Weise folgende SQL-Anweisung möglichst gut und berechnen Sie wie stark sich die Datenmenge durch jede Optimierung reduziert (nehmen Sie für Ihre Berechnung an, dass Herr Mustermann genau zwei Vorlesungen hält). Geben Sie jeweils den Operatorbaum vor und nach Ihren jeweiligen Optimierungen an.

SELECT Titel FROM Professor, Vorlesung WHERE Name = Mustermann’ AND PersNr = gelesenVon;

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-2/Aufgabe-5.tex`

Examensaufgabe „Universitätsschema“ (66116-2020-F.T1-TA2-A3)

Relationale Algebra
Tupelkalkül

Gegeben sei ein Universitätsschema.

(a) Finden Sie alle Studierenden, die keine Vorlesung hören. Formulieren Sie die Anfrage im Tupelkalkül.

Lösungsvorschlag

$$\{s \in \text{Studierende} \wedge h \in \text{hören} \mid \neg\exists \text{s.MatrNr} = \text{h.MatrNr}\}$$

(b) Geben Sie einen Ausdruck an, der die Relation $\neg$`hören` erzeugt. Diese enthält für jeden Studierenden und jede Vorlesung, die der Studierende **nicht** hört, einen Eintrag mit Matrikelnummer und Vorlesungsnummer. Formulieren Sie die Anfrage in **relationaler Algebra**.

Lösungsvorschlag

$$\rho_{\neg\text{hören}}\Big(\big(\pi_{\text{MatrNr}}(\text{Studierende}) \times \pi_{\text{VorlNr}}(\text{Vorlesungen})\big) - \text{hören}\Big)$$

(c) Finden Sie alle Studierenden, die **keine** Vorlesung hören. Formulieren Sie die Anfrage in **relationaler Algebra**.

Lösungsvorschlag

$$\pi_{\text{MatrNr}}(\text{Studierende}) - \pi_{\text{MatrNr}}(\text{hören})$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Relationale Algebra und Optimierung“ (66116-2020-H.T2-TA2-A3)

SQL mit Übungsdatenbank
Relationale Algebra

```sql
CREATE TABLE V (
  Name VARCHAR(1),
  Jahr integer
);

CREATE TABLE S (
  Jahr integer
);

INSERT INTO V VALUES
  ('A', 2019),
  ('A', 2020),
  ('B', 2018),
  ('B', 2019),
  ('B', 2020),
  ('C', 2017),
  ('C', 2018),
  ('C', 2020);

INSERT INTO S VALUES
  (2018),
  (2019),
  (2020);
```

(a) Betrachten Sie die Relation V. Sie enthält eine Spalte *Name* sowie ein dazugehörendes Jahr.

Name	Jahr
A	2019
A	2020
B	2018
B	2019
B	2020
C	2017
C	2018
C	2020

(i) Gesucht ist eine Relation S, die das folgende Ergebnis von $V \div S$ berechnet ($\div$ ist die Division der relationalen Algebra):
$V \div S$

Name
B

Welche der nachstehenden Ausprägungen für die Relation liefert das gewünschte Ergebnis? Geben Sie eine Begründung an.

i.

Jahr
2017
2018
2019
2020

ii.

Jahr
2018
2019
2020

iii.

Jahr
2017
2019
2020

iv. ii.,

Lösungsvorschlag

iv) also weder i., noch ii., noch iii.

i.

Name

ii.

Name
C

iii.

Name
B
C

(ii) Formulieren Sie die Divisions-Query aus Teilaufgabe i. in SQL.

```
SELECT DISTINCT v1.Name FROM V as v1
WHERE NOT EXISTS (
  (SELECT s.Jahr FROM S as s)
  EXCEPT
  (SELECT v2.Jahr FROM V as v2 WHERE v2.Name = v1.Name)
);
```

1

[1] https://www.geeksforgeeks.org/sql-division/

(b) Gegeben sind die Tabellen R(A, B) und S(C, D) sowie die folgende View:

1 CREATE VIEW mv (A,C,D) AS

, SELECT DISTINCTA,C,D

» FROMR,S

« WHEREB=DANDA <> 10;

Auf dieser View wird die folgende Query ausgeführt:

, SELECT DISTINCT A , FROM mv ;» WHEREC>D:

Konvertieren Sie die Query und die zugrundeliegenden View in einen Ausdruck der relationalen Algebra in Form eines Operatorbaums. Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und begründen Sie dabei kurz jeden durchgeführten Schritt.

(c) Gegeben sind die Relationen R, S und U sowie deren Kardinalitäten Tr, Ts und Tr:

R (al, a2, a3) Tr = 200 S (al, a2, a3) Ts = 100 U (ul, u2) Iv = 50

Bei der Ausführung des folgenden Query-Plans wurden die Kardinalitäten der Zwischenergebnisse mitgezählt und an den Kanten notiert.

Leiten Sie aus den Angaben im Ausführungsplan den Anteil der qualifizierten Tupel aller Prädikate her und geben Sie diese an.

Tx s0| N Ral > Vu

N R.a3 = S.a3 U N OR.al > 100 OS.al < 10

R 5

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Relationen R1 und R2“ (66116-2021-F.T1-TA2-A3)

Relationale Algebra

(a) Gegeben seien die folgenden beiden Relationen R1 und R2:

R1

P	Q	S
10	einfach	5
15	b	8
13	einfach	6

R2

A	B	C
10	b	6
13	c	3
10	b	5

Geben Sie das Ergebnis des folgenden relationalen Ausdrucks an:

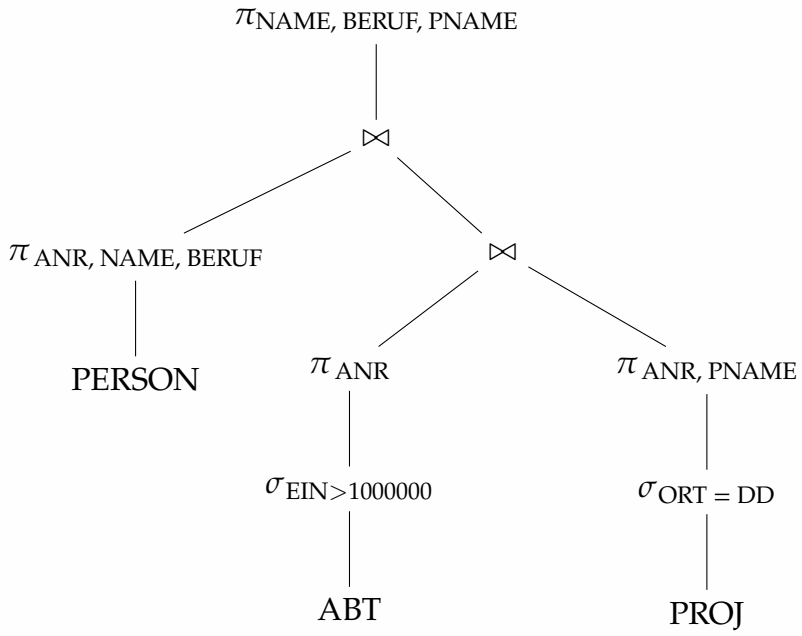
$$R_1 \bowtie_{R_1.P=R_2.A} R_2 \text{ (Equi-Join)}$$

Lösungsvorschlag

P	Q	S	A	B	C
10	einfach	5	10	b	6
10	einfach	5	10	b	5
13	einfach	6	13	c	3

(b) Zeichnen Sie den Operatorbaum zu folgender Abfrage in relationaler Algebra:

Lösungsvorschlag



(c) Ist der linke (bzw. rechte) Verbundoperator (Left- bzw. Right-Outer Join) assoziativ? Falls ja, beweisen Sie die Aussage, falls nein, geben Sie ein Gegenbeispiel an.

Lösungsvorschlag

Nein. Beleg durch Gegenbeispiel:

R1

A	B
1	2
2	15

R2

A	C
1	35
2	12
13	5

R3

B	C
2	35
100	35

(R1 LEFT OUTER JOIN R2) LEFT OUTER JOIN R3

R1.A	R1.B	R2.A	R2.C	R3.B	R3.C
1	2	1	35	2	35
2	15	2	12	NULL	NULL

R1 LEFT OUTER JOIN (R2 LEFT OUTER JOIN R3)

R1.A	R1.B	R2.A	R2.C	R3.B	R3.C
1	2	1	35	2	35
2	15	NULL	NULL	NULL	NULL

(Nur wenn beide Tabellen leer sind, wären auch die Outer Joins assoziativ).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Autoverleih“ (66116-2021-F.T2-TA2-A3)

Relationenmodell

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema (in dritter Normalform, 3. NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation: Primärschlüssel werden durch Unterstreichen gekennzeichnet, Fremdschlüssel durch die Nennung der Relation, auf die sie verweisen, in eckigen Klammern hinter dem Fremdschlüsselattribut. Attribute zusammengesetzter Fremdschlüssel werden durch runde Klammern als zusammengehörig markiert. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als UNIQUE oder NOT NULL ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2, Fremdschlüsselattribut1[Relation1], (Fremdschlüssel2
Attribut1, Fremdschlüsssel2 Attribut2) [Relation2]); Attribut1 UNIQUE Attribut2 NOT NULL
```

> **Exkurs: Enhanced entity-relationship model**
> https://en.wikipedia.org/wiki/Enhanced_entity-relationship_model https://www.tutorialride.com/dbms/enhanced-entity-relationship-model-eer-model.htm

Lösungsvorschlag

```
Automarke : {[ AName, Name[Firma] ]}

Fahrzeug : {[ Kennzeichen, Mieter-Nr[Mieter], Name[Firma], Modell, Farbe ]}

Firma : {[ Name, Umsatz, Mieter-Nr ]}

Hersteller : {[ Name[Firma] ]}

Mieter : {[ Mieter-Nr ]}

Person : {[ Handynummer, Vorname, Nachname, Mieter-Nr[Mieter] ]}
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex

Examensaufgabe „Mitarbeiter einer Abteilung“ (66116-2021-F.T2-TA2-A5)

RelaX - relational algebra calculator
Relationale Algebra

Formulieren Sie basierend auf den in der letzten Aufgabe gegebenen Relationen die geforderten Anfragen in der Relationenalgebra.

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter]
NOT NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)
```

(a) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in der Buchhaltung arbeiten.

Lösungsvorschlag

$$\pi_{\text{Vorname, Nachname}}\Big(\sigma_{\text{Bezeichnung = 'Buchhaltung'}}\big(\text{Mitarbeiter} \bowtie_{\text{Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID}} \text{Abteilung}\big)\Big)$$

(b) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in keiner Abteilung arbeiten.

Lösungsvorschlag

$$\pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter}) - \pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter} \bowtie \text{Abteilung})$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-5.tex

Übungsaufgabe „Bus-Unternehmen“ (Relationenmodell)

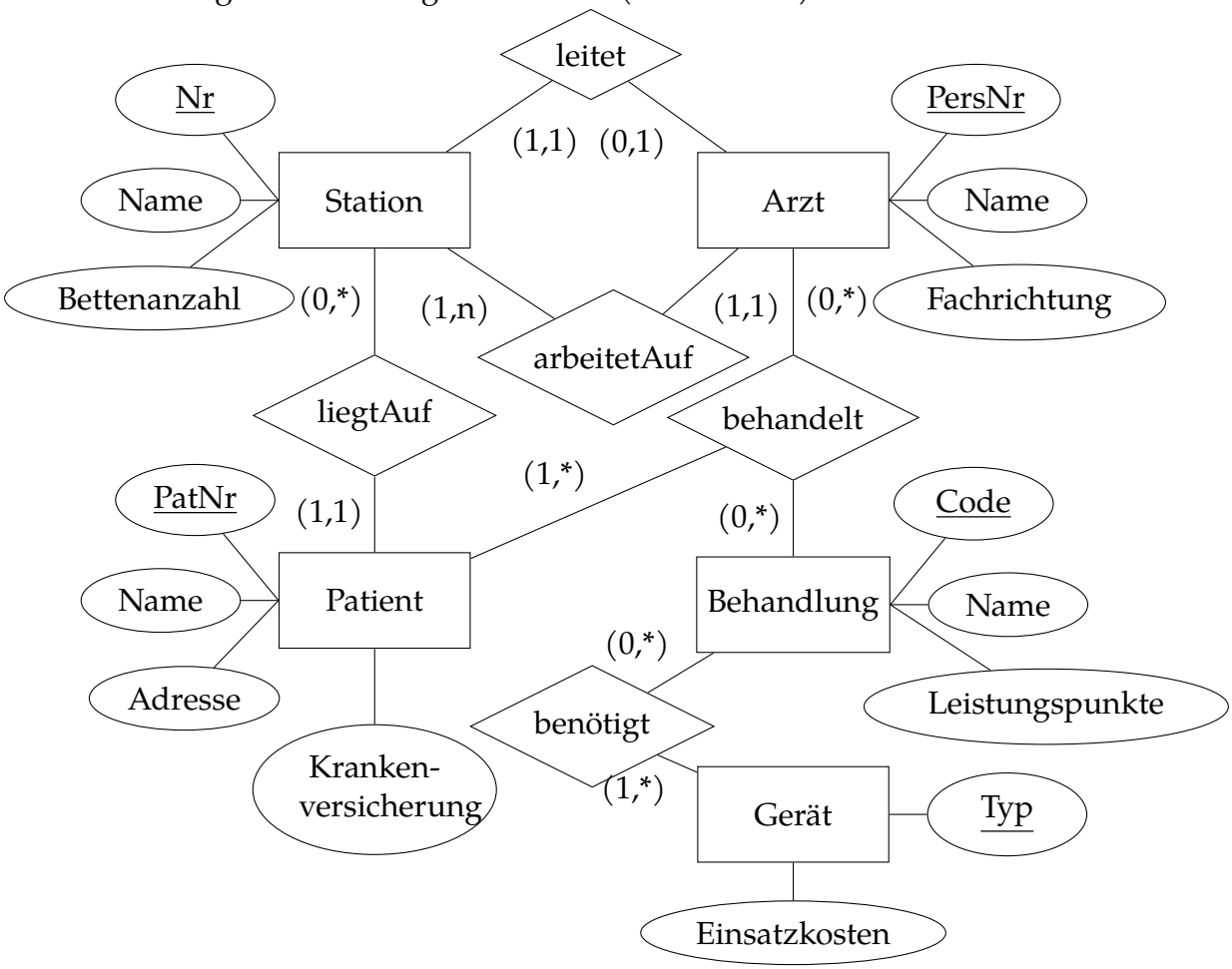
Relationenmodell

Konvertieren Sie das folgende ER-Modell in ein relationales DB-Schema. Hinweis: Die Angabe der Domänen ist nicht notwendig!

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/10_Relationenmodell/Aufgabe_Bus-Unternehmen.tex`

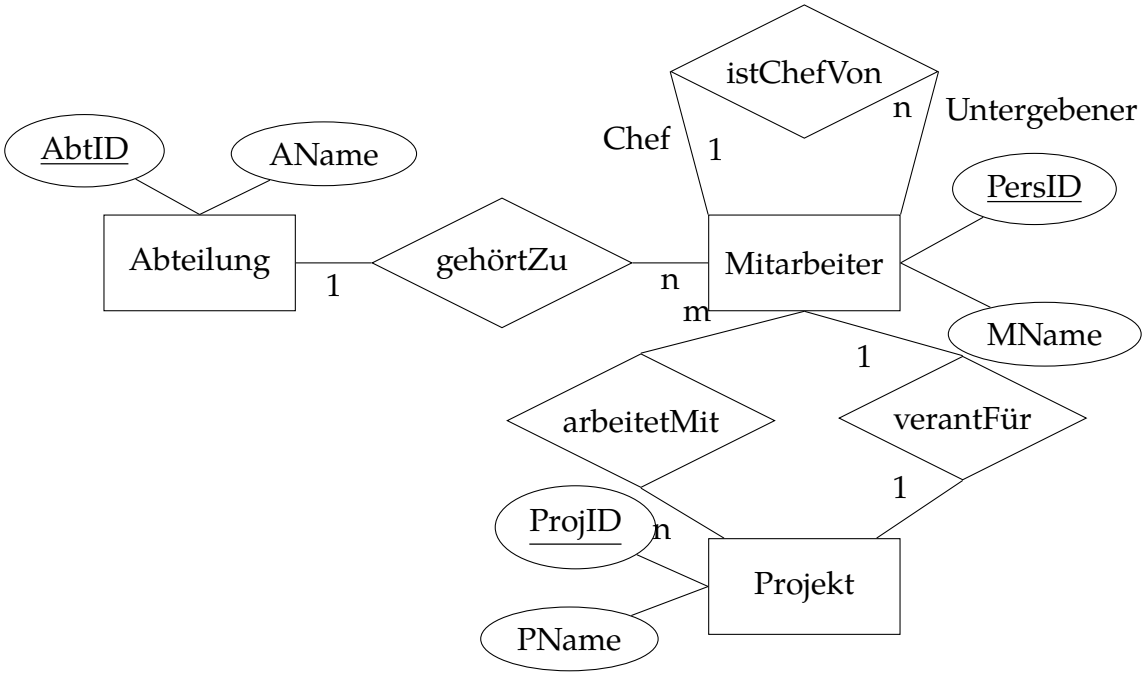
Übungsaufgabe „Krankenhaus“ (Verfeinertes Relationenmodell)

Überführen Sie folgendes ER-Diagramm in ein (verfeinertes) Relationenschema!



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/10_Relationenmodell/Aufgabe_Krankenhaus.tex`

Übungsaufgabe „Mitarbeiter-Projekte einer Abteilung“ (Relationenmodell, Verfeinertes Relationenmodell)



(a) Übertragen Sie das gegebene ER-Modell in ein relationales Schema! Geben Sie in geeigneter Weise Schlüssel an.

Lösungsvorschlag

```
Abteilung : {[ AbtID, AName ]}

Mitarbeiter : {[ PersID, MName ]}

Projekt : {[ ProjID, PName ]}

gehörtZu : {[ PersID, AbtID ]}

arbeitetMit : {[ PersID, ProjID ]}

istChefvon : {[ PersID, VorID ]}

verantfür : {[ PersID, ProjID ]}
```

(b) Verfeinern Sie das Relationenschema!

Lösungsvorschlag

```
Abteilung : {[ AbtID, AName ]}

Mitarbeiter : {[ PersID, MName, AbtID, VerantwortlicherID ]}
```

→ `gehörtZu` und `istChefVon` wurden berücksichtigt

`Projekt : {[ ProjID, PName, VerantwortlicherID ]}`

→ zur Vermeidung von NULL-Werten wurde hier `verantFür` berücksichtigt

`arbeitetMit : {[ PersID, ProjID ]}`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/10_Relationenmodell/Aufgabe_Mitarbeiter-Projekte.tex`

Übungsaufgabe „Süße Produktion“ (Relationenmodell, Kartesisches Produkt)

Relationenmodell
Kartesisches Produkt

Vorgegeben sind die Domänen Abteilung = Verwaltung, Lager, Produktion und Mitarbeiter = Fent, Süß, Dobler.

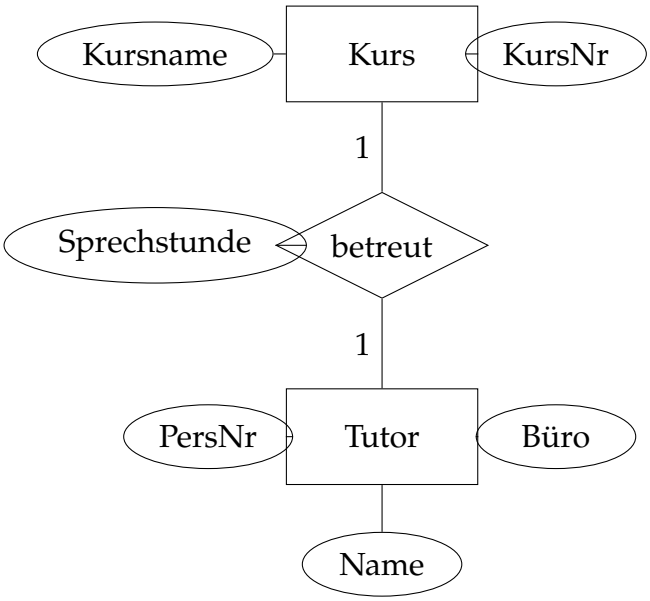
(a) Bestimmen Sie Mitarbeiter x Abteilung! Welche Aussage liefert dieses kartesische Produkt?

(b) Geben Sie – orientiert an Aufgabe a) - eine mögliche Interpretation der Menge (Fent,Produktion), (Süß, Lager) an.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/10_Relationenmodell/Aufgabe_Suesse-Produktion.tex`

Übungsaufgabe „Tutor“ (Relationenmodell)

Relationenmodell



Beim obenstehenden ER-Modell gilt:

- Ein Tutor kann durch seine Personalnummer oder durch die Kombination aus Name und Büro(-nummer) identifiziert werden.
- Bei den Kursen ist sowohl Kursname als auch Kursnummer eindeutig.

(a) Überführen Sie die Entity-Typen Kurs und Tutor in entsprechende Relationen. Legen Sie für jede der Relationen einen Primärschlüssel fest.

Lösungsvorschlag

```
Kurs : {[ KursNr, Kursname ]}

Tutor : {[ PersNr, Name, Buero ]}
```

bzw.

```
Kurs : {[ KursNr, Kursname ]}

Tutor : {[ PersNr, Name, Buero ]}
```

(b) Für die Konvertierung des Relationship-Typen betreut gibt es – unabhängig von den gewählten Primärschlüsseln in Aufgabe a – mehrere Möglichkeiten. Geben Sie alle möglichen Relationen (mit Festlegung des Primärschlüssels) an.

Lösungsvorschlag

Bei der Konvertierung des Relationship-Typ betreut enthält die entsprechende Relation jeweils einen Schlüsselkandidaten der Relationen Kurs und Tutor. Aufgrund der Aufgabenstellung besitzen beide Relationen zwei Schlüssel-

kandidaten

- Kurs: KursNr und Kursname
- Tutor: PersNr und Name, Büro

Da es sich um eine 1:1-Beziehung handelt, sind die ausgewählten Schlüsselkandidaten gleichzeitig Schlüsselkandidaten von betreut. Einer der beiden wird als Primärschlüssel ausgewählt. Es gibt folglich acht mögliche Relationen:

```
betreut : {[ PersNr, KursNr, Sprechstunde ]}

betreut : {[ PersNr, KursNr, Sprechstunde ]}

betreut : {[ PersNr, Kursname, Sprechstunde ]}

betreut : {[ PersNr, Kursname, Sprechstunde ]}

betreut : {[ Name, Büro, KursNr, Sprechstunde ]}

betreut : {[ Name, Büro, KursNr, Sprechstunde ]}

betreut : {[ Name, Büro, Kursname, Sprechstunde ]}

betreut : {[ Name, Büro, Kursname, Sprechstunde ]}
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/10_Relationenmodell/Aufgabe_Tutor.tex

Übungsaufgabe „Division“ (Relationale Algebra, Division)

Gegeben sind zwei Relationen, repräsentiert als Tabellen. Bestimmen Sie die Ergebnisrelation $R_1 \div R_2$!

R_1

A	B	C	D	E
a	r	4	3	t
c	r	2	3	t
b	w	d	4	s
a	b	r	3	t
b	r	d	3	t
a	b	w	4	s
a	w	4	4	s
b	k	d	2	s
c	w	3	4	s

R_2

B	D	E
r	3	t
w	4	s

Lösungsvorschlag

R_1

A	B	C	D	E
a	r	4	3	t
c	r	2!	3	t
b	w	d	4	s
a	b	r	3	t
b	r	d	3	t
a	b	w	4	s
a	w	4	4	s
b	k	d	2	s
c	w	3!	4	s

R_2

B	D	E
r	3	t
w	4	s

$R_1 \div R_2$

A	**C**
a	4
b	d

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Division.tex`

Übungsaufgabe „Freizeitcenter“ (Relationale Algebra)

Gegeben ist das Datenbankschema „Freizeitcenter“ mit folgender Ausprägung:

Hinweise:

- Die Courts werden immer für eine Stunde gebucht. Gespeichert ist der Buchungsbeginn.
- Die Tabelle „Buchung“ enthält die Daten eines Tages.
- Angabe der Attributwerte von Betrag in Euro
- Court 1-10: Squash, Court 11-20: Badminton, Court 21-30: Tischtennis

(a) Interpretieren Sie folgende Terme in natürlicher Sprache und geben Sie die Ergebnisrelation an!

(i) $\pi_{\text{Name}}(\sigma_{\text{Beruf}=\text{'Student'}}(\text{Spieler}))$

Lösungsvorschlag

Es sollen die Nachnamen aller Studenten ausgegeben werden, die im Freizeitcenter registriert sind.

Name
Klein

(ii) $\pi_{\text{Beruf,von,bis,Betrag}}(\sigma_{\text{Beruf}=\text{'Schüler'}}(\text{Spieler}) \bowtie \text{Preis})$

Lösungsvorschlag

Der gegebene Term gibt nichts aus, da die Relation *Spieler* und *Preis* kein gemeinsames Attribut haben. Es kann kein Naturaljoin statt finden.

Beruf	**von**	**bis**	**Betrag**

Müsste es nicht so heißen?

$\pi_{\text{Beruf,von,bis,Betrag}}(\sigma_{\text{Beruf}=\text{'Schüler'}}(\textbf{Preisstufe}) \bowtie \text{Preis})$

Beruf	**von**	**bis**	**Betrag**
Schüler	07:00	12:00	10
Schüler	12:00	17:00	15
Schüler	17:00	22:00	20

(iii) $\pi_{\text{Name,Vorname}}(\sigma_{\text{Betrag}\geq 10 \wedge \text{Betrag}\leq 20}(\text{Preis}) \bowtie \text{Preisstufe} \bowtie \text{Spieler})$

Lösungsvorschlag

Es werden der Nachname und der Vorname von allen Mitglieder des Freizeitcenters eingezeigt, die der Preisstufe 1 und 2 angehören und deshalb nicht mehr als 20 Euro zahlen müssen.

Name	**Vorname**
Klein	Mathias
Müller	Inge
Deckard	Klara
Beutlin	Hein

(iv)

$$\pi_{\text{Name,Buchung,Zeit}}(\sigma_{\text{Typ='Tischtennis'}}(\text{Court}) \bowtie_{\text{Court.ID=Buchung.Court-ID}} \text{Buchung} \bowtie_{\text{Buchung.Spieler=Spieler.Spieler-ID}} \text{Spieler})$$

Lösungsvorschlag

Es wird der Name der/des SpielerIn, die Buchung (Court-ID) und die Zeit von allen Tischtennis-Buchungen ausgegeben.

Zwischenschritt:

$\sigma_{\text{Typ='Tischtennis'}}(\text{Court}) \bowtie_{\text{Court.ID=Buchung.Court-ID}} \text{Buchung}$

Court-ID	**Zeit**	**Spieler**	**Typ**
21	16:00	5	Tischtennis
24	12:00	1	Tischtennis

Ergebnis-Relation:

Name	**Buchung**	**Zeit**
Beutlin	NULL	16:00
Klein	NULL	12:00

Müsste es nicht so heißen?

$$\pi_{\text{Name},\textbf{Court-ID},\text{Zeit}}\Big(\sigma_{\text{Typ}=\text{'Tischtennis'}}(\text{Court}) \bowtie_{\text{Court.ID}=\text{Buchung.Court-ID}} \text{Buchung} \bowtie_{\text{Buchung.Spieler}=\text{Spieler.Spieler-ID}} \text{Spieler}\Big)$$

Name	**Court-ID**	**Zeit**
Beutlin	21	16:00
Klein	24	12:00

(b) Formulieren Sie folgende Anfragen in relationaler Algebra!

(i) Gesucht sind die Spieler-IDs der Personen, die einen Squash-Court gebucht haben.

Lösungsvorschlag

$$\pi_{\text{Spieler}}(\sigma_{\text{Typ}=\text{'Squash'}}(\text{Court}) \bowtie_{\text{Court.ID}=\text{Buchung.Court-ID}} \text{Buchung})$$

(ii) In welche Preisstufe fällt Frau Tyrell?

Lösungsvorschlag

$$\pi_{\text{PS}}(\sigma_{\text{Name}=\text{'Tyrell'}}(\text{Spieler}) \bowtie \text{Preisstufe})$$

(iii) Gesucht sind die Nummern der Courts, die nicht benutzt werden.

Lösungsvorschlag

$$\pi_{\text{ID}}(\text{Court}) - \pi_{\text{Court-ID}}(\text{Buchung})$$

(iv) Welche Berufe üben die Personen aus, die zwischen 9 und 12 Uhr einen Court gebucht haben?

Lösungsvorschlag

$$\pi_{\text{Beruf}}(\sigma_{\text{Zeit}\geq 9 \wedge \text{Zeit}\leq 11}(\text{Buchung}) \bowtie_{\text{Buchung.Spieler}=\text{Spieler.Spieler-ID}} \text{Spieler})$$

(v) Gesucht sind Name und Vorname der Spieler, die für mehr als eine Stunde gebucht haben.

Lösungsvorschlag

$$\pi_{\text{Name, Vorname}}\Big(\text{Buchung} \bowtie_{\text{left.Spieler}=\text{right.Spieler} \wedge \text{left.Zeit} \neq \text{right.Zeit}} \text{Buchung} \bowtie_{\text{Buchung.Spieler}=\text{Spieler.Spieler-ID}} \text{Spieler}\Big)$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Freizeitcenter.tex`

Übungsaufgabe „Universität“ (Relationale Algebra)


```
Studenten {[MatrNr:integer, Name:string, Semester:integer]}
Vorlesungen {[VorlNr:integer, Titel:string, SWS:integer, gelesenVon:integer] }
Professoren {[PersNr:integer, Name:string, Rang:string, Raum:integer] }
hoeren {[MatrNr:integer, VorlNr:integer]}
voraussetzen {[VorgaengerVorlNr:integer, NachfolgerVorlNr:integer]}
pruefen {[MatrNr:integer, VorlNr:integer, PrueferPersNr:integer, Note:decimal]}
```

(a) Geben Sie verbal an, welches Ergebnis folgende SQL-Anfrage liefert:

Lösungsvorschlag

Liste mit zwei unterschiedliche Studenten, die in derselben Vorlesung waren.

(b) Geben Sie einen Relationenalgebra-Ausdruck für diese Anfrage an. Dieser Ausdruck sollte keine Kreuzprodukte (nur Joins) enthalten.

Lösungsvorschlag

$$\pi_{s_1.\text{Name},s_2.\text{Name}}\Big(\big(\rho_{s_1}(\text{Studenten}) \bowtie \rho_{h_1}(\text{hoeren})\big) \bowtie_{(h_1.\text{VorlNr}=h_2.\text{VorlNr} \wedge s_1.\text{MatrNr}<>s_2.\text{MatrNr})} \big(\rho_{s_2}(\text{Studenten}) \bowtie \rho_{h_2}(\text{hoeren})\big)\Big)$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Universitaet.tex

Übungsaufgabe „Vater-Muter-Kind“ (Division)

Division

R

[2]

Vater	Mutter	Kind	Alter
Hans	Helga	Harald	5
Hans	Helga	Maria	4
Hans	Ursula	Sabine	2
Martin	Melanie	Gertrud	7
Martin	Melanie	Maria	4
Martin	Melanie	Sabine	2
Peter	Christina	Robert	9

S

Kind	Alter
Maria	4
Sabine	2

$R \div S$

Kind	Alter
Maria	4
Sabine	2

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Vater-Muter-Kind.tex`

[2] `https://de.wikibooks.org/wiki/Relationenalgebra_und_SQL:_Division`

Übungsaufgabe „Wassned“ (Relationale Algebra)

Gegeben ist folgende Datenbank-Anfrage:

$$\pi_{\text{Bezeichnung}}\left(\sigma_{\text{SWS}=2\wedge\neg(\text{Name}=\text{'Wassned'})}\left(\text{Vorlesung} \bowtie \text{Professor}\right)\right)$$

(a) Geben Sie eine umgangssprachliche Formulierung der Anfrage an!

Lösungsvorschlag

Eine Liste mit den Bezeichnungen der Vorlesungen, die 2 Semester Wochenstunden dauern und die nicht vom dem Professor „Wassned“ gelesen werden.

(b) Geben Sie die Ergebnistabelle an!

Lösungsvorschlag

Bezeichnung
Japanische Malerei
Chinesische Schrift

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/20_Relationale-Algebra/Aufgabe_Wassned.tex`

Übungsaufgabe „Xenokrates“ (Tupelkalkül)

Tupelkalkül

Lösen Sie die Aufgaben im Tupel- und Domänenkalkül:

(a) Geben Sie alle Vorlesungen an, die der Student *Xenokrates* gehört hat.

Lösungsvorschlag

$$\{v | v \in Vorlesungen \wedge \exists h \in hoeren(v.VorlNr = h.VorlNr \wedge \exists s \in Studenten(h.MatrNr = s.MatrNr \wedge s.Name = \text{‘}Xenokrates\text{’}))\}$$

(b) Geben Sie die Titel der direkten Voraussetzungen für die Vorlesung Wissenschaftstheorie an.

(c) Geben Sie Paare von Studenten(-Namen) an, die sich aus der Vorlesung Grundzüge kennen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/30_Relationales-Modell/30_Tupelkalkuel/Aufgabe_Xenokrates.tex`

Übungsaufgabe „Kaufhaus“ (SQL, SQL mit Übungsdatenbank, Relationale Algebra) SQL

Die relationale Datenbank eines Kaufhauses enthält folgende Tabellen:

Artikel

ArtNr	Bezeichnung	Verkaufspreis	Einkaufspreis
95	Kamm	1.25	0.80
97	Kamm	0.99	0.75
507	Seife	3.93	2.45
1056	Zwieback	1.20	0.90
1401	Räucherlachs	4.90	3.60
2045	Herrenhose	37.25	24.45
2046	Herrenhose	20.00	17.00
2340	Sommerkleid	94.60	71.50

Abteilung

Abteilungsname	Stockwerk	Abteilungsleiter
Lebensmittel	I	Josef Kunz
Lebensmittel	EG	Monika Stiehl
Textilien	II	Monika Stiehl

Bestand

Abteilungsname	ArtNr	Vorrat
Lebensmittel	1056	129
Lebensmittel	1401	200
Textilien	2045	14

```
CREATE TABLE Artikel (
  ArtNr INTEGER PRIMARY KEY NOT NULL,
  Bezeichnung VARCHAR(100) NOT NULL,
  Verkaufspreis FLOAT(2),
  Einkaufspreis FLOAT(2)
);

CREATE TABLE Abteilung (
  Abteilungsname VARCHAR(60) NOT NULL,
  Stockwerk VARCHAR(10) NOT NULL,
  Abteilungsleiter VARCHAR(100),
  PRIMARY KEY (Abteilungsname, Stockwerk)
);
```

```
CREATE TABLE Bestand (
  Abteilungsname VARCHAR(100) REFERENCES Abteilung(Abteilungsname),
  ArtNr INTEGER REFERENCES Artikel(ArtNr),
  Vorrat INTEGER,
  PRIMARY KEY (Abteilungsname, ArtNr)
);

INSERT INTO Artikel VALUES
  (95, 'Kamm', 1.25, 0.80),
  (97, 'Kamm', 0.99, 0.75),
  (507, 'Seife', 3.93, 2.45),
  (1056, 'Zwieback', 1.20, 0.90),
  (1401, 'Räucherlachs', 4.90, 3.60),
  (2045, 'Herrenhose', 37.25, 24.45),
  (2046, 'Herrenhose', 20.00, 17.00),
  (2340, 'Sommerkleid', 94.60, 71.50);

INSERT INTO Abteilung VALUES
  ('Lebensmittel', 'I', 'Josef Kunz'),
  ('Lebensmittel', 'EG', 'Monika Stiehl'),
  ('Textilien', 'II', 'Monika Stiehl');

INSERT INTO Bestand VALUES
  ('Lebensmittel', 1056, 129)
  ('Lebensmittel', 1401, 200)
  ('Textilien', 2045, 14);
```

Formulieren Sie mit Hilfe von SQL folgende Anfragen:

(a) Gesucht sind alle Informationen über Herrenhose und Sommerkleid!

Lösungsvorschlag

```
SELECT *
FROM Artikel
WHERE
  Bezeichnung = 'Herrenhose' OR
  Bezeichnung = 'Sommerkleid';
```

(b) Welche Artikelnummer hat der Zwieback?

Lösungsvorschlag

```
SELECT ArtNr
FROM Artikel
WHERE
  Bezeichnung = 'Zwieback';
```

(c) Welche Waren (Artikelnummer und Verkaufspreis) werden für mehr als 25€ verkauft?

Lösungsvorschlag

```
SELECT ArtNr, Verkaufspreis
FROM Artikel
```

```
WHERE Verkaufspreis > 25.00;
```

(d) Welche Artikel (Angabe der Bezeichnung) bietet das Kaufhaus an?

Lösungsvorschlag

```
SELECT DISTINCT Bezeichnung
FROM Artikel;
```

(e) Gesucht sind die Artikelnummern aller Artikel mit Ausnahme der Artikelnummer 2046.

Lösungsvorschlag

```
SELECT ArtNr
FROM Artikel
WHERE NOT (ArtNr = 2046);
```

(f) Gib die Artikelnummern und die Verkaufspreise aller Herrenhosen aus, die für höchstens 25€ verkauft werden! Der Spaltenname für die Verkaufspreise soll in der Ergebnistabelle „Sonderangebot“ heißen.

Lösungsvorschlag

```
SELECT ArtNr, Verkaufspreis AS Sonderangebot
FROM Artikel
WHERE Bezeichnung = 'Herrenhose' AND Verkaufspreis <= 25;
```

(g) Gib Artikelnummer und Verkaufspreis aller Artikel aus, die im Einkauf zwischen 80 Cent und 5€ kosten.

Lösungsvorschlag

```
SELECT ArtNr, Verkaufspreis
FROM Artikel
WHERE Einkaufspreis BETWEEN 0.80 AND 5.00;
```

Teilaufgabe 2

(a) Geben Sie die SQL-Befehle an, mit der die Tabellenschemata von Artikel und Bestand erzeugt werden können. Wählen Sie dabei geeignete Domänen.

Lösungsvorschlag

```
CREATE TABLE Artikel (
  ArtNr INTEGER PRIMARY KEY NOT NULL,
  Bezeichnung VARCHAR(100) NOT NULL,
  Verkaufspreis FLOAT(2),
  Einkaufspreis FLOAT(2)
);

CREATE TABLE Bestand (
  Abteilungsname VARCHAR(100) REFERENCES Abteilung(Abteilungsname),
  ArtNr INTEGER REFERENCES Artikel(ArtNr),
  Vorrat INTEGER,
  PRIMARY KEY (Abteilungsname, ArtNr)
```

```
);
```

(b) Es treten nun nacheinander die folgenden Änderungen auf. Aktualisieren Sie den Tabellenbestand mit den entsprechenden SQL-Befehlen:

(i) Ein Sommerkleid mit der Artikelnummer 2341, dem Einkaufspreis 70€ und dem Verkaufspreis 90,75€ wird in das Artikelsortiment aufgenommen.

Lösungsvorschlag

```
INSERT INTO Artikel
VALUES (2341, 'Sommerkleid', 90.75, 70.00);
```

(ii) Der Artikel mit der Nummer 2341 wird wieder aus dem Sortiment genommen, da er den Qualitätsstandards nicht entsprochen hat.

Lösungsvorschlag

```
DELETE FROM Artikel WHERE ArtNr = 2341;
```

(iii) Eine Bürste mit der Artikelnummer 2 wird in das Sortiment aufgenommen. Einkaufs- bzw. Verkaufspreis sind noch nicht festgelegt.

Lösungsvorschlag

```
INSERT INTO Artikel (ArtNr, Bezeichnung)
VALUES (2, 'Bürste');
```

(iv) Eine Damenhose (Verkaufspreis 89€, Einkaufspreis: 60,50€) wird neu in das Sortiment aufgenommen. Eine Artikelnummer wurde noch nicht festgelegt.

Lösungsvorschlag

`ArtNr` ist der Primärschlüssel der Tabelle `Artikel`. Bei Eingabe eines neuen Datensatzes müssen mindestens die Werte aller Attribute, die zum Primärschlüssel gehören, angegeben werden. Da aber im Fall der Damenhose die Artikelnummer noch nicht festgelegt ist, ist eine Eingabe der Damenhose-Daten in die Tabelle Artikel nicht möglich. Hinweis: Denken Sie also immer daran, dass bei Einfügen von Datensätzen der Primärschlüssel keine NULL-Werte enthalten darf!

(v) Die Herrenhosen werden aus dem Sortiment genommen und deshalb aus der Tabelle Artikel gelöscht.

Lösungsvorschlag

```
DELETE FROM Bestand WHERE ArtNr = 2045;
DELETE FROM Artikel WHERE Bezeichnung = 'Herrenhose';
```

(vi) Die neue Abteilungsleiterin der Lebensmittelabteilung heißt Elvira Sommer.

Lösungsvorschlag

```
UPDATE Abteilung
SET Abteilungsleiter = 'Elvira Sommer'
WHERE Abteilungsnahme = 'Lebensmittel';
```

(vii) Die Abteilung Feinkost hat einen Bestand von 150 Räucherlachspackungen.

Lösungsvorschlag

Die Attribute ArtNr und Abteilungsname der Tabelle Bestand sind Fremdschlüssel. Ein neuer Datensatz darf in die Tabelle nur eingefügt werden, wenn die Fremdschlüsselwerte in den entsprechenden (Primärschlüssel-)Attribute der referenzierten Tabelle auch existieren. Die Abteilung Feinkost, genauer gesagt den Abteilungsnamen 'Feinkost' gibt es in Abteilung aber noch nicht.

i. Lösungsmöglichkeit: Die Aktualisierung kann nicht durchgeführt werden.
ii. Lösungsmöglichkeit: Die entsprechende Abteilung Feinkost wird – natürlich in „Absprache“ mit der Kaufhausleitung – eingeführt und ein dementsprechender Datensatz in Abteilung eingefügt.

```
INSERT INTO Abteilung (Abteilungsname) VALUES ('Feinkost');
INSERT INTO Bestand VALUES ('Feinkost', 1401, 150);
```

(c) Formulieren Sie folgende Anfragen in SQL:

(i) Gesucht sind Artikelnummer und Vorrat aller Artikel aus der Textil-Abteilung.

Lösungsvorschlag

```
SELECT ArtNr, Vorrat FROM Bestand WHERE Abteilungsname = 'Textilien';
```

(ii) Gesucht sind alle Informationen über die Abteilungen, die im zweiten Stock platziert sind oder von Frau Stiehl geleitet werden.

Lösungsvorschlag

```
SELECT * FROM Abteilung
WHERE Stockwerk = 'II' OR Abteilungsleiter = 'Monika Stiehl';
```

(d) Formulieren Sie folgende SQL-Anfragen umgangssprachlich:

(i) SQL-Anfrage:

```
SELECT DISTINCT Abteilungsleiter
FROM Abteilung
WHERE NOT (Abteilungsname = 'Kosmetik');
```

Lösungsvorschlag

Gesucht sind die Namen aller Abteilungsleiter mit Ausnahme der Kosmetik-Abteilung. Duplikate sollen eliminiert werden.

(ii) SQL-Anfrage:

```
SELECT ArtNr
FROM Bestand
WHERE Abteilungsname = "Lebensmittel" AND Vorrat <= 100;
```

Lösungsvorschlag

Gesucht sind die Nummern der Artikel, von denen in der Lebensmittelabteilung maximal 100 vorrätig sind.

(e) Interpretieren Sie nun die obigen Tabellen als Repräsentationen der drei Relationen Artikel, Abteilung und Bestand. Bestimmen Sie die Ergebnisrelationen folgender relationaler Ausdrücke:

(i) $\pi_{ArtNr,Bezeichnung}(Artikel)$

Lösungsvorschlag

95	Kamm
97	Kamm
507	Seife
1056	Zwieback
1401	Räucherlachs
2045	Herrenhose
2046	Herrenhose
2340	Sommerkleid

(ii) $\pi_{Abteilungsname}(Bestand)$

Lösungsvorschlag

Lebensmittel

Textilien

(iii) $\sigma_{((Vorrat<100 \wedge ArtNr>1500) \vee ArtNr<1100)}(Bestand)$

Lösungsvorschlag

Lebensmittel	1056	129
Textilien	2045	14

(iv) $\sigma_{((Vorrat<100 \wedge (ArtNr>1500 \vee ArtNr<1100)}(Bestand)$

Lösungsvorschlag

Textilien 2045 14

(v) $\pi_{ArtNr}(\sigma_{Bezeichnung=Herrenhose}(Artikel))$

Lösungsvorschlag

2045

2046

(vi) $\pi_{Abteilungsname}(Abteilung) - \pi_{Abteilungsname}(Bestand)$

Lösungsvorschlag

Kosmetik

(vii) $\pi_{Bezeichnung,Einkaufspreis}(\sigma_{Einkaufspreis<2.50}(Artikel)) \cup$
$\pi_{Bezeichnung,Einkaufspreis}(\sigma_{Einkaufsreis>20.00}(Artikel))$

Lösungsvorschlag

Die letzten Zeile ist nicht in der Musterlösung dabei. Ich glaube aber es müsste so stimmen.

Bezeichnung	Einkaufspreis
Kamm	0.80
Kamm	0.75
Seife	2.45
Zwieback	0.90
Herrenhose	24.45
Sommerkleid	71.50

Teilaufgabe 2

(a) Formulieren Sie nachfolgende Anfragen in SQL mit Hilfe von Joins!

- Wie viele Packungen Zwieback sind noch vorrätig?

Lösungsvorschlag

Hinweis: In obigem Lösungsansatz wird berücksichtigt, dass ein Artikel, hier der Zwieback, in mehreren Abteilungen verkauft werden kann. Geht man davon aus, dass Zwieback nur in einer Abteilung verkauft wird, kann man die Aggregatfunktion SUM weglassen.

```
SELECT SUM(b.Vorrat)
FROM Bestand b, Artikel a
```

```
WHERE b.ArtNr = a.ArtNr AND a.Bezeichnung = 'Zwieback';
```

- In welchem Stockwerk wird Räucherlachs verkauft?

Lösungsvorschlag

```
SELECT Abteilung.Stockwerk
FROM Artikel, Abteilung, Bestand
WHERE Artikel.ArtNr = Bestand.ArtNr AND
Bestand.Abteilungsname = Abteilung.Abteilungsname AND
Artikel.Bezeichnung = 'Räucherlachs';
```

(b) Formulieren Sie folgende Anfragen an die Kaufhaus-Datenbank unter Verwendung von geschachtelten SELECT-Anweisungen!

- Gib die Bezeichnungen und die Artikelnummern aller Artikel aus, die nicht mehr als der Artikel mit der Artikelnummer 1401 kosten!

Lösungsvorschlag

Hinweis: Durch Hinzufügen der Bedingung NOT(ArtNr=1401) wird der Artikel mit der Nummer 1401 in der Ergebnistabelle nicht aufgeführt

```
SELECT Bezeichnung, ArtNr AS Artikelnummer
FROM Artikel
WHERE Verkaufspreis <= (
  SELECT Verkaufspreis FROM Artikel WHERE ArtNr = 1401
);
```

- Gesucht sind Bezeichnung und Verkaufspreis aller Artikel, die in der Textilienabteilung verkauft werden!

Lösungsvorschlag

```
SELECT Bezeichnung, Verkaufspreis
FROM Artikel WHERE ArtNr in (
  SELECT ArtNr FROM Bestand WHERE Abteilungsname = 'Textilien'
);
```

- Welche Produkte (Angabe der Bezeichnung) werden im Erdgeschoss verkauft?

Lösungsvorschlag

```
SELECT DISTINCT Bezeichnung
FROM Artikel
WHERE ArtNr in (
  SELECT ArtNr
  FROM Bestand
  WHERE Abteilungsname IN (
    SELECT Abteilungsname
    FROM Abteilung
    WHERE Stockwerk = 'EG'
  )
);
```

- Gib die Namen aller Abteilungsleiter aus, in deren Abteilungen von jedem Artikel weniger als 100 Exemplare vorrätig sind!

Lösungsvorschlag

```
SELECT DISTINCT Abteilungsleiter
FROM Abteilung
WHERE NOT EXISTS (
  SELECT *
  FROM Bestand
  WHERE (Abteilung.Abteilungsname =
  Bestand.Abteilungsname) AND Vorrat >= 100
);
```

(c) Lösen Sie die Aufgabe 1b) Punkt 1 ohne Verwendung einer geschachtelten SQL Anfrage! (Gib die Bezeichnungen und die Artikelnummern aller Artikel aus, die nicht mehr als der Artikel mit der Artikelnummer 1401 kosten!)

Lösungsvorschlag

```
SELECT a.Bezeichnung, a.ArtNr as Artikelnummer
FROM Artikel a, Artikel b
WHERE
  a.Verkaufspreis <= b.Verkaufspreis AND
  b.ArtNr = 1401;
```

(d) Formulieren Sie nachfolgende Anfragen mit Mengenoperatoren!

- Gibt es registrierte Artikel, die noch nicht im Bestand aufgeführt sind?

Lösungsvorschlag

```
SELECT ArtNr FROM Artikel
EXCEPT
SELECT ArtNr FROM Bestand;
```

- Welche Artikel (Artikelnummer) sind registriert und bereits im Bestand aufgeführt?

Lösungsvorschlag

```
SELECT ArtNr FROM Artikel
INTERSECT
SELECT ArtNr FROM Bestand;
```

- Welche Artikel (Bezeichnung und Artikelnummer) sind bereits registriert und im Bestand aufgeführt?

Lösungsvorschlag

```
SELECT Bezeichnung, ArtNr FROM Artikel WHERE ArtNr IN (
  SELECT ArtNr FROM Artikel
  INTERSECT
  SELECT ArtNr FROM Bestand
);
```

(e) Formulieren Sie folgende Anfragen in SQL:

- Welche Artikel mit dem Anfangsbuchstaben “S” gibt es?

Lösungsvorschlag SQL

```sql
SELECT Bezeichnung FROM Artikel WHERE Bezeichnung LIKE 'S%';
```

- Welche Artikel haben an der 3. Stelle ein “i”?

Lösungsvorschlag

```sql
SELECT Bezeichnung FROM Artikel WHERE Bezeichnung LIKE '__i%';
```

- Heißt der Artikel “Zwieback” oder “Zweiback”?

Lösungsvorschlag

```sql
SELECT Bezeichnung FROM Artikel WHERE Bezeichnung LIKE 'Zw__back';
```

Teilaufgabe 4

(a) Welche Artikel (Artikelnummer, Abteilungsname) werden in den Abteilungen angeboten? Die Ausgabe soll absteigend nach der Artikelnummer sortiert werden. Bei gleicher Artikelnummer sollen die betroffenen Abteilungen alphabetisch aufgelistet werden.

Lösungsvorschlag

```sql
SELECT ArtNr, Abteilungsname
FROM Bestand
ORDER BY ArtNr DESC, Abteilungsname;
```

(b) Wie viele verschiedene Waren werden in der Lebensmittelabteilung verkauft?

Lösungsvorschlag

```sql
SELECT COUNT(*)
FROM Bestand
WHERE Abteilungsname = 'Lebensmittel';
```

(c) Wie viele verschiedene Waren werden in den einzelnen Abteilungen verkauft?

Lösungsvorschlag

```sql
SELECT Abteilungsname, COUNT(*)
FROM Bestand
GROUP BY Abteilungsname;
```

(d) Wie viel kostet der billigste, wie viel der teuerste Artikel?

Lösungsvorschlag

```sql
SELECT MIN(Verkaufspreis), MAX(Verkaufspreis)
FROM Artikel;
```

(e) Gib die Namen aller Abteilungen aus, deren Gesamtvorrat an Artikel kleiner als 100 ist!

Lösungsvorschlag

```sql
SELECT Abteilungsname
FROM Bestand
GROUP BY Abteilungsname
HAVING COUNT(Vorrat) < 100;
```

(f) Gesucht sind Bezeichnung und Verkaufspreis aller in der Datenbank gespeicherten Artikel. Die Ausgabe soll alphabetisch aufgelistet werden. Bei gleicher Bezeichnung sollen die teureren Artikel zuerst aufgelistet werden.

Lösungsvorschlag

```sql
SELECT Bezeichnung, Verkaufspreis
FROM Artikel
ORDER BY Bezeichnung, Verkaufspreis DESC;
```

(g) Gib für alle Artikel, von denen (unabhängig von der Abteilung) noch mindestens 130 Exemplare vorrätig sind, die Artikelnummer und den aktuellen Vorrat aus!

Lösungsvorschlag

```sql
SELECT ArtNr, SUM(Vorrat)
FROM Bestand
GROUP BY ArtNr
HAVING SUM(Vorrat) >= 130;
```

Teilaufgabe 5

(a) Sicht `view1`: Gesucht sind alle Informationen zu Artikeln, an denen das Kaufhaus mehr als 35% verdient.

Lösungsvorschlag

```sql
CREATE VIEW view1 AS
SELECT *
FROM Artikel
WHERE Verkaufspreis > 1.35 * Einkaufspreis;
```

(b) Sicht `view2`: Gesucht sind alle Informationen zu Artikeln, an denen das Kaufhaus mehr als 35% verdient und die für höchstens 50 € verkauft werden.

Lösungsvorschlag

```sql
CREATE VIEW view2 AS
SELECT *
FROM view1
WHERE Verkaufspreis <= 50;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/40_Relationale-Anfragesprachen/10_SQL/Aufgabe_Kaufhaus.tex

Relationale Anfragesprachen

Examensaufgabe „Personalverwaltung“ (46116-2012-F.T1-TA1-A3)

Aufgabe 3

Gegeben ist folgendes einfache Datenbankschema zur Personalverwaltung (Schlüsselattribute unterstrichen, Fremdschlüssel kursiv): `Angestellter (PersNr, Name, Gehalt, Beruf, AbtNr, Ort)`
`Abteilung (AbtNr, Name, Ort)`
Formulieren Sie folgende Datenbankoperationen in SQL:

(a) Welche Angestellten sind von Beruf Koch und verdienen mehr als 5000 €?

Lösungsvorschlag

```
SELECT name
FROM Angestellter
WHERE Beruf = 'Koch' AND Gehalt > 5000:
```

(b) Welche Angestellten der Abteilung B17 sind aus München?

Lösungsvorschlag

```
SELECT Name
FROM Angestellter, Abteilung
WHERE
  Angestellter.AbtNr = Abteilung.AbtNr AND
  Abteilung.Name = 'B17' AND
  Abgesteller.Ort = 'München';
```

(c) Hans Meier aus der Abteilung C4 zieht von München nach Erlangen um.

Lösungsvorschlag

```
UPDATE Angestellter SET Ort = "Erlangen"
WHERE Name = "Hans Meier" AND (
  SELECT ab.AbtNr
  FROM Abteilung ab
  WHERE ab.Name = "C4"
) = AbtNr;
```

(d) Abteilung C4 wird aufgelöst.

(e) Formulieren Sie folgende SQL-Anfrage umgangssprachlich und so exakt wie möglich.

```sql
SELECT AbtNr
FROM Abteilung a, (
  SELECT PersNr, Name, AbtNr
  FROM Angestellter
  WHERE Gehalt < 2000) t
WHERE
a.AbtNr = t.AbtNr AND a.Ort = "Nuernberg";
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2012/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex

Examensaufgabe „Mitfahrgelegenheiten“ (46116-2014-F.T2-TA2-A3)

SQL

„Kunde”:

KID	Name	Vorname	Stadt
K1	Meier	Stefan	S3
K2	Müller	Peta	S3
K3	Schmidt	Christine	S2
K4	Schulz	Michael	S4

„Stadt”

SID	SName	Bundesland
S1	Berlin	Berlin
S2	Nürnberg	Bayern
S3	Köln	Nordrhein-Wesffalen
S4	Stuttgart	Baden-Württemberg
S5	München	Bayern

„Angebot”:

KID	Start	Ziel	Datum	Plätze
K4	S4	S5	08.07.2011	3
K4	S5	S4	10.07.2011	3
K1	S1	S5	08.07.2011	3
K3	S2	S3	15.07.2011	1
K4	S4	S1	15.07.2011	3
K1	S5	S5	09.07.2011	2

„Anfrage”:

KID	Start	Ziel	Datum
K2	S4	S5	08.07.2011
K2	S5	S4	10.07.2011
K3	S2	S3	08.07.2011
K3	S3	S2	10.07.2011
K2	S4	S5	05.07.2011
K2	S5	S4	17.07.2011

```
CREATE TABLE Stadt (
  SID VARCHAR(100) NOT NULL PRIMARY KEY,
  SName VARCHAR(100) NOT NULL,
  Bundesland VARCHAR(100) NOT NULL
);

CREATE TABLE Anfrage (
  KID VARCHAR(100) NOT NULL,
  Start VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
  Ziel VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
  Datum date NOT NULL,
  PRIMARY KEY (KID, Start, Ziel, Datum)
);

CREATE TABLE Angebot (
  KID VARCHAR(100) NOT NULL,
  Start VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
  Ziel VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
  Datum date NOT NULL,
  Plätze integer DEFAULT NULL,
  PRIMARY KEY (Datum, KID)
);

CREATE TABLE Kunde (
  KID VARCHAR(100) NOT NULL PRIMARY KEY,
  Name VARCHAR(100) DEFAULT NULL,
  Vorname VARCHAR(100) DEFAULT NULL,
  Stadt VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID)
);
```

```
INSERT INTO Stadt (SID, SName, Bundesland) VALUES
  ('S1', 'Berlin', 'Berlin'),
  ('S2', 'Nürnberg', 'Bayern'),
  ('S3', 'Köln', 'NRW'),
  ('S4', 'Stuttgart', 'BW'),
  ('S5', 'München', 'Bayern');

INSERT INTO Anfrage (KID, Start, Ziel, Datum) VALUES
  ('K2', 'S4', 'S5', '2011-07-05'),
  ('K2', 'S4', 'S5', '2011-07-08'),
  ('K3', 'S2', 'S3', '2011-07-08'),
  ('K2', 'S5', 'S4', '2011-07-10'),
  ('K3', 'S3', 'S2', '2011-07-10'),
  ('K2', 'S5', 'S4', '2011-07-17');

INSERT INTO Kunde (KID, Name, Vorname, Stadt) VALUES
  ('K1', 'Meier', 'Stefan', 'S3'),
  ('K2', 'Müller', 'Petra', 'S3'),
  ('K3', 'Schmidt', 'Christine', 'S2'),
  ('K4', 'Schulz', 'Michael', 'S4');

INSERT INTO Angebot (KID, Start, Ziel, Datum, Plätze) VALUES
  ('K1', 'S1', 'S5', '2011-07-08', 3),
  ('K4', 'S4', 'S5', '2011-07-08', 3),
  ('K1', 'S5', 'S4', '2011-07-09', 2),
  ('K4', 'S5', 'S4', '2011-07-10', 3),
  ('K3', 'S2', 'S3', '2011-07-15', 1),
  ('K4', 'S4', 'S1', '2011-07-15', 3);
```

(a) Formulieren Sie die folgenden Anfragen in SQL:

(i) Geben Sie alle Attribute aller Anfragen aus, für die passende Angebote existieren! Ein Angebot ist passend zu einer Anfrage, wenn Start, Ziel und Datum identisch sind!

Lösungsvorschlag

```
SELECT Anfrage.KID, Anfrage.Start, Anfrage.Ziel, Anfrage.Datum
FROM Anfrage, Angebot
WHERE
  Anfrage.Start = Angebot.Start AND
  Anfrage.Ziel = Angebot.Ziel AND
  Anfrage.Datum = Angebot.Datum;
```

(ii) Finden Sie Nachnamen und Vornamen aller Kunden, für die kein Angebot existiert!

Lösungsvorschlag

```
SELECT k.Name, k.Vorname
FROM Kunde k
WHERE NOT EXISTS ( SELECT * FROM Angebot a WHERE a.KID = k.KID )
```

oder:

```
SELECT k.Name, k.Vorname
FROM Kunde k
```

```
WHERE k.KID NOT IN ( SELECT KID FROM Angebot );
```

GROUP BY
HAVING

(iii) Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus und sortieren Sie das Ergebnis aufsteigend!

```
SELECT Datum
FROM Angebot, Stadt
WHERE
  (SID = Start OR
  SID = Ziel)
  AND
  (SName = 'München' OR SName = 'Stuttgart')
```

(iv) Geben Sie für jeden Startort einer Anfrage den Namen der Stadt und die Anzahl der Anfragen aus.

```
SELECT SName, COUNT(*)
FROM Anfrage, Stadt
WHERE SID = Start
GROUP BY SID;
```

(b) Wie sieht die Ergebnisrelation zu folgenden Anfragen auf den Beispieldaten aus?

```
SELECT *
FROM
Stadt
WHERE
NOT EXISTS ( SELECT *
FROM Anfrage
WHERE Start = SID OR Ziel = SID ) ;
```

Lösungsvorschlag

S1 Berlin Berlin

```
SELECT KID, SUM (Plätze)
FROM Angebot
WHERE Plätze > 2
GROUP BY KID
HAVING SUM (Plätze) > 4;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex

Examensaufgabe „Turmspringen“ (46116-2017-H.T2-TA2-A4)

SQL
GROUP BY

Für die bayerische Meisterschaft im Turmspringen ist folgendes Datenbankschema angelegt:

```
Springer : {[ Startnummer, Nachname, Vorname, Geburtsdatum, Körpergröße ]}

Sprung : {[ SID, Beschreibung, Schwierigkeit ]}

springt : {[ SID, Startnummer, Durchgang ]}
FK (SID) referenziert Sprung (SID)
FK (Startnummer) referenziert Springer (Startnummer)
```

Das Attribut Schwierigkeit kann die Werte 1 bis 10 annehmen, das Attribut Durchgang ist positiv und ganzzahlig. Die Körpergröße der Springer ist in Zentimeter angegeben.

(a) Welche Springer sind größer als 1,80 m? Schreiben Sie eine SQL-Anweisung, welche in der Ausgabe mit dem größten Springer beginnt.

Lösungsvorschlag

```sql
SELECT Vorname, Nachname, Körpergröße
FROM Springer
WHERE Körpergröße > 180
ORDER BY Körpergröße DESC;
```

(b) Welche Springer haben im ersten Durchgang einen Sprung mit einer Schwierigkeit von unter 6 gezeigt? Schreiben Sie eine SQL-Anweisung, welche Startnummer und Nachname dieser Springer ausgibt.

Lösungsvorschlag

```sql
SELECT Springer.Startnummer, Springer.Nachname
FROM Springer, Sprung, springt
WHERE
  Sprung.SID = springt.SID AND
  Springer.Startnummer = springt.Startnummer AND
  springt.Durchgang = 1 AND
  Sprung.Schwierigkeit < 6;
```

(c) Formulieren Sie in Umgangssprache, aber trotzdem möglichst präzise, wonach mit folgender Abfrage gesucht wird:

```sql
SELECT springt.Startnummer, s.Nachname, s.Vorname, MAX(springt.Durchgang)
FROM springt, Springer s
WHERE springt.Startnummer = s.Startnummer
GROUP BY springt.Startnummer, s.Nachname, s.Vorname
```

Lösungsvorschlag

Die Abfrage gibt die Startnummer, den Nachnamen, den Vornamen und die Anzahl der Sprünge, d. h. die Anzahl der Durchgänge der einzelnen Springer an.

(d) Gesucht ist die *„durchschnittliche Körpergröße“* all der Springer, die vor dem 01.01.2000 geboren wurden. Formulieren Sie eine SQL-Anweisung, wobei die Spalte mit der durchschnittlichen Körpergröße genau diesen Namen *„durchschnittliche Körpergröße“* haben soll.

Lösungsvorschlag

Umlaute und Leerzeichen sind bei Spaltenbeschriftungen nicht erlaubt.

```
SELECT AVG(Körpergröße) AS durchschnittliche_Koerpergroesse
FROM SPRINGER
WHERE Geburtsdatum < DATE('2000-01-01');
```

oder

Lösungsvorschlag

```
SELECT AVG(Körpergröße) AS durchschnittliche_Koerpergroesse
FROM SPRINGER
WHERE Geburtsdatum < '01.01.2000';
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/09/Thema-2/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Kundenverwaltungssystem“ (46116-2018-H.T1-TA1-A4)

SQL CREATE TABLE

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

```
Kunde : {[ ID, Vorname, Nachname, PLZ ]}

Produkt : {[ GTIN, Bezeichnung, Bruttopreis, MWStSatz) ]}

Kauf : {[ ID[Kunde], GTIN[Produkt], Datum, Menge ]}
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *„Kauf“* anlegt. Gehen Sie davon aus, dass die Tabellen *„Kunde“* und *„Produkt“* bereits existieren.

Lösungsvorschlag

```sql
CREATE TABLE IF NOT EXISTS Kauf (
  ID INTEGER REFERENCES Kunde(ID),
  GTIN INTEGER REFERENCES Produkt(GTIN),
  Datum DATE,
  Menge INTEGER,
  PRIMARY KEY (ID, GTIN, Datum)
);
```

(b) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller *Kunden* mit der *Postleitzahl 20251* ausgibt, absteigend sortiert nach *Nachname* und bei gleichen *Nachnamen*, absteigend nach *Vorname*.

Lösungsvorschlag

```sql
SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname DESC, Vorname DESC;
```

```
 vorname | nachname
---------+----------
 Hanna   | Winter
 Jakob   | Sommer
 Bert    | Sommer
(3 rows)
```

(c) Schreiben Sie eine SQL-Anweisung, die zu jedem Einkauf mit mehr als 10 unterschiedlichen Produkten den *Nachnamen* des *Kunden* und den *Bruttogesamtpreis* des Einkaufs ausgibt. Ein Einkauf ist definiert als Menge aller Produkte, die ein bestimmter Kunde an einem bestimmten Datum kauft.

Lösungsvorschlag HAVING EXCEPT

```
SELECT Nachname, SUM(Bruttopreis * Menge)
FROM Kunde k, Produkt p, Kauf x
WHERE k.ID = x.ID AND p.GTIN = x.GTIN
GROUP BY Datum, Nachname, k.ID
HAVING COUNT (*) > 10;
```

```
  nachname   |  sum
-------------+--------
 Mustermann | 713.86
(1 row)
```

(d) Schreiben Sie eine SQL-Anweisung, die die *GTINs* aller Produkte ausgibt, die an mindestens einen in der Datenbank enthaltenen PLZ-Bereich noch nie verkauft worden sind. Als in der Datenbank enthaltener PLZ-Bereich gelten alle in der Tabelle *„Kunde“* enthaltenen PLZs. Ein Produkt gilt als an einen PLZ-Bereich verkauft, sobald es von mindestens einem Kunden aus diesem PLZ-Bereich gekauft wurde. Produkte, die bisher noch gar nicht verkauft worden sind, müssen nicht berücksichtigt werden.

Lösungsvorschlag

Die beiden Lösungswege liefern leider unterschiedliche Ergebnisse.

```
WITH tmp AS (
  SELECT x.GTIN, k.PLZ
  FROM Kunde k, Kauf x
  WHERE x.ID = k.ID
  GROUP BY x.GTIN, k.PLZ
)

SELECT DISTINCT GTIN
FROM tmp
WHERE EXISTS (
  SELECT Kunde.PLZ
  FROM Kunde LEFT OUTER JOIN tmp
  ON Kunde.PLZ = tmp.PLZ
  WHERE tmp.PLZ IS NULL
)
ORDER BY GTIN;
```

```
 gtin
------
    4
   23
  112
  113
  123
  124
  125
```

Top-N-Query
WITH

```
  155
  189
  324
  453
  765
(12 rows)
```

oder

```
SELECT DISTINCT GTIN FROM (
  (
    SELECT GTIN, PLZ
    FROM Kunde, Produkt
  )
  EXCEPT
  (
    SELECT x.GTIN, k.PLZ
    FROM Kunde k, Kauf x
    WHERE x.ID = k.ID
    GROUP BY x.GTIN, k.PLZ
  )
) as tmp
ORDER BY GTIN;
```

```
 gtin
------
    4
   23
  112
  113
  123
  124
  125
  155
  189
  324
  453
  765
  889
(13 rows)
```

(e) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der am meisten verkauften Produkte ausgibt. Ausgegeben werden sollen der Rang (1 bis 10) und die Bezeichnung des Produkts. Gehen Sie davon aus, dass es keine zwei Produkte mit gleicher Verkaufszahl gibt und verwenden Sie keine produktspezifischen Anweisungen wie beispielsweise `ROWNUM`, `TOP` oder `LIMIT`.

Lösungsvorschlag DELETE

```
WITH Gesamtverkauf AS (
  SELECT k.GTIN, Bezeichnung, SUM(Menge) AS Gesamtmenge
  FROM Produkt p, Kauf k
  WHERE p.GTIN = k.GTIN
  GROUP BY k.GTIN, Bezeichnung
)

SELECT g1.Bezeichnung, COUNT (*) AS Rang
FROM Gesamtverkauf g1, Gesamtverkauf g2
WHERE g1.Gesamtmenge <= g2.Gesamtmenge
GROUP BY g1.GTIN, g1.Bezeichnung
HAVING COUNT (*) <= 10
ORDER BY Rang;
```

```
 bezeichnung  | rang
--------------+------
 Topf         |    1
 Kaffee       |    2
 Sonnenbrille |    3
 T-Shirt      |    4
 Klopapier    |    5
 Duschgel     |    6
 Hammer       |    7
 Heft         |    8
(8 rows)
```

(f) Schreiben Sie eine SQL-Anweisung, die alle Produkte löscht, die noch nie gekauft wurden.

Lösungsvorschlag

```
 count
-------
    13
(1 row)

SELECT COUNT(*) FROM Produkt;

DELETE FROM Produkt
WHERE GTIN NOT IN
(
  SELECT DISTINCT GTIN
  FROM Kauf
);

SELECT COUNT(*) FROM Produkt;


 count
-------
    12
```

```
(1 row)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-1/Teilaufgabe-1/Aufgabe-4.tex

Examensaufgabe „Kundenverwaltungssystem“ (46116-2018-H.T1-TA2-A4)

CREATE TABLE

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

```
Kunde (ID, Vorname, Nachname, PLZ)
Produkt (GTIN, Bezeichnung, Bruttopreis, MWStSatz)
Kauf (ID[Kunde], GTIN[Produkt], Datum, Menge)
```

(a) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller *Kunden* mit der Postleitzahl *20251* ausgibt, *absteigend* sortiert nach *Nachname* und bei gleichen Nachnamen absteigend nach *Vorname*.

ASC (ascending) = aufsteigend DESC (descending) = absteigend

Lösungsvorschlag

```sql
SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname, Vorname DESC;
```

(b) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung aller Produkte ausgibt, deren Bruttopreis größer ist als 10 €.

Lösungsvorschlag

```sql
SELECT Bezeichnung
FROM Produkt
WHERE Bruttopreis > 10;
```

(c) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Kauf“ anlegt. Gehen Sie davon aus, dass die Tabellen „Kunde“ und „Produkt“ bereits existieren.

Lösungsvorschlag

```sql
CREATE TABLE Kauf (
  ID INTEGER REFERENCES Kunde(ID),
  GTIN INTEGER REFERENCES Produkt(GTIN),
  Datum DATE,
  Menge INTEGER,
  PRIMARY KEY (ID, GTIN, Datum)
);
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-1/Teilaufgabe-2/Aufgabe-4.tex

Examensaufgabe „Schuldatenbank“ (46116-2018-H.T2-TA2-A3)

SQL
CREATE TABLE

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

Schüler:

SchülerID	**Vorname**	**Nachname**	**Klasse**
1	Laura	Müller	4A
2	Linus	Schmidt	4A
3	Jonas	Schneider	4A
4	Liam	Fischer	4B
5	Tim	Weber	4B
6	Lea	Becker	4B
7	Emilia	Klein	4C
8	Julia	Wolf	4C

Noten:

SchülerID[Schüler]	**Schulfach**	**Note**	**Gewicht**	**Datum**
1	Mathematik	3	2	23.09.2017
1	Mathematik	1	1	03.10.2017
1	Mathematik	2	2	15.10.2017
1	Mathematik	4	1	11.11.2017

(a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Schüler (
  SchülerID INTEGER PRIMARY KEY NOT NULL,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20),
  Klasse VARCHAR(5)
);

CREATE TABLE IF NOT EXISTS Noten (
  SchülerID INTEGER NOT NULL,
```

```
  Schulfach VARCHAR(20),
  Note INTEGER,
  Gewicht INTEGER,
  Datum DATE,
  PRIMARY KEY (SchülerID, Schulfach, Datum),
  FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);
```

(b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägungen) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

```
INSERT INTO Schüler
  (SchülerID, Vorname, Nachname, Klasse)
VALUES
  (6, 'Johannes', 'Schmied', '4C');
```

Lösungsvorschlag

Nein. Ein/e Schüler/in mit der ID 6 existiert bereits. Primärschlüssel müssen eindeutig sein.

```
INSERT INTO Noten VALUES (6, 'Chemie', 1, 2, '1.4.2020');
```

Lösungsvorschlag

Nein. Ein *Datum* ist zwingend notwendig. Da *Datum* im Primärschlüssel enthalten ist, darf es nicht NULL sein. Es gibt auch keine/n Schüler/in mit der ID 9. Der/die Schüler/in müsste vorher angelegt werden, da die Spalte *SchülerID* von der Tabelle *Noten* auf den Fremdschlüssel *SchülerId* aus der Schülertabelle verweist.

(c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbankzustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.

  Lösungsvorschlag

  ```
  ALTER TABLE Schüler
  ADD CONSTRAINT eindeutiger_Vorname UNIQUE (Vorname, Klasse);
  ```

- Der Schüler *Tim Weber* (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse *„4C“* überführt.

Lösungsvorschlag

```
UPDATE Schüler
SET Klasse = '4C'
WHERE
  Vorname = 'Tim' AND
  Nachname = 'Weber' AND
  SchülerID = 5;
```

- Die Schülerin *Laura Müller* (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie einen möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.

Lösungsvorschlag

Alle Noten von *Laura Müller* werden gelöscht, falls `ON DELETE CASCADE` gesetzt ist. Oder es müssen erst alle Fremdschlüsselverweise auf diese *SchülerID* in der Tabelle *Noten* gelöscht werden

```
DELETE FROM Noten
WHERE SchülerID = 1;
```

- Erstellen Sie eine View *„DurchschnittsNoten“*, die die folgenden Spalten beinhaltet: *Klasse*, *Schulfach*, *Durchschnittsnote*

  Hinweis: Beachten Sie die Gewichte der Noten.

Lösungsvorschlag

```
CREATE VIEW DurchschnittsNoten AS (
  (SELECT s.Klasse, n.Schulfach, (SUM(n.Note * n.Gewicht) /
↪  SUM(n.Gewicht)) AS Durchschnittsnote
  FROM Noten n, Schüler s
  WHERE s.SchülerID = n.SchülerID
  GROUP BY s.Klasse, n.Schulfach)
);

SELECT * FROM DurchschnittsNoten;
```

```
 klasse | schulfach  | durchschnittsnote
--------+------------+-------------------
 4A     | Mathematik |                 2
(1 row)
```

- Geben Sie den Befehl an, der die komplette Tabelle *„Noten“* löscht.

Lösungsvorschlag

```
DROP TABLE Noten;
```

(d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass sie SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird.

  Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant

Lösungsvorschlag

```
SELECT AVG(Note)
FROM Noten
WHERE Schulfach = 'Mathematik';
```

- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (ðGewicht=2) geschrieben haben.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Schüler
FROM Noten
WHERE Datum = '23.09.2017' AND Gewicht = 2 AND Schulfach = 'Mathematik';
```

```
 anzahl_schüler
----------------
              1
(1 row)
```

- Geben Sie die *SchülerID* aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.

Lösungsvorschlag

```
SELECT SchülerID
FROM Noten
WHERE Schulfach = 'Mathematik' AND Note = 6
GROUP BY SchülerID
HAVING COUNT(*) >= 3;
```

```
 schülerid
-----------
(0 rows)
```

- Gesucht ist der Notendurchschnitt bezüglich jedes Fachs der Klasse „4A“.

Lösungsvorschlag

```
SELECT n.Schulfach, AVG(n.Note)
FROM Schüler s, Noten n
WHERE s.SchülerID = n.SchülerID AND s.Klasse = '4A'
GROUP BY n.Schulfach;
```

```
schulfach  |        avg
-----------+--------------------
 Mathematik | 2.5000000000000000
(1 row)
```

(e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

```
SELECT COUNT(DISTINCT Klasse)
FROM
Schüler NATURAL JOIN Noten;
```

Lösungsvorschlag

```
 count
-------
     1
(1 row)
```

4A von Laura Müller. Ohne `DISTINCT` wäre das Ergebnis 4.

```
SELECT COUNT(ALL Klasse)
FROM
Noten, Schüler;
```

Lösungsvorschlag

```
 count
-------
    32
(1 row)
```

Es entsteht das Kreuzprodukt $(8 \cdot 4 = 32)$.

```
SELECT COUNT(Note)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

```
SELECT * FROM Schüler NATURAL LEFT OUTER JOIN Noten;
```

ergibt:

```
 schülerid | vorname | nachname  | klasse | schulfach  | note | gewicht |   datum
-----------+---------+-----------+--------+------------+------+---------+------------
         1 | Laura   | Müller    | 4A     | Mathematik |    3 |       2 | 2017-09-23
         1 | Laura   | Müller    | 4A     | Mathematik |    1 |       1 | 2017-10-03
         1 | Laura   | Müller    | 4A     | Mathematik |    2 |       2 | 2017-10-15
         1 | Laura   | Müller    | 4A     | Mathematik |    4 |       1 | 2017-11-11
         2 | Linus   | Schmidt   | 4A     |            |      |         |
         5 | Tim     | Weber     | 4B     |            |      |         |
         8 | Julia   | Wolf      | 4C     |            |      |         |
         6 | Lea     | Becker    | 4B     |            |      |         |
         4 | Liam    | Fischer   | 4B     |            |      |         |
         3 | Jonas   | Schneider | 4A     |            |      |         |
         7 | Emilia  | Klein     | 4C     |            |      |         |
(11 rows)

 count
-------
     4
(1 row)
```

`COUNT` zählt die `{NULL}-Werte` nicht mit. Die *Laura Müller* hat 4 Noten.

```
SELECT COUNT(*)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

```
 count
-------
    11
(1 row)
```

Siehe Zwischenergebistabelle in der obenstehenden Antwort. Alle Schüler und die Laura 4-mal, weil sie 4 Noten hat.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-2/Teilaufgabe-2/Aufgabe-3.tex

Examensaufgabe „Mitarbeiterverwaltung“ (66113-2003-F.T1-A5)

SQL

Gegeben seien die folgenden drei Relationen. Diese Relationen erfassen die Mitarbeiterverwaltung eines Unternehmens. Schlüssel sind fett dargestellt und Fremdschlüssel sind kursiv dargestellt. So werden Mitarbeiter, Abteilungen und Unternehmen jeweils durch ihre Nummer identifiziert. AbtNr ist die Nummer der Abteilung, in der ein Mitarbeiter arbeitet. Manager ist die Nummer des Mitarbeiters, der die Abteilung leitet. UntNr ist die Nummer des Unternehmens, dem eine Abteilung zugeordnet ist.

```
Mitarbeiter(Nummer, Name, Alter, Gehalt, AbtNr)
Abteilung(Nummer, Name, Budget, Manager, UntNr)
Unternehmen(Nummer, Name, Adresse)
```

```
CREATE TABLE unternehmen (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) DEFAULT NULL,
  Adresse VARCHAR(50) DEFAULT NULL
);

CREATE TABLE abteilung (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) DEFAULT NULL,
  Budget float DEFAULT NULL,
  Manager VARCHAR(20) NOT NULL,
  UntNr integer DEFAULT NULL REFERENCES unternehmen (Nummer)
);

CREATE TABLE mitarbeiter (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) NOT NULL,
  Alter integer NOT NULL,
  Gehalt float NOT NULL,
  AbtNr integer NOT NULL REFERENCES abteilung (Nummer)
);

INSERT INTO unternehmen (Nummer, Name, Adresse) VALUES
  (1, 'Test.com', 'Alter Hafen 11'),
  (2, 'Party.de', 'Technostraße 3'),
  (3, 'IT.ch', 'Sequelweg 1');

INSERT INTO abteilung (Nummer, Name, Budget, Manager, UntNr) VALUES
  (1, 'Personal_Care', 20000, 'Huber', 1),
  (11, 'Tequilla_Mix', 50000, 'Taylor', 2),
  (21, 'Nerds', 500, 'Gates', 3);

INSERT INTO mitarbeiter (Nummer, Name, Alter, Gehalt, AbtNr) VALUES
  (1, 'Müller', 30, 30000, 1),
  (2, 'Huber', 45, 80000, 1),
  (3, 'Habermeier', 62, 40000, 1),
  (4, 'Leifsson', 27, 50000, 1),
  (5, 'Taylor', 37, 85000, 11),
  (6, 'Smith', 61, 34000, 11),
  (7, 'Pitt', 36, 40000, 11),
  (8, 'Thompson', 54, 52000, 11),
```

```
(9, 'Gates', 69, 15000000, 21),
(10, 'Zuckerberg', 36, 10000000, 21),
(11, 'Jobs', 99, 14000000, 21),
(12, 'Nakamoto', 66, 5000000, 21);
```

(a) Wie hoch ist das Durchschnittsalter der Abteilung „Personal Care“ im Unternehmen „Test.com“?

Lösungsvorschlag

`GROUP BY` nicht nötig, `AS` nicht vergessen.

```
SELECT AVG(m.Alter) AS Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
  a.Name = 'Personal Care' AND
  u.Name = 'Test.com' AND
  u.Nummer = a.UntNr AND
  m.AbtNr = a.Nummer;
```

(b) Geben Sie für jedes Unternehmen das Durchschnittsalter der Mitarbeiter an!

Lösungsvorschlag

Statt `a.UntNr` kann `u.Nummer` verwendet werden. `a.UntNr` nur deshalb, weil man dann eventuell den Join über die Unternehmenstabelle sparen kann.

Alles was ausgegeben werden soll, muss auch in `GROUP BY` enthalten sein.

```
SELECT a.UntNr, u.Name, AVG(m.Alter) as Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
  u.Nummer = a.UntNr AND
  m.AbtNr = a.Nummer
GROUP BY a.UntNr, u.Name;
```

(c) Wie viele Mitarbeiter im Unternehmen „Test.com“ sind älter als ihr Chef? (D.h. sind älter als der Manager der Abteilung, in der sie arbeiten.)

Lösungsvorschlag

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com'
AND m.Alter > (
  SELECT ma.Alter
  FROM Mitarbeiter ma, Abteilung ab
  WHERE
    ma.Nummer = ab.Manager AND
    a.Nummer = ab.Nummer
);
```

oder einfacher:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com'
AND m.Alter > (
  SELECT ma.Alter
  FROM Mitarbeiter ma
  WHERE ma.Nummer = a.Manager
);
```

Alternativ Lösung ohne Unterabfragem, mit Self join:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u, Mitarbeiter m2
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com' AND
  a.Manager = m2.Nummer AND
  m.Alter > m2.Atler;
```

(d) Welche Abteilungen haben ein geringeres Budget als die Summe der Gehälter der Mitarbeiter, die in der Abteilung arbeiten?

Lösungsvorschlag

```
SELECT a.Name, a.Nummer
FROM Abteilung a
WHERE a.Budget < (
  SELECT SUM(m.Gehalt)
  FROM Mitarbeiter m
  WHERE a.Nummer = m.AbtNr
);
```

Ohne Unterabfrage

```
SELECT a.Name, a.Nummer
FROM Abteilung a, Mitarbeiter m
WHERE a.Nummer = m.AbtNr
GROUP BY a.Nummer, a.Name, a.Budget
HAVING a.Budget < SUM(m.Gehalt);
```

(e) Versetzen Sie den Mitarbeiter „Wagner“ in die Abteilung „Personal Care“!

Lösungsvorschlag

```
UPDATE Mitarbeiter m
SET AbtNr = (
  SELECT a.Nummer FROM
  Abteilung a
  WHERE a.Name = 'Personal Care'
```

```
)
WHERE m.Name = 'Wagner';
```

(f) Löschen Sie die Abteilung „Personal Care“ mit allen ihren Mitarbeitern!

Lösungsvorschlag

```
DELETE FROM Mitarbeiter
WHERE AbtNr = (
  SELECT a.Nummer
  FROM Abteilung a
  WHERE a.Name = 'Personal Care'
);

DELETE FROM Abteilung
WHERE Name = 'Personal Care';
```

(g) Geben Sie den Managern aller Abteilungen, die ihr Budget nicht überziehen, eine 10 Prozent Gehaltserhöhung. (Das Budget ist überzogen, wenn die Gehälter der Mitarbeiter höher sind als das Budget der Abteilung.) Zusatzfrage: Was passiert mit Mitarbeitern, die Manager von mehreren Abteilungen sind?

Lösungsvorschlag

```
CREATE VIEW LowBudget AS (
  SELECT Nummer
  FROM Abteilung
  WHERE Nummer NOT IN (
    SELECT a.Nummer
    FROM Abteilung a
    WHERE
      a.Budget < (
        SELECT SUM(Gehalt)
        FROM Mitarbeiter m Abteilung A
        WHERE m.AbtNr = A.Nummer AND
        a.Nummer = A.Nummer
    )
  )
)

UPDATE Mitarbeiter
SET Gehalt = 1.1 * Gehalt
WHERE Nummer IN (
  SELECT Manager
  FROM LowBudget, Abteilung
  WHERE LowBudget.Manager = Abteilung.Nummer
)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66113/2003/03/Thema-1/Aufgabe-5.tex

Examensaufgabe „Universitätsverwaltung“ (66113-2003-F.T2-A3)

SQL

Gegeben sei folgendes relationales Schema, das eine Universitätsverwaltung modelliert:

```
Studenten {[MatrNr:integer, Name:string, Semester:integer]}
Vorlesungen {[VorlNr:integer, Titel:string, SWS:integer, gelesenVon:integer]}
Professoren {[PersNr:integer, Name:string, Rang:string, Raum:integer] }
hoeren {[MatrNr:integer, VorlNr:integer]}
voraussetzen {[VorgaengerVorlNr:integer, NachfolgerVorlNr:integer]}
pruefen {[MatrNr:integer, VorlNr:integer, PrueferPersNr:integer, Note:decimal]}
```

Formulieren Sie die folgenden Anfragen in SQL:

(a) Alle Studenten, die den Professor *Kant* aus einer Vorlesung kennen.

Lösungsvorschlag

```
SELECT DISTINCT s.Name
FROM Studenten s, Professoren p, hoeren h, Vorlesungen v
WHERE
  p.Name = 'Kant' AND
  v.gelesenVon = p.PersNr AND
  s.MatrNr = h.MatrNr AND
  h.VorlNr = h.VorlNr
```

(b) Geben Sie eine Liste der Professoren (Name, PersNr) mit ihrem Lehrdeputat (Summe der SWS der gelesenen Vorlesungen) aus. Ordnen Sie diese Liste so, dass sie absteigend nach Lehrdeputat sortiert ist! Bei gleicher Lehrtätigkeit dann noch aufsteigend nach dem Namen des Professors/der Professorin.

Lösungsvorschlag

```
SELECT p.Name, p.PersNr, SUM(v.SWS) AS Lehrdeputat
FROM Vorlesung v, Professoren p
WHERE v.gelesenVon = p.PersNr
GROUP BY p.Name, p.PersNr
ORDER BY Lehrdeputat DESC, p.Name ASC;
```

(c) Geben Sie eine Liste der Studenten (Name, MatrNr, Semester) aus, die mindestens zwei Vorlesungen bei *Kant* gehört haben.

Lösungsvorschlag

Mit einer VIEW

```
CREATE VIEW hoertKant AS
  SELECT s.Name, s.MatrNr, s.Semester, v.VorlNr
  FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
  WHERE
    s.MatrNr = h.MatrNr AND
    h.VorlNr = v.VorlNr AND
    v.gelesenVon = p.PersNr AND
    p.Name = 'Kant';
```

```sql
SELECT DISTINCT h1.Name, h2.MatrNr, h1.Semester
FROM hoertKant h1, hoertKant h2
WHERE h1.MatrNr = h2.MatrNr AND h1.VorlNr <> h2.VorlNr;
```

oder:

```sql
SELECT DISTINCT Name, MatrNr, Semester
FROM hoertKant
GROUP BY Name, MatrNr, Semester
HAVING COUNT(VorlNr) > 1;
```

In einer Abfrage

```sql
SELECT s.Name, s.MatrNr, s.Semester
FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
WHERE
  s.MatrNr = h.MatrNr AND
  h.VorlNr = v.VorlNr AND
  v.gelesenVon = p.PersNr AND
  p.Name = 'Kant'
GROUP BY s.MatrNr, s.Name, s.Semster
HAVING COUNT(s.MatrNr) > 1;
```

(d) Geben Sie eine Liste der Semesterbesten (MatrNr und Notendurchschnitt) aus.

Lösungsvorschlag

```sql
CREATE VIEW Notenschnitte AS (
  SELECT p.MatrNr, s.Name, s.Semester, AVG(Note) AS Durchschnitt
  FROM Studenten s, pruefen p
  WHERE s.MatrNr = p.MatrNr
  GROUP BY p.MatrNr, s.Name, s.Semester
);

SELECT a.Durchschnitt, a.MatrNr, a.Semester
FROM Notenschnitte a, Notenschnitte b
WHERE
  a.Durchschnitt >= b.Durchschnitt
  a.Semster = b.Semster
GROUP BY a.Durchschnitt, a.MatrNr, a.Semester
HAVING COUNT(*) < 2;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66113/2003/03/Thema-2/Aufgabe-3.tex

Examensaufgabe „Gebrauchtwagen“ (66116-2012-F.T1-TA1-A3)

SQL

Gegeben sei das folgende Relationenschema:

```
Fahrzeug : {[ MNR[Modell], FZGNR, Baujahr, KMStand, Preis ]}

Modell : {[ MNR, HNR[Hersteller], Typ, Neupreis, ps ]}

Hersteller : {[ HNR, Name ]}
```

Dabei sind die Schlüsselattribute jeweils unterstrichen und zusätzlich für alle Attribute die Typen angegeben. Formulieren Sie die folgenden Anfragen bzw. Anweisungen in SQL.

(a) Geben Sie die Anweisungen in SQL-DDL an, die notwendig sind, um die Relationen „Fahrzeug“, „Modell“ und „Hersteller“ zu erzeugen. Achten Sie dabei darauf, die Primärschlüssel der Relationen zu kennzeichnen.

Lösungsvorschlag

```sql
CREATE TABLE IF NOT EXISTS Hersteller (
  HNR INTEGER PRIMARY KEY,
  Name CHAR(20)
);

CREATE TABLE IF NOT EXISTS Modell (
  MNR INTEGER PRIMARY KEY,
  HNR INTEGER REFERENCES Hersteller(HNR),
  Typ CHAR(20),
  Neupreis INTEGER,
  ps INTEGER
);

CREATE TABLE IF NOT EXISTS Fahrzeug (
  MNR INTEGER REFERENCES Modell(MNR),
  FZGNR CHAR(12) PRIMARY KEY,
  Baujahr INTEGER,
  KMStand INTEGER,
  Preis INTEGER
);
```

(b) Bestimmen Sie die Typen aller Modelle des Herstellers mit Namen BMW.

Lösungsvorschlag

```sql
SELECT m.Typ
FROM Modell m, Hersteller h
WHERE h.HNR = m.HNR AND h.Name = 'BMW'
GROUP BY m.Typ;
```

(c) Bestimmen Sie den Mindestpreis, bezogen auf das Attribut „Preis“, der Fahrzeuge eines jeden Herstellers.

Lösungsvorschlag

```
SELECT h1.Name AS Hersteller, (
  SELECT MIN(f.Preis)
  FROM Fahrzeug f, Modell m, Hersteller h2
  WHERE
    f.MNR = m.MNR AND
    m.HNR = h2.HNR AND
    H2.HNR = h1.HNR
) AS Mindestpreis
FROM Hersteller h1;
```

(d) Bestimmen Sie die Namen der Hersteller, für die von jedem ihrer Modelle mindestens ein Fahrzeug in der Datenbank gespeichert ist.

Lösungsvorschlag

```
SELECT h.Name AS Hersteller
FROM Fahrzeug f, Modell m, Hersteller h
WHERE
  f.MNR = m.MNR AND
  m.HNR = h.HNR
GROUP BY h.Name;
```

(e) Bestimmen Sie die Namen aller Hersteller, von denen mindestens fünf Fahrzeuge eines beliebigen Modells in der Datenbank gespeichert sind.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2012/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex

Examensaufgabe „Musik-CDs“ (66116-2015-F.T1-TA1-A2)

SQL
GROUP BY
MAX
OR
DISTINCT

Formulieren Sie in SQL die folgenden Anfragen, Views bzw. Datenmanipulations-Statements an Teile der Musik-Datenbank aus Teilaufgabe DB.1:

Interpret (Interpreten_ID, Name, Bühnenstart, Geschäftsadresse), CD (CD_ID, Name, Interpreten_ID, Erscheinungsdatum), Musikstück (CD_ID, Position, Titel, Länge), Auszeichnung_CD (CD_ID, Typ),

Auszeichnung_Stück (CD_ID, Position, Typ).

(a) Welche CDs hat „Adele“ herausgebracht? Geben Sie die Namen der CDs aus.

Lösungsvorschlag

```
SELECT c.Name DISTINICT
FROM CD c, Interpret i
WHERE
  i.Interpreten_ID = c.Interpreten_ID AND
  i.name = 'Adele';
```

(b) Geben Sie für alle Interpreten - gegeben durch die ID und den Namen - die Anzahl ihrer veröffentlichten CDs an.

Lösungsvorschlag

```
SELECT i.Interpreten_ID, i.Name, COUNT(*)
FROM Interpret i, CD c
WHERE
  i.Interpreten_ID = c.Interpreten_ID
GROUP BY i.Interpreten_ID, i.Name;
```

(c) Geben Sie die Länge des längsten Musikstücks auf der CD mit dem Namen „Thriller“ des Interpreten „Michael Jackson“ an.

Lösungsvorschlag

```
SELECT MAX(m.Länge)
FROM Interpret i, CD c, Musikstück m
WHERE
  m.CD_ID = c.CD_ID AND
  i.Name = 'Michael Jackson' AND
  c.Name = 'Thriller';
```

(d) Geben Sie die Namen aller Interpreten aus, die eine Auszeichnung für eine CD oder eines ihrer Musikstücke bekommen haben.

Lösungsvorschlag

```
SELECT i.Name
FROM Auszeichnung_CD acd, Auszeichnung_Stück ast, CD c, Interpret i
WHERE
  (acd.CD_ID = c.CD_ID OR ast.CD_ID = c.CD_ID) AND
  c.Interpreten_ID = i.Interpreten_ID;
```

(e) Fügen Sie ein, dass „Adele“ einen „Emmy“ für ihre CD mit dem Namen „Adele 21“ bekommen hat.

Lösungsvorschlag UPDATE

```
INSERT INTO Auszeichnung_CD VALUES
  (
    (
      SELECT c.CD_ID
      FROM CD c, Interpret i
      WHERE
        c.Name = 'Adele 21' AND
        i.Interpreten_ID = c.Interpreten_ID AND
        i.Name = 'Adele'
    ),
    'Emmy'
  );

-- Test
SELECT * from Auszeichnung_CD;
```

(f) Ändern Sie die Geschäftsadresse von „Genesis“ auf „Hollywood Boulevard 13, Los Angeles“.

Lösungsvorschlag

```
-- Test
SELECT * FROM Interpret;

UPDATE Interpret
SET Geschäftsadresse = 'Hollywood Boulevard 13, Los Angeles'
WHERE Name = 'Gensis';

-- Test
SELECT * FROM Interpret;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex

Examensaufgabe „Vater und Mutter“ (66116-2015-H.T1-TA1-A3)

SQL

Gegeben seien folgende Relationen:

`Mensch : {[ ID, MutterID, VaterID ]}`

`Mann : {[ ID ]}`

`Frau : {[ ID ]}`

Das zugehörige ER-Modell für dieses relationale Datenbankschema sieht folgendermaßen aus:

Bearbeiten Sie folgende Teilaufgaben:

(a) Finden Sie die Töchter der Frau mit ID 42.

Lösungsvorschlag

```sql
SELECT Mensch.ID
FROM Mensch, Frau
WHERE
  Mensch.MutterID = Frau.id AND
  Frau.ID = 42;
```

(b) Gibt es Männer, die ihre eigenen Großväter sind? Formulieren Sie eine geeignete SQL-Anfrage.

Lösungsvorschlag

```sql
SELECT Mensch.ID
FROM Mann, Mensch
WHERE
  Mensch.ID = Mann.id AND (
    Mensch.VaterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID = Mensch.ID)
    OR
    Mensch.MutterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID =
↪   Mensch.ID)
  );
```

(c) Definieren Sie eine View VaterKind (VaterID; KindID), die allen Vätern (VaterID) ihre Kinder (KinderID) zuordnet. Diese View darf keine NULL-Werte enthalten.

Lösungsvorschlag

```sql
-- Wir erzeuge bereits beim Erstellen der Datenbank diese View, damit
-- sie für spätere Aufgaben zur Verfügung steht.
DROP VIEW IF EXISTS VaterKind;
CREATE VIEW VaterKind AS
SELECT Mensch.VaterID, Mensch.ID as KindID
FROM Mensch
WHERE
  Mensch.VaterID IS NOT NULL;
SELECT * FROM VaterKind;
```

(d) Verwenden Sie die View aus c), um alle Väter zurückzugeben, absteigend geordnet nach der Anzahl ihrer Kinder.

Lösungsvorschlag

```sql
SELECT VaterID, COUNT(VaterID) as Anzahl
FROM VaterKind
GROUP BY VaterID
ORDER BY Anzahl DESC;
```

(e) Hugo möchte mit folgender Anfrage auf Basis der View aus c) alle kinderlosen Männer erhalten:

```sql
SELECT VaterID
FROM VaterKind
GROUP BY VaterID
HAVING COUNT(KindID) = 0
```

(i) Was ist das Ergebnis von Hugos Anfrage und warum?

Lösungsvorschlag

Die Anfrage liefert kein Ergebnis. Da die View laut Angabe keine Null-Werte enthalten darf, sind in der View nur Männer verzeichnet, die wirklich Väter sind.

(ii) Formulieren Sie eine Anfrage, die tatsächlich alle kinderlosen Männer zurückliefert.

Lösungsvorschlag

```sql
SELECT * FROM Mann
EXCEPT
SELECT VaterID
FROM VaterKind
GROUP BY VaterID;
```

Hinweis: Denken Sie daran, dass SQL auch Mengenoperationen kennt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex

Examensaufgabe „Personalverwaltung“ (66116-2016-H.T1-TA1-A4)

SQL

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter : {[ MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter], Abteilungs
ID[Abteilung], Telefonnummer, Gehalt ]}

Abteilung : {[ AbteilungsID, Bezeichnung ]}
```

(a) Schreiben Sie eine SQL-Anfrage, die *Vor-* und *Nachnamen* der *Mitarbeiter* aller *Abteilungen* mit der Bezeichnung *„Buchhaltung“* ausgibt, absteigend sortiert nach *Mitarbeiter-ID*.

Lösungsvorschlag

```
SELECT Vorname, Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
  m.AbteilungsID = a.AbteilungsID AND
  a.Bezeichnung = 'Buchhaltung'
ORDER BY m.MitarbeiterID DESC;


 vorname | nachname
---------+----------
 Till    | Fuchs
 Hans    | Meier
(2 rows)
```

(b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

Lösungsvorschlag

```
SELECT m.Nachname AS Mitarbeiter, v.Nachname AS Vorgesetzter
FROM Mitarbeiter m LEFT OUTER JOIN Mitarbeiter v
ON m.Vorgesetzter = v.MitarbeiterID;


 mitarbeiter | vorgesetzter
-------------+--------------
 Meier       | Müller
 Wolitz      | Müller
 Müller      |
 Fuchs       | Wolitz
 Hase        | Müller
 Navratil    |
 Schmidt     | Navratil
(7 rows)
```

(c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung. Gehen Sie davon dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch keine datenbanksystemspezifischen Erweiterungen wie `limit` oder `rownum`. VIEW WITH

Lösungsvorschlag

```
CREATE VIEW Durchschnittsgehälter AS
SELECT Abteilung.AbteilungsID, Bezeichnung,
  AVG (Gehalt) AS Durchschnittsgehalt
FROM Mitarbeiter, Abteilung
WHERE Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID
GROUP BY Abteilung.AbteilungsID, Bezeichnung;

SELECT a.Bezeichnung, a.Durchschnittsgehalt, COUNT (*) AS Rang
FROM Durchschnittsgehälter a, Durchschnittsgehälter b
WHERE a.Durchschnittsgehalt <= b.Durchschnittsgehalt
GROUP BY a.AbteilungsID, a.Bezeichnung, a.Durchschnittsgehalt
HAVING COUNT(*) <= 10
ORDER BY Rang ASC;


 bezeichnung | durchschnittsgehalt | rang
-------------+---------------------+------
 Managment   |              6514.5 |    1
 Buchhaltung |                2340 |    2
 Vertrieb    |              1283.5 |    3
 Produktion  |                 654 |    4
(4 rows)
```

(d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit der AbteilungsID 42 um 5% erhöht.

Lösungsvorschlag

```
 vorname | nachname | gehalt
---------+----------+--------
 Lea     | Müller   |   5875
 Gerd    | Navratil |   7154
(2 rows)

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;

UPDATE Mitarbeiter
SET Gehalt = 1.05 * Gehalt
WHERE AbteilungsID = 42;
```

DELETE

```
SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;

 vorname | nachname |      gehalt
---------+----------+-------------------
 Lea     | Müller   |           6168.75
 Gerd    | Navratil | 7511.700000000001
(2 rows)
```

(e) Alle *Abteilungen* mit Bezeichnung *„Qualitätskontrolle“* sollen zusammen mit den Datensätzen ihrer *Mitarbeiter* gelöscht werden. `ON DELETE CASCADE` ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

Lösungsvorschlag

```
 vorname | nachname
---------+----------
 Hans    | Meier
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Fred    | Hase
 Gerd    | Navratil
 Jürgen  | Schmidt
(7 rows)

 abteilungsid |    bezeichnung
--------------+--------------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
            4 | Qualitätskontrolle
            5 | Produktion
(5 rows)
```

```
SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

DELETE FROM Mitarbeiter
WHERE AbteilungsID IN (
  SELECT a.AbteilungsID
  FROM Abteilung a
  WHERE a.Bezeichnung = 'Qualitätskontrolle'
);

DELETE FROM Abteilung
WHERE Bezeichnung = 'Qualitätskontrolle';
```

```
SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

 vorname | nachname
---------+----------
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Gerd    | Navratil
 Jürgen  | Schmidt
(5 rows)

 abteilungsid | bezeichnung
--------------+-------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
            5 | Produktion
(4 rows)
```

(f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die Tabellen als abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Lösungsvorschlag

Wir könnten eine VIEW erstellen, die zwar Namen und ID der anderen Mitarbeiter, sowie ihre Telefonnummern enthält (evtl. auch Abteilungsbezeichnung und ID), aber eben nicht das Gehalt: Mitarbeiter arbeiten auf eingeschränkter Sicht.

Alternativ mit `GRANT`:

explizit mit `SELECT` die Spalten auswählen, die man lesen können soll (auf nicht angegebene Spalten ist kein Zugriff möglich)

```
GRANT SELECT (Vorname, Nachname, Telefonnummer)
ON Mitarbeiter TO postgres;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-1/Teilaufgabe-1/Aufgabe-4.tex

Examensaufgabe „Schulverwaltung“ (66116-2016-H.T2-TA1-A2)

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

```
Person : {[
  ID : INTEGER,
  Name : VARCHAR(255),
  Wohnort : VARCHAR(255),
  Typ : CHAR(1)
]}

Unterricht : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Lehrer : INTEGER,
  Fach : VARCHAR(100)
]}

Klasse : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Klassenlehrer : INTEGER
]}

Klassenverband : {[
  Schüler : INTEGER,
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER
]}
```

```sql
CREATE TABLE Person (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255),
  Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);

CREATE TABLE Klasse (
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Klassenlehrer INTEGER REFERENCES Person(ID),
  PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);

CREATE TABLE Klassenverband (
  Schüler INTEGER REFERENCES Person(ID),
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  PRIMARY KEY (Schüler, Schuljahr)
);
```

SQL mit Übungsdatenbank
SQL
CREATE TABLE

```
CREATE TABLE Unterricht (
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Lehrer INTEGER REFERENCES Person(ID),
  Fach VARCHAR(100),
  CONSTRAINT Unterricht_PK
    PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);

INSERT INTO Person VALUES
  (1, 'Lehrer Ludwig', 'München', 'L'),
  (2, 'Schüler Max', 'München', 'S'),
  (3, 'Schülerin Maria', 'München', 'S'),
  (4, 'Schülerin Eva', 'Starnberg', 'S'),
  (5, 'Lehrerin Walter', 'München', 'L'),
  (6, 'Schüler Karl', 'München', 'S');

INSERT INTO Klasse VALUES
  ('1a', 2015, 1),
  ('1a', 2014, 1),
  ('1b', 2015, 5);

INSERT INTO Klassenverband VALUES
  (2, '1a', 2015),
  (3, '1a', 2015),
  (4, '1b', 2015),
  (6, '1a', 2015),
  (6, '1a', 2014);
```

Hierbei enthält die Tabelle *Person* Informationen über Lehrer (Typ 'L') und Schüler (Typ 'S'); andere Werte für Typ sind nicht zulässig. *Klasse* beschreibt die Klassen, die in jedem Schuljahr gebildet wurden, zusammen mit ihrem Klassenlehrer. In *Unterricht* wird abgelegt, welcher Lehrer welches Fach in welcher Klasse unterrichtet; es ist möglich, dass derselbe Lehrer mehr als ein Fach in einer Klasse unterrichtet. *Klassenverband* beschreibt die Zuordnung der Schüler zu den Klassen.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Unterricht* mit allen ihren Constraints (einschließlich Fremdschlüsselconstraints) anlegt.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Person(
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255),
  Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);

CREATE TABLE IF NOT EXISTS Klasse(
  Klassenbezeichnung VARCHAR(20),
  Schuljahr INTEGER,
  Klassenlehrer INTEGER REFERENCES Person(ID),
  PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);
```

```
CREATE TABLE IF NOT EXISTS Unterricht (
  Klassenbezeichnung VARCHAR(20) REFERENCES Klasse(Klassenbezeichnung),
  Schuljahr INTEGER REFERENCES Klasse(Schuljahr),
  Lehrer INTEGER REFERENCES Person(ID),
  Fach VARCHAR(100),
  CONSTRAINT Unterricht_PK
    PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);
```

(b) Definieren Sie ein geeignetes Constraint, das sicherstellt, dass nur zulässige Werte im Attribut Typ der (bereits angelegten) Tabelle *Person* eingefügt werden können.

Lösungsvorschlag

Ich habe `REFERENCES` bei Unterricht Schuljahr vergessen, die referenzierten Tabellen `Person` und `Klasse` wurden in der Musterlösung auch nicht angelegt.

```
ALTER TABLE Person
  ADD CONSTRAINT TypLS
    CHECK(Typ IN ('S', 'L'));
```

(c) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung der Klassen bestimmt, die im Schuljahr 2015 die meisten Schüler haben.

Lösungsvorschlag

Falsch: `ORDER BY Anzahl;`. `DESC` vergessen.

```
SELECT k.Klassenbezeichnung, COUNT(*) AS Anzahl
FROM Klasse k, Klassenverband v
WHERE
  k.Schuljahr = 2015 AND
  k.Klassenbezeichnung = v.Klassenbezeichnung
GROUP BY k.Klassenbezeichnung
ORDER BY COUNT(*) DESC;
```

(d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lehrer bestimmt, die nur Schüler aus ihrem Wohnort unterrichtet haben.

Lösungsvorschlag

```
SELECT DISTINCT l.Name
FROM Person l
WHERE NOT EXISTS(
  SELECT DISTINCT *
  FROM Unterricht u, Klassenverband v, Person s
  WHERE
    u.Lehrer = l.ID AND
    u.Klassenbezeichnung = v.Klassenbezeichnung AND
    v.Schüler = s.ID AND
    l.Wohnort != s.Wohnort
);
```

(e) Schreiben Sie eine SQL-Anweisung, die die Namen aller Schüler bestimmt, die immer den gleichen Klassenlehrer hatten. HAVING

Lösungsvorschlag

```
SELECT s.Name
FROM Person s, Klasse k, Klassenverband v
WHERE
  v.Klassenbezeichnung = k.Klassenbezeichnung AND
  k.Schuljahr = v.Schuljahr AND
  v.Schüler = s.ID
GROUP BY s.ID, s.Name
HAVING COUNT(*) = 1
```

(f) Schreiben Sie eine SQL-Anweisung, die alle Paare von Schülern bestimmt, die mindestens einmal in der gleichen Klasse waren. Es genügt dabei, wenn Sie die ID der Schüler bestimmen.

Lösungsvorschlag

```
SELECT DISTINCT s1.ID, s2.ID
FROM Klassenverband s1, Klassenverband s2
WHERE
  s1.Schuljahr = s2.Schuljahr AND
  s1.Schueler <> s2.Schueler AND
  s1.Klassenbezeichung = s2.Klassenbezeichnung;
```

(g) Formulieren Sie eine Anfrage in der relationalen Algebra, die die ID aller Schüler bestimmt, die mindestens einmal von „Ludwig Lehrer“ unterrichtet wurden.

Lösungsvorschlag

$$\pi_{\text{Schüler}}\Big(\sigma_{\text{Name}=\text{'Ludwig Lehrer'}}(\text{Person}) \bowtie_{\text{Person.ID}=\text{Unterricht.Lehrer}} \text{Unterricht} \bowtie_{\text{Unterricht.Klassenbezeichnung}=\text{Klassenverband.Klassenbezeichnung}} \text{Klassenverband}\Big)$$

(h) Formulieren Sie eine Anfrage in der relationalen Algebra, die Namen und ID der Schüler bestimmt, die von allen Lehrern unterrichtet wurden.

Lösungsvorschlag

$$\pi_{\text{Name,ID}}\Big(\Big(\pi_{\text{Lehrer,Schueler}}(\text{Unterricht} \bowtie \text{Klassenverband}) \div \pi_{\text{ID}}(\sigma_{\text{Typ}='\text{L}'}(\text{Person}))\Big) \bowtie \text{Person}\Big)$$

Beachten Sie bei der Formulierung der SQL-Anfragen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „SQL-Syntax-Überprüfung“ (66116-2017-H.T1-TA1-A5)

SQL
CREATE TABLE
DELETE
INSERT

Gegeben ist die folgende Definition zweier Tabellen:

```
CREATE TABLE R2 (
  b integer not null,
  c integer unique,
  primary key (b)
);

CREATE TABLE R1 (
  a integer not null,
  b integer references R2,
  primary key (a)
);
```

Geben Sie jeweils an, ob das Statement syntaktisch korrekt ist und ob es von der gegebenen Datenbank ausgeführt werden kann.

Beantworten Sie jede der folgenden Fragen unabhängig von allen anderen, ðes liegt immer das hier gezeigte Schema vor undd alle Relationen sind leer.

(a)
```
DELETE FROM R1;
```

Lösungsvorschlag

korrekt

(b)
```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R1 VALUES (1,1);
INSERT INTO R1 VALUES (2,1);
INSERT INTO R1 VALUES (3,1);
```

Lösungsvorschlag

korrekt

(c)
```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R2 VALUES (2,2);
INSERT INTO R1 VALUES (1,1);
DELETE FROM R2 WHERE b=a;
```

Lösungsvorschlag

falsch: Fehlermeldung `column ä"does not exist`

```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R2 VALUES (2,2);
INSERT INTO R1 VALUES (1,1);
-- Wir löschen von R1 weil R2 auf R1 referenziert
-- b kann nur mit Integer verglichen werden.
```

```
DELETE FROM R1 WHERE b=1;
```

DROP TABLE

(d)

```
INSERT INTO R1 SELECT * FROM R1;
```

Lösungsvorschlag

korrekt

(e)

```
DROP TABLE R2 FROM DATABASE;
```

Lösungsvorschlag

falsch: Fehlermeldung `ERROR: syntax error at or near "FROM"`

Müsste so lauten:

```
-- Zuerst R1 löschen, wegen der Referenz
DROP TABLE R1;
DROP TABLE R2;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/09/Thema-1/Teilaufgabe-1/Aufgabe-5.tex

Examensaufgabe „Fluginformationssystem“ (66116-2017-H.T1-TA1-A6)[SQL]

Folgende Tabellen veranschaulichen eine Ausprägung eines Fluginformationssystems:

Flughäfen

Code	Stadt	Transferzeit (min)
LHR	London	30
LGW	London	20
JFK	New York City	60
EWR	New York City	35
MUC	München	30
FRA	Frankfurt	45

Verbindungen

ID	Von	Nach	Linie	Abflug (MEZ)	Ankunft (MEZ)
410	MUC	FRA	LH	2016-02-24 07:00:00	2016-02-24 08:10:00
411	MUC	FRA	LH	2016-02-24 08:00:00	2016-02-24 09:10:00
412	FRA	JFK	LH	2016-02-24 10:50:00	2016-02-24 19:50:00

Hinweise

- Formulieren Sie alle Abfragen in SQL-92 (insbesondere sind LIMIT, TOP, FETCH FIRST, ROWNUM und dergleichen nicht erlaubt).
- Alle Datum/Zeit-Angaben erlauben arithmetische Operationen, beispielsweise wird bei der Operation `ankunf + transferzeit` die `transferzeit` auf den Zeitstempel `ankunft` addiert.
- Es müssen keine Zeitverschiebungen berücksichtigt werden. Alle Zeitstempel sind in MEZ.

```
CREATE TABLE Flughaefen (
  Code VARCHAR(3) PRIMARY KEY,
  Stadt VARCHAR(20),
  Transferzeit integer
);

CREATE TABLE Verbindungen (
  ID integer PRIMARY KEY,
  Von VARCHAR(3) REFERENCES Flughaefen(Code),
  Nach VARCHAR(3) REFERENCES Flughaefen(Code),
  Linie VARCHAR(20),
  Abflug timestamp,
  Ankunft timestamp
```

```
);

INSERT INTO Flughaefen VALUES
  ('LHR', 'London', 30),
  ('LGW', 'London', 20),
  ('JFK', 'New York City', 60),
  ('EWR', 'New York City', 35),
  ('MUC', 'München', 30),
  ('FRA', 'Frankfurt', 45);

INSERT INTO Verbindungen VALUES
  (410, 'MUC', 'FRA', 'LH', '2016-02-24 07:00:00', '2016-02-24 08:10:00'),
  (411, 'MUC', 'FRA', 'LH', '2016-02-24 08:00:00', '2016-02-24 09:10:00'),
  (412, 'FRA', 'JFK', 'LH', '2016-02-24 10:50:00', '2016-02-24 19:50:00'),
  (413, 'MUC', 'LHR', 'LH', '2016-02-24 10:00:00', '2016-02-24 12:10:00'),
  (414, 'MUC', 'LGW', 'LH', '2016-02-24 11:00:00', '2016-02-24 13:20:00'),
  (415, 'MUC', 'LHR', 'LH', '2016-02-24 12:00:00', '2016-02-24 14:00:00');
```

(a) Ermitteln Sie die Städte, in denen es mehr als einen Flughafen gibt.

Lösungsvorschlag

```
SELECT Stadt FROM Flughaefen
GROUP BY Stadt
HAVING count(Stadt) > 1;
```

(b) Ermitteln Sie die Städte, in denen man mit der Linie „LH“ an mindestens zwei verschiedenen Flughäfen landen kann.

Lösungsvorschlag

```
SELECT Stadt FROM Flughaefen
WHERE Stadt IN (
  SELECT Stadt FROM Flughaefen, Verbindungen
  WHERE
    Code = Nach AND
    Linie = 'LH'
  GROUP BY Stadt
)
GROUP BY Stadt
HAVING COUNT(Stadt) > 1;
```

(c) Ermitteln Sie die Flugzeit des kürzesten Direktflugs von München nach London.

Lösungsvorschlag

```
CREATE VIEW Flugdauer AS
  SELECT ID, Ankunft - Abflug AS Dauer FROM Flughaefen v, Flughaefen n,
↪  Verbindungen
  WHERE
    n.Code = Nach AND
    v.Code = Von AND
    v.Stadt = 'München' AND
    n.Stadt = 'London';

SELECT a.Dauer FROM Flugdauer a, Flugdauer b
```

```
WHERE a.Dauer >= b.Dauer
GROUP BY a.Dauer
HAVING COUNT(*) <= 1;
```

(d) Ermitteln Sie die kürzeste Roundtrip-Zeit (nur Direktflüge) zwischen den Flughäfen FRA und JFK (Transferzeit am Flughafen JFK beachten).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/09/Thema-1/Teilaufgabe-1/Aufgabe-6.tex

Examensaufgabe „Triathlon“ (66116-2018-H.T1-TA2-A4)

SQL mit Übungsdatenbank
SQL

Gegeben sind folgende Relationen aus einer Datenbank zur Verwaltung von Triathlon-Wettbewerben.

```
Athlet : {[ ID, Vorname, Nachname ]}

Ergebnis : {[ Athlet[Athlet], Wettbewerb[Wettbewerb], Schwimmzeit, Radzeit, Laufzeit ]}

Wettbewerb : {[ Name, Jahr ]}
```

```sql
CREATE TABLE Athlet (
  ID INTEGER PRIMARY KEY,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20)
);

CREATE TABLE Wettbewerb (
  Name VARCHAR(40) PRIMARY KEY,
  Jahr INTEGER
);

CREATE TABLE Ergebnis (
  Athlet INTEGER REFERENCES Athlet(ID),
  Wettbewerb VARCHAR(40) REFERENCES Wettbewerb(Name),
  Schwimmzeit INTEGER NOT NULL,
  Radzeit INTEGER,
  Laufzeit INTEGER,
  PRIMARY KEY (Athlet, Wettbewerb)
);

INSERT INTO Athlet VALUES
  (1, 'Boris', 'Stein'),
  (2, 'Trevor', 'Wurtele'),
  (3, 'Reichelt', 'Horst'),
  (12, 'Mitch', 'Kibby');

INSERT INTO Wettbewerb VALUES
  ('Zürichsee', 2018),
  ('Ironman Vichy', 2018),
  ('Challenge Walchsee', 2018),
  ('Triathlon Alpe d’Huez', 2017);

INSERT INTO Ergebnis VALUES
  (1, 'Zürichsee', 14, 10, 11),
  (2, 'Zürichsee', 13, 10, 11),
  (3, 'Zürichsee', 12, 10, 11),
  (12, 'Zürichsee', 11, 10, 11),
  (2, 'Challenge Walchsee', 12, 10, 11),
  (3, 'Challenge Walchsee', 11, 10, 11),
  (12, 'Triathlon Alpe d’Huez', 9, 10, 11);
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

UPDATE
DESC
AVG
GROUP BY
HAVING
EXCEPT

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Ergebnis“ anlegt. Gehen Sie davon aus, dass die Tabellen „Athlet“ und „Wettbewerb“ bereits existieren. Verwenden Sie sinnvolle Datentypen.

Lösungsvorschlag

```sql
CREATE TABLE IF NOT EXISTS Ergebnis (
  Athlet INTEGER REFERENCES Athlet(ID),
  Wettbewerb INTEGER REFERENCES Wettbewerb(Name),
  Schwimmzeit INTEGER NOT NULL,
  Radzeit INTEGER,
  Laufzeit INTEGER,
  PRIMARY KEY (Athlet, Wettbewerb)
);
```

(b) Schreiben Sie eine SQL-Anweisung, die die Radzeit des Teilnehmers mit der ID 12 beim Wettbewerb „Zürichsee“ um eins erhöht.

Lösungsvorschlag

```sql
-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

UPDATE Ergebnis
SET Radzeit = Radzeit + 1
WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';
```

(c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe des Jahres 2018 ausgibt, absteigend sortiert nach Name.

Lösungsvorschlag

```sql
SELECT Name
FROM Wettbewerb
WHERE Jahr = 2018
ORDER BY Name DESC;
```

(d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe ausgibt, in der die durchschnittliche Schwimmzeit größer als 10 ist.

Lösungsvorschlag

```sql
SELECT Wettbewerb, AVG(Schwimmzeit)
FROM Ergebnis
GROUP BY Wettbewerb
HAVING AVG(Schwimmzeit) > 10;
```

(e) Schreiben Sie eine SQL-Anweisung, die die IDs aller Athleten ausgibt, die im Jahr 2017 an keinem Wettbewerb teilgenommen haben.

Lösungsvorschlag

```
(SELECT DISTINCT Athlet FROM Ergebnis)
EXCEPT
(SELECT DISTINCT Athlet FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.name AND w.Jahr = 2017);
```

(f) Schreiben Sie eine SQL-Anweisung, die die Nachnamen aller Athleten ausgibt, die mindestens 10 Wettbewerbe gewonnen haben, das heißt im jeweiligen Wettbewerb die kürzeste Gesamtzeit erreicht haben. Die Gesamtzeit ist die Summe aus Schwimmzeit, Radzeit und Laufzeit. Falls zwei Athleten in einem Wettbewerb die gleiche Gesamtzeit erreichen, sind beide Sieger.

Lösungsvorschlag

vermutlich falsch

```
CREATE VIEW Gesamtzeiten AS
SELECT e.Athlet AS NameAthlet, e.Radzeit + e.Schwimmzeit + e.Laufzeit
AS Gesamtzeit, w.NameWettbewerb
FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.Name
CREATE VIEW Sieger AS
SELECT g1.NameAthlet
FROM Gesamtzeiten g1, Gesamtzeiten g2
GROUP BY g1.NameWettbewerb
HAVING g1.Gesamtzeit < g2.Gesamtzeit
SELECT NameAthlet
FROM Sieger
GROUP BY NameAthlet
HAVING count(*) > 10;
```

(g) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der Athleten mit der schnellsten Schwimmzeit des Wettbewerbs „Paris“ ausgibt. Ausgegeben werden sollen die Platzierung (1 bis 10) und der Nachname des Athleten, aufsteigend sortiert nach Platzierung. Gehen Sie davon aus, dass keine zwei Athleten die gleiche Schwimmzeit haben und verwenden Sie keine produktspezifischen Anweisungen wie beispielsweise rownum, top oder limit.

Lösungsvorschlag

```
CREATE VIEW AthletenParis AS
SELECT a.Nachname, e.Schwimmzeit
FROM Athlet a, Ergebnis e INNER JOIN Wettbewerb W ON e.Wettbewerb = w.Name
WHERE w.Name = "Paris" AND a.ID = e.Athlet
SELECT a.Nachname COUNT(*) + 1 AS Platzierung
FROM AthletenParis a, AthletenParis b
WHERE a.Schwimmzeit < b.Schwimmzeit
GROUP BY a.Nachname
HAVING Platzierung <= 10;
```

(h) Schreiben Sie einen Trigger, der beim Einfügen neuer Tupel in die Tabelle „Ergebnis“ die Schwimmzeit auf den Wert 0 setzt, falls diese negativ ist.

Lösungsvorschlag

```
CREATE TRIGGER update_Ergebnis AFTER UPDATE ON Ergebnis AS
IF(UPDATE Schwimmzeit AND Schwimmzeit < 0) BEGIN UPDATE Ergebnis
SET Schwimmzeit = 0
WHERE (Athlet, Wettbewerb) IN (SELECT DISTINCT (Athlet, Wettbewerb) FROM
↪  inserted)
END;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/09/Thema-1/Teilaufgabe-2/Aufgabe-4.tex

Examensaufgabe „App für konfizierte Schüler-Smartphones“ (66116-2019-F.T2-TA1-A5) SQL

Wir betrachten erneut das gegebene Schema aus Aufgabe 4. Primärschlüssel sind unterstrichen, Fremdschlüssel sind überstrichen und der Text in den darauf folgenden eckigen Klammern benennt die Relation, auf die verwiesen wird.

Der Notenschnitt, der Preis und die Bewertung werden als Kommazahl dargestellt, wobei die Bewertung die Anzahl der Sterne angibt, also maximal 5 und mindestens 0. Die Modellnummer kann sowohl aus Zahlen und Buchstaben bestehen, ist jedoch nie länger als 50 Zeichen. ID, RAM, Bildschirmdiagonale und Datum sind ganze Zahlen. Die restlichen Attribute sind Strings der Länge 15.

Lehrer (Name, Fachl, Fach2, Fach3)
Schüler (Vorname, Nachname, Notenschnitt)
Smartphone (ID, Modellnr, RAM, Bildschirmdiagonale)
App (Name, Bewertung, Preis)
Eingesammelt (Vorname [Schüler], Nachname [Schüler], Name [Lehrer], ID [Smartphone], Datum)
Installiert (ID [Smartphone], Name [App])

(a) Geben Sie die Anweisung in SOL-DDL an, die notwendig ist, um die Relation ’App’ zu erzeugen.

Lösungsvorschlag

```
CREATE TABLE App(
Name VARCHAR(50) PRIMARY KEY,
Bewertung VARCHAR(255),
Preis INTEGER);
```

(b) Geben Sie die Anweisung in SOL-DDL an, die notwendig ist, um die Relation ’Installiert’ zu erzeugen.

Lösungsvorschlag

```
CREATE TABLE Installiert(
ID INTEGER REFERENCES Smartphone(ID),
Name VARCHAR(50) App(Name),
PRIMARY KEY(ID,Name));
```

(c) Formulieren Sie die folgende Anfrage in SQL: Gesucht sind die Namen der Apps zusammen mit ihrer Bewertung, die auf den Smartphones installiert sind, die Lehrer Keating eingesammelt hat. Sortieren Sie das Ergebnis nach Bewertung absteigend. Hinweis: Achten Sie auf gleichnamige Attribute.

Lösungsvorschlag

```
SELECT DISTINCT a.Name, a.Bewertung
FROM Eingesammelt e, Installiert i, App a
WHERE e.ID =i.ID AND i.Name = a.Name AND e.Name = "Keating"
ORDER BY a.Bewertung DESC;
```

(d) Formulieren Sie die folgende Anfrage in SQL:

Gesucht ist der durchschnittliche Notenschnitt der Schüler, denen ein iPhone 65 abgenommen wurde. Ein iPhone 6s kann A1633 als Modellnummer haben oder A1688.

Lösungsvorschlag

```
SELECT DISTINCT AVG(s.Notenschnitt)
FROM Schüler s, Eingesammelt e, Smartphone sm
WHERE s.Vorname = e.Vorname AND s.Nachname = e.Nachname
AND e.ID=sm.ID AND(sm.Modellnr = 'A1633' OR sm.Modellnr = 'A1688');
```

(e) Formulieren Sie die folgende Anfrage in SQL:

Gesucht ist die Modellnummer der Smartphones, die durchschnittlich die meisten Apps installiert haben.

Tipp: Die Verwendung von Views kann die Aufgabe vereinfachen.

Lösungsvorschlag

```
CREATE VIEW NumberApps AS
SELECT s.ID, s.Modellnr, COUNT(i.Name) AS number
FROM Smartphone s, Installiert i
WHERE s.ID = i.ID
GROUP BY s.ID
SELECT ModellNr, AVG(number)
FROM NumberApps
GROUP BY ModellNr
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-2/Teilaufgabe-1/Aufgabe-5.tex

Examensaufgabe „Game of Thrones“ (66116-2019-H.T1-TA2-A3)

SQL

Formulieren Sie folgende Anfragen in SQL gegen die angegebene Datenbank aus einer imaginären Serie.

```
Figur : {[ Id, Name, Schwertkunst, Lebendig, Titel ]}

gehört_zu : {[ Id, Familie, FK (Id) references Figur(Id), FK (Familie) references Familie(Id)

]}

Familie : {[ Id, Name, Reichtum, Anführer ]}

Drache : {[ Name, Lebendig ]}

besitzt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Drache(Name) ]}

Festung : {[ Name, Ort, Ruine ]}

besetzt : {[ Familie, Festung, FK (Familie) references Familie(Id), FK (Festung) references

Festung(Name) ]}

lebt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Festung(Name) ]}
```

```sql
CREATE TABLE Figur (
  Id integer PRIMARY KEY,
  Name VARCHAR(20),
  Schwertkunst integer,
  Lebendig boolean,
  Titel VARCHAR(50)
);

CREATE TABLE Familie (
  Id integer PRIMARY KEY,
  Name VARCHAR(20),
  Reichtum numeric(11,2),
  Anführer VARCHAR(20)
);

CREATE TABLE gehört_zu (
  Id integer REFERENCES Figur(id),
  Familie integer REFERENCES Familie(id)
);

CREATE TABLE Drache (
  Name VARCHAR(20) PRIMARY KEY,
  Lebendig boolean
);

CREATE TABLE besitzt (
  Id integer REFERENCES Figur(Id),
  Name VARCHAR(20) REFERENCES Drache(Name)
);
```

```
CREATE TABLE Festung (
  Name VARCHAR(20) PRIMARY KEY,
  Ort VARCHAR(30),
  Ruine boolean
);

CREATE TABLE besetzt (
  Familie integer REFERENCES Familie(Id),
  Festung VARCHAR(20) REFERENCES Festung(Name)
);

CREATE TABLE lebt (
  Id integer REFERENCES Figur(Id),
  Name VARCHAR(20) REFERENCES Festung(Name)
);

INSERT INTO Figur VALUES
  (1, 'Eddard Stark', 5, FALSE, 'Lord von Winterfell'),
  (2, 'Rodd Stark', 4, FALSE, 'Lord von Winterfell'),
  (3, 'Tywin Lennister', 5, FALSE, 'Lord von Casterlystein'),
  (4, 'Cersei Lennister', 2, TRUE, 'Lady von Casterlystein'),
  (5, 'Brandon Stark', 0, TRUE, 'König der Andalen'),
  (6, 'Jon Schnee', 5, TRUE, 'König-jenseits-der-Mauer');

INSERT INTO Familie VALUES
  (1, 'Haus Stark', 76873.12, 'Eddard Stark'),
  (2, 'Haus Lennister', 82345.43, 'Tywin Lennister');

INSERT INTO gehört_zu VALUES
  (1, 1),
  (2, 1),
  (3, 2),
  (4, 2),
  (5, 1),
  (6, 1);

INSERT INTO Festung VALUES
  ('Roter Bergfried', 'Westeros', FALSE),
  ('Casterlystein', 'Westeros', FALSE),
  ('Winterfell', 'Westeros', FALSE);

INSERT INTO besetzt VALUES
  (1, 'Winterfell'),
  (2, 'Roter Bergfried'),
  (2, 'Casterlystein');

INSERT INTO lebt VALUES
  (1, 'Winterfell'),
  (2, 'Winterfell'),
  (3, 'Casterlystein'),
  (4, 'Roter Bergfried'),
  (5, 'Winterfell'),
  (6, 'Winterfell');
```

(a) Geben Sie für alle Figuren an, wie oft alle vorhandenen Titel vorkommen.

Korrelierte Anfrage
Korrelierte Anfrage

Lösungsvorschlag

```
SELECT count(*) AS Anzahl, f.Titel
FROM Figur f
GROUP BY f.Titel;

 anzahl |          titel
--------+--------------------------
      1 | König der Andalen
      2 | Lord von Winterfell
      1 | Lord von Casterlystein
      1 | König-jenseits-der-Mauer
      1 | Lady von Casterlystein
(5 rows)
```

(b) Welche Figuren (Name ist gesucht) kommen aus „Kings Landing“?

Lösungsvorschlag

```
SELECT
  f.Name
FROM
  Figur f,
  Festung b,
  lebt l
WHERE
  b.Name = l.Name AND
  f.Id = l.Id AND
  b.Ort = 'Königsmund';
```

(c) Geben Sie für jede Familie (Name) die Anzahl der zugehörigen Charaktere und Festungen an.

Lösungsvorschlag

```
SELECT
  f.Name,
  (SELECT COUNT(*) FROM Figur fi, gehört_zu ge WHERE fi.id = ge.id AND
↪  ge.Familie = f.id) AS Anzahl_Charaktere,
  (SELECT COUNT(*) FROM Festung fe, besetzt be WHERE fe.Name = be.Festung AND
↪  be.Familie = f.id) AS Anzahl_Festungen
FROM
  Familie f;
```

(d) Gesucht sind die besten fünf Schwertkämpfer aus Festungen aus dem Ort „Westeros“. Es soll der Name, die Schwertkunst und die Platzierung ausgegeben werden. Die Ausgabe soll nach der Platzierung sortiert erfolgen.

Lösungsvorschlag

```
-- Problem: Es gibt 3 mal 3. Platz und nicht 3 mal 1. Platz
SELECT f1.Name, f1.Schwertkunst, COUNT(*) FROM Figur f1, Figur f2
WHERE f1.Schwertkunst <= f2.Schwertkunst
GROUP BY f1.Name, f1.Schwertkunst
```

```
ORDER BY COUNT(*)
LIMIT 5;
```

DELETE
DROP COLUMN
ALTER TABLE
BETWEEN
CHECK
REFERENCES
NOT NULL

(e) Schreiben Sie eine Anfrage, die alle Figuren löscht, die tot sind. Das Attribut *Lebendig* kann dabei die Optionen „ja“ und „nein“ annehmen.

Lösungsvorschlag

```
-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;

-- PostgreSQL unterstützt kein DELETE JOIN
-- DELETE f, g, b, l FROM Figur AS f
--  JOIN gehört_zu AS g ON f.id = g.id
--  JOIN besitzt AS b ON f.id = b.id
--  JOIN lebt AS l ON f.id = l.id
-- WHERE f.Lebendig = FALSE;

DELETE FROM gehört_zu WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
↪  FALSE);
DELETE FROM besitzt WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
↪  FALSE);
DELETE FROM lebt WHERE id IN (SELECT id FROM Figur WHERE Lebendig = FALSE);
DELETE FROM Figur WHERE Lebendig = FALSE;

-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;
```

(f) Löschen Sie die Spalten „Lebendig“ aus der Datenbank.

Lösungsvorschlag

```
ALTER TABLE Figur DROP COLUMN Lebendig;
ALTER TABLE Drache DROP COLUMN Lebendig;
```

(g) Erstellen Sie eine weitere Tabelle mit dem Namen *Waffen*, welche genau diese auflistet. Eine Waffe ist genau einer Figur zugeordnet, hat einen eindeutigen Namen und eine Stärke zwischen 0 und 5. Wählen Sie sinnvolle Typen für die Attribute.

Lösungsvorschlag

```
CREATE TABLE Waffen (
  Name VARCHAR(20) PRIMARY KEY,
  Figur integer NOT NULL REFERENCES Figur(Id),
  Stärke integer NOT NULL CHECK(Stärke BETWEEN 0 AND 5)
);

-- Sollte funktionieren:
INSERT INTO Waffen VALUES
  ('Axt', 1, 5);

-- Sollte Fehler ausgeben:
```

```
-- INSERT INTO Waffen VALUES
--  ('Schleuder', 1, 6);
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Formel-1-Rennen“ (66116-2019-H.T2-TA2-A7)

SQL

Gegeben sind folgende Relationen aus einem Verwaltungssystem für die jährlichen Formel-1-Rennen:

```
Strecke(Strecken_ID, Streckenname, Land, Länge)

Fahrer(Fahrer_ID, Fahrername, Nation, Rennstall)

Rennen(Strecken_ID[Strecke], Jahr, Wetter)

Rennteilnahme(Fahrer_ID[Fahrer], Strecken_ID[Rennen], Jahr[Rennen], Rundenbestzeit, Gesamtzeit,

disqualifiziert)
FK (Strecken_ID, Jahr) referenziert Rennen (Strecken_ID, Jahr)
```

```sql
CREATE TABLE Fahrer (
  Fahrer_ID INTEGER PRIMARY KEY,
  Fahrername VARCHAR(100) NOT NULL,
  Nation VARCHAR(100) NOT NULL,
  Rennstall VARCHAR(100) NOT NULL
);

CREATE TABLE Strecke (
  Strecken_ID INTEGER PRIMARY KEY,
  Streckenname VARCHAR(100) NOT NULL,
  Land VARCHAR(100) NOT NULL,
  Länge NUMERIC(5,3) NOT NULL
);

CREATE TABLE Rennen (
  Strecken_ID INTEGER REFERENCES Strecke(Strecken_ID),
  Jahr INTEGER NOT NULL,
  Wetter VARCHAR(10) NOT NULL,
  PRIMARY KEY (Strecken_ID, Jahr)
);

CREATE TABLE Rennteilnahme (
  Fahrer_ID INTEGER REFERENCES Fahrer(Fahrer_ID),
  Strecken_ID INTEGER REFERENCES Strecke(Strecken_ID),
  Jahr INTEGER NOT NULL,
  Rundenbestzeit NUMERIC(5,3) NOT NULL,
  Gesamtzeit NUMERIC(5,3) NOT NULL,
  disqualifiziert BOOLEAN NOT NULL,
  PRIMARY KEY (Fahrer_ID, Strecken_ID, Jahr)
);

INSERT INTO Fahrer VALUES
  (1, 'Kimi Räikkönen', 'Finnland', 'Alfa Romeo'),
  (2, 'Rubens Barrichello', 'Brasilien', 'Brawn'),
  (3, 'Fernando Alonso', 'Spanien', 'Ferrari'),
  (4, 'Michael Schumacher', 'Deutschland', 'Ferrari'),
  (5, 'Jenson Button', 'Vereinigtes Königreich Großbritannien', 'McLaren'),
  (6, 'Felipe Massa', 'Brasilien', 'Ferrari'),
```

```
  (7, 'Lewis Hamilton', 'Vereinigtes Königreich Großbritannien', 'Williams'),
  (8, 'Riccardo Patrese', 'Italien', 'Williams'),
  (9, 'Sebastian Vettel', 'Deutschland', 'Ferrari'),
  (10, 'Jarno Trulli', 'Italien', 'Toyota');

INSERT INTO Strecke VALUES
  (1, 'Autodromo Nazionale Monza', 'Italien', 5.793),
  (2, 'Circuit de Monaco', 'Monaco', 3.340),
  (3, 'Silverstone Circuit', 'Vereinigtes Königreich', 5.891),
  (4, 'Circuit de Spa-Francorchamps', 'Belgien', 7.004),
  (5, 'Circuit Gilles-Villeneuve', 'Kanada', 4.361),
  (6, 'Nürburgring', 'Deutschland', 5.148),
  (7, 'Hockenheimring', 'Deutschland', 4.574),
  (8, 'Interlagos', 'Brasilien', 4.309),
  (9, 'Hungaroring', 'Ungarn', 4.381),
  (10, 'Red Bull Ring', 'Österreich', 5.942),
  (11, 'Abu Dhabi', 'Abu Dhabi', 5.554);

INSERT INTO Rennen VALUES
  (11, 2011, 'sonnig'),
  (10, 2006, 'sonnig'),
  (9, 2007, 'regnerisch'),
  (8, 2008, 'regnerisch'),
  (7, 2009, 'sonnig'),
  (6, 2010, 'regnerisch'),
  (5, 2011, 'sonnig'),
  (4, 2012, 'sonnig'),
  (3, 2013, 'sonnig'),
  (2, 2014, 'regnerisch'),
  (1, 2015, 'regnerisch');

INSERT INTO Rennteilnahme VALUES
  (1, 11, 2011, 2.001, 90.001, FALSE),
  (2, 11, 2011, 2.002, 90.002, FALSE),
  (3, 11, 2011, 2.003, 90.003, FALSE),
  (4, 11, 2011, 2.004, 89.999, FALSE),
  (5, 11, 2011, 2.005, 90.005, FALSE),
  (6, 11, 2011, 2.005, 99.009, FALSE),
  (4, 10, 2006, 2.782, 90.005, TRUE),
  (3, 10, 2006, 2.298, 90.005, TRUE),
  (3, 9, 2009, 2.253, 90.005, TRUE),
  (2, 10, 2006, 2.005, 90.005, TRUE),
  (2, 9, 2009, 3.298, 90.342, TRUE),
  (2, 8, 2008, 4.782, 78.005, TRUE);
```

Der Einfachheit halber wird angenommen, dass Fahrer den Rennstall nicht wechseln können. Das Attribut „disqualifiziert“ kann die Ausprägungen „ja“ und „nein“ haben. Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

(a) Geben Sie für jeden Fahrer seine ID sowie die Anzahl seiner Disqualifikationen in den Jahren 2005 bis 2017 aus. Ordnen Sie die Ausgabe absteigend nach der Anzahl der Disqualifikationen.

Lösungsvorschlag EXCEPT

```
SELECT Fahrer_ID, COUNT(disqualifiziert) as anzahl_disqualifikationen FROM
↪  Rennteilnahme
WHERE disqualifiziert = TRUE
GROUP BY Fahrer_ID, disqualifiziert
ORDER BY anzahl_disqualifikationen DESC;
```

(b) Gesucht sind alle Länder, aus denen noch nie ein Fahrer disqualifiziert wurde.

Lösungsvorschlag

```
SELECT Nation FROM Fahrer GROUP BY Nation
EXCEPT
SELECT f.Nation FROM Fahrer f, Rennteilnahme t
WHERE f.Fahrer_ID = t.Fahrer_ID AND t.disqualifiziert = TRUE
GROUP BY f.Nation;
```

(c) Gesucht sind die ersten fünf Plätze des Rennens von 2011 in „Abu Dhabi“ (Streckenname). Die Ausgabe soll nach der Platzierung absteigend erfolgen. Geben Sie Fahrer_ID, Fahrername, Nation und Rennstall mit aus.

Lösungsvorschlag

Mit LIMIT

```
SELECT f.Fahrer_ID, f.Fahrername, f.Nation, f.Rennstall
FROM Fahrer f, Rennteilnahme t, Strecke s
WHERE
  f.Fahrer_ID = t.Fahrer_ID AND
  s.Strecken_ID = t.Strecken_ID AND
  s.Streckenname = 'Abu Dhabi' AND
  t.Jahr = 2011
ORDER BY t.Gesamtzeit ASC LIMIT 5;
```

Als Top-N-Query:

```
CREATE VIEW Rennen_Abu_Dhabi AS
  SELECT f.Fahrer_ID, f.Fahrername, f.Nation, f.Rennstall, t.Gesamtzeit
  FROM Fahrer f, Rennteilnahme t, Strecke s
  WHERE
    f.Fahrer_ID = t.Fahrer_ID AND
    s.Strecken_ID = t.Strecken_ID AND
    s.Streckenname = 'Abu Dhabi' AND
    t.Jahr = 2011
  ORDER BY t.Gesamtzeit ASC;

SELECT a.Fahrer_ID, a.Fahrername, a.Nation, a.Rennstall
FROM Rennen_Abu_Dhabi a, Rennen_Abu_Dhabi b
WHERE
  a.Gesamtzeit >= b.Gesamtzeit
GROUP BY a.Fahrer_ID, a.Fahrername, a.Nation, a.Rennstall, a.Gesamtzeit
HAVING COUNT(*) <= 5
ORDER BY a.Gesamtzeit;
```

(d) Führen Sie eine neue Spalte `Gehalt` in die Tabelle `Fahrer` ein. Da sich die Prämien für die Fahrer nach einem Rennstallwechsel ändern, soll ein Trigger geschrieben werden, mit dem das Gehalt des betreffenden Fahrers um 10% angehoben wird.

ALTER TABLE
TRIGGER

Lösungsvorschlag

Lösung für PostgreSQL:

```
ALTER TABLE Fahrer ADD Gehalt numeric(12,2);

UPDATE Fahrer SET Gehalt = 1000000 WHERE Fahrer_ID = 1;
UPDATE Fahrer SET Gehalt = 2000000 WHERE Fahrer_ID = 2;
UPDATE Fahrer SET Gehalt = 3000000 WHERE Fahrer_ID = 3;
UPDATE Fahrer SET Gehalt = 4000000 WHERE Fahrer_ID = 4;
UPDATE Fahrer SET Gehalt = 5000000 WHERE Fahrer_ID = 5;
UPDATE Fahrer SET Gehalt = 6000000 WHERE Fahrer_ID = 6;
UPDATE Fahrer SET Gehalt = 7000000 WHERE Fahrer_ID = 7;
UPDATE Fahrer SET Gehalt = 8000000 WHERE Fahrer_ID = 8;
UPDATE Fahrer SET Gehalt = 9000000 WHERE Fahrer_ID = 9;
UPDATE Fahrer SET Gehalt = 10000000 WHERE Fahrer_ID = 10;

CREATE FUNCTION trigger_function()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
  BEGIN
    UPDATE Fahrer
      SET gehalt = gehalt * 1.1
      WHERE Fahrer_ID = NEW.Fahrer_ID;
    RETURN NEW;
  END;
$$;

CREATE TRIGGER mehr_gehalt
  AFTER UPDATE OF Rennstall ON Fahrer
  FOR EACH ROW EXECUTE PROCEDURE trigger_function();

-- Test:
SELECT * FROM FAHRER WHERE Fahrer_ID = 1;
UPDATE Fahrer SET Rennstall = 'Red Bull' WHERE Fahrer_ID = 1;
SELECT * FROM FAHRER WHERE Fahrer_ID = 1;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-2/Aufgabe-7.tex`

Examensaufgabe „Zehnkampf“ (66116-2020-F.T1-TA2-A7)

SQL
SQL mit Übungsdatenbank

Gegeben sei die Relation *Zehnkampf*, welche die Ergebnisse eines Zehnkampfwettkampfes verwaltet. Eine beispielhafte Ausprägung ist in nachfolgender Tabelle gegeben.

Hinweise: Jeder Athlet kann in jeder Disziplin maximal ein Ergebnis erzielen. Außerdem können Sie davon ausgehen, dass jeder Name eindeutig ist.

Name	Disziplin	Leistung	Einheit	Punkte
John	100m	10.21	Sekunden	845
Peter	Hochsprung	213	Zentimeter	812
Peter	100m	10.10	Sekunden	920
Hans	100m	10.21	Sekunden	845
Hans	400m	44.12	Sekunden	910
...	...	...	...	...

```sql
CREATE TABLE Zehnkampf (
  Name VARCHAR(30),
  Disziplin VARCHAR(30),
  Leistung FLOAT,
  Einheit VARCHAR(30),
  Punkte INTEGER,
  PRIMARY KEY(Name, Disziplin, Leistung)
);

INSERT INTO Zehnkampf VALUES
  ('John', '100m', 10.21, 'Sekunden', 845),
  ('Peter', 'Hochsprung', 213, 'Zentimeter', 812),
  ('Peter', '100m', 10.10, 'Sekunden', 920),
  ('Hans', '100m', 10.21, 'Sekunden', 845),
  ('Hans', '400m', 44.12, 'Sekunden', 910);
```

(a) Bestimmen Sie alle funktionale Abhängigkeiten, die **sinnvollerweise** in der Relation Zehnkampf gelten.

Lösungsvorschlag

$$\mathrm{FA} = \Big\{ \begin{aligned} \{\, \mathit{Disziplin} \,\} &\to \{\, \mathit{Einheit} \,\}, \\ \{\, \mathit{Disziplin}, \mathit{Leistung} \,\} &\to \{\, \mathit{Punkte} \,\}, \\ \{\, \mathit{Name}, \mathit{Disziplin} \,\} &\to \{\, \mathit{Leistung} \,\}, \end{aligned} \Big\}$$

(b) Normalisieren Sie die Relation Zehnkampf unter Beachtung der von Ihnen identifzierten funktionalen Abhängigkeiten. Unterstreichen Sie alle Schlüssel des resultierenden Schemas.

Lösungsvorschlag Top-N-Query

R_1 : {[Disziplin, Einheit]}
R_2 : {[Disziplin, Leistung, Punkte]}
R_3 : {[Name, Disziplin, Leistung]}

(c) Bestimmen Sie in SQL den Athleten (oder bei Punktgleichheit, die Athleten), der in der Summe am meisten Punkte in allen Disziplinen erzielt hat. Benutzen Sie dazu die noch nicht normalisierte Ausgangsrelation *Zehnkampf*.

Lösungsvorschlag

```
CREATE VIEW GesamtPunkte AS
  SELECT Name, SUM(Punkte) As Punkte
  FROM Zehnkampf
  GROUP BY Name;

SELECT g1.Name, g1.Punkte, COUNT(*) AS Rang
FROM GesamtPunkte g1, GesamtPunkte g2
WHERE g1.Punkte <= g2.Punkte
GROUP BY g1.Name, g1.Punkte
HAVING COUNT(*) = 1;

 name | punkte | rang
------+--------+------
 Hans |   1755 |    1
(1 row)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-7.tex

Examensaufgabe „Universitätssschema“ (66116-2020-F.T1-TA2-A8)

SQL

Gegeben sei das Universitätssschema. Formulieren Sie folgende Anfragen in SQL-92:

```sql
CREATE TABLE Studierende (
  MatrNr INTEGER PRIMARY KEY,
  Name VARCHAR(15),
  Semester INTEGER
);

CREATE TABLE Professoren (
  PersNr INTEGER PRIMARY KEY,
  Name VARCHAR(30),
  Rang VARCHAR(30),
  Raum INTEGER
);

CREATE TABLE Assistenten (
  PersNr INTEGER PRIMARY KEY,
  Name VARCHAR(20),
  Fachgebiet VARCHAR(30),
  Boss INTEGER,
  FOREIGN KEY (Boss) REFERENCES Professoren(PersNr)
);

CREATE TABLE Vorlesungen (
  VorlNr INTEGER PRIMARY KEY,
  Titel VARCHAR(30),
  SWS INTEGER,
  gelesenVon INTEGER,
  FOREIGN KEY (gelesenVon) REFERENCES Professoren(PersNr)
);

CREATE TABLE hören (
  MatrNr INTEGER,
  VorlNr INTEGER,
  PRIMARY KEY(MatrNr, VorlNr),
  FOREIGN KEY (MatrNr) REFERENCES Studierende(MatrNr),
  FOREIGN KEY (VorlNr) REFERENCES Vorlesungen(VorlNr)
);

CREATE TABLE prüfen (
  MatrNr INTEGER,
  VorlNr INTEGER,
  PersNr INTEGER,
  Note INTEGER,
  PRIMARY KEY(MatrNr, VorlNr, PersNr),
  FOREIGN KEY (MatrNr) REFERENCES Studierende(MatrNr),
  FOREIGN KEY (VorlNr) REFERENCES Vorlesungen(VorlNr),
  FOREIGN KEY (PersNr) REFERENCES Professoren(PersNr)
);

CREATE TABLE voraussetzen (
  Vorgänger INTEGER,
  Nachfolger INTEGER,
```

```
  PRIMARY KEY(Vorgänger, Nachfolger),
  FOREIGN KEY (Vorgänger) REFERENCES Vorlesungen(VorlNr),
  FOREIGN KEY (Nachfolger) REFERENCES Vorlesungen(VorlNr)
);

INSERT INTO Studierende
  (MatrNr, Name, Semester)
VALUES
  (24002, 'Xenokrates', 18),
  (25403, 'Jonas', 12),
  (26120, 'Fichte', 10),
  (26830, 'Aristoxenos', 8),
  (27550, 'Schopenhauer', 6),
  (28106, 'Carnap', 3),
  (29120, 'Theophrastos', 2),
  (29555, 'Feuerbach', 2);

INSERT INTO Professoren
  (PersNr, Name, Rang, Raum)
VALUES
  (2125, 'Sokrates', 'C4', 226),
  (2126, 'Russel', 'C4', 226),
  (2127, 'Kopernikus', 'C3', 226),
  (2133, 'Popper', 'C3', 226),
  (2134, 'Augustinus', 'C3', 226),
  (2136, 'Curie', 'C4', 226),
  (2137, 'Kant', 'C4', 226);

INSERT INTO Assistenten
  (PersNr, Name, Fachgebiet, Boss)
VALUES
  (3002, 'Platon', 'Ideenlehre', 2125),
  (3003, 'Aristoteles', 'Syllogistik', 2125),
  (3004, 'Wittgenstein', 'Sprachtheorie', 2126),
  (3005, 'Rhetikus', 'Planetenbewegung', 2127),
  (3006, 'Newton', 'Kaplersche Gesetze', 2127),
  (3007, 'Spinosa', 'Gott und Natur', 2134);

INSERT INTO Vorlesungen
  (VorlNr, Titel, SWS, gelesenVon)
VALUES
  (4052, 'Logik', 4, 2125),
  (4630, 'Die 3 Kritiken', 4, 2137),
  (5001, 'Grundzüge', 4, 2137),
  (5022, 'Glaube und Wissen', 2, 2134),
  (5041, 'Ethik', 4, 2125),
  (5043, 'Erkenntnisstheorie', 3, 2126),
  (5049, 'Mäeutik', 2, 2125),
  (5052, 'Wissenschaftstheorie', 3, 2126),
  (5216, 'Bioethik', 2, 2126),
  (5259, 'Der Wiener Kreis', 2, 2133);

INSERT INTO hören
  (MatrNr, VorlNr)
VALUES
```

```
  (25403, 5022),
  (26120, 5001),
  (27550, 4052),
  (27550, 5001),
  (28106, 5041),
  (28106, 5052),
  (28106, 5216),
  (28106, 5259),
  (29120, 5001),
  (29120, 5041),
  (29120, 5049),
  (29555, 5001),
  (29555, 5022),
  (28106, 5001),
  (28106, 5022);

INSERT INTO prüfen
  (MatrNr, VorlNr, PersNr, Note)
VALUES
  (28106, 5001, 2126, 1),
  (25403, 5041, 2125, 2),
  (27550, 4630, 2137, 2),
  (25403, 4630, 2137, 5);

INSERT INTO voraussetzen
  (Vorgänger, Nachfolger)
VALUES
  (5001, 5041),
  (5001, 5043),
  (5001, 5049),
  (5041, 5216),
  (5043, 5052),
  (5041, 5052),
  (5052, 5259);
```

(a) Welche Vorlesungen liest der Boss des Assistenten *Platon* (nur Vorlesungsnummer und Titel ausgeben)?

Lösungsvorschlag

```
SELECT v.VorlNr, v.Titel
FROM Vorlesungen v, Assistenten a
WHERE a.Boss = v.gelesenVon AND a.Name = 'Platon';

 vorlnr |  titel
--------+---------
   4052 | Logik
   5041 | Ethik
   5049 | Mäeutik
(3 rows)
```

(b) Welche Studierende haben sich schon in mindestens einer direkten Voraussetzung von *Wissenschaftstheorie* prüfen lassen?

Lösungsvorschlag

Wissenschaftstheorie (5052) → Erkenntnistheorie (5043) Ethik (5041) → Jonas (25403)

```
SELECT s.Name
FROM Vorlesungen l, voraussetzen a, prüfen p, Studierende s
WHERE
  l.Titel = 'Wissenschaftstheorie' AND
  l.VorlNr = a.Nachfolger AND
  a.Vorgänger = p.VorlNr AND
  p.MatrNr = s.MatrNr;

 name
-------
 Jonas
(1 row)
```

(c) Wie viele Studierende hören *Ethik*?

Lösungsvorschlag

```
SELECT COUNT(*)
FROM Vorlesungen v, hören h
WHERE
  v.Titel = 'Ethik' AND
  v.VorlNr = h.VorlNr;

 count
-------
     2
(1 row)
```

(d) Welche Studierende sind im gleichen Semester? — Geben Sie Paare von Studierenden aus.

Achten Sie darauf, dass ein/e Studierende/r mit sich selbst kein Paar bildet. — Achten Sie auch darauf, dass kein Paar doppelt ausgeben wird: wenn das Paar *StudentA*, *StudentB* im Ergebnis enthalten ist, soll nicht auch noch das Paar *StudentB*, *StudentA* ausgegeben werden.

Lösungsvorschlag

```
SELECT s1.Name, s2.Name
FROM Studierende s1, Studierende s2
WHERE
  s1.Semester = s2.Semester AND
  s1.MatrNr < s2.MatrNr;

     name     |   name
--------------+-----------
 Theophrastos | Feuerbach
(1 row)
```

(e) In welchen Fächern ist die Durchschnittsnote schlechter als 2? Geben Sie die Vorlesungsnummer und den Titel aus.

Lösungsvorschlag

```
SELECT v.VorlNr, v.Titel
FROM prüfen p, Vorlesungen v
WHERE p.VorlNr = v.VorlNr
GROUP BY v.VorlNr, v.Titel
HAVING AVG(p.Note) >  2;

 vorlnr |     titel
--------+----------------
   4630 | Die 3 Kritiken
(1 row)
```

(f) Finden Sie alle Paare von Studierenden (*MatrNr* duplikatfrei ausgeben), die mindestens zwei Vorlesungen gemeinsam hören.

Lösungsvorschlag

```
SELECT h1.MatrNr, h2.MatrNr
FROM hören h1, hören h2
WHERE
  h1.VorlNr = h2.VorlNr AND
  h1.MatrNr < h2.MatrNr
GROUP BY h1.MatrNr, h2.MatrNr
HAVING COUNT(*) > 1;

 matrnr | matrnr
--------+--------
  28106 |  29120
  28106 |  29555
(2 rows)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-8.tex

Examensaufgabe „Mode-Kollektionen“ (66116-2020-F.T2-TA2-A2)

SQL

Gegeben sei der folgende Ausschnitt eines Schemas für die Verwaltung von Kollektionen:

Die Tabelle *Promi* beschreibt Promis über ihren eindeutigen Namen, ihr Alter und ihren Wohnort. Kollektion enthält Informationen über *Kollektionen*, nämlich deren eindeutigen Namen, das Jahr und die Saison. Die Tabelle *promotet* verwaltet über Referenzen, welcher Promi welche Kollektion promotet. *Kleidungsstück* speichert die IDs von Kleidungsstücken zusammen mit dem Hauptbestandteil und einer Referenz auf die zugehörige Kollektion. Die Tabelle *hat_getragen* verwaltet über Referenzen, welcher Promi welches Kleidungsstück an welchem Datum getragen hat.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

```
CREATE TABLE Promi (
  Name VARCHAR(255) PRIMARY KEY,
  Alter INTEGER,
  Wohnort VARCHAR(255)
);

CREATE TABLE Kollektion (
  Name VARCHAR(255) PRIMARY KEY,
  Jahr INTEGER,
  Saison VARCHAR(255)
);

CREATE TABLE Kleidungsstueck (
  ID INTEGER PRIMARY KEY,
  Hauptbestandteil VARCHAR(255),
  gehoert_zu VARCHAR(255) REFERENCES Kollektion(Name)
);

CREATE TABLE hat_getragen (
  PromiName VARCHAR(255) REFERENCES Promi(Name),
  KleidungsstueckID INTEGER REFERENCES Kleidungsstueck(ID),
  Datum DATE,
  PRIMARY KEY(PromiName, KleidungsstueckID, Datum)
);

CREATE TABLE promotet (
  PromiName VARCHAR(255),
  KollektionName VARCHAR(255),
  PRIMARY KEY(PromiName, KollektionName)
);

INSERT INTO Promi
  (Name, Alter, Wohnort)
VALUES
  ('Till Schweiger', 52, 'Dortmund'),
  ('Lena Meyer-Landrut', 30, 'Hannover');

INSERT INTO Kollektion VALUES
```

```
  ('Gerry Weber', 2020, 'Sommer'),
  ('H & M', 2020, 'Sommer');

INSERT INTO promotet
  (PromiName, KollektionName)
VALUES
  ('Till Schweiger', 'Gerry Weber'),
  ('Lena Meyer-Landrut', 'H & M');

INSERT INTO Kleidungsstueck
  (ID, Hauptbestandteil, gehoert_zu)
VALUES
  (1, 'Hose', 'Gerry Weber');

INSERT INTO hat_getragen
  (PromiName, KleidungsstueckID, Datum)
VALUES
  ('Till Schweiger', 1, '2021-08-03');
```

(a) Schreiben Sie SQL-Anweisungen, welche die Tabelle `hat_getragen` inklusive aller benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalte `Datum` Teil des Primärschlüssels ist.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS hat_getragen (
  PromiName VARCHAR(255) REFERENCES Promi(Name),
  KleidungsstueckID INTEGER REFERENCES Kleidungsstueck(ID),
  Datum DATE,
  PRIMARY KEY(PromiName, KleidungsstueckID, Datum)
);
```

Das Datenbanksystem achtet selbst darauf, dass Felder des Primärschlüssels nicht NULL sind. Deshalb muss man NOT NULL bei keinem der drei Felder angeben.

(b) Schreiben Sie eine SQL-Anweisung, welche die Namen der Promis ausgibt, die eine Sommer-Kollektion promoten (Saison ist „Sommer“).

Lösungsvorschlag

```
SELECT p.PromiName
FROM promotet p, Kollektion k
WHERE
  k.Saison = 'Sommer' AND
  p.KollektionName = k.Name;

     prominame
--------------------
 Till Schweiger
 Lena Meyer-Landrut
(2 rows)
```

(c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Promis und der Kollektionen bestimmt, welche der Promi zwar promotet, aber daraus noch kein Kleidungsstück getragen hat.

Lösungsvorschlag

Lösung mit NOT IN:

```
SELECT * FROM promotet p
WHERE p.KollektionName NOT IN (
  SELECT k.Name FROM Kollektion k
  INNER JOIN Kleidungsstueck s ON s.gehoert_zu = K.Name
  INNER JOIN hat_getragen t ON t.KleidungsstueckID = s.ID
  WHERE t.PromiName = p.PromiName
);
```

Mit EXCEPT:

```
(
  SELECT p.PromiName, p.KollektionName FROM promotet p
)
EXCEPT
(
  SELECT p.PromiName, p.KollektionName FROM promotet p
  INNER JOIN Kollektion k ON p.KollektionName = k.Name
  INNER JOIN Kleidungsstueck s ON s.gehoert_zu = k.Name
  INNER JOIN hat_getragen t ON t.KleidungsstueckID = s.ID
  WHERE t.PromiName = p.PromiName
);
```

(d) Bestimmen Sie für die folgenden SQL-Anweisungen die minimale und maximale Anzahl an Tupeln im Ergebnis. Beziehen Sie sich dabei auf die Größe der einzelnen Tabellen.

Verwenden Sie für die Lösung folgende Notation: – Promi – beschreibt die Größe der Tabelle Promi.

(i)

```
SELECT k.Name
FROM Kollektion k, Kleidungsstueck kl
WHERE k.Name = kl.gehoert_zu and k.Jahr = 2018
GROUP BY k.Name
HAVING COUNT(kl.Hauptbestandteil) > 10;
```

Lösungsvorschlag

– Kollektion – beschreibt die Anzahl der Tupel in der Tabelle Kollektion. Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion den Anforderungen `k.Jahr = 2018` oder `HAVING COUNT(...)` genügt. Die maximale Anzahl an Tupeln im Ergebnis ist – Kollektion –, da nur Namen aus Kollektion ausgewählt werden.

(ii)

```
SELECT DISTINCT k.Jahr
FROM Kollektion k
WHERE k.Name IN (
  SELECT pr.KollektionName
  FROM Promi p, promotet pr
  WHERE p.Alter < 30 AND pr.PromiName = p.Name
);
```

Lösungsvorschlag

Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion promotet wird. Die maximale Anzahl ist das Minimum von – Kollektion – und 30, da die Promis, die Kollektionen beworben haben, die älter als 30 Jahre sind, selbst mindestens 30 Jahre alt sein müssen.

Zugrundeliegende Annahmen: Kollektionen werden nur im Erscheinungsjahr von Promis beworben; Neugeborene, die Kollektionen bewerben, werden ggf. Promis.

(e) Beschreiben Sie den Effekt der folgenden SQL-Anfrage in natürlicher Sprache

```
SELECT pr.KollektionName
FROM promotet pr, Promi p
WHERE pr.PromiName = p.Name
GROUP BY pr.KollektionName
HAVING COUNT (*) IN (
  SELECT MAX(anzahl)
  FROM (
    SELECT k.Name, COUNT(*) AS anzahl
    FROM Kollektion k, promotet pr
    WHERE k.Name = pr.KollektionName
    GROUP BY k.Name
  ) as tmp
);
```

Lösungsvorschlag

Die Abfrage gibt die Namen aller Kollektionen aus, bei denen die Anzahl der bewerbenden Promis am größten ist.

(f) Formulieren Sie folgende SQL-Anfrage in relationaler Algebra. Die Lösung kann in Baum- oder in Term-Schreibweise angegeben werden, wobei eine Schreibweise genügt.

```
SELECT p.Wohnort
FROM Promi p, promotet pr, Kollektion k
WHERE
  p.Name = pr.PromiName AND
  k.Name pr.KollektionName AND
  k.Jahr 2018;
```

(i) Konvertieren Sie zunächst die gegebene SQL-Anfrage in die zugehörige Anfrage in relationaler Algebra nach Standard-Algorithmus.

Lösungsvorschlag

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name}=\text{promotet.PromiName} \wedge \text{Kollektion.Name}=\text{promotet.KollektionName} \wedge \text{Kollektion.Jahr}=2018}(\text{Promi} \times \text{promotet} \times \text{Kollektion}))$$

(ii) Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und begründen Sie dabei kurz jeden durchgeführten Schritt.

Lösungsvorschlag

1. Schritt: Um die kartesischen Produkte in Joins zu verwandeln, muss die Selektion in drei Selektionen zerlegt werden.

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name}=\text{promotet.PromiName}}(\sigma_{\text{Kollektion.Name}=\text{promotet.KollektionName}}(\sigma_{\text{Kollektion.Jahr}=2018}(\text{Promi} \times (\text{promotet} \times \text{Kollektion})))))$$

2. Schritt: Man kann die Selektion nach dem Jahr ausführen, bevor die kartesischen Produkte ausgeführt werden. Dadurch werden von vornherein nur die Kollektionen betrachtet, die in dem entsprechenden Jahr aufgetreten sind.

$$\pi_{\text{Promi.Wohnort}}(\sigma Promi.Name = promotet.PromiName(\sigma Kollektion.Name = promotet.Koll$$

3. Schritt: Die zweitinnerste Selektion kann direkt nach dem innersten kartesischen Produkt ausgeführt werden; dadurch wird das Gesamtprodukt nicht so groß. Man benötigt also weniger Rechenzeit und Speicher.

$$\pi_{\text{Promi.Wohnort}}(\sigma Promi.Name = promotet.PromiName(Promi \times \sigma Kollektion.Name = prom$$

4. Schritt: Beide Selektionen in Verbindung mit dem jeweiligen kartesischen Produkt können in einen Join verwandelt werden. So wird

nicht zuerst das gesamte Produkt berechnet und danach die passenden Eintraege ausgewaehlt, sondern gleich nur die zusammengehörigen Eintraege kombiniert. Auch dies spart Rechenzeit und Speicher.

$$\pi_{\text{Promi.Wohnort}}(Promi \bowtie Promi.Name = promotet.PromiName(promotet \bowtie Kollektion.Na$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Restaurant“ (66116-2020-H.T2-TA2-A2)

SQL

Gegeben ist der folgende Ausschnitt eines Schemas für die Verwaltung eines Restaurants.

Hinweis: Unterstrichene Attribute sind Primärschlüsselattribute, kursiv geschriebene Attribute

sind Fremdschlässelattribute.

Restaurant : | RestaurantID : INTEGER, RestaurantName : VARCHAR (255), StadtName : VARCHAR(255),

PLZ: INTEGER, 1

Küche: |

RestaurontID : INTEGER, Art : VARCHAR(255), KochPersonID : INTEGER

Straße : VARCHAR (255), Hausnummer: INTEGER, Kategorie : VARCHAR (255)

1

Stadt : [Person : |

StadtName : VARCHAR(255), PersonID : INTEGER,

Land : VARCHAR(255) Name : VARCHAR(255), 1 Vorname : VARCHAR (255), StadtName : VARCHAR(255), PLZ: INTEGER, Straße : VARCHAR(255),

Hausnummer: INTEGER

bevorzugt : | PersonID : INTEGER, Art : VARCHAR(255)

I

Die Tabelle Restaurant beschreibt Restaurants eindeutig durch ihre ID. Zudem wird der Name, die Adresse des Restaurants und die (Sterne-)Kategorie gespeichert. Küche enthält u. a. Informationen zu der Art der Küche. Dabei kann ein Restaurant mehrere Arten anbieten, z. B. italienisch, deutsch, etc. In der Tabelle Stadt werden der Name der Stadt sowie das Land verwaltet, in dem die Stadt liegt. Wir gehen davon aus, dass eine Stadt eindeutig durch ihren Namen gekennzeichnet ist. Person beschreibt Personen mit Name, Vorname und Adresse. Personen werden eindeutig durch eine ID identifiziert. Die Tabelle bevorzugt gibt an, welche Person welche Art der Küche präferiert.

Bearbeiten Sie die folgenden Teilaufgaben:

(a) Erläutern Sie kurz, warum das Attribut Art in Küche Teil des Primärschlüssels ist.

Lösungsvorschlag

> Es kann mehr als eine Küche pro Restaurant geben.

(b) Schreiben Sie eine SQL-Anweisung, welche alle Städte findet, in denen man “deutsch” (Art der Küche) essen kann.

Lösungsvorschlag

```sql
SELECT DISTINCT s.StadtName
FROM Stadt s, Restaurant r, Küche k
WHERE s.Stadtname = r.StadtName AND r.RestaurantID = k.RestaurantID AND Art =
↪ 'deutsch';
```

(c) Schreiben Sie eine SQL-Abfrage, die alle Personen (Name und Vorname) liefert, die kein deutsches Essen bevorzugen. Verwenden Sie keinen Join.

Lösungsvorschlag

```sql
SELECT Name, Vorname
FROM Person
WHERE PersonID IN (SELECT DISTINCT PersonID
FROM bevorzugt
EXCEPT
SELECT DISTINCT PersonID
FROM bevorzugt
WHERE Art = 'deutsch');
```

(d) Schreiben Sie eine SQL-Abfrage, die für jede Stadt (StadtName) und Person (PersonID) die Anzahl der Restaurants ermittelt, in denen diese Person bevorzugt essen gehen würde. Es sollen nur Städte ausgegeben werden, in denen es mindestens drei solche Restaurants gibt.

Lösungsvorschlag

```sql
SELECT r.StadtName, b.PersonID, count(DISTINCT r.RestaurantID) as Anzahl
FROM Restaurant r, bevorzugt b, Kueche k
WHERE r.RestaurantID = k.RestaurantID AND k.Art = b.Art
GROUP BY r.StadtName, b.PersonID
HAVING count(r.RestaurantID) >= 3;
```

(e) Schreiben Sie eine SQL-Abfrage, die die Namen aller Restaurants liefert, in denen sich die Personen mit den IDs 1 und 2 gemeinsam zum Essen verabreden können, und beide etwas zum Essen finden, das sie bevorzugen. Es sollen keine Duplikate ausgegeben werden.

Lösungsvorschlag

```sql
CREATE VIEW Person1 AS
SELECT DISTINCT r.RestaurantID, r.RestaurantName
FROM Person p, bevorzugt b, Restaurant r, Küche k
WHERE r.StadtName = p.StadtName AND p.PersonID = b.PersonID
AND r.RestaurantID = k.RestaurantID AND k.Art = b.Art
AND p.PersonID = 1;
CREATE VIEW Person2 AS
SELECT DISTINCT r.RestaurantID, r.RestaurantName
FROM Person p, bevorzugt b, Restaurant r, Küche k
WHERE r.StadtName = p.StadtName AND p.PersonID = b.PersonID
AND r.RestaurantID = k.RestaurantID AND k.Art = b.Art
AND p.PersonID = 2;
SELECT * FROM Person1
INTERSECT
SELECT * FROM Person2;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-2.tex

Examensaufgabe „Fußballweltmeisterschaft“ (66116-2021-F.T1-TA2-A6)[SQL]

Gegeben ist folgendes Relationenschema zur Verwaltung von Daten aus der Fußballweltmeisterschaft:

Die Tabelle Match wurde in Spiel umgenannt, da es sonst zu Konfliken mit der SQL-Syntax kommt, da match ein SQL Schlüsselwort ist.

Nation (Land, Kapitän, Trainer) Kapitän ist Fremdschlüssel zu Spieler_ID in Spieler.

Spiel (Spiel_ID, Ort, Datum, Team1, Team2, ToreTeam1, ToreTeam2) Team1 ist Fremdschlüssel zu Land in Nation. Team2 ist Fremdschlüssel zu Land in Nation.

Spieler (Spieler_ID, Name, Vorname, Wohnort, Land) Land ist Fremdschlüssel zu Land in Nation.

Platzverweise (Platzverweis_ID,Spiel_ID, Spieler_ID, Spielminute) Spiel_ID ist Fremdschlüsssel zu Spiel_ID in Spiel. Spieler_ID ist Fremdschlüssel zu Spieler_ID in Spieler.

Die Primärschlüssel der Relationen sind wie üblich durch Unterstreichen gekennzeichnet. Pro Ort und Datum findet jeweils nur ein Spiel statt.

Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

(a) Ermitteln Sie die Anzahl der Platzverweise pro Spieler und geben Sie jeweils Name und Vorname des Spielers mit aus. Die Ausgabe soll nach der Anzahl der Platzverweise absteigend erfolgen.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl, s.Name, s.Vorname
FROM Platzverweise p, Spieler s
WHERE p.Spieler_ID = s.Spieler_ID
GROUP BY s.Name, s.Vorname
ORDER BY Anzahl DESC;

 anzahl |    name    | vorname
--------+------------+---------
      2 | Matthäus   | Lothar
      1 | Bodden     | Olaf
      1 | Beckham    | David
      1 | Rizzitelli | Luca
      1 | Babel      | Markus
      1 | Häßler     | Thomas
(6 rows)
```

(b) Welches ist die maximale Anzahl an Toren, die eine Mannschaft insgesamt im Turnier erzielt hat? (Sie dürfen der Einfachheit halber annehmen, dass jede Mannschaft jeweils mindestens einmal als Team1 und Team2 angetreten ist.)

Lösungsvorschlag

```
SELECT MAX(tmp2.Summe) FROM (
  SELECT Team, SUM(Summe) as Summe FROM (
    SELECT Team1 AS Team, SUM(ToreTeam1) AS Summe
      FROM Spiel
      GROUP BY Team1, ToreTeam1
    UNION
    SELECT Team2 AS Team, SUM(ToreTeam2) AS Summe
```

Top-N-Query

```
      FROM Spiel
      GROUP BY Team2, ToreTeam2
  ) AS tmp
  GROUP BY Team
) as tmp2;

 max
-----
   9
(1 row)
```

(c) Wie viele Tore sind im Turnier insgesamt gefallen?

Lösungsvorschlag

```
SELECT SUM(ToreTeam1 + ToreTeam2) AS GesamtanzahlTore
FROM Spiel;
```

(d) Ermitteln Sie die Namen und Länder der fünf Spieler, die nach der kürzesten Spielzeit einen Platzverweis erhielten. Die Ausgabe soll nummeriert erfolgen (beginnend bei 1 für die kürzeste Spielzeit).

Lösungsvorschlag

```
SELECT s.Name, s.Land, COUNT(*) AS Rang
FROM Spieler s, Platzverweise p1, Platzverweise p2
WHERE
  s.Spieler_ID = p2.Spieler_ID AND
  p1.Spielminute <= p2.Spielminute
GROUP BY s.Name, s.Land, p2.Spieler_ID
HAVING COUNT(*) < 6
ORDER BY Rang;
```

Der Erstplatzierte kommt durch die WHERE-Bedingungen nur einmal in der Relation vor, weil sein Eintrag genau einmal mit sich selbst vorkommt. Alle anderen Einträge, bei denen die `p2.Spieler_ID`, der Spieler mit der „geringsten Minute“ ist, werden ja eliminiert, da ja nur die Einträge behalten werden, die der Bedingung `p1.Spielminute <= p2.Spielminute` entsprechen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-6.tex`

Examensaufgabe „Mitarbeiter einer Abteilung“ (66116-2021-F.T2-TA2-A4)

SQL
SQL mit Übungsdatenbank

Gegeben sind folgende Relationen:

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter]
NOT NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

```sql
CREATE TABLE Abteilung(
  AbteilungsID INTEGER PRIMARY KEY,
  Bezeichnung VARCHAR(30) UNIQUE NOT NULL
);

CREATE TABLE Mitarbeiter(
  MitarbeiterID INTEGER PRIMARY KEY,
  Vorname VARCHAR(30),
  Nachname VARCHAR(30),
  Adresse VARCHAR(60),
  Gehalt DECIMAL(7, 2),
  Vorgesetzter INTEGER NOT NULL REFERENCES Mitarbeiter(MitarbeiterID),
  AbteilungsID INTEGER REFERENCES Abteilung(AbteilungsID)
);

INSERT INTO Abteilung VALUES
  (1, 'Buchhaltung'),
  (42, 'Vertrieb');

INSERT INTO Mitarbeiter VALUES
  (1, 'Karl', 'Landsbach', 'Sigmaringstraße 4, 87153 Farnbach', 2467.23, 1, 42),
  (2, 'Lisa', 'Grätzner', 'Scheidplatz 6, 18434 Tullach', 5382.2, 1, 42),
  (3, 'Sarah', 'Riedel', 'Am Angera 3, 79527 Töll', 7382.2, 1, 42),
  (4, 'Franz', 'Rudolf', 'Strewitzstraße 4, 45507 Strewith', 2382.2, 1, 42),
  (5, 'Sergej', 'Puschkin', 'Radolf 4, 12507 Radstadt', 1382.2, 1, 1);
```

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle Mitarbeiter anlegt. Gehen Sie davon aus, dass die Tabelle Abteilung bereits existiert.

Lösungsvorschlag

Siehe oben

(b) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen der Mitarbeiter der Abteilung mit der Bezeichnung Vertrieb ausgibt, absteigend sortiert nach MitarbeiterID.

Lösungsvorschlag Top-N-Query

```
SELECT m.Vorname, m.Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
  a.AbteilungsID = m.AbteilungsID AND
  a.Bezeichnung = 'Vertrieb'
ORDER BY m.MitarbeiterID DESC;
```

(c) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen sowie das Gehalt von Mitarbeitern ausgibt, die mehr verdienen als ihr Chef. Sortieren Sie die Ausgabe absteigend nach dem Gehalt.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n
WHERE
  m.Vorgesetzter = n.MitarbeiterID AND
  m.Gehalt > n.Gehalt
ORDER BY m.Gehalt DESC;
```

(d) Schreiben Sie eine SQL-Anweisung, die das Gehalt von allen Mitarbeitern aus der Abteilung mit der ID 42 um 10% erhöht.

Lösungsvorschlag

```
UPDATE Mitarbeiter
SET Gehalt = Gehalt * 1.1
WHERE AbteilungsID = 42;
SELECT * FROM Mitarbeiter;
```

(e) Schreiben Sie eine SQL-Anweisung, welche den Vornamen, die Nachnamen und das Gehalt der sieben bestbezahlten Mitarbeiter aus der Buchhaltung ausgibt. Standardkonforme Sprachkonstrukte, die eine Beschränkung der Ausgabe bewirken, sind erlaubt.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
  m.Gehalt <= n.Gehalt AND
  a.AbteilungsID = m.AbteilungsID AND
  a.AbteilungsID = n.AbteilungsID AND
  a.Bezeichnung = 'Buchhaltung'
GROUP BY m.Vorname, m.Nachname, m.Gehalt
HAVING COUNT(*) <= 7
ORDER BY m.Gehalt DESC;
```

(f) Schreiben Sie eine SQL-Anweisung, die für jede Abteilung die Mitarbeiter ermittelt, die am wenigsten verdienen. Dabei sollen Vorname, Nachname und die Abteilungsbezeichnung der Mitarbeiter ausgegeben werden.

Lösungsvorschlag

```sql
SELECT m.Vorname, m.Nachname, m.Gehalt, a.Bezeichnung
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
  m.Gehalt >= n.Gehalt AND
  m.AbteilungsID = n.AbteilungsID AND
  m.AbteilungsID = a.AbteilungsID
GROUP BY m.Vorname, m.Nachname, m.Gehalt, m.AbteilungsID, a.Bezeichnung
HAVING COUNT(*) <= 1
ORDER BY m.Gehalt DESC;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-4.tex

Übungsaufgabe „Bands“ (SQL, SQL mit Übungsdatenbank)

SQL

Gegeben ist folgende Datenbank:

```
Album (Titel, Typ, Firma, Preis, ANr)
herausgegeben (BName, ANr, Jahr)
Band (BName, Musikrichtung, Gruendungsjahr, Aktiv)
Musiker (Vorname, Name, GebJahr, BName, ID)
```

```sql
CREATE TABLE Album (
  ANr integer PRIMARY KEY,
  Titel VARCHAR(20) NOT NULL,
  Typ VARCHAR(20),
  Firma VARCHAR(20),
  Preis decimal(5,0)
);

CREATE TABLE Band (
  BName VARCHAR(20) PRIMARY KEY,
  Musikrichtung VARCHAR(20),
  Gruendungsjahr integer,
  Aktiv smallint
);

CREATE TABLE herausgegeben (
  ANr integer REFERENCES Album(ANr),
  BName VARCHAR(20) REFERENCES Band(BName),
  Jahr integer,
  PRIMARY KEY (ANr, BName)
);

CREATE TABLE Musiker (
  ID integer PRIMARY KEY,
  Vorname VARCHAR(20),
  Name VARCHAR(20),
  GebJahr integer,
  BName VARCHAR(20) REFERENCES Band(BName)
);

INSERT INTO Album (ANr, Titel, Typ, Firma, Preis) VALUES
  (1, 'Sin after sin', NULL, 'CBS', '16'),
  (2, 'Highway to hell', NULL, 'Atlantic Records', '20'),
  (3, 'Metallica', NULL, 'Electra Records', '17'),
  (4, 'Paranoid', NULL, 'Vertigo Records', '15'),
  (5, 'High Hopes', NULL, 'Col', '14'),
  (6, 'Tyr', NULL, 'I.R.S. Records', '9');

INSERT INTO Band (BName, Musikrichtung, Gruendungsjahr, Aktiv) VALUES
  ('ACDC', 'Hardrock', 1973, 1),
  ('Black Sabbath', 'Hardrock', 1969, 0),
  ('Bruce Springsteen', 'Rock', 1971, 1),
  ('Judas Priest', 'Heavy Metal', 1969, 1),
  ('Lynyrd Skynyrd', 'Southern Rock', 1964, 1),
```

```
  ('Metallica', 'Heavy Metal', 1981, 1);

INSERT INTO herausgegeben (ANr, BName, Jahr) VALUES
  (1, 'Judas Priest', 1977),
  (2, 'ACDC', 1979),
  (3, 'Metallica', 1999),
  (4, 'Black Sabbath', 1970),
  (6, 'Black Sabbath', 1990);

INSERT INTO Musiker (ID, Vorname, Name, GebJahr, BName) VALUES
  (1, 'Ozzy', 'Osbourne', 1948, 'Black Sabbath'),
  (2, 'Bruce', 'Springsteen', 1949, 'Bruce Springsteen'),
  (3, 'Matt', 'Chamberlain', 1967, 'Bruce Springsteen'),
  (4, 'Angus', 'Young', 1955, 'ACDC'),
  (5, 'Kirk', 'Hammett', 1962, 'Metallica'),
  (6, 'Malcom', 'Young', 1953, 'ACDC'),
  (7, 'Robert', 'Trujillo', 1964, 'Metallica');
```

Beantworten Sie folgende Fragen durch geeignete SQL-Anfragen.

(a) Welche Alben wurden von der Firma „Col“ herausgegeben?

Lösungsvorschlag

```
SELECT a.Titel
FROM Album a
WHERE a.Firma = 'Col';
```

(b) Welche Alben wurden 1990 von *„Black Sabbath“* veröffentlicht?

Lösungsvorschlag

```
SELECT a.Titel
FROM ALBUM a, herausgegeben h
WHERE
  a.ANr = h.ANr AND
  h.BName = 'Black Sabbath' AND
  h.Jahr = 1990;
```

(c) Welche Band veröffentlichte das Album *„Sin After Sin“*?

Lösungsvorschlag

```
SELECT h.BName
FROM herausgegeben h, Album a
WHERE
  a.ANr = h.ANr AND
  a.Titel = 'Sin After Sin';
```

(d) In welcher Band spielt „Ozzy Osbourne“?

Lösungsvorschlag

```
SELECT BName
FROM Musiker
WHERE
```

```
  Name = 'Osbourne' AND
  Vorname = 'Ozzy';
```

(e) Welche Bands wurden vor *1980* gegründet, spielen *Hardrock* und sind nicht bei *Col* unter Vertrag?

Lösungsvorschlag

Joins sind teuer: zuerst Bedingungen

```
SELECT DISTINCT b.BName
FROM Band b, herausgegeben h, Album a
WHERE
  b.Musikrichtung = 'Hardrock' AND
  a.Firma != 'Col' AND
  b.Gruendungsjahr < 1980 AND
  b.BName = h.BName AND
  h.ANr = a.ANr;
```

(f) Wie viele Alben mit einem Preis von unter 10€ sind gelistet?

Lösungsvorschlag

Nicht `COUNT(a.Titel)`: doppelte werden nur 1 mal gezählt

```
SELECT COUNT(*) AS Anzahl
FROM Album a
WHERE a.Preis < 10;
```

(g) Welche Musiker spielen in einer Hardrock Band (alphabetisch nach Name)?

Lösungsvorschlag

`DISTINCT`: keine Duplicate. `DISTINCT` ist `GROUP BY` vorzuziehen

```
SELECT DISTINCT m.Name, m.Vorname
FROM Musiker m, Band b
WHERE
  m.BName = b.BName AND
  b.Musikrichtung = 'Hardrock'
ORDER BY m.Name, m.Vorname;
```

(h) Wie viele Alben hat jede Band veröffentlicht (Bandname, Anzahl der Alben)?

Lösungsvorschlag

```
SELECT BName, COUNT(*) AS AnzahlAlben
FROM herausgegeben
GROUP by BName;
```

(i) Gib alle verschiedenen *Namen* der *Musiker* aufsteigend sortiert aus, die in *aktiven* Bands spielen.

Lösungsvorschlag

```
SELECT DISTINCT m.Name, m.Vorname
FROM Band b, Musiker m
WHERE
  b.BName = m.BName AND
  b.aktiv = 1
ORDER By m.Name, m.Vorname ASC;
```

(j) Welche Musiker spielen in einer Band, die keine Alben vor *1970* veröffentlicht hat?

Lösungsvorschlag

```
SELECT DISTINCT m.Name
FROM Musiker m
WHERE
  m.BName NOT IN (SELECT BName FROM herausgegeben WHERE Jahr < 1970);
```

(k) Welche *Musiker* spielen in einer Band, in der es mindestens ein *jüngeres* Bandmitglied gibt?

Lösungsvorschlag

```
SELECT DISTINCT a.Vorname, a.Name
FROM Musiker a, Musiker b
WHERE a.BName = b.BName
AND a.GebJahr < b.GebJahr;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/40_Relationale-Anfragesprachen/10_SQL/Aufgabe_Bands.tex`

Übungsaufgabe „SQL abstrakt“ (SQL)

SQL

Gegeben sind die drei Tabellen TAB1, TAB2 und TAB3:

TAB1

A	B	C
1	2	3
4	5	6
7	8	9

TAB2

A	B
1	4
3	7

TAB3

A	B
1	2
9	6
3	3
9	2
2	7
7	4
9	4
3	7

Geben Sie die Ergebnistabellen der folgenden Aussagen an:

(a)
```
SELECT A FROM TAB3 WHERE B = 4 OR B = 7 ORDER BY A;
```

Lösungsvorschlag

A
2
7
9
3

(b)
```
SELECT * FROM TAB3 WHERE NOT (B = 7) ORDER BY A ASC, B
DESC;
```

Lösungsvorschlag

A	B
1	2
3	3
7	4
9	6
9	4
9	2

(c)
```
SELECT COUNT(DISTINCT A) FROM TAB3 WHERE B >= 3;
```

Lösungsvorschlag

4

(d)
```
SELECT A, COUNT(*), SUM (B), MAX(A), AVG(B) FROM TAB3
GROUP BY A;
```

Lösungsvorschlag

A	Count	SumB	MaxA	AvgB
1	1	2	1	2
2	1	7	2	7
3	2	10	3	5
7	1	4	7	4
9	3	12	9	4

(e)
```
SELECT TAB1.*, TAB2.* FROM TAB1, TAB2;
```

Lösungsvorschlag

TAB1.A	TAB1.B	TAB1.C	TAB2.A	TAB2.B
1	2	3	1	4
4	5	6	1	4
7	8	9	1	4
1	2	3	3	7
4	5	6	3	7
7	8	9	3	7

(f)
```
SELECT TAB1.*, TAB2.* FROM TAB1, TAB2 WHERE TAB1.A =
TAB2.B;
```

Lösungsvorschlag

TAB1.A	TAB1.B	TAB1.C	TAB2.A	TAB2.B
4	5	6	1	4
7	8	9	3	7

(g)
```
SELECT TAB1.*, TAB2.* FROM TAB1, TAB2 WHERE TAB1.A =
TAB2.B AND TAB2.A = 3;
```

Lösungsvorschlag

TAB1.A	TAB1.B	TAB1.C	TAB2.A	TAB2.B
7	8	9	3	7

(h)
```
SELECT TAB1.A, TAB1.C, TAB2.A FROM TAB1, TAB2 WHERE
TAB1.A = TAB2.B AND TAB2.A = 3;
```

Lösungsvorschlag

TAB1.A	TAB1.C	TAB2.A
7	9	3

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/40_Relationale-Anfragesprachen/10_SQL/Aufgabe_SQL-abstract.tex

Relationale Entwurfstheorie

Examensaufgabe „Studentenbibliothek“ (66111-1994-F.A7)

Betrachten Sie das relationale Schema

$$R(\ \mathit{Signatur, Titel, Fachgebiet, Art, ErschOrt, MatrNr, StudName, Gebdatum, StudWohnort, StudFachrichtung, AutNr, AutName, AutWohnort, AutBuchHonorar}\ )$$

und die Menge

$$\begin{aligned}\mathrm{FA} = \Big\{ & \\ & \{\ \mathit{Signatur}\ \} \to \{\ \mathit{Titel, Fachgebiet, Art, ErschOrt}\ \}, \\ & \{\ \mathit{Signatur}\ \} \to \{\ \mathit{MatrNr}\ \}, \\ & \{\ \mathit{MatrNr}\ \} \to \{\ \mathit{StudName, Gebdatum, StudWohnort, StudFachrichtung}\ \}, \\ & \{\ \mathit{AutNr}\ \} \to \{\ \mathit{AutName, AutWohnort}\ \}, \\ & \{\ \mathit{AutNr, Signatur}\ \} \to \{\ \mathit{AutBuchHonorar}\ \}, \\ & \Big\}\end{aligned}$$

Geben Sie eine abhängigkeitserhaltende und verlustfreie Zerlegung von R in 3. Normalform an!

Lösungsvorschlag

(a) **Linksreduktion**

— *Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq \mathit{AttrHülle}(F, \alpha - A)$.* ———————

$\mathrm{AttrHülle}(\mathrm{F}, \{\ \mathit{Autnr}\ \}) = \{\ \mathit{Autnr, AutName, AutWohnort}\ \}$

$\mathrm{AttrHülle}(\mathrm{F}, \{\ \mathit{Signatur}\ \}) = \{\ \mathit{Signatur, Titel, Fachgebiet, Art, ErschOrt, MatrNr, StudName, Gebdatum, StudWohnort, StudFachrichtung}\ \}$

(b) **Rechtsreduktion**

— *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in \mathit{AttrHülle}(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, $\delta\alpha \to \beta$ wird durch $\alpha \to (\beta - B)$ ersetzt.* ———————

$\mathrm{AttrHülle}(\mathrm{F} - \{\ \{\ \mathit{Signatur}\ \} \to \{\ \mathit{MatrNr}\ \}\ \}, \{\ \mathit{Signatur}\ \}) = \{\ \mathit{Signatur, Titel, Fachgebiet, Art, ErschOrt}\ \}$

Es kann nichts weggelassen werden

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66111/1994/03/Aufgabe-7.tex`

Examensaufgabe „Funktionale Abhängigkeiten, Normalisierung“ (66113-2002-H.T2-A2)

Gegeben sei ein Relationenschema R mit Attributen A, B, C, D. Für dieses Relationenschema seien die folgenden Mengen an funktionalen Abhängigkeiten (FDs) gegeben:

(a) $\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B\}, \\ \{B\} \to \{C\}, \\ \{A\} \to \{D\}, \end{array} \right\}$

(b) $\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B\}, \\ \{B\} \to \{C\}, \\ \{C\} \to \{D\}, \\ \{C\} \to \{A\}, \end{array} \right\}$

(c) $\text{FA} = \left\{ \begin{array}{r} \{A, B\} \to \{C\}, \\ \{B\} \to \{D\}, \end{array} \right\}$

(d) $\text{FA} = \left\{ \begin{array}{r} \{A, B\} \to \{C\}, \\ \{A, C\} \to \{D\}, \\ \{A, D\} \to \{B\}, \end{array} \right\}$

(e) $\text{FA} = \left\{ \begin{array}{r} \{A, B\} \to \{C\}, \\ \{A\} \to \{D\}, \\ \{C, D\} \to \{A\}, \end{array} \right\}$

(a) Bestimmen Sie für das Relationschema R für jede der angegebenen Mengen an funktionalen Abhängigkeiten jeweils alle möglichen Schlüssel(-kandidaten)'

Lösungsvorschlag

Abkürzung

A kommt auf keiner rechten Seite der FD's vor. Man kann es über FD's nicht erreichen. A muss also Teil des Schlüsselkandidaten sein.

$$\text{AttrHülle}(F, \{A\}) = R \to \textit{Superschlüssel}$$

A ist minimal, deshalb handelt es bei A um einen Schlüsselkandidat. Jeder weitere Schlüsselkandidat muss ebenfalls minimal sein und zudem A enthalten. Daraus folgt, dass A der einzige Schlüsselkandidat ist.

Mit Hilfe des Algorithmus:

$Test = \{\{ A, B, C, D \}\}$ $Erg = \{\}$

(i) $K = \{A, B, C, D\}$
$K\backslash A$: AttrHülle$(F, \{B, C, D\}) = \{B, C, D\}$!

$K\backslash B$: AttrHülle$(F, \{A, C, D\}) = R$
$\rightarrow Test = \{\{ A, C, D \}\}$

$K\backslash C$: AttrHülle$(F, \{A, B, D\}) = R$
$\rightarrow Test = \{\{ A, C, D \}, \{ A, B, D \}\}$

$K\backslash D$: AttrHülle$(F, \{A, B, C\}) = R$
$\rightarrow Test = \{\{ A, C, D \}, \{ A, B, D \}, \{ A, B, C \}\}$

(ii) $K = \{A, C, D\}$
$K\backslash A$: AttrHülle$(F, \{C, D\}) = \{C, D\}$!

$K\backslash C$: AttrHülle$(F, \{A, D\}) = R$
$\rightarrow Test = \{\{ A, D \}, \{ A, C, D \}, \{ A, B, D \}, \{ A, B, C \}\}$

$K\backslash C$: AttrHülle$(F, \{A, C\}) = R$
$\rightarrow Test = \{\{ A, C \}, \{ A, D \}, \{ A, C, D \}, \{ A, B, D \}, \{ A, B, C \}\}$

(iii) $K = \{A, C\}$
$K\backslash A$: AttrHülle$(F, \{C\}) = \{C\}$!

$K\backslash C$: AttrHülle$(F, \{A\}) = R$
$\rightarrow Test = \{\{ A \}, \{ A, D \}, \{ A, C, D \}, \{ A, B, D \}, \{ A, B, C \}\}$

(iv) $K = \{A\}$
$K\backslash A$: ! $\rightarrow$ kein Superschlüssel ohne A mehr möglich
$\rightarrow$ dieses K wandert in Ergebnis und wird in Test gelöscht
$\rightarrow Test = \{\{ A, D \}, \{ A, C, D \}, \{ A, B, D \}, \{ A, B, C \}\}$
$\rightarrow Erg = \{\{ A \}\}$

analog verfahren wir mit den übrigen Mengen in Test, wie man bereits sieht bleibt $\{A\}$ einziger Schlüsselkandidat.

(b) Geben Sie für jede der Mengen an funktionalen Abhängigkeiten an, ob das Relationenschema R in 2. Normalform (2NF) und ob es in 3. Normalform (3NF) ist. Begründen Sie dies jeweils kurz!

(c) Für die Fälle, in denen R nicht in 2NF bzw. 3NF ist, geben Sie bitte neue Relatio-

nenschemata in 3NF an! Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz!

(d) Untersuchen Sie für die Fälle d) und e), ob das Relationenschema in Boyce-Codd-Normalform (BCNEF) ist! Geben Sie jeweils eine kurze Begründung an! Wenn das Relationenschema nicht in BCNF ist, erläutern Sie, ob eine Zerlegung in eine seman- tisch äquivalente Menge an Relationenschemata in BCNF möglich ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66113/2002/09/Thema-2/Aufgabe-2.tex`

Examensaufgabe „Nachteile vollständige Normalisierung“ (66113-2003-H.T2-A1)

Normalformen

Gegeben sei die folgenden Datenbank mit den offenen Rechnungen der Kunden eines Versandhauses:

RNR	KDNR	Name	Adresse	Positionen	Datum	Betrag
1	1	Müller	München	3	01.11.2002	60
2	1	Müller	München	2	23.05.2003	90
3	2	Huber	Nürnberg	3	09.03.2003	90
4	2	Huber	Nürnberg	8	14.02.2003	70
5	3	Meier	Augsburg	7	20.06.2003	110
6	4	Meier	München	12	07.04.2003	90

(a) Erläutern Sie, warum nur Relationen mit einem zusammengesetzten Schlüsselkandidaten die 2. Normalform verletzen können!

Lösungsvorschlag

Ist der Schlüsselkandidat ein-elementig, so müssen sämtliche Attribute zwangsläufig voll funktional von diesem Schlüsselkandidaten abhängig sein. Dies ist genau die Voraussetzung für die 2. NF, sodass die 2. NF bei atomaren Attributwerten und nur ein-elementigen Schlüsselkandidaten immer gegeben ist. Bei zusammengesetzten Schlüsselkandidaten kann die 2. NF hingegen verletzt werden, da es sein kann, dass ein Nicht-Schlüsselattribut nur von Schlüsselkandidaten abhängig ist.

(b) Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten (einschließlich der transitiven) an?

Lösungsvorschlag

- `RNR` $\rightarrow$ `KDNR, Name, Adresse, Positionen, Datum, Betrag`
- `KDNR` $\rightarrow$ `Name, Adresse`

(c) Erläutern Sie, inwiefern obiges Schema die 3. Normalform verletzt! Zeigen Sie anhand obiger Relation „Rechnung“ zwei mögliche Anomalien auf, die bei fehlender Normalisierung auftreten können.

Lösungsvorschlag

Die Attribute `Name` und `Adresse` sind transitiv (`RNR` $\rightarrow$ `KDNR` $\rightarrow$ `Name, Adresse`) vom Schlüssel `RNR` abhängig!

Mögliche Anomalien:

UPDATE-Anomalie: Müller zieht nach Regensburg, müsste in jedem Tupel geändert werden, wird aber bei `RNR` 2 vergessen $\rightarrow$ Inkonsistenz

Synthese-Algorithmus

INSERT-Anomalie: Neuer (potentieller) Kunde Schmidt kann erst eingefügt werden, wenn auch eine offene Rechnung vorliegt

DELETE-Anomalie: Wird `RNR 6` gelöscht, gehen auch die Kundendaten von Meier aus München verloren.

(d) Überführen Sie das obige Relationenschema in die 3. Normalform! Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz!

Lösungsvorschlag

Rechnung:

RNR	KDNR	Positionen	Datum	Betrag
1	1	3	01.11.2002	60
2	1	2	23.05.2003	90
3	2	3	09.03.2003	90
4	2	8	14.02.2003	70
5	3	7	20.06.2003	110
6	4	12	07.04.2003	90

Kunde:

KDNR	Name	Adresse
1	Müller	München
1	Müller	München
2	Huber	Nürnberg
2	Huber	Nürnberg
3	Meier	Augsburg
4	Meier	München

Die transitiven Abhängigkeiten sind zu entfernen, dadurch wird die neue Relation „Kunde“ mit KDNR als Primärschlüssel geschaffen.

Erläutern Sie, inwiefern sich eine vollständige Normalisierung nachteilig auf die Geschwindigkeit der Anfragebearbeitung auswirken kann und wie darauf reagiert werden kann!

Lösungsvorschlag

Durch die vielen Tabellen sind schon bei einfacheren Anfragen schnell Joins notwendig, was bei komplexeren Anfragen und großen Datenmengen zu einigem Rechenaufwand führen kann. Hier ist es sinnvoll, zuerst eine Selektion zu treffen, anstatt in einem einfachen Kreuzprodukt auch sämtliche sinnlose Tupel miteinander zu verknüpfen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66113/2003/09/Thema-2/Aufgabe-1.tex`

Examensaufgabe „Wareneingänge“ (66116-2012-F.T1-TA1-A2)

Normalformen

Gegeben sei folgende Datenbank für Wareneingänge eines Warenlagers. Die Primärschlüssel-Attribute sind unterstrichen.

ZulieferungsNr	ArtikelNr	Datum	Artikelname	Menge
1	1	01.01.2009	Handschuhe	5
1	2	01.01.2009	Mütze	10
2	3	05.01.2009	Schal	2
2	1	05.01.2009	Handschuhe	18
3	4	06.01.2009	Jacke	2

(a) Erläutern Sie, inwiefern obiges Schema die 3. Normalform verletzt.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(b) Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten (einschließlich der transitiven) an.

Lösungsvorschlag

Lösungsvorschlag

Exkurs: Voll funktionale Abhängigkeit

Eine vollständig funktionale Abhängigkeit liegt dann vor, wenn dass Nicht-Schlüsselattribut nicht nur von einem Teil der Attribute eines zusammengesetzten Schlüsselkandidaten funktional abhängig ist, sondern von allen Teilen eines Relationstyps. Die vollständig funktionale Abhängigkeit wird mit der 2. Normalform (2NF) erreicht. [a]

[a] datenbank-verstehen.de

Lösungsvorschlag

Exkurs: Transitive Abhängigkeit

Eine transitive Abhängigkeit liegt dann vor, wenn Y von X funktional abhängig und Z von Y, so ist Z von X funktional abhängig. Diese Abhängigkeit ist transitiv. Die transitive Abhängigkeit wird mit 3. Normalform (3NF) erreicht. [a]

[a] datenbank-verstehen.de

$$\text{FA} = \left\{$$

$$
\begin{aligned}
\{\, \mathit{ZulieferungsNr} \,\} &\rightarrow \{\, \mathit{Datum} \,\}, \\
\{\, \mathit{ArtikelNr} \,\} &\rightarrow \{\, \mathit{Artikelname} \,\}, \\
\{\, \mathit{ZulieferungsNr}, \mathit{ArtikelNr} \,\} &\rightarrow \{\, \mathit{Menge} \,\}, \\
&\Big\}
\end{aligned}
$$

(c) Überführen Sie das obige Relationenschema in die 3. Normalform. Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2012/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Relation A-F“ (66116-2015-F.T1-TA1-A3)

Normalformen

Gegeben sei das Relationenschema R=(U, F) mit der Attributmenge

$$\{ U \}\text{A, B, C, D, E}$$

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$\begin{aligned} \text{FA} = \Big\{ & \\ \{ A \} &\rightarrow \{ B \}, \\ \{ A, B, C \} &\rightarrow \{ D \}, \\ \{ D \} &\rightarrow \{ B, C \}, \\ & \Big\} \end{aligned}$$

(a) Geben Sie alle Schlüssel für das Relationenschema R (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.

(b) Ist R in 3NF bzw. in BCNF? Geben Sie jeweils eine Begründung an.

(c) Geben Sie eine Basis G von F an. Zerlegen Sie R mittels des Synthesealgorithmus in ein 3NF-Datenbankschema. Es genügt, die resultierenden Attributmengen anzugeben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Relation A-H“ (66116-2015-H.T1-TA1-A2)

Normalformen

Gegeben sei folgendes verallgemeinerte Relationenschema in 1. Normalform:

$$R(\,A,B,C,D,E,F,G,H\,)$$

Für R soll die folgende Menge FD von funktionalen Abhängigkeiten gelten:

$$\mathrm{FA} = \Big\{ \begin{aligned} \{\,F\,\} &\to \{\,E\,\},\\ \{\,A\,\} &\to \{\,B, D\,\},\\ \{\,A, E\,\} &\to \{\,D\,\},\\ \{\,A\,\} &\to \{\,E, F\,\},\\ \{\,A, G\,\} &\to \{\,H\,\}, \end{aligned} \Big\}$$

Bearbeiten Sie mit diesen Informationen folgende Teilaufgaben. Vergessen Sie dabei nicht Ihr Vorgehen stichpunktartig zu dokumentieren und zu begründen.

(a) Bestimmen Sie alle Schlüsselkandidaten von R. Begründen Sie stichpunktartig, warum es außer den von Ihnen gefundenen Schlüsselkandidaten keine weiteren geben kann.

(b) Ist R in 2NF, 3NF?

(c) Berechnen Sie eine kanonische Überdeckung von FD. Es genügt, wenn Sie für jeden der vier Einzelschritte die Menge der funktionalen Abhängigkeiten als Zwischenergebnis angeben.

(d) Bestimmen Sie eine Zerlegung von R in 3NF. Wenden Sie hierfür den Synthesealgorithmus an.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/09/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Entwurfstheorie“ (66116-2017-F.T2-TA1-A5)

Synthese-Algorithmus

In der folgenden Datenbank sind die Ausleihvorgänge einer Bibliothek gespeichert:

| Ausleihe | LNr | Name | Adresse | BNr | Titel | Kategorie | ExemplarNr | 1 | Müller | Winklerstr. 1 Datenbanksysteme | Informatik 1 1 | Miller | Winklerstr. 1 | Datenbanksysteme | Informatik 2 2 | Huber | Friedrichstr. | 2 Anatomie I Medizin 5 2 Huber | Friedrichstr. 3 Harry Potter Literatur 20 3 Meier | Bismarkstr. 4 OODBS Informatik 1 4 Meier Marktpl. 5 | Pippi Langstrumpf | Literatur 1

Für die Datenbank gilt:

Jeder Leser hat eine eindeutige Lesernummer (LNr), einen Namen und eine Adresse. Ein Buch hat eine Buchnummer (BNr), einen Titel und eine Kategorie. Es kann mehrere Exemplare eines Buches geben, welche durch eine, innerhalb einer Buchnummer eindeutigen, Exemplarnummer unterschieden werden.

(a) Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.

(b) Nachfolgend sind alle nicht-trivialen funktionalen Abhängigkeiten, welche in der obigen Datenbank gelten, angegeben:

$$
\begin{aligned}
\mathrm{FA} = \Big\{ & \\
& \{\, \mathit{LNr} \,\} \rightarrow \{\, \mathit{Name} \,\}, \\
& \{\, \mathit{LNr} \,\} \rightarrow \{\, \mathit{Adresse} \,\}, \\
& \{\, \mathit{BNr} \,\} \rightarrow \{\, \mathit{Titel} \,\}, \\
& \{\, \mathit{BNr} \,\} \rightarrow \{\, \mathit{Kategorie} \,\}, \\
& \{\, \mathit{LNr, BNr, ExemplarNr} \,\} \rightarrow \{\, \mathit{Name, Adresse, Titel, Kategorie} \,\}, \\
& \Big\}
\end{aligned}
$$

Einziger Schlüsselkandidat ist { *LNr, BNr, ExemplarNr* }. Überführen Sie das Schema mit Hilfe des Synthesealgorithmus für 3NF in die dritte Normalform.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \rightarrow \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq \mathit{AttrHülle}(F, \alpha - A)$. ———

$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus LNr\}) = \{Titel, Kategorie\}$$
$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus BNr\}) = \{Name, Adresse\}$$
$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus \boldsymbol{ExemplarNr}\}) = \{\boldsymbol{Name, Adresse, Titel, Kategorie}\}$$

$$\text{FA} = \Big\{ \{LNr\} \to \{Name\}, \{LNr\} \to \{Adresse\}, \{BNr\} \to \{Titel\}, \{BNr\} \to \{Kategorie\}, \{LNr, BNr\} \to \{Name, Adresse, Titel, Kategorie\}, \Big\}$$

ii. **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt. ———

$$\text{AttrHülle}(FA - (\{LNr\} \to \{Name\}) \cup (\{LNr\} \to \{\varnothing\}), \{LNr\}) = \{Adresse\}$$
$$\text{AttrHülle}(FA - (\{LNr\} \to \{Adresse\}) \cup (\{LNr\} \to \{\varnothing\}), \{LNr\}) = \{Name\}$$
$$\text{AttrHülle}(FA - (\{BNr\} \to \{Titel\}) \cup (\{BNr\} \to \{\varnothing\}), \{BNr\}) = \{Kategorie\}$$
$$\text{AttrHülle}(FA - (\{BNr\} \to \{Kategorie\}) \cup (\{BNr\} \to \{\varnothing\}), \{BNr\}) = \{Titel\}$$
$$\text{AttrHülle}(FA - (\{LNr, BNr\} \to \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \to \{Adresse, Titel, \{Name, Adresse, Titel, Kategorie\}$$
$$\text{AttrHülle}(FA - (\{LNr, BNr\} \to \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \to \{Name, Titel, K \{Name, Adresse, Titel, Kategorie\}$$
$$\text{AttrHülle}(FA - (\{LNr, BNr\} \to \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \to \{Name, Adresse \{Name, Adresse, Titel, Kategorie\}$$
$$\text{AttrHülle}(FA - (\{LNr, BNr\} \to \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \to \{Name, Adresse \{Name, Adresse, Titel, Kategorie\}$$

$$\text{FA} = \Big\{ \begin{aligned} \{ LNr \} &\to \{ Name \}, \\ \{ LNr \} &\to \{ Adresse \}, \\ \{ BNr \} &\to \{ Titel \}, \\ \{ BNr \} &\to \{ Kategorie \}, \\ \{ LNr, BNr \} &\to \{ \varnothing \}, \end{aligned} \Big\}$$

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* —

$$\text{FA} = \Big\{ \begin{aligned} \{ LNr \} &\to \{ Name \}, \\ \{ LNr \} &\to \{ Adresse \}, \\ \{ BNr \} &\to \{ Titel \}, \\ \{ BNr \} &\to \{ Kategorie \}, \end{aligned} \Big\}$$

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* —

$$\text{FA} = \Big\{ \begin{aligned} \{ LNr \} &\to \{ Name, Adresse \}, \\ \{ BNr \} &\to \{ Titel, Kategorie \}, \end{aligned} \Big\}$$

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

(iii) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* —

(iv) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* —

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/03/Thema-2/Teilaufgabe-1/Aufgabe-5.tex`

Examensaufgabe „Relation A-F“ (66116-2019-F.T1-TA1-A3)

Gegeben sei folgendes relationales Schema R in erster Normalform:

$$\mathrm{R} : \{[\ \mathrm{A, B, C, D, E, F}\ ]\}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:

$$\mathrm{FA} = \Big\{ \begin{array}{r} \{ A, D, F \} \to \{ E \}, \\ \{ B, C \} \to \{ A, E \}, \\ \{ D \} \to \{ B \}, \\ \{ D, E \} \to \{ C, B \}, \\ \{ A \} \to \{ F \}, \end{array} \Big\}$$

(a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von R mit FD. *Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.*

Lösungsvorschlag

- $\{ D, A \}$
- $\{ D, C \}$
- $\{ D, E \}$

(b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

Lösungsvorschlag

R ist in 1NF, da $\{ d \} \to \{ b \}$

(c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FD_C von FD:

(i) Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD_L).

Lösungsvorschlag

Linksreduktion

— *Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$.* ————————

$\{ A, D, F \} \to \{ E \}$

$E \notin \text{AttrHülle}(F, \{A, D, F \setminus A\}) = \{D, F, B\}$
$E \notin \text{AttrHülle}(F, \{A, D, F \setminus D\}) = \{A, F\}$
$E \in \text{AttrHülle}(F, \{A, D, F \setminus F\}) = \{A, B, D, \boldsymbol{F}\}$

$\{ B, C \} \to \{ A, E \}$

$\{A, E\} \notin \text{AttrHülle}(F, \{B, C \setminus B\}) = \{C\}$
$\{A, E\} \notin \text{AttrHülle}(F, \{B, C \setminus C\}) = \{B\}$

$\{\, \boldsymbol{D, E} \,\} \rightarrow \{\, \boldsymbol{C, B} \,\}$

$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus D\}) = \{E\}$
$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus E\}) = \{B, D\}$

$$\text{FA} = \Big\{ \begin{array}{r} \{\, A, D \,\} \rightarrow \{\, E \,\}, \\ \{\, B, C \,\} \rightarrow \{\, A, E \,\}, \\ \{\, D \,\} \rightarrow \{\, B \,\}, \\ \{\, D, E \,\} \rightarrow \{\, C, B \,\}, \\ \{\, A \,\} \rightarrow \{\, F \,\}, \end{array} \Big\}$$

(ii) Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD_L) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD_R).

Lösungsvorschlag

Rechtsreduktion

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \rightarrow \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in \mathit{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \rightarrow (\beta - B)$ ersetzt. —

E

$E \notin \text{AttrHülle}(F \setminus \{\, A, D \,\} \rightarrow \{\, E \,\}, \{A, D\}) = \{A, B, D, F\}$
$E \notin \text{AttrHülle}(F \setminus \{\, B, C \,\} \rightarrow \{\, A, E \,\} \cup \{\, B, C \,\} \rightarrow \{\, A \,\}, \{B, C\}) = \{A, B, C, F\}$

B

$B \notin \text{AttrHülle}(F \setminus \{\, D \,\} \rightarrow \{\, B \,\}, \{D\}) = \{D\}$
$B \in \text{AttrHülle}(F \setminus \{\, D, E \,\} \rightarrow \{\, C, B \,\} \cup \{\, D, E \,\} \rightarrow \{\, C \,\}, \{D, E\}) = \{\boldsymbol{B}, D, E\}$

$$\text{FA} = \Big\{ \begin{array}{r} \{\, A, D \,\} \rightarrow \{\, E \,\}, \\ \{\, B, C \,\} \rightarrow \{\, A, E \,\}, \\ \{\, D \,\} \rightarrow \{\, B \,\}, \\ \{\, D, E \,\} \rightarrow \{\, C \,\}, \\ \{\, A \,\} \rightarrow \{\, F \,\}, \end{array} \Big\}$$

(iii) Bestimmen Sie eine kanonische Überdeckung $FD.$ von FD auf Basis des Ergebnisses der Rechtsreduktion (FD_R). Synthese-Algorithmus

Lösungsvorschlag

- **Löschen leerer Klauseln**
  — *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* —

  $\varnothing$ Nichts zu tun
- **Vereinigung**
  — *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* —

  $\varnothing$ Nichts zu tun

(d) Zerlegen Sie R mit FD_C mithilfe des Synthesealgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

Lösungsvorschlag

- **Relationsschemata formen**
  — *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

  $R_1(\underline{A, D}, E)$
  $R_2(\underline{B, C}, A, E)$
  $R_3(\underline{D}, B)$
  $R_4(\underline{D, E}, C)$
  $R_5(\underline{A}, F)$
- **Schlüssel hinzufügen**
  — *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* —

  $\varnothing$ Nichts zu tun
- **Entfernung überflüssiger Teilschemata**
  — *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* —

  $\varnothing$ Nichts zu tun

(e) Prüfen Sie für alle Relationen der Zerlegung aus d), ob sie jeweils in BCNF sind.

Lösungsvorschlag

R1 und R4 sind in BCNF, weil ihre Determinanten Schlüsselkandidaten sind.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Normalisierung“ (66116-2019-F.T2-TA1-A6)

Normalformen
Schlüsselkandidat
Synthese-Algorithmus

Gegeben sei das Relationsschema $R(A, B, C, D, E, F)$, sowie die Menge der zugehörigen funktionalen Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{aligned} \{B\} &\to \{F\}, \\ \{C, D\} &\to \{E\}, \\ \{C\} &\to \{A\}, \\ \{C, D\} &\to \{A\}, \\ \{D\} &\to \{F\}, \\ \{D\} &\to \{B\}, \end{aligned} \Big\}$$

(a) Bestimmen Sie den Schlüsselkandidaten der Relation R und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

Lösungsvorschlag

Der Schlüsselkandidat ist $\{C, D\}$, da $\{C, D\}$ auf keiner rechten Seiten der Funktionalen Abhängigkeiten vorkommt. Außerdem ist $\{C, D\}$ ein Superschlüssel da gilt: $\text{AttrHülle}(F, \{C, E\}) = \{A, B, C, D, E, G\} = R$

(b) Überführen Sie das Relationsschema R mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist. Benennen Sie alle Schritte und begründen Sie eventuelle Reduktionen.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$. ———

$$\text{FA} = \Big\{ \begin{aligned} \{B\} &\to \{F\}, \\ \{C, D\} &\to \{E\}, \\ \{C\} &\to \{A\}, \\ \{C\} &\to \{A\}, \\ \{D\} &\to \{F\}, \\ \{D\} &\to \{B\}, \end{aligned}$$

$\}$

ii. **Rechtsreduktion**

— *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.*

$$\text{FA} = \Big\{ \begin{aligned} \{ B \} &\to \{ F \}, \\ \{ C, D \} &\to \{ E \}, \\ \{ C \} &\to \{ \varnothing \}, \\ \{ C \} &\to \{ A \}, \\ \{ D \} &\to \{ \varnothing \}, \\ \{ D \} &\to \{ B \}, \end{aligned} \Big\}$$

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.*

$$\text{FA} = \Big\{ \begin{aligned} \{ B \} &\to \{ F \}, \\ \{ C, D \} &\to \{ E \}, \\ \{ C \} &\to \{ A \}, \\ \{ D \} &\to \{ B \}, \end{aligned} \Big\}$$

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.*

$\varnothing$ Nichts zu tun

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

- $R_1(\, B, F \,)$
- $R_2(\, C, D, E \,)$
- $R_3(\, C, A \,)$
- $R_4(\, D, B \,)$

(iii) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$*

$\varnothing$ Nichts zu tun

(iv) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* —

∅ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-2/Teilaufgabe-1/Aufgabe-6.tex`

Examensaufgabe „Sekretäre“ (66116-2020-F.T1-TA2-A2)

Relation „Sekretäre“

PersNr	Name	Boss	Raum
4000	Freud	2125	225
4000			225
4020	Röntgen	2163	6
4020	Röntgen		26
4030	Galileo	2127	
	Freud	2137	80

Gegeben sei oben stehenden (lückenhafte) Relationenausprägung **Sekretäre** sowie die folgenden funktionalen Abhängigkeiten:

$$\mathrm{FA} = \Big\{ \begin{aligned} \{\, \mathit{PersNr} \,\} &\rightarrow \{\, \mathit{Name} \,\}, \\ \{\, \mathit{PersNr, Boss} \,\} &\rightarrow \{\, \mathit{Raum} \,\}, \end{aligned} \Big\}$$

Geben Sie für alle leeren Zellen Werte an, so dass keine funktionalen Abhängigkeiten verletzt werden. (Hinweis: Es gibt mehrere richtige Antworten.)

Lösungsvorschlag

PersNr	Name	Boss	Raum
4000	Freud	2125	225
4000	*Freud*	*2143*[a]	225
4020	Röntgen	2163	6
4020	Röntgen	*2163*[b]	26
4030	Galileo	2127	*27*[c]
4000	Freud	2137	80

[a] Muss eine andere Boss-ID sein, sonst gäbe es zwei identische Zeilen.
[b] anderer Boss
[c] anderer Raum

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Relation A-F“ (66116-2020-F.T1-TA2-A4)

Synthese-Algorithmus

Gegeben sei die Relation

$$R(\,A, B, C, D, E, F\,)$$

mit den FDs

$$\text{FA} = \Big\{ \begin{aligned} \{\,A\,\} &\to \{\,B, C, F\,\}, \\ \{\,B\,\} &\to \{\,A, B, F\,\}, \\ \{\,C, D\,\} &\to \{\,E, F\,\}, \end{aligned} \Big\}$$

(a) Geben Sie alle Kandidatenschlüssel an.

Lösungsvorschlag

- $\{\,A, D\,\}$
- $\{\,B, D\,\}$

(b) Überführen Sie die Relation mittels Synthesealgorithmus in die 3. NF. Geben Sie alle Relationen in der 3. NF an und **unterstreichen Sie in jeder einen Kandidatenschlüssel.** — Falls Sie Zwischenschritte notieren, machen Sie das Endergebnis **klar kenntlich.**

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———————

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$. ———

$\boldsymbol{\{\,C, D\,\} \to \{\,E, F\,\}}$

$\{E, F\} \notin \text{AttrHülle}(F, \{C, D \setminus D\}) = \{C\}$
$\{E, F\} \notin \text{AttrHülle}(F, \{C, D \setminus C\}) = \{D\}$

$$\text{FA} = \Big\{ \begin{aligned} \{\,A\,\} &\to \{\,B, C, F\,\}, \\ \{\,B\,\} &\to \{\,A, B, F\,\}, \\ \{\,C, D\,\} &\to \{\,E, F\,\}, \end{aligned} \Big\}$$

ii. **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt. ———————

F

$F \in \text{AttrHülle}(F \setminus \{A\} \to \{B, C, F\} \cup \{A\} \to \{B, C\}, \{A\}) = \{A, B, C, \boldsymbol{F}\}$

$$\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B, C\}, \\ \{B\} \to \{A, B, F\}, \\ \{C, D\} \to \{E, F\}, \end{array} \right\}$$

$F \notin \text{AttrHülle}(F \setminus \{B\} \to \{A, B, F\} \cup \{B\} \to \{A, B\}, \{B\}) = \{A, B, C\}$
$F \notin \text{AttrHülle}(F \setminus \{C, D\} \to \{E, F\} \cup \{C, D\} \to \{E\}, \{C, D\}) = \{C, D, E\}$

B

$B \notin \text{AttrHülle}(F \setminus \{A\} \to \{B, C\} \cup \{A\} \to \{C\}, \{A\}) = \{A, C\}$
$B \in \text{AttrHülle}(F \setminus \{B\} \to \{A, B, F\} \cup \{B\} \to \{A, F\}, \{B\}) = \{A, \boldsymbol{B}, F\}$

$$\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B, C\}, \\ \{B\} \to \{A, F\}, \\ \{C, D\} \to \{E, F\}, \end{array} \right\}$$

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ———

$\varnothing$ Nichts zu tun

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ———

$\varnothing$ Nichts zu tun

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

$R_1(\underline{A, B}, C)$
$R_2(\underline{A, B}, F)$
$R_3(\underline{C, D}, E, F)$

(iii) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ———

$R_1(\underline{A, B}, C)$
$R_2(\underline{A, B}, F)$

$R_3(\underline{C, D}, E, F)$
$R_4(\underline{A, D})$

(iv) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* —

∅ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Schlüssel“ (66116-2020-F.T1-TA2-A5)

Schlüssel

Gegeben sei die Relation $R(A,B,C)$

(a) Schreiben Sie eine SQL-Anfrage, mit der sich zeigen lässt, ob das Paar A, B ein Superschlüssel der Relation R ist. Beschreiben Sie ggf. textuell - falls nicht eindeutig ersichtlich - wie das Ergebnis Ihrer Anfrage interpretiert werden muss, um zu erkennen ob A, B ein Superschlüssel ist.

Lösungsvorschlag

Diese Anfrage darf keine Ergebnisse liefern, dann ist das Paar A, B ein Superschlüssel.

```
SELECT *
FROM R
GROUP BY A, B
HAVING COUNT(*) > 1;
```

(b) Erläutern Sie den Unterschied zwischen einem Superschlüssel und einem Kandidatenschlüssel. Tipp: Was muss gelten, damit A, B ein Kandidatenschlüssel ist und nicht nur ein Superschlüssel?

Lösungsvorschlag

Ein Superschlüssel ist ein Attribut oder eine Attributkombination, von der *alle Attribute* einer Relation funktional *abhängen*.

Ein Kandidatenschlüssel ist ein *minimaler* Superschlüssel. Keine Teilmenge dieses Superschlüssels ist ebenfalls Superschlüssels.

(c) Sei A, B der Kandidatenschlüssel für die Relation R. Geben Sie eine minimale Ausprägung der Relation R an, die diese Eigenschaft erfüllt.

Lösungsvorschlag

A	B	C
1	2	3
2	1	4
1	1	5
2	2	5

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-5.tex

Examensaufgabe „Relation A-F“ (66116-2020-F.T2-TA2-A3)

Gegeben sei folgendes relationales Schema R in erster Normalform:

$$\text{R:[A,B,C,D,E,F]}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{array}{c} \{A\} \to \{F\}, \\ \{C, E, F\} \to \{A, B\}, \\ \{A, E\} \to \{B\}, \\ \{B, C\} \to \{D\}, \\ \{A, F\} \to \{C\}, \end{array} \Big\}$$

(a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von R mit FD. Begründen Sie Ihre Antwort. Begründen Sie zudem, warum es keine weiteren Kandidatenschlüssel/Schlüsselkandidaten gibt.

Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

Lösungsvorschlag

E muss in allen Superschlüsseln enthalten sein, denn es steht nicht auf der rechten Seite von FD (*).

D kann in keinem Schlüsselkandidaten vorkommen, denn es steht nur auf der rechten Seite von FD (**).

E allein ist kein Schlüsselkandidat (***).

AE führt über FD zu B, A zu F, AF zu C und BC zu D, also ist AE ein Superschlüssel und damit wegen (*) und (***) ein Schlüsselkandidat. Wegen (*) enthält jeder Superschlüssel, der A enthält, AE. Also ist kein weiterer Superschlüssel, der A enthält, ein Schlüsselkandidat (****).

BE, CE und EF sind keine Superschlüssel, also auch keine Schlüsselkandidaten.

BCE ist kein Superschlüssel, da A und F nicht erreicht werden können.

BEF ist kein Superschlüssel, da A, D und F nicht erreicht werden können.

CEF führt über FD zu AB, BC führt dann zu D, also ist CEF ein Superschlüssel. Wegen (*), (**) und weil CE und EF keine Superschlüssel sind, ist CEF ein Schlüsselkandidat.

Das waren alle dreielementigen Buchstabenkombinationen, die (*), (**) und (****) genügen. Vierelementig ist nur BCEF und das enthält CEF, ist also kein Schlüsselkandidat.

Die einzigen Schlüsselkandidaten sind folglich AE und CEF.

(b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

Lösungsvorschlag

R mit FD ist nicht in 2NF, denn bei Wahl des Schlüsselkandidaten AE hängt F von A, also nur einem Teil des Schlüssels, ab. Also ist AE $\rightarrow$ F nicht voll funktional. Damit ist R mit FD auch nicht in 3NF, denn 3NF $\subseteq$ 2NF.

(c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FDc von FD. Begründen Sie jede Ihrer Entscheidungen:

(i) Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD;).

Lösungsvorschlag

$$\text{FA} = \Big\{ \begin{array}{r} \{A\} \rightarrow \{F\}, \\ \{C, E, F\} \rightarrow \{A, B\}, \\ \{A, E\} \rightarrow \{B\}, \\ \{B, C\} \rightarrow \{D\}, \\ \{A\} \rightarrow \{C\}, \end{array} \Big\}$$

(ii) Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD;) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD).

Lösungsvorschlag

$$\text{FA} = \Big\{ \begin{array}{r} \{A\} \rightarrow \{F\}, \\ \{C, E, F\} \rightarrow \{A\}, \\ \{A, E\} \rightarrow \{B\}, \\ \{B, C\} \rightarrow \{D\}, \\ \{A\} \rightarrow \{C\}, \end{array} \Big\}$$

(iii) Bestimmen Sie eine kanonische Überdeckung FD. von FD auf Basis des Ergebnisses der Rechtsreduktion (FD).

Lösungsvorschlag

$$\mathrm{FA} = \Big\{ \begin{aligned} \{ A \} &\to \{ F, C \}, \\ \{ C, E, F \} &\to \{ A \}, \\ \{ A, E \} &\to \{ B \}, \\ \{ B, C \} &\to \{ D \}, \end{aligned} \Big\}$$

(d) Zerlegen Sie R mit FDc mithilfe des Synthesealgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

(e) Prüfen Sie für alle Relationen der Zerlegung aus 4., ob sie jeweils in BCNF sind.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Entwurfstheorie“ (66116-2020-H.T2-TA2-A4)

Normalformen

Gegeben ist das folgende Relationenschema R in erster Normalform.
R:[A,B, C, D, E, F]
Für R gelte folgende Menge FD funktionaler Abhängigkeiten:

$$\mathrm{FA} = \Big\{ \begin{aligned} \{ AC \} &\rightarrow \{ DE \}, \\ \{ ACE \} &\rightarrow \{ B \}, \\ \{ E \} &\rightarrow \{ B \}, \\ \{ D \} &\rightarrow \{ F \}, \\ \{ AC \} &\rightarrow \{ F \}, \\ \{ AD \} &\rightarrow \{ F \}, \end{aligned} \Big\}$$

(a) R mit FD hat genau einen Kandidatenschlüssel X. Bestimmen Sie diesen und begründen Sie Ihre Antwort.

Lösungsvorschlag

AC ist der Kandidatenschlüssel. AC kommt in keiner rechten Seite der Funktionalen Abhängigkeiten vor.

(b) Berechnen Sie Schritt für Schritt die Hülle X^+ von $X := \{K\}$.

Lösungsvorschlag

(i) $AC \cup DE$
(ii) $ACDE \cup B$ (ACE -> B)
(iii) $ACDEB$ (E -> B)
(iv) $ACDEB \cup F$ (D -> F)
(v) $ACDEBF$ (AC -> F)
(vi) $ACDEBF$ (AD -> F)

(c) Nennen Sie alle primen und nicht-primen Attribute.

Lösungsvorschlag

prim: AC
nicht-prim: BDEF

(d) Geben Sie die höchste Normalform an, in der sich die Relation befindet. Begründen Sie.

Lösungsvorschlag

2NF

D –> F hängt transitiv von AC ab: AC -> D, D-> F

(e) Gegeben ist die folgende Zerlegung von R:

R1 (A, C, D, E) R2 (B, E) R3 (D, F)

Weisen Sie nach, dass es sich um eine verlustfreie Zerlegung handelt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-4.tex

Examensaufgabe „Normalisierung“ (66116-2021-F.T1-TA2-A4)

Normalformen

Gegeben ist das folgende Relationenschema in erster Normalform, bestehend aus zwei Relationen:

```
Relation1(A, B, C, D, E)
Relation2(F, G, H, A, E)
```

In diesem Schema gelten die folgenden funktionalen Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{aligned} \{A, B\} &\to \{C\}, \\ \{A, B, C\} &\to \{E\}, \\ \{A\} &\to \{D\}, \\ \{F, G\} &\to \{H, A\}, \\ \{G, H\} &\to \{E\}, \end{aligned} \Big\}$$

(a) Nennen Sie die Bedingungen, damit ein Schema in erster Normalform ist.

Lösungsvorschlag

Ein Schema ist in erster Normalform, wenn es ausschließlich atomare Attributwerte aufweist.

(b) Überprüfen Sie, ob das Schema in zweiter Normalform ist.

Lösungsvorschlag

Eine Relation ist in 2NF, wenn sie in 1NF ist und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist. Der Schlüsselkandidat ist (A, B) in Relation 1 sowie (F, G) in Relation 2.

Das Nichtschlüsselartribut D in Relation 1 ist nicht voll funktional abhängig von (A, B), sondern nur von A. Somit ist das Schema nicht in 2NF. Alle anderen Nichtschlüsselattribute sind voll funktional abhängig.

(c) Wenden Sie den Synthesealgorithmus an, um das Schema in ein Schema in dritter Normalform zu überführen.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$. ———

$$\text{FA} = \Big\{$$

$$\left.\begin{aligned} \{A, B\} &\to \{C\}, \\ \{A, B\} &\to \{E\}, \\ \{A\} &\to \{D\}, \\ \{F, G\} &\to \{H, A\}, \\ \{G, H\} &\to \{E\}, \end{aligned}\right\}$$

ii. **Rechtsreduktion**

— *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.* ———

nichts zu tun

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ———

nichts zu tun

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ———

$$\mathrm{FA} = \left\{\begin{aligned} \{A, B\} &\to \{C, E\}, \\ \{A\} &\to \{D\}, \\ \{F, G\} &\to \{H, A\}, \\ \{G, H\} &\to \{E\}, \end{aligned}\right\}$$

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E)

(iii) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ———

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E) R5 (B, F, G)
als Verbindung von R1 bis R4 (Attributhülle erhält alle Attribute, ist daher Schlüsselkandidat)

(iv) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* ———

nichts zu tun

(d) Sei nun das Relationenschema R(A,B,C,D) in erster Normalform gegeben. In R gelten die folgenden funktionalen Abhängigkeiten:

$$\mathrm{FA} = \left\{ \begin{array}{r} \{ A, B \} \rightarrow \{ D \}, \\ \{ B \} \rightarrow \{ C \}, \\ \{ C \} \rightarrow \{ B \}, \end{array} \right\}$$

Welches ist die höchste Normalform, in der sich das Schema R befindet? Begründen Sie Ihre Entscheidung.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Relation Prüfung“ (66116-2021-F.T2-TA2-A6)

Gegeben ist die Relation Prüfung (Prüfungsnummer, Fakultät, Prüfungsname, Dozent, Prüfungstyp, ECTS) mit den beiden Schlüsselkandidaten (Prüfungsnummer, Fakultät) und (Fakultät, Prüfungsname, Dozent).

Alle Attributwerte sind atomar. Es gelten nur die durch die Schlüsselkandidaten implizierten funktionalen Abhängigkeit.

Geben Sie die höchste Normalform an, die die Relation-Prüfung erfüllt. Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind und dass mindestens eine Bedingung der nächsthöheren Normalform verletzt ist. Beziehen Sie sich bei der Begründung auf die gegebene Relation und nennen Sie nicht nur die allgemeinen Definitionen der Normalformen.

Lösungsvorschlag

$$\text{FA} = \Big\{ \{ \textit{Prüfungsnummer, Fakultät} \} \rightarrow \{ \textit{Prüfungsname, Dozent, Prüfungstyp, ECTS} \}, \{ \textit{Fakultät, Prüfungsname, Dozent} \} \rightarrow \{ \textit{Prüfungsnummer, Prüfungstyp, ECTS} \}, \Big\}$$

Höchste Normalform: 4NF
Siehe Taschenbuch Seite 449

1NF Alle Werte sind atomar.

2NF Ist in 1NF und jedes Attribut ist Teil des Schlüsselkandidaten (`Prüfungsnummer`, `Fakultät` oder `Fakultät, Prüfungsname, Dozent`) oder das Attribut ist von einem Schlüsselkandidaten voll funktional abhängig (`Prüfungsname, Dozent, Prüfungstyp, ECTS` oder `Prüfungsnummer, Prüfungstyp, ECTS`).

3NF Ist in 2NF und ein Nichtschlüsselattribut darf nur vom Schlüsselkandidaten abhängen (`Prüfungsname, Dozent, Prüfungstyp, ECTS` hängt von `Prüfungsnummer, Fakultät` ab) und (`Prüfungsnummer, Prüfungstyp, ECTS` hängt von `Fakultät, Prüfungsname, Dozent` ab).

BCNF Jede Determinate ist Schlüsselkandidat (`Prüfungsnummer, Fakultät` und `Fakultät, Prüfungsname, Dozent`).

4NF keine paarweise Unabhängigkeiten mehrwertigen Abhängigkeiten zwischen ihren Attributen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-6.tex`

Übungsaufgabe „Schlüsselkandidat von R“ (Schlüsselkandidat)

Schlüsselkandidat

Gegeben sei die Relation *Abstrakt* mit dem Schema $Abstrakt(A, B, C, D, E)$ und die Menge der funktionalen Abhängigkeiten

$$F = \Big\{ \begin{array}{r} \{A\} \to \{B, C\}, \\ \{C, D\} \to \{E\}, \\ \{A, C\} \to \{E\}, \\ \{B\} \to \{D\}, \end{array} \Big\}$$

.

Bestimmen Sie die Schlüsselkandidaten von Abstrakt!

Lösungsvorschlag

Das Attribut A kommt auf keiner rechten Seite der Funktionalen Abhängigkeiten aus F vor und kann deshalb in keinem Fall durch ein anderes Attribut bestimmt werden. Damit muss A in jedem Schlüsselkandidaten von Abstrakt enthalten sein. Ist A bereits ein Superschlüssel, ist die Menge folglich der (einzig mögliche) Schlüsselkandidat. Wir überprüfen die Superschlüsseleigenschaft mit dem Attributhüllenalgorithmus:

ERG	Begründung
ERG $= \{A\}$	Initialisierung
ERG $= \{A\} \cup \{B, C\}$	$\{A\} \to \{B, C\}$
ERG $= \{A, B, C\}$	$\{C, D\} \to \{E\}$
ERG $= \{A, B, C\} \cup \{E\}$	$\{A, C\} \to \{E\}$
ERG $= \{A, B, C, E\} \cup \{D\}$	$\{B\} \to \{D\}$
ERG $= \{A, B, C, D, E\}$	

ERG $= \{A, B, C, D, E\}$ kann bei einem zweiten Durchlauf nicht mehr ändern, da die Menge bereits alle Attribute von Abstrakt enthält. Die Attributhülle von A über F entspricht der Attributmenge von *Abstrakt*.

$\text{AttrHülle}(F, \{A\}) = \{A, B, C, D, E\} = R$

$\to \{A\}$ ist der Schlüsselkandidat von *Abstrakt*.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/10_Schluessel/Aufgabe_Schlüsselkandidat-von-R.tex

Übungsaufgabe „Arbeitsvermittler“ (Normalformen, Funktionale Abhängigkeiten, Schlüsselkandidat, Synthese-Algorithmus)

Normalformen
Funktionale Abhängigkeiten

Ein privater Arbeitsvermittler legt eine relationale Datenbank an, in der u.a. die folgenden Informationen gespeichert werden:

Zu jeder gemeldeten offenen Stelle werden der Name und die Adresse des Arbeitgebers gespeichert, ebenso die genaue Stellenbezeichnung, das Jahresgehalt und das Datum, ab dem die Stelle zu besetzen ist. Außerdem wird vermerkt, ob es sich um eine befristete oder unbefristete Anstellung handelt. Verschiedene Arbeitgeber können Stellen mit derselben Stellenbezeichnung anbieten, ebenfalls kann ein Arbeitgeber mehrere Niederlassungen haben.

Aus der Adresse kann nicht auf den Namen des Arbeitgebers geschlossen werden (Beispiel Bürohochhaus). Ein Arbeitgeber hat jedoch nicht mehrere Niederlassungen am selben Ort. Außerdem sei der Einfachheit halber vorausgesetzt, dass jeder Arbeitgeber pro Niederlassung nur eine Stelle mit einer bestimmten Bezeichnung zu vergeben hat, und jeder Bewerber nur einmal vermittelt wird. Aus der Stellenbezeichnung lässt sich bereits ersehen, ob die Stelle befristet ist oder nicht. Des Weiteren weist eine Stellenbezeichnung darauf hin, welcher Branchenbezeichnung sie zuzuordnen ist.

Von jedem Arbeitsuchenden werden der Name und die Adresse, die Telefonnummer, der erlernte Beruf und das Geburtsdatum des Bewerbers gespeichert. Zusätzlich soll direkt abrufbar sein, ob der Bewerber bereits älter als 25 Jahre (schwer zu vermitteln!) ist. Als Kriterium gilt dabei der Zeitpunkt der Meldung beim Arbeitsvermittler und nicht der Zeitpunkt eines möglichen Stellenantritts.

Aus diesen Vorgaben ergibt sich (beispielsweise) folgendes relationales Schema (Relationen mindestens in 1.NF):

- Stellen : {[Stellenbezeichnung, AG-Name, AG-Adresse, Gehalt, Einstellungsdatum, befristet]}
- Bewerber : {[Name, Adresse, Gebdatum, Beruf, TelNr, Antrittsdatum, aelter25]}
- Branche : {[Branchenbezeichnung, Bedarf]}
- gehoert_zu : {[Stellenbezeichnung, AG-Name, AG-Adresse, Branchenbezeichnung]}
- vermittelt : {[Stellenbezeichnung, AG-Name, AG-Adresse, Name, Adresse]}

(a) Welche funktionalen Abhängigkeiten gibt es bzgl. der einzelnen Relationen?

Lösungsvorschlag

- **Stellen**:
  $\{ \textit{Stellenbezeichnung}, \textit{AG-Name}, \textit{AG-Adresse} \} \rightarrow \{ \textit{Gehalt} \}$
  $\{ \textit{Stellenbezeichnung}, \textit{AG-Name}, \textit{AG-Adresse} \} \rightarrow \{ \textit{Einstellungsdatum} \}$
  $\{ \textit{Stellenbezeichnung} \} \rightarrow \{ \textit{befristet} \}$
- **Bewerber**:
  $\{ \textit{Name}, \textit{Adresse} \} \rightarrow \{ \textit{Gebdatum} \}$
  $\{ \textit{Name}, \textit{Adresse} \} \rightarrow \{ \textit{Beruf} \}$

Schlüsselkandidat
Synthese-Algorithmus

$\{ Name, Adresse \} \rightarrow \{ TelNr \}$
$\{ Name, Adresse \} \rightarrow \{ Antrittsdatum \}$
$\{ Name, Adresse \} \rightarrow \{ aelter25 \}$
$\{ Name, Gebdatum \} \rightarrow \{ aelter25 \}$

- **Branche**:
  $\{ Branchenbezeichnung \} \rightarrow \{ Bedarf \}$
- **gehoert_zu**:
  $\{ Stellenbezeichnung \} \rightarrow \{ Branchenbezeichnung \}$
- **vermittelt**:
  $\{ Stellenbezeichnung, AG\text{-}Name, AG\text{-}Adresse \} \rightarrow \{ Name \}$
  $\{ Stellenbezeichnung, AG\text{-}Name, AG\text{-}Adresse \} \rightarrow \{ Adresse \}$
  $\{ Name, Adresse \} \rightarrow \{ Stellenbezeichnung \}$
  $\{ Name, Adresse \} \rightarrow \{ AG\text{-}Name \}$
  $\{ Name, Adresse \} \rightarrow \{ AG\text{-}Adresse \}$

(b) Wie lauten die Schlüsselkandidaten der einzelnen Relationen?

Lösungsvorschlag

- **Stellen**:
  $\{ Stellenbezeichnung, AG\text{-}Name, AG\text{-}Adresse \}$
- **Bewerber**:
  $\{ Name, Adresse \}$
- **Branche**:
  $\{ Branchenbezeichnung \}$
- **gehoert_zu**:
  $\{ Stellenbezeichnung, AG\text{-}Name, AG\text{-}Adresse \}$
- **vermittelt**:
  $\{ Stellenbezeichnung, AG\text{-}Name, AG\text{-}Adresse \}, \{ Name, Adresse \}$

(c) Überprüfen Sie, welche Normalformen bei den einzelnen Relationenschemata vorliegen!

(d) Überführen Sie die Relationenschemata in die 3. Normalform!

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/10_Synthesealgorithmus/Aufgabe_Arbeitsvermittler.tex`

Übungsaufgabe „Drei-Schemata“ (Boyce-Codd-Normalform, Dritte Normalform, Zweite Normalform, Synthese-Algorithmus)

Es seien folgende Relationenschemata mit den jeweiligen Mengen funktionaler Abhängigkeiten gegeben:

$S_1(P, Q, R)$ mit

$$F_1 = \left\{ \begin{array}{c} \{P, Q\} \to \{R\}, \\ \{P, R\} \to \{Q\}, \\ \{Q, R\} \to \{P\}, \end{array} \right\}$$

$S_2(P, R, S, T)$ mit

$$F_2 = \left\{ \begin{array}{c} \{P, S\} \to \{T\}, \end{array} \right\}$$

$S_3(P, S, U)$ mit

$$F_3 = \left\{ \begin{array}{c} \end{array} \right\}$$

(a) Welche der drei Schemata sind in BCNF, welche in 3NF, welche in 2NF? Begründe!

Lösungsvorschlag

S_1**:** BCNF

S_2**:** 1NF aber nicht 2NF

S_3**:** BCNF

(S_1, F_1) und (S_3, F_3) sind offenbar in BCNF und daher auch in 3NF und 2NF. (S_2, F_2) ist offenbar nicht in 2NF, da der Schlüsselkandidat PRS ist und T von einem Teil dieser Schlüsselkandidaten, nämlich PS, abhängig ist und daher auch nicht in 3NF oder BCNF.

(b) Wenden Sie auf (S_2, F_2) den Synthesealgorithmus an, und bestimmen Sie auch die Mengen aller nichttrivialen einfachen funktionalen Abhängigkeiten, die über den erhaltenen Teilrelationen gelten. Ihr Lösungsweg muss nachvollziehbar sein.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———

$$F_2 = \left\{ \{ P, S \} \rightarrow \{ T \}, \right\}$$

(ist schon in der kanonische Überdeckung)

(ii) **Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit $\alpha \rightarrow \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.

$R_{21}(P, S, T)$

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$ ———

$R_{21}(\underline{P, S}, T)$ mit

$$F_{21} = \left\{ \{ PS \} \rightarrow \{ T \}, \right\}$$

$R_{22}(\underline{P, S, R})$ mit

$$F_{22} = \left\{ \right\}$$

(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$. ———

$\varnothing$ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/10_Synthesealgorithmus/Aufgabe_Drei-Schemata.tex

Übungsaufgabe „Relation A-H“ (Synthese-Algorithmus)

Synthese-Algorithmus

Überführen Sie das Relationenschema mit Hilfe des Synthesealgorithmus in die 3. Normalform!

$$R(A, B, C, D, E, F, G, H)$$

$$\text{FA} = \Big\{ \begin{array}{r} \{F\} \to \{E\}, \\ \{A\} \to \{B, D\}, \\ \{A, E\} \to \{D\}, \\ \{A\} \to \{E, F\}, \\ \{A, G\} \to \{H\}, \end{array} \Big\}$$

Lösungsvorschlag

(a) Kanonische Überdeckung

(i) **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$. ———

Wir betrachten nur die zusammengesetzten Attribute:

$\{\boldsymbol{A, E}\} \to \{\boldsymbol{D}\}$

$D \in \text{AttrHülle}(F, \{A, E \setminus E\}) = \{A, E, F, B, \boldsymbol{D}\}$
$D \notin \text{AttrHülle}(F, \{A, E \setminus A\}) = \{E\}$

$\{\boldsymbol{A, G}\} \to \{\boldsymbol{H}\}$

$H \notin \text{AttrHülle}(F, \{A, G \setminus G\}) = \{A, E, F, B, D\}$
$H \notin \text{AttrHülle}(F, \{A, G \setminus A\}) = \{G\}$

$$\text{FA} = \Big\{ \begin{array}{r} \{F\} \to \{E\}, \\ \{A\} \to \{B, D\}, \\ \{A\} \to \{D\}, \\ \{A\} \to \{E, F\}, \\ \{A, G\} \to \{H\}, \end{array} \Big\}$$

(ii) **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt. ———

Nur die Attribute betrachten, die rechts doppelt vorkommen:

E

$\text{AttrHülle}(F \setminus \{ F \} \to \{ E \}, \{F\}) = \{F\}$
$\text{AttrHülle}(F \setminus \{ A \} \to \{ E, F \} \cup \{ A \} \to \{ E \}, \{A\}) = \{A, B, D, F, \boldsymbol{E}\}$

D

$\text{AttrHülle}(F \setminus \{ A \} \to \{ D \}, \{A\}) = \{A, B, \boldsymbol{D}, F, E\}$
$\{ A \} \to \{ D \}$ kann wegen der Armstrongschen Dekompositionsregel weggelassen werden. Wenn gilt $\{ A \} \to \{ B, D \}$, dann gilt auch $\{ A \} \to \{ B \}$ und $\{ A \} \to \{ D \}$

$$\text{FA} = \left\{ \begin{array}{r} \{ F \} \to \{ E \}, \\ \{ A \} \to \{ B, D \}, \\ \{ A \} \to \{ \varnothing \}, \\ \{ A \} \to \{ F \}, \\ \{ A, G \} \to \{ H \}, \end{array} \right\}$$

(iii) **Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.

$$\text{FA} = \left\{ \begin{array}{r} \{ F \} \to \{ E \}, \\ \{ A \} \to \{ B, D \}, \\ \{ A \} \to \{ F \}, \\ \{ A, G \} \to \{ H \}, \end{array} \right\}$$

(iv) **Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.

$$\text{FA} = \left\{ \begin{array}{r} \{ F \} \to \{ E \}, \\ \{ A \} \to \{ B, D, F \}, \\ \{ A, G \} \to \{ H \}, \end{array} \right\}$$

(b) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.* —

$R_1(\underline{E}, E)$
$R_2(\underline{A}, B, D, F)$
$R_3(\underline{A, G}, H)$

(c) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ———————————

$R_1(\underline{E}, E)$
$R_2(\underline{A}, B, D, F)$
$R_3(\underline{A, G}, H)$
$R_4(\underline{A, C, G})$

(d) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* ———————————

$\varnothing$ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/10_Synthesealgorithmus/Aufgabe_Relation-A-H.tex`

Übungsaufgabe „Relation-MNVTPPN“ (Synthese-Algorithmus, Kanonische Überdeckung)

Betrachten Sie ein abstraktes Relationenschema $R(M, N, V, T, P, PN)$ mit den Funktionalen Abhängigkeiten [1]

$$\text{FA} = \Big\{ \begin{array}{l} \{M\} \to \{M\}, \\ \{M\} \to \{N\}, \\ \{V\} \to \{T, P, PN\}, \\ \{P\} \to \{PN\}, \end{array} \Big\}$$

(a) Bestimmen Sie alle Kandidatenschlüssel.

Lösungsvorschlag

V kommt auf keiner rechten Seite der Funktionalen Abhängigkeiten vor.

$\text{AttrHülle}(R, \{V\}) = \{V, T, P, PN\} \neq R$

$\text{AttrHülle}(R, \{V, M\}) = \{V, M, N, T, P, PN\} = R$

$\text{AttrHülle}(R, \{V, P\}) = \{V, P, T, PN\} \neq R$

$\{ V, M \}$ ist Schlüsselkandidat

(b) In welcher Normalform befindet sich die Relation?

Lösungsvorschlag

Die Relation befindet sich in der 1. Normalform weil, nichtprimäre Attribute von einer echten Teilmenge des Schlüsselkandidaten abhängen (z. B. $\{M\} \to \{N\}$).

(c) Bestimmen Sie zu den gebenen Funktionalen Abhängigkeiten die kanonische Überdeckung.

Lösungsvorschlag

(i) **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq \mathit{AttrHülle}(F, \alpha - A)$. —

$\varnothing$ Nichts zu tun

(ii) **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in \mathit{AttrHülle}(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt. —

M

[1] `https://db.in.tum.de/teaching/ws1415/grundlagen/Loesung08.pdf`

$M \in \text{AttrHülle}(F \setminus \{M\} \to \{M\}, \{M\}) = \{\boldsymbol{M}, N\}$

$$\text{FA} = \Big\{ \{M\} \to \{\varnothing\}, \quad \{M\} \to \{N\}, \quad \{V\} \to \{T, P, PN\}, \quad \{P\} \to \{PN\}, \Big\}$$

PN

$PN \in \text{AttrHülle}(F \setminus \{V\} \to \{T, P, PN\} \cup \{V\} \to \{T, P\}, \{V\}) = \{V, T, P, \boldsymbol{PN}\}$

$$\text{FA} = \Big\{ \{M\} \to \{\varnothing\}, \quad \{M\} \to \{N\}, \quad \{V\} \to \{T, P\}, \quad \{P\} \to \{PN\}, \Big\}$$

(iii) **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* —

$\varnothing$ Nichts zu tun

(iv) **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* —

$$\text{FA} = \Big\{ \{M\} \to \{N\}, \quad \{V\} \to \{T, P\}, \quad \{P\} \to \{PN\}, \Big\}$$

(d) Falls nötig, überführen Sie die Relation verlustfrei und abhängigkeitsbewahrend in die dritte Normalform.

Lösungsvorschlag

(i) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

$R_1(\underline{M}, N)$

$R_2(\underline{V}, T, P)$
$R_3(\underline{P}, PN)$

(ii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$ ———————

$R_1(\underline{M}, N)$
$R_2(\underline{V}, T, P)$
$R_3(\underline{P}, PN)$
$R_4(\underline{V, M})$

(iii) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$. ———————

$\varnothing$ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/10_Synthesealgorithmus/Aufgabe_Relation-MNVTPPN.tex`

Übungsaufgabe „Supermarkt“ (Zweite Normalform, Schlüsselkandidat, Update-Anomalie, Delete-Anomalie, Synthese-Algorithmus)

Zweite Normalform
Schlüsselkandidat

Ein Supermarkt speichert seine Bestellungen in nachfolgender Tabelle:

$ARTBEST(\ ArtNr, ArtName, ArtArt, Hersteller, HerstTel, Lieferant, LiefTel, BestDat, Anzahl, EP\ )$

Es existieren folgende funktionale Abhängigkeiten:

$$
\mathrm{F} = \Big\{
\begin{aligned}
\{\ ArtNr\ \} &\rightarrow \{\ ArtName, ArtArt, Hersteller, HerstTel\ \},\\
\{\ Hersteller\ \} &\rightarrow \{\ HerstTel\ \},\\
\{\ Lieferant\ \} &\rightarrow \{\ LiefTel\ \},\\
\{\ ArtNr, Lieferant, BestDat\ \} &\rightarrow \{\ Anzahl\ \},
\end{aligned}
\Big\}
$$

„Bestdat“ steht für Bestelldatum, „EP“ für Einkaufspreis

(a) Erläutern Sie, warum nur Relationen mit einem zusammengesetzten Schlüsselkandidaten die 2. Normalform verletzen können!

Lösungsvorschlag

Eine Relation ist genau dann in der zweiten Normalform, wenn kein Nichtprimärattribut funktional von einer echten Teilmenge eines Schlüsselkandidaten abhängt.

Anders gesagt: Jedes nicht-primäre Attribut ist jeweils von allen ganzen Schlüsseln abhängig, nicht nur von einem Teil eines Schlüssels.

Bei nicht zusammengesetzten Schlüsselkandidaten, d. h. Schlüsselkandidaten mit nur einem Attribut, können Nichtprimärattribute nur von diesem einen Schlüsselkandidaten abhängen, sonst wäre es ja kein Schlüsselkandidat / Primärschlüssel.

(b) Finden Sie den einzigen Schlüsselkandidaten von ARTBEST.

Lösungsvorschlag

Ich wähle $\{\ ArtNr, Lieferant, BestDat\ \}$ aus, da diese Attribute auf keiner rechten Seite einer FD vorkommen. Außerdem wähle ich $\{\ EP\ \}$ aus, da $\{\ EP\ \}$ in keiner FD vorkommt.

$$
\begin{aligned}
&\mathrm{AttrHülle}(F, \{ArtNr, Lieferant, BestDat, EP\}) =\\
&\{\ ArtNr, Lieferant, BestDat, EP, ArtName, ArtArt, Hersteller, HerstTel, LiefTel, Anzahl\ \} = R
\end{aligned}
$$

Damit ist gezeigt, dass $\{\ ArtNr, Lieferant, BestDat, EP\ \}$ ein Superschlüssel ist.

Ich teste mit Hilfe der Attributhülle, ob man den Superschlüssel noch weiter

verkleinern kann.

ohne `ArtNr`

$$\text{AttrHülle}(F, \{\textit{Lieferant, BestDat, EP}\}) = \{\ \textit{Lieferant, BestDat, EP, LiefTel}\ \} \neq R$$

ohne `Lieferant`

$$\text{AttrHülle}(F, \{\textit{ArtNr, BestDat, EP}\}) = \{\ \textit{ArtNr, BestDat, EP, ArtName, ArtArt, Hersteller, HerstTel}\ \} \neq R$$

ohne `BestDat`

$$\text{AttrHülle}(F, \{\textit{ArtNr, Lieferant, EP}\}) = \{\ \textit{ArtNr, Lieferant, EP, ArtName, ArtArt, Hersteller, HerstTel, LiefTel}\ \} \neq R$$

ohne `EP`

$$\text{AttrHülle}(F, \{\textit{ArtNr, Lieferant, BestDat}\}) = \{\ \textit{ArtNr, Lieferant, BestDat, ArtName, ArtArt, Hersteller, HerstTel, LiefTel, Anzahl}\ \} \neq R$$

Der Superschlüssel { *ArtNr, Lieferant, BestDat, EP* } kann nicht mehr weiter verkleinert werden. Er ist bereits minimal. { *ArtNr, Lieferant, BestDat, EP* } ist der einzige Schlüsselkandidat und damit der Primärschlüssel.

(c) Erläutern Sie, inwiefern obiges Schema die 3. Normalform verletzt! Zeigen Sie anhand obiger Relation `ARTBEST` zwei mögliche Anomalien auf, die bei fehlender Normalisierung auftreten können.

Lösungsvorschlag

In der dritten Normalform darf kein Nichtschlüsselattribut von einem Schlüsselkandidaten transitiv abhängen. In der Relation `ARTBEST` hängt { *HerstTel* }

funkional von { *Hersteller* } und { *Hersteller* } hängt wiederum funktional von dem Primärschlüssel / Schlüsselkandidaten { *ArtNr, Lieferant, BestDat, EP* } ab.

{ *ArtNr, Lieferant, BestDat, EP* } $\rightarrow$ { *Hersteller* } $\rightarrow$ { *HerstTel* }

Update-Anomalie

Es kann zur Update-Anomalie kommen. Ändert sich zum Beispiel die Telefonnummer eines Herstellers, so müssen in allen Datensätzen die Telefonnummer geändert werden.

Delete-Anomalie

Wird die Datenbank aufgeräumt, d. h. alte Bestellungen gelöscht, so verschwindet auch die Hersteller-Telefonnumer von manchen Herstelllern.

(d) Überführen Sie das obige Relationenschema schrittweise in die 3. Normalform! Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz!

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. ———

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \rightarrow \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$. ———

Die einzige FD mit einer Determinante bestehtend aus mehrere Attributen, ist { *ArtNr, Lieferant, BestDat* } $\rightarrow$ { *Anzahl* }

- ohne { *ArtNr* }

$$\{Anzahl\} \notin \text{AttrHülle}(F, \{Lieferant, BestDat\}) = \{ Lieferant, BestDat, LiefTel \}$$

- ohne { *Lieferant* }

$$Anzahl \notin \text{AttrHülle}(F, \{ArtNr, BestDat\}) = \{ ArtNr, BestDat, ArtName, ArtArt, Hersteller, HerstTel \}$$

- ohne { *BestDat* }

$$\textit{Anzahl} \notin \text{AttrHülle}(F, \{\textit{ArtNr, Lieferant}\}) = \{\ \textit{ArtNr, Lieferant, ArtName, ArtArt, Hersteller, HerstTel, LiefTel}\ \}$$

Die linke Seiten der FDs können nicht reduziert werden.

ii. **Rechtsreduktion**
— *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in \textit{AttrHülle}(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.* ——————————

Das einzige Attribut, dass auf der rechten Seite der FDs doppelt vorkommt ist { *HerstTel* }

$$\{\textit{HerstTel}\} \in \text{AttrHülle}(F - \{\ \textit{ArtNr}\ \} \to \{\ \textit{HerstTel}\ \}, \{\textit{ArtNr}\}) = \{\ \textit{ArtNr, ArtName, ArtArt, Hersteller, HerstTel}\ \}$$

iii. **Löschen leerer Klauseln**
— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ——————————

$\varnothing$ Nichts zu tun

iv. **Vereinigung**
— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ——————————

$\varnothing$ Nichts zu tun

Kanonische Überdeckung:

$$F_c = \Big\{ \begin{aligned} \{\ \textit{ArtNr}\ \} &\to \{\ \textit{ArtName, ArtArt, Hersteller}\ \}, \\ \{\ \textit{Hersteller}\ \} &\to \{\ \textit{HerstTel}\ \}, \\ \{\ \textit{Lieferant}\ \} &\to \{\ \textit{LiefTel}\ \}, \\ \{\ \textit{ArtNr, Lieferant, BestDat}\ \} &\to \{\ \textit{Anzahl}\ \}, \end{aligned} \Big\}$$

(ii) **Relationsschemata formen**
— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

R_1(*ArtNr, ArtName, ArtArt, Hersteller*)
R_2(*Hersteller, HerstTel*)
R_3(*Lieferant, LiefTel*)
R_4(*ArtNr, Lieferant, BestDat, Anzahl*)

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$ ———

Es muss noch eine Relation hinzugefügt werden, nämlich kommt das Attribut { *EP* } bisher in keiner Relation vor.

R_1(*ArtNr, ArtName, ArtArt, Hersteller*)
R_2(*Hersteller, HerstTel*)
R_3(*Lieferant, LiefTel*)
R_4(*ArtNr, Lieferant, BestDat, Anzahl*)
R_5(*ArtNr, Lieferant, BestDat, EP*)

(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$. ———

Nicht zu tun.

Ergebnis:

R_1(*ArtNr, ArtName, ArtArt, Hersteller*)
R_2(*Hersteller, HerstTel*)
R_3(*Lieferant, LiefTel*)
R_4(*ArtNr, Lieferant, BestDat, Anzahl*)
R_5(*ArtNr, Lieferant, BestDat, EP*)

(e) Erläutern Sie, inwiefern sich eine vollständige Normalisierung nachteilig auf die Geschwindigkeit der Anfragebearbeitung auswirken kann und wie darauf reagiert werden kann!

Lösungsvorschlag

Eine vollständige Normalisierung hat den Effekt, dass die Daten auf mehr Relation bzw. Tabellen aufgeteilt werden. In der Regel geht damit einher, dass bei Abfragen mehr Joins durchgeführt werden müssen, was in der Regel mit mehr Speicherbedarf und Rechenzeit der Anfragen einhergeht.

Man könnte auf eine Normalisierung verzichten, oder nur teilweise normalisieren und somit zwischen Performance und Redundanz abwägen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/10_Synthesealgorithmus/Aufgabe_Supermarkt.tex`

Übungsaufgabe „Abstraktes R“ (Schlüsselkandidat, Zweite Normalform, Kanonische Überdeckung)

Schlüsselkandidat
Zweite Normalform

Gegeben sei das Relationenschema $R(A, B, C, D, E, G)$ mit

$$F = \left\{ \begin{array}{r} \{E\} \rightarrow \{D\}, \\ \{C\} \rightarrow \{B\}, \\ \{C, E\} \rightarrow \{G\}, \\ \{B\} \rightarrow \{A\}, \end{array} \right\}$$

(a) Zeigen Sie: C, E ist der einzige Schlüsselkandidat von R.

Lösungsvorschlag

C und E kommen auf keiner rechten Seite der Funktionalen Abhängigkeiten aus F vor, ðC und E müssen Teil jedes Schlüsselkandidaten sein.

Außerdem gilt: $\text{AttrHülle}(F, \{C, E\}) = \{A, B, C, D, E, G\} = R$

$\{ C, E \}$ ist somit Superschlüssel von R. Zudem ist $\{ C, E \}$ minimal, da beide Attribute Teil jedes Schlüsselkandidaten sein müssen.

$\Rightarrow \{C, E\}$ ist damit der einzige Schlüsselkandidat von R (da kein Schlüssel ohne C und E möglich ist).

Anmerkung:

- Man könnte hier auch einen Algorithmus zur Bestimmung der Schlüsselkandidaten verwenden, dessen einziges Ergebnis wäre dann $\{C, E\}$. In diesem Fall lässt sich die Schlüsselkandidateneigenschaft jedoch einfacher zeigen, sodass man den Algorithmus und somit Zeit sparen kann.
- Achtung! $\{ C, E \}$ ist zwar der einzige Schlüsselkandidat, aber nicht der einzige Superschlüssel, auch $\{ A, B, C, D, E, G \}$ wäre ein Superschlüssel!

(b) Ist R in 2NF?

Lösungsvorschlag

R ist nicht in 2NF, denn:

Betrachte $\{ E \} \rightarrow \{ D \}$: D ist ein Nicht-Schlüsselattribut und E ist echt Teilmenge des Schlüsselkandidaten $\{C, E\}$. Ebenso ist B nicht voll funktional abhängig vom Schlüsselkandidaten, sondern nur von einer echten Teilmenge des Schlüsselkandidaten, nämlich C.

Anmerkung:

- Ob alle Attributwerte atomar sind, können wir in einem abstrakten Schema wie diesem nicht wirklich sagen, daher kann dies Annahme in der

Regel nicht getroffen werden.

- Dass A von B abhängig ist, spielt bei der Entscheidung über die 2. NF keine Rolle, da B selbst (genauso wie A) ein Nicht-Schlüsselattribut ist. Wichtig ist nur, ob es Abhängigkeiten zwischen einem Teil der Schlüsselkandidaten (also einem Schlüsselattribut) und einem Nicht-Schlüsselattribut gibt.
- Um der 2NF zu genügen, müsste in folgenden Relationen aufgeteilt werden:
  $R_1(\,C, E, G\,)\quad R_2(\,C, B, A\,)\quad R_2(\,E, D\,)$

(c) Ist F minimal?

$$\text{FA} = \left\{ \begin{array}{r} \{\,E\,\} \rightarrow \{\,D\,\}, \\ \{\,C\,\} \rightarrow \{\,B\,\}, \\ \{\,CE\,\} \rightarrow \{\,G\,\}, \\ \{\,B\,\} \rightarrow \{\,A\,\}, \end{array} \right\}$$

Lösungsvorschlag

Kanonische Überdeckung

(i) Linksreduktion
$AttrHul\{F, \{\,C\,\}\} = \{C, B\} \rightarrow G$ nicht enthalten
$AttrHul\{F, \{\,E\,\}\} = \{E, D\} \rightarrow G$ nicht enthalten

(ii) Rechtsreduktion
Kein Attribut auf einer rechten Seite ist redundant: Da das einzelne Attribut, das die rechte Seite einer FD aus F bildet, bei keiner anderen FD auf der rechten Seite auftritt, kann die rechte Seite einer FD nicht unter ausschließlicher Verwendung der restlichen FD aus der entsprechenden linken Seite abgeleitet werden.

Lösungsvorschlag

Vorgehen: Entsprechen die hier abgebildeten Funktionalen Abhängigkeiten bereits einer kanonischen Überdeckung von F oder nicht?

- Eliminierung redundanter Attribute auf der linken Seite: Die Attributmenge auf den linken Seiten der FDs sind bereits bis auf $\{\,C, E\,\} \rightarrow \{\,G\,\}$ einelementig. Bei $\{\,C, E\,\} \rightarrow \{\,G\,\}$ ist $\{\,CE\,\}$ der Schlüsselkandidat, also kann kein redundantes Attribut vorliegen.
- Eliminierung redundanter Attribute auf der rechten Seite (hier müssen auch alle einelementigen FA's betrachtet werden)
  - $\{\,E\,\} \rightarrow \{\,D\,\}$: AttrHülle$(F - \{E \rightarrow D\}, \{E\}) = \{E\}, \eth D \notin$ AttrHülle$(F -$

$\{E \rightarrow D\}, \{E\})$

- $\{C\} \rightarrow \{B\}$: AttrHülle$(F - \{C \rightarrow B\}, \{C\}) = \{C\}$, ðB $\notin$ AttrHülle$(F - \{C \rightarrow B\}, \{E\})$
- $\{CE\} \rightarrow \{G\}$: AttrHülle$(F - \{CE \rightarrow G\}, \{C, E\}) = \{A, B, C, D, E\}$, ðG $\notin$ AttrHülle$(F - \{CE \rightarrow G\}, \{E\}) \Rightarrow CE \rightarrow G$ ist nicht redundant
- $\{B\} \rightarrow \{A\}$: AttrHülle$(F - \{B\} \rightarrow \{A\}, \{B\}) = \{B\}$, ð $A \notin$ AttrHülle$(F - \{B \rightarrow A\}, \{E\}) \Rightarrow B \rightarrow A$ ist nicht redundant

F ist bereits minimal.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/Aufgabe_Abstraktes-R.tex

Übungsaufgabe „Anomalien Abhängigkeiten“ (Update-Anomalie, Delete-Anomalie, Insert-Anomalie, Funktionale Abhängigkeiten, Attributhüllen-Algorithmus, Attributhülle, Superschlüssel)

Gegeben ist die Relation *Abteilungsmitarbeiter*, repräsentiert durch folgende Tabelle. Es sei angenommen, dass innerhalb einer Abteilung keine Mitarbeiter mit identischem Namen existieren. Die Abteilungsnummer ist eindeutig, es kann aber durchaus sein, dass mehrere Abteilungen die gleiche Bezeichnung tragen.

Name	**Straße**	**Ort**	**AbtNr**	**Bezeichnung**
Schweizer	Hauptstraße	Zürich	A3	Finanzen
Deutscher	Lindenstraße	Passau	A4	Informatik
Österreicher	Nebenstraße	Wien	A4	Informatik

(a) Geben Sie - orientiert an der obigen Tabelle - ein Beispiel für eine mögliche Änderungsanomalie an!

Lösungsvorschlag

Update-Anomalie Die Abteilung *A4* wird umbenannt, beispielsweise in *Softwareabteilung*. Die Änderung wird aus Versehen nicht in allen Tupeln mit $AbtNr = A4$ vollzogen.

Delete-Anomalie Herr *Schweizer* (aus der Abteilung A4) verlässt die Firma und wird aus der Datenbank gelöscht. Damit gehen auch die Daten über die Abteilung *A3* verloren.

Insert-Anomalie Es wird eine neue Abteilung *A5* (*Hardwareabteilung*) geschaffen, der aber noch keine Mitarbeiter zugeteilt sind. Damit müsste ein Tupel (NULL, NULL, NULL, A5, Hardwareabteilung) in die Datenbank eingefügt werden. Da aber das Attribut *Name* sicher in jedem Schlüsselkandidaten enthalten sein muss, kann der Wert von Name keinen Nullwert enthalten. Das Tupel kann nicht eingefügt werden.

(b) Bestimmen Sie eine Menge F der funktionalen Abhängigkeiten, die sich aus Ihrer Analyse des Anwendungsbereiches ergeben. (Triviale Abhängigkeiten brauchen nicht angegeben werden.) Begründen Sie Ihre Entscheidung kurz.

Lösungsvorschlag

- $\{ AbtNr \} \rightarrow \{ Bezeichnung \}$
  Die *Abteilungsnummer* ist eindeutig (als „künstliches“ Unterscheidungsmerkmal für *Abteilungen*) und legt damit die *Abteilung* eindeutig fest.
- $\{ Name, AbtNr \} \rightarrow \{ Strasse, Ort \}$
  Da der *Name* innerhalb der *Abteilung* eindeutig ist, ist damit der *Mitarbeiter* und folglich auch die *Adressdaten* eindeutig festgelegt. Da es sich bei dieser Attributkombination um den Primärschlüssel handelt, bestimmt diese Attributkombination auch das Attribut *Bezeichnung*, allerdings darf

Attributhüllen-Algorithmus
Attributhülle
Superschlüssel

es nicht in diese Funktionale Abhängigkeit aufgenommen werden, da die Abteilungsbezeichnung nicht von der Kombination aus *Name* & *AbtNr* abhängig, sondern nur von der *AbtNr* allein, somit muss dies als einzelne Funktionale Abhängigkeit formuliert werden und kann hier nicht aufgenommen werden

$\rightarrow$ der Rückschluss daraus wäre nämlich, dass sich die *Bezeichnung* der *Abteilung* nur aus der Kombination von *Mitarbeiter* und *AbtNr* erkennen lässt und nicht allein aus der *AbtNr* und das wäre ja nicht korrekt. Grundsätzlich gilt: Primärschlüssel und Funktionale Abhängigkeiten müssen getrennt betrachtet werden.

$$\mathrm{F} = \Big\{ \{ \textit{Name}, \textit{AbtNr} \} \rightarrow \{ \textit{Strasse}, \textit{Ort} \}, \quad \{ \textit{AbtNr} \} \rightarrow \{ \textit{Bezeichnung} \}, \Big\}$$

(c) Bestimmen Sie z. B. mit Hilfe des Attributhüllen-Algorithmus die Attributhülle

(i) $\mathrm{AttrHülle}(F, \{\textit{Name}, \textit{Bezeichnung}\})$

Lösungsvorschlag

$\mathrm{AttrHülle}(F, \{\textit{Name}, \textit{Bezeichnung}\}) = \{\textit{Name}, \textit{Bezeichnung}\}$

(ii) $\mathrm{AttrHülle}(F, \{\textit{Name}, \textit{AbtNr}\})$

Lösungsvorschlag

$\mathrm{AttrHülle}(F, \{\textit{Name}, \textit{AbtNr}\}) = \{\textit{Name}, \textit{AbtNr}, \textit{Strasse}, \textit{Ort}, \textit{Bezeichnung}\}$

(d) Ist { *Name*, *Bezeichnung* } bzw. { *Name*, *AbtNr* } ein Superschlüssel der Relation Abteilungsmitarbeiter? Kurze Begründung!

Lösungsvorschlag

{ *Name*, *AbtNr* }, da die Attributhülle von { *Name*, *AbtNr* } alle Attribute der Relation umfasst und { *Name*, *Bezeichnung* } nicht.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/Aufgabe_Anomalien-Abhaengigkeiten.tex`

Übungsaufgabe „Kanonische Überdeckung (Kemper)“ (Kanonische Überdeckung)

$$\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B\}, \\ \{B\} \to \{C\}, \\ \{A, B\} \to \{C\}, \end{array} \right\}$$

Lösungsvorschlag

(a) **Linksreduktion**

— Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$.

$$\text{AttrHülle}(F, \{A, B\} - \{B\}) = \{A, B, C\}$$
$$\text{AttrHülle}(F, \{A, B\} - \{A\}) = \{C\}$$

$$\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B\}, \\ \{B\} \to \{C\}, \\ \{A\} \to \{C\}, \end{array} \right\}$$

(b) **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.

$$\text{FA} = \left\{ \begin{array}{r} \{A\} \to \{B\}, \\ \{B\} \to \{C\}, \\ \{A\} \to \{\varnothing\}, \end{array} \right\}$$

(c) **Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.

$$\text{FA} = \left\{$$

$$\{ A \} \rightarrow \{ B \},$$
$$\{ B \} \rightarrow \{ C \},$$
$$\}$$

(d) **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \rightarrow \beta_1, \ldots, \alpha \rightarrow \beta_n$, so dass $\alpha \rightarrow \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* —

Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/Aufgabe_Kanonische-Ueberdeckung-Kemper
tex

Übungsaufgabe „Mietwagenfirma“ (Zweite Normalform, Delete-Anomalie, Update-Anomalie, Insert-Anomalie, Dritte Normalform)

Gegeben sei die folgende relationale Datenbank der Mietwagenfirma „MobilRent“, Station „München-Mitte“.

KdNr	Name	Wohnort	Buchungsdatum	Aktion	Fahrzeug	Typ	Tarif	Tage	Rueckgabestation	Stationsleiter
123	Chomsky	Nürnberg	23.01.2004	0	Cala	Klein	1	2	Nürnberg-Nord	Backus
123	Chomsky	Nürnberg	07.10.2003	-25%	Vanny	Transp	5	1	Nürnberg-Nord	Hoare
220	Neumann	München	02.04.2004	0	Baro	Klein	1	2	München-Mitte	Zuse
710	Turing	München	20.02.2004	-10%	Cala	Klein	1	2	München-Mitte	Zuse
888	Neumann	Passau	07.10.2003	-25%	Lux	Mittelkl	3	3	München-Mitte	Zuse

KdNr steht für die Kundennummer der Kunden. An bestimmten Tagen gewährt die Firma Rabatt. Folgende funktionale Abhängigkeiten seien vorgegeben.

$$
\begin{aligned}
\mathrm{FA} = \Big\{ & \\
& \{\, \mathit{KdNr} \,\} \to \{\, \mathit{Name, Wohnort} \,\}, \\
& \{\, \mathit{KdNr, Buchungsdatum} \,\} \to \{\, \mathit{Fahrzeug, Typ, Tarif, Tage, Rueckgabe\text{-}Station} \,\}, \\
& \{\, \mathit{Buchungsdatum} \,\} \to \{\, \mathit{Aktion} \,\}, \\
& \{\, \mathit{Fahrzeug} \,\} \to \{\, \mathit{Typ, Tarif} \,\}, \\
& \{\, \mathit{Typ} \,\} \to \{\, \mathit{Tarif} \,\}, \\
& \{\, \mathit{Rueckgabestation} \,\} \to \{\, \mathit{Stationsleiter} \,\}, \\
& \Big\}
\end{aligned}
$$

Der Primärschlüssel besteht aus den Attributen *KdNr* und *Buchungsdatum*.

(a) Begründe, dass diese Tabelle in 1. Normalform vorliegt.

Lösungsvorschlag

> Es gibt nur atomare Werte. Alle Attributewerte sind atomar.

(b) Erläutere, warum nur Tabellen mit zusammengesetzem Primärschlüssel die zweite Normalform verletzen können.

Lösungsvorschlag

> Nur bei Tabellen mit zusammengesetzten Primärschlüssel kann es vorkommen, dass ein Nichtschlüssel-Attribut von einer echten Teilmenge des Primärschlüssel voll funktional abhängt.

(c) Zeige mögliche Anomalien auf, die hier auftreten können.

Lösungsvorschlag

> **Delete-Anomalie** z. B. Kunde *888* wird gelöscht, gleichzeitig gehen alle Informationen über das Fahrzeug *A3* verloren.
>
> **Update-Anomalie** z. B. *Chomsky* zieht um, Wohnort wird nicht in allen Tupeln geändert.
>
> **Insert-Anomalie** z. B. Eine neue Rückgabestation kann erst eröffnet werden, wenn ein Kunde ein bereits gebuchtes Auto auch an dieser Station zu-

rückgeben will.

(d) Überführe das Schema in die 2. Normalform.

Lösungsvorschlag

Der *Name* ist voll funktional abhängig von *KdNr*, nicht aber von *Buchungsdatum*. Deswegen verletzt die FD $\{ KdNr \} \rightarrow \{ Name \}$ die 2. Normalform (und also auch die 3. Normalform). *Stationsleiter* ist nur transitiv von *KdNr*, *Buchungsdatum* abhängig (über *Rückgabestation*). Dies verletzt die 3. Normalform. Überführung in 2NF: Entfernung der vom Primärschlüssel nicht voll funktional abhängigen Attribute.

MobilRent(KdNr, Buchungsdatum, Fahrzeug, Typ, Tarif, Tage, Rückgabestation, Stationsleiter)

Kunden(Kdnr, Name, Wohnort)

Aktion(Buchungsdatum, Aktion)

(e) Überführe das Schema in die 3. Normalform.

Lösungsvorschlag

Für die 3. NF: Entfernung der transitiven Abhängigkeiten. Die Tabellen „*Kunde*“ und „*Sonderaktionen*“ bleiben erhalten.

Kunden(Kdnr, Name, Wohnort)

Aktion(Buchungsdatum, Aktion)

Fahrzeugtypen(Fahrzeug, Typ)

Tarife(Typ, Tarif)

Stationsleiter(Rückgabestation, Stationsleiter)

MobilRent(KdNr, Buchungsdatum, Fahrzeug, Tage, Rückgabestation)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/Aufgabe_Mietwagenfirma.tex`

Übungsaufgabe „Minimale Überdeckung“ (Kanonische Überdeckung)

Kanonische Überdeckung

Gegeben ist die Menge

$$F = \Big\{ \begin{aligned} \{A\} &\to \{B, C\}, \\ \{C\} &\to \{D, A\}, \\ \{E\} &\to \{A, C\}, \\ \{C, D\} &\to \{B, E\}, \end{aligned} \Big\}$$

. Bestimmen Sie eine minimale Überdeckung von F.

Lösungsvorschlag

(a) Linksreduktion

$\text{AttrHülle}(F, \{D\}) = \{D\}$
$\text{AttrHülle}(F, \{C\}) = \{C, D, A, B, E\}$

$$F' = \Big\{ \begin{aligned} \{A\} &\to \{B, C\}, \\ \{C\} &\to \{D, A\}, \\ \{E\} &\to \{A, C\}, \\ \{C\} &\to \{B, E\}, \end{aligned} \Big\}$$

(b) Rechtsreduktion

$\text{AttrHülle}(F - \{A\} \to \{B, C\}, \{A\}) = \{A\}$
$\text{AttrHülle}(F - \{C \to DA\}, \{C\}) = \{C, B, E, A\}$
$\text{AttrHülle}(F - \{E \to AC\}, \{E\}) = \{E\}$
$\text{AttrHülle}(F - \{C \to BE\}, \{C\}) = \{C, D, A, B\}$
Keine Rechtsreduktion möglich

schon minimal

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/50_Relationale-Entwurfstheorie/30_Normalformen/Aufgabe_Minimale-Ueberdeckung.tex

Transaktionsverwaltung

Examensaufgabe „Transaktionen“ (46116-2016-F.T1-TA1-A5)

(a) Nennen Sie die vier wesentlichen Eigenschaften einer Transaktion und erläutern Sie jede Eigenschaft kurz (ein Satz pro Eigenschaft).

Lösungsvorschlag

Atomicity Eine Transaktion ist atomar, ðvon den vorgesehenen Änderungsoperationen auf die Datenbank haben entweder alle oder keine eine Wirkung auf die Datenbank.

Consistency Eine Transaktion überführt einen korrekten (konsistenten) Datenbankzustand wieder in einen korrekten (konsistenten) Datenbankzustand.

Isolation Eine Transaktion bemerkt das Vorhandensein anderer (parallel ablaufender) Transaktionen nicht und beeinflusst auch andere Transaktionen nicht.

Durability Die durch eine erfolgreiche Transaktion vorgenommenen Änderungen sind dauerhaft (persistent).

(b) Gegeben ist die folgende Historie (Schedule) dreier Transaktionen:

$r_1(B) \rightarrow w_1(C) \rightarrow r_3(C) \rightarrow r_1(A) \rightarrow c_1 \rightarrow r_2(C) \rightarrow r_3(C) \rightarrow r_2(C) \rightarrow w_2(B) \rightarrow c_2 \rightarrow c_3$

Zeichnen Sie den Serialisierbarkeitsgraphen zu dieser Historie und begründen Sie, warum die Historie serialisierbar ist oder nicht.

Exkurs: Historie

In Transaktionssystemen existiert ein Ausführungsplan für die parallele Ausführung mehrerer Transaktionen. Der Plan wird auch Historie genannt und gibt an, in welcher Reihenfolge die einzelnen Operationen der Transaktion ausgeführt werden. Als serialisierbar bezeichnet man eine Historie, wenn sie zum selben Ergebnis führt wie eine nacheinander (seriell) ausgeführte Historie über dieselben Transaktionen.

Lösungsvorschlag

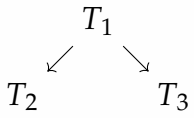
Der Algorithmus geht schrittweise durch den Ablaufplan unten und hebt die Abhängigkeiten der aktiven Transaktion zu allen anderen Transaktionen hervor. Hierfür werden in allen nachfolgenden Schritten solche Operationen gesucht, die einen Konflikt mit der aktuellen Operation hervorrufen. Konflikt-Operationen sind: `read after write`, `write after read` und `write after write` auf denselben Datenobjekt.

T_1	T_2	T_3
$r_1(B)$		
$w_1(C)$		
		$r_3(C)$
$r_1(A)$		
c_1		
	$r_2(C)$	
		$r_3(C)$
	$r_2(C)$	
	$w_2(B)$	
	c_2	
		c_3

Konfliktoperation

- $r_1(B) < w_2(B)$: Kante von T_1 nach T_2
- $w_1(C) < r_3(C)$: Kante von T_1 nach T_3
- $w_1(C) < r_2(C)$: Kante von T_1 nach T_2

Serialisierbarkeitsgraph



Es gibt keinen Zyklus im Graph. Er ist deshalb serialisierbar. Wenn ein Zyklus auftreten würde, dann wäre er nicht serialisierbar.

(c) Geben Sie an, wodurch die erste und die zweite Phase des Zwei-Phasen-Sperrprotokolls jeweils charakterisiert sind (ein Satz pro Phase).

Lösungsvorschlag

Die zwei Phasen des Protokolls bestehen aus einer *Sperrphase*, in der alle benötigten Objekte für die Transaktion gesperrt werden. In der zweiten Phase werden die *Sperren wieder freigegeben*, sodass die Objekte von anderen Transaktionen genutzt werden können.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2016/03/Thema-1/Teilaufgabe-1/Aufgabe-5.tex

Examensaufgabe „Schedule S“ (66116-2020-F.T2-TA2-A4)

(a) Betrachten Sie den folgenden Schedule S:

T_1	T_2	T_3
	$r_2(z)$	
		$w_3(y)$
	$r_2(x)$	
$w_1(x)$		
	$w_2(x)$	
		$r_3(z)$
		c_3
	$w_2(z)$	
$w_1(y)$		
c_1		
	c_2	

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im rigorosen Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.

Lösungsvorschlag

T_1	T_2	T_3
	$\text{rlock}_2(z)$	
	$r_2(z)$	
		$\text{xlock}_3(y)$
		$w_3(y)$
	$\text{rlock}_2(x)$	
	$r_2(x)$	
	$\text{xlock}_2(x)$	
	$w_2(x)$	
		$\text{rlock}_3(z)$
		$r_3(z)$
		c_3
		$\text{unlock}_2(y, z)$
	$\text{xlock}_2(z)$	
	$w_2(z)$	
	c_2	
	$\text{unlock}_2(x, z)$	
$\text{xlock}_1(x)$		
$w_1(x)$		
$\text{xlock}_1(y)$		
$w_1(y)$		
c_1		
$\text{unlock}_2(x, z)$		

(b) Beschreiben Sie den Unterschied zwischen dem herkömmlichen Zweiphasen-Sperrprotokoll (2PL) und dem rigorosen Zweiphasen-Sperrprotokoll. Warum wird in der Praxis häufiger das rigorose Zweiphasen-Sperrprotokoll verwendet?

Lösungsvorschlag

Bei beiden Protokollen fordert die Transaktion erst alle Sperren an (Anforderungsphase) und gibt sie später frei (Freigabephase).

- Beim strengen oder rigorosen 2PL werden die Sperren dann angefordert, wenn sie benötigt werden, beim konservativen 2PL werden alle Sperren zu Beginn gemeinsam angefordert.
- Beim strengen oder rigorosen 2PL werden die Lesesperren bis zum Com-

mit, die Schreibsperren sogar bis nach dem Commit gehalten. Beim konservativen 2PL werden dagegen die Sperren freigegeben, wenn sie nicht mehr benötigt werden.

Das rigorose 2PL wird der Praxis häufiger verwendet, weil dabei nicht zu Beginn der Transaktion bekannt sein muss, welche Sperren benötigt werden, und durch die schrittweise Anforderung der Sperren unter Umständen ein höheres Maß an Parallelität erreicht werden kann.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-2/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Transaktionen T1 und T2“ (66116-2021-F.T1-TA2-A5) [Transaktionen]

Gegeben sind die folgenden transaktionsähnlichen Abläufe. (Zunächst wird auf das Setzen von Sperren verzichtet.) Hierbei steht R(X) für ein Lesezugriff auf X und W(X) für einen Schreibzugriff auf X.

T1	T2
R(A)	R(D)
A := A-10	D := D-20
W(A)	W(D)
R(C)	R(A)
R(B)	A := A+20
B := B+10	W(A)
W(B)	

Betrachten Sie folgenden Schedule:

T1	T2
R(A)	
	R(D)
	D := D-20
	W(D)
	R(A)
	A := A+20
	W(A)
A := A-10	
W(A)	
R(C)	
R(B)	
B := B+10	
W(B)	

(a) Geben Sie die Werte von A, B, C und D nach Ablauf des Schedules an, wenn mit $A = 100$, $B = 200$, $C =$ true und $D = 150$ begonnen wird.

Lösungsvorschlag

A 90 (A := A - 10 := 100 - 10) T2 schreibt 120 in A, was aber von T1 wieder-überschrieben wird.

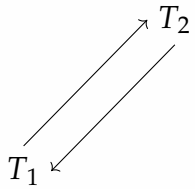
B 210 (B wird nur in T1 gelesen, verändert und geschrieben)

C true (C wird nur in T1 gelesen)

D 130 (D wird nur in T2 gelesen, verändert und geschrieben)

(b) Geben Sie den Dependency-Graphen des Schedules an.

Lösungsvorschlag



(c) Geben Sie alle auftretenden Konflikte an.

Lösungsvorschlag

$R_1(A) < W_2(A)$ resultierende Kante $T_1 \rightarrow T_2$,

$R_2(A) < W_1(A)$ resultierende Kante $T_2 \rightarrow T_1$,

$W_2(A) < W_1(A)$ resultierende Kante $T_2 \rightarrow T_1$ (bereits vorhanden)

(d) Begründen Sie, ob der Schedule serialisierbar ist.

Lösungsvorschlag

Nicht ohne den Einsatz der Lese- und Schreibsperren, denn der Dependency Graph enthält einen Zyklus, womit er nicht konfliktserialisierbar ist.

(e) Beschreiben Sie, wie die beiden Transaktionen mit LOCK Aktionen erweitert werden können, so dass nur noch serialisierbare Schedules ausgeführt werden können. Die Angabe eines konkreten Schedules ist nicht zwingend notwendig.

Lösungsvorschlag

Hier führt die Verwendung des Zwei-Phasen-Sperrpotokolls zur gewünschten Serialisierbarkeit. (Es muss dabei weder die konservative, noch die strenge Variante verwendet werden, damit es funktioniert). T1 würde zu Beginn die und Lese- und Schreibsperre für A anfordern, den Wert verändern, zurückschreiben und anschließend die Sperren für A zurückgeben. Währendessen könnte T2 „ungestört“ die Schreib- und Lesesperren für D anfordern, D lesen, verändern und schreiben, und die Sperren zurückgeben. T2 bemüht sich nun um die Lesesperre für A, muss aber nun so lange warten, bis T1 die Schreibsperre zurückgegen hat. Dadurch kann man den Lost-Update-Fehler vermeiden und erhält allgemein einen serialisierbaren Schedule.

Beispiel für einen konkreten Schedule mit LOCKs (auch wenn nicht zwingend gefordert in der Aufgabenstellung):

T1	T2
rLock(A)	
xLock(A)	
	rLock(D)
	xLock(D)
R1(A)	
	R2(D)
A := A-10	
	D := D-20
	W2(D)
	unLock(D)
	rLock(A) DELAY
W1(A)	
unLock(A)	
	R2(A)
	A := A+20
	W2(A)
	unLock(A)
rLock(C)	
R1(D)	
unLock(C)	
	commit
rLock(B)	
xLock(B)	
R1(B)	
B := B+10	
W1(B)	
unLock(B)	
commit	

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-5.tex`

Übungsaufgabe „ACID“ (Transaktionen, ACID)

Transaktionen
ACID

Beurteilen Sie kurz folgende Aussagen oder Fragen unter ACID-Gesichtspunkten! [1]

(a) Seit dem Abort meiner Transaktion sind deren Änderungen überhaupt nicht mehr vorhanden!

Lösungsvorschlag

Das entspricht der Forderung *Atomicity*, da entweder alle Aktionen oder keine Aktion einer Transaktion ausgeführt werden soll(en). Bei Abbruch einer Transaktion werden die bereits abgearbeiteten Aktionen dieser Transaktion zurückgesetzt.

(b) Leider wurde die erfolgreich abgeschlossene Transaktion zurückgesetzt, da das DBS abgestürzt ist.

Lösungsvorschlag

Das widerspricht der Forderung *Durability*, da erfolgreich abgeschlossene Transaktionen permanent, ðdauerhaft, abgespeichert werden müssen.

(c) Eine andere Transaktion hat Änderungen meiner Transaktion überschrieben. Darf ich jetzt meine Transaktion überhaupt noch beenden oder muss ich sie abbrechen?

Lösungsvorschlag

Das widerspricht der Forderung *Isolation*. Parallel ablaufende Transaktionen dürfen sich nicht beeinflussen, wenn dies also wie hier geschildert der Fall ist, dann muss die Transaktion abgebrochen werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/60_Transaktionsverwaltung/Aufgabe_ACID.tex`

[1] `http://wwwlgis.informatik.uni-kl.de/archiv/wwwdvs.informatik.uni-kl.de/courses/DBSREAL/SS2003/Uebungen/Blatt.01.half.pdf`

Übungsaufgabe „PKW“ (Transaktionen, Deadlock)

Transaktionen

(a) Gegeben ist folgende Situation (die nichts mit einer Datenbank zu tun hat!): Vier PKWs kommen gleichzeitig an eine Kreuzung, an der die Rechts-vor-Links-Vorfahrtsregelung gilt. Welches Problem tritt hier auf?

Lösungsvorschlag

Es tritt eine sogenannte Deadlock-Situation auf. Rein theoretisch müsste der Verkehr zum Erliegen kommen, denn jedes Auto müsste einem anderen Auto die Vorfahrt gewähren. Jedes KFZ ist mit einem Verkehrsteilnehmer konfrontiert, der von rechts kommt.

(b) Gegeben sind die Transaktionen T_1 und T_2.

T_1	T_2
BOT	BOT
...	...
SELECT F1 FROM TAB	SELECT F2 FROM TAB
...	...
SELECT F2 FROM TAB	SELECT F1 FROM TAB
...	...
COMMIT WORK	COMMIT WORK

TAB	
F1	F2
2	3

Geben Sie eine quasiparallele Verarbeitung von T_1 und T_2 an, bei der es zum „gleichen“ Problem wie in Aufgabe a) kommt.

Hinweis: Wir nehmen an, dass eine Spalte F der Tabelle TAB durch `rlock(F)` bzw. `xlock(F)` gesperrt werden kann.

Lösungsvorschlag

In der 8. Zeile entsteht ein Deadlock, da von verschiedenen Transaktionen `rlocks` auf `F2` gesetzt wurden. Jetzt will T_1 auf `F2` einen `xlock` setzten, was nicht möglich ist, weil der `rlock` von T_2 noch nicht frei gegeben wurde.

	T_1	T_2	
1	BOT		
2		BOT	
3	rlock(F1)		
4		rlock(F2)	
5	SELECT F1 FROM TAB		
6	rlock(F2)		
7	SELECT F2 FROM TAB		
8	xlock(F2)		$\leftarrow$ Deadlock
9		SELECT F2 FROM TAB	
10	UPDATE TAB SET F2 = F1		

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/60_Transaktionsverwaltung/Aufgabe_PKW.tex`

Übungsaufgabe „Tabelle TAB“ (Transaktionsverwaltung, Lost-Update, Dirty-Read)

Transaktionsverwaltung
Lost-Update
Dirty-Read

Die Transaktionen eines Transaktionsprogramms besteht aus SQL-Befehlen. Die Transaktionen T_1 und T_2 arbeiten auf der Tabelle TAB.

Transaktion T_1	Transaktion T_2
BOT	BOT
SELECT FROM TAB	SELECT FROM TAB
NEUF := F+5	NEUF := F*2
UPDATE TAB SET F=NEUF	UPDATE TAB SET F=NEUF
COMMIT WORK	COMMIT WORK

TAB
F
2

Die quasiparallele Abarbeitung erfolgt in folgenden Schritten:

	T_1	T_2
1		BOT
2	BOT	
3	SELCT F FROM TAB	
4		SELECT F FROM TAB
5		NEUF := F*2
6	NEUF := F+5	
7	UPDATE TAB SET F=NEUF	
8	COMMIT WORK	
9		UPDATE TAB SET F=NEUF
10		COMMIT WORK

(a) Ist die (quasiparallele) Bearbeitung der Transaktionen korrekt? Begründung!

Lösungsvorschlag

Nein, es liegt ein Lost-Update-Fehlerfall vor. In Schritt 3 bzw. 4 lesen T_1 bzw. T_2 denselben Wert aus der Tabelle TAB. Der von T_1 in Schritt 7 in die Tabelle zurückgeschriebene Wert wird in Schritt 9 von T_2 überschrieben.

(b) Konstruieren Sie unter Verwendung von T_1 und T_2 einen Dirty-Read-Fehlerfall.

Lösungsvorschlag

	T_1	T_2
1		BOT
2	BOT	
3	SELCT F FROM TAB	
4	NEUF := F+5	
5	UPDATE TAB SET F=NEUF	
6		SELECT F FROM TAB
7		NEUF := F*2
8	ABORT	
9		UPDATE TAB SET F=NEUF
10		COMMIT WORK

Dirty-Read bedeutet, dass von zwei gleichzeitig ablaufenden Transaktionen die eine Transaktion Daten liest, die von der anderen Transaktion geschrieben bzw. geändert werden, jedoch noch nicht bestätigt (committed) sind. Somit ist noch nicht sichergestellt, dass diese Daten permanent in die Datenbank übernommen werden. Findet dann ein Abort statt, hat die eine Transaktion Daten ausgelesen, die am Ende nicht in der Datenbank ankommen.

Sobald T_1 committed hat, kann es dazu nicht mehr kommen, da dann die Änderungen von T_1 permanent in der Datenbank festgeschrieben sind und T_2 auf sichere, garantierte Werte zugreift.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/10_DB/60_Transaktionsverwaltung/Aufgabe_Tabelle-TAB.tex`

Sonstige

Examensaufgabe „Wissensfragen“ (66116-2019-H.T1-TA2-A1)

Antworten Sie kurz und prägnant.

(a) Nennen Sie einen Vorteil und einen Nachteil der Schichtenarchitektur.

Lösungsvorschlag

Vorteil

Physische Datenunabhängigkeit:

Die interne Ebene ist von der konzeptionellen und externen Ebene getrennt.

Physische Änderungen, z. B. des Speichermediums oder des Datenbankprodukts, wirken sich nicht auf die konzeptionelle oder externe Ebene aus.

Logische Datenunabhängigkeit:

Die konzeptionelle und die externe Ebene sind getrennt. Dies bedeutet, dass Änderungen an der Datenbankstruktur (konzeptionelle Ebene) keine Auswirkungen auf die externe Ebene, also die Masken-Layouts, Listen und Schnittstellen haben.

[a]

Nachteil

Overhead durch zur Trennung der Ebenen benötigten Schnittstellen

[a] `https://de.wikipedia.org/wiki/ANSI-SPARC-Architektur`

(b) Wie ermöglicht es ein Datenbankensystem, verschiedene Sichten darzustellen?

Lösungsvorschlag

Die Sichten greifen auf die zwei darunterliegenden Abstraktionsebenen eines Datenbanksystems zu, nämlich auf die logische Ebene und die logische Ebene greift auf die physische Ebene zu.

(c) Was beschreibt das Konzept der Transitiven Hülle? Erklären Sie dies kurz und nennen Sie ein Beispiel für (1) die Transitive Hülle eines Attributes bei funktionalen Abhängigkeiten und (2) die Transitive Hülle einer SQL-Anfrage.

Lösungsvorschlag

Die transitive Hülle einer Relation R mit zwei Attributen A und B gleichen Typs ist definitert als

Sie enthält damit alle Tupel (a, b), für die ein Pfad beliebiger Länge k in R

existiert.

Berechnung rekursiver Anfragen (z. B. transitive Hülle) über rekursiv definierte Sichten (Tabellen)[a]

[a] https://dbs.uni-leipzig.de/file/dbs2-ss16-kap4.pdf

(d) Nennen Sie zwei Indexstrukturen und beschreiben Sie jeweils ihren Vorteil.

Lösungsvorschlag

In Hauptspeicher-Datenbanksystemen werden oft Hashtabellen verwendet, um effizierte Punkt-Abfragen (exact Match) zu unterstützen.

Wenn auch Bereichs-Abfragen (range queries) vorkommen, werden zumeister balancierte Suchbäume - AVL- oder rot/schwarz-Bäume - verwendet.

a

[a] https://de.wikipedia.org/wiki/Indexstruktur

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-2/Aufgabe-1.tex

Examensaufgabe „Vermischte Fragen“ (66116-2021-F.T2-TA2-A1)

DB

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

(a) Kann ein Tupel mehrfach im Ergebnis einer SQL-Anfrage enthalten sein?

Lösungsvorschlag

Ja. Geben wir nur eine Teilmenge an Attributen aus (z. B. ohne Primärschlüssel), so kann ein Tupel mehrfach in der Ausgabe erscheinen.

Außerdem ist es möglich eine Tabelle ohne `PRIMARY KEY` anzulegen. In so einer Tabelle kann dann eine Tupel mehrmals gespeichert werden und über `SELECT * FROM ...` mehrmals ausgegeben werden. (getestet in MySQL und in PostgreSQL).

```
CREATE TABLE tmp (
  tmp INTEGER
);

INSERT INTO tmp VALUES
  (1),
  (1),
  (1);

SELECT * FROM tmp;

+------+
| tmp  |
+------+
|    1 |
|    1 |
|    1 |
+------+
```

Um die mehrfache Ausgabe zu verhindert, gibt es in SQL das Schlüsselwort `DISTINCT`. In der Relationalen Algebra hingegeben sind die Tupel einer Relation eindeutig.

(b) Was ist der Unterschied zwischen einem `INNER JOIN` und einem `OUTER JOIN`?

Lösungsvorschlag

Ein `INNER JOIN` entspricht der Schnittmenge $A \cap B$.

Ein `OUTER JOIN` entspricht der Vereinigung $A \cup B$. Bei `OUTER JOIN`s können auch NULL-Werte vorkommen. [a]

[a] `https://stackoverflow.com/a/38578`

(c) Welche Auswirkung hat die Verwendung von `ON DELETE CASCADE` bei einem Fremdschlüsselattribut?

Lösungsvorschlag

Ist `ON DELETE CASCADE` bei einem Fremdschlüsselattribut gesetzt, so wird der referenzierte Datensatz bei einem Löschvorgang mitgelöscht.

(d) Kann eine abgebrochene (aborted) Transaktion wieder fortgesetzt werden?

Lösungsvorschlag

Eine Transaktion kann nicht fortgesetzt werden. Sie muss zurückgesetzt und wiederholt werden.

(e) Was versteht man unter einer `stored procedure` im Kontext einer Programmierschnittstelle für relationale Datenbanken (z.B JDBC)?

Lösungsvorschlag

Eine `stored procedure` bildet eine Gruppe von SQL-Befehlen, die eine logische Einheiten bildet und einer bestimmten Aufgabe zugeordnet sind. `stored procedure` werden dazu benutzt eine mehrere Anweisungen und Abfragen zu koppeln

Beispielsweise können bei einer Angestellten-Datenbank die Aufgaben „einstellen“, „entlassen“, „befördern“ als `stored procedure` kompiliert werden und dann mit unterschiedlichen Parametern ausgeführt werden. [a]

[a] `https://docs.oracle.com/javase/tutorial/jdbc/basics/storedprocedures.html`

(f) Was sind `check constraints` und wie wirken sich diese aus?

Lösungsvorschlag

Constraints definieren Bedingungen, die beim Einfügen, Ändern und Löschen von Datensätzen in der Datenbank erfüllt werden müssen. Wird beispielsweise eine Bedinung bei Einfügen eines Datensatzen nicht erfüllt, so kann dieser Datensatz nicht gespeichert werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Optimierung“ (66116-2021-F.T2-TA2-A7)

DB

(a) Erläutern Sie kurz, was Indizes sind und warum diese in Datenbanksystemen verwendet werden.

Lösungsvorschlag

Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.

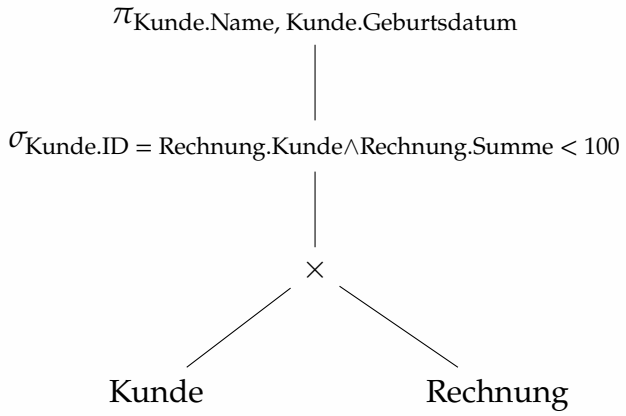
Ein Index besteht aus einer Ansammlung von Zeigern (Verweisen), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren. Wird bei einer Abfrage eine indizierte Spalte als Suchkriterium herangezogen, sucht das Datenbankmanagementsystem (DBMS) die gewünschten Datensätze anhand dieser Zeiger. In der Regel finden hier B+-Bäume Anwendung. Ohne Index müsste die Spalte sequenziell durchsucht werden, während eine Suche mit Hilfe des Baums nur logarithmische Komplexität hat. [a]

[a] `https://de.wikipedia.org/wiki/Datenbankindex`

(b) Übertragen Sie folgendes SQL-Statement in einen nicht optimierten algebraischen Term oder in einen Anfragegraphen.

```
SELECT Kunde.Name, Kunde.Geburtsdatum
FROM Kunde, Rechnung
WHERE Kunde.ID = Rechnung.Kunde
AND Rechnung.Summe < 100;
```

Lösungsvorschlag

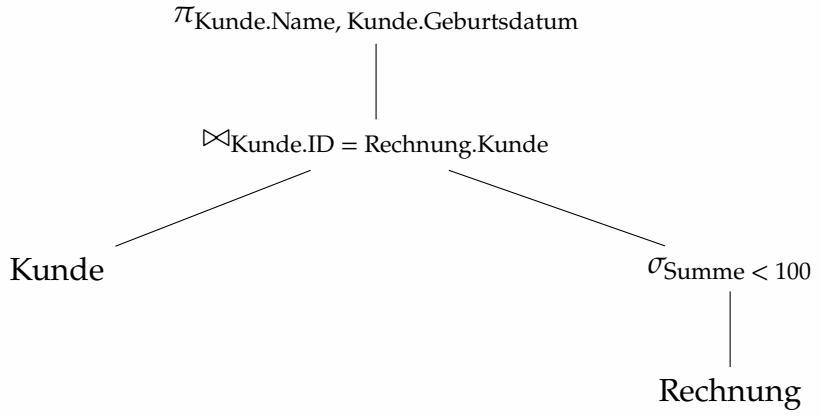


(c) Nennen Sie zwei Möglichkeiten, den algebraischen Term bzw. den Anfragegraphen aus der vorhergehenden Teilaufgabe logisch (d. h. algebraisch) zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten algebraischen Ausdrucks.

Lösungsvorschlag

Zuerst Selection und dann Join

Theta-Join



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-7.tex`

Teil II

Algorithmen und Datenstrukturen (AUD)

Rekursion

Examensaufgabe „Binomialkoeffizient“ (46115-2014-F.T2-A4)

Aufgabe 4

Für Binomialkoeffizienten $\binom{n}{k}$ gelten neben den grundlegenden Beziehungen $\binom{n}{0} = 1$ und $\binom{n}{n} = 1$ auch folgende Formeln:

> **Exkurs: Binomialkoeffizient**
> Der Binomialkoeffizient ist eine mathematische Funktion, mit der sich eine der Grundaufgaben der Kombinatorik lösen lässt. Er gibt an, auf wie viele verschiedene Arten man k bestimmte Objekte aus einer Menge von n verschiedenen Objekten auswählen kann (ohne Zurücklegen, ohne Beachtung der Reihenfolge). Der Binomialkoeffizient ist also die Anzahl der k-elementigen Teilmengen einer n-elementigen Menge. [a]
>
> [a] `https://de.wikipedia.org/wiki/Binomialkoeffizient`

A $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$

B $\binom{n}{k} = \binom{n-1}{k-1} \cdot \frac{n}{k}$

(a) Implementieren Sie unter Verwendung von Beziehung (A) eine rekursive Methode `binRek(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

Lösungsvorschlag

Zuerst verwandeln wir die Beziehung (A) geringfügig um, indem wir n durch $n-1$ ersetzen:
$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

```java
public static int binRek(int n, int k) {
  if (k == 0 || k == n) {
    return 1;
  } else {
    return binRek(n - 1, k - 1) + binRek(n - 1, k);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

(b) Implementieren Sie unter Verwendung von Beziehung (B) eine iterative Methode `binIt(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

Lösungsvorschlag

```java
public static int binIt(int n, int k) {
  // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
  // Zwischenergebnisse ganze Zahlen sind.
  double ergebnis = 1;
  while (k > 0) {
    ergebnis = ergebnis * n / k;
    n--;
    k--;
  }
  // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
  // umgewandelt werden.
  return (int) ergebnis;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

(c) Geben Sie die Laufzeitkomplexität der Methoden `binRek(n, k)` und `binIt(n, k)` aus den vorhergehenden beiden Teilaufgaben in O-Notation an!

Komplette Java-Klasse

```java
/**
 * <a href="https://www.studon.fau.de/file2889270_download.html">Angabe:
↪  PUE_AUD_WH.pdf</a>
 * <a href="https://www.studon.fau.de/file3081306_download.html">Lösung:
↪  PUE_AUD_WH_Lsg.pdf</a>
 */
public class Binomialkoeffizient {

  /**
   * Berechnet rekursiv den Binominalkoeffizienten „n über k". Dabei muss gelten:
   * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
   *
   * @param n Ganzzahl n
   * @param k Ganzzahl k
   *
   * @return Eine Ganzzahl.
   */
  public static int binRek(int n, int k) {
    if (k == 0 || k == n) {
      return 1;
    } else {
      return binRek(n - 1, k - 1) + binRek(n - 1, k);
    }
  }

  /**
   * Berechnet iterativ den Binominalkoeffizienten „n über k". Dabei muss gelten:
   * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
   *
   * @param n Ganzzahl n
   * @param k Ganzzahl k
```

```
   *
   * @return Eine Ganzzahl.
   */
  public static int binIt(int n, int k) {
    // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
    // Zwischenergebnisse ganze Zahlen sind.
    double ergebnis = 1;
    while (k > 0) {
      ergebnis = ergebnis * n / k;
      n--;
      k--;
    }
    // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
    // umgewandelt werden.
    return (int) ergebnis;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

Test

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BinomialkoeffizientTest {

  public void testeRek(int n, int k, int ergebnis) {
    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
  }

  public void testeIt(int n, int k, int ergebnis) {
    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
  }

  public void teste(int n, int k, int ergebnis) {
    testeRek(n, k, ergebnis);
    testeIt(n, k, ergebnis);
  }

  @Test
  public void teste() {
    teste(0, 0, 1);

    teste(1, 0, 1);
    teste(1, 1, 1);

    teste(2, 0, 1);
    teste(2, 1, 2);
    teste(2, 2, 1);

    teste(3, 0, 1);
    teste(3, 1, 3);
    teste(3, 2, 3);
```

```
    teste(3, 3, 1);

    teste(4, 0, 1);
    teste(4, 1, 4);
    teste(4, 2, 6);
    teste(4, 3, 4);
    teste(4, 4, 1);
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/BinomialkoeffizientTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2014/03/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Klasse „LeftFactorial“ und Methode „lfBig()““ (66115-2014-F.T1-A1)

(a) Gegeben sei die Methode `BigInteger lfBig(int n)` zur Berechnung der eingeschränkten Linksfakultät:

$$!n := \begin{cases} n!(n-1) - (n-1)!(n-2) & \text{falls } 1 < n < 32767 \\ 1 & \text{falls } n = 1 \\ 0 & \text{sonst} \end{cases}$$

```
import java.math.BigInteger;
import static java.math.BigInteger.ZERO;
import static java.math.BigInteger.ONE;

public class LeftFactorial {

  BigInteger sub(BigInteger a, BigInteger b) {
    return a.subtract(b);
  }

  BigInteger mul(BigInteger a, BigInteger b) {
    return a.multiply(b);
  }

  BigInteger mul(int a, BigInteger b) {
    return mul(BigInteger.valueOf(a), b);
  }

  // returns the left factorial !n
  BigInteger lfBig(int n) {
    if (n <= 0 || n >= Short.MAX_VALUE) {
      return ZERO;
    } else if (n == 1) {
      return ONE;
    } else {
      return sub(mul(n, lfBig(n - 1)), mul(n - 1, lfBig(n - 2)));
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

Implementieren Sie unter Verwendung des Konzeptes der *dynamischen Programmierung* die Methode `BigInteger dp(int n)`, die jede $!n$ auch bei mehrfachem Aufrufen mit dem gleichen Parameter höchstens einmal rekursiv berechnet. Sie dürfen der Klasse `LeftFactorial` genau ein Attribut beliebigen Datentyps hinzufügen und die in `lfBig(int)` verwendeten Methoden und Konstanten ebenfalls nutzen.

Lösungsvorschlag

Wir führen ein Attribut mit dem Namen `store` ein und erzeugen ein Feld vom Typ `BigInteger` mit der Länge $n + 1$. Die Länge des Feld $n + 1$ hat den

Vorteil, dass nicht ständig $n - 1$ verwendet werden muss, um den gewünschten Wert zu erhalten.

In der untenstehenden Implementation gibt es zwei Methoden mit dem Namen `dp`. Die untenstehende Methode ist nur eine Hüllmethode, mit der nach außen hin die Berechnung gestartet und das `store`-Feld neu gesetzt wird. So ist es möglich `dp()` mehrmals hintereinander mit verschiedenen Werten aufzurufen (siehe `main()`-Methode).

```
BigInteger[] store;

BigInteger dp(int n, BigInteger[] store) {
  if (n > 1 && store[n] != null) {
    return store[n];
  }
  if (n <= 0 || n >= Short.MAX_VALUE) {
    return ZERO;
  } else if (n == 1) {
    return ONE;
  } else {
    BigInteger result = sub(mul(n, dp(n - 1, store)), mul(n - 1, dp(n - 2,
↪ store)));
    store[n] = result;
    return result;
  }
}

BigInteger dp(int n) {
  store = new BigInteger[n + 1];
  return dp(n, store);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

(b) Betrachten Sie nun die Methode `lfLong(int)` zur Berechnung der vorangehend definierten Linksfakultät ohne obere Schranke. Nehmen Sie im Folgenden an, dass der Datentyp `long` unbeschränkt ist und daher kein Überlauf auftritt.

```
long lfLong(int n) {
  if (n <= 0) {
    return 0;
  } else if (n == 1) {
    return 1;
  } else {
    return n * lfLong(n - 1) - (n - 1) * lfLong(n - 2);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

Beweisen Sie *formal* mittels *vollständiger Induktion*:

$$\forall n \geq 0 : \text{lfLong}(n) \equiv \sum_{k=0}^{n-1} k!$$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$n = 1 \Rightarrow \texttt{lfLong(1)} = 1 = \sum_{k=0}^{n-1} k! = 0! = 1$$

$$\begin{aligned} n = 2 \Rightarrow & \text{lfLong}(2) \\ & = (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! \\ & = \texttt{2 * lfLong(1) - 1 * lfLong(0)} \\ & = 2 \\ & = \sum_{k=0}^{1} k! \\ & = 1! + 0! \\ & = 1 + 1 \\ & = 2 \end{aligned}$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\texttt{lfLong(n)} = \sum_{k=0}^{n-1} k!$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss.

$$
\begin{aligned}
&A(n+1) \\
&\quad = \texttt{lfLong(n + 1)} \\
&\quad = \texttt{(n + 1) * lfLong(n) - n * lfLong(n - 1)} \\
&\quad = (n+1)\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{(n-1)-1} k! && \text{Formel eingesetzt} \\
&\quad = (n+1)\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{subtrahiert} \\
&\quad = n\sum_{k=0}^{n-1} k! + \sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{ausmultipliziert mit } (n+1) \\
&\quad = \sum_{k=0}^{n-1} k! + n\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{Reihenfolge der Terme geändert} \\
&\quad = \sum_{k=0}^{n-1} k! + n\left((n-1)! + \sum_{k=0}^{n-2} k!\right) - n\sum_{k=0}^{n-2} k! && (n-1)! \text{ aus Summenzeichen entfernt} \\
&\quad = \sum_{k=0}^{n-1} k! + n\left((n-1)! + \sum_{k=0}^{n-2} k! - \sum_{k=0}^{n-2} k!\right) && \text{Distributivgesetz } ac - bc = (a-b)c \\
&\quad = \sum_{k=0}^{n-1} k! + n(n-1)! && +\Sigma - \Sigma = 0 \\
&\quad = \sum_{k=0}^{n-1} k! + n! && \text{Fakultät erhöht} \\
&\quad = \sum_{k=0}^{n} k! && \text{Element zum Summenzeichen hinzugefügt} \\
&\quad = \sum_{k=0}^{(n+1)-1} k! && \text{mit } (n+1) \text{ an der Stelle von } n
\end{aligned}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/03/Thema-1/Aufgabe-1.tex`

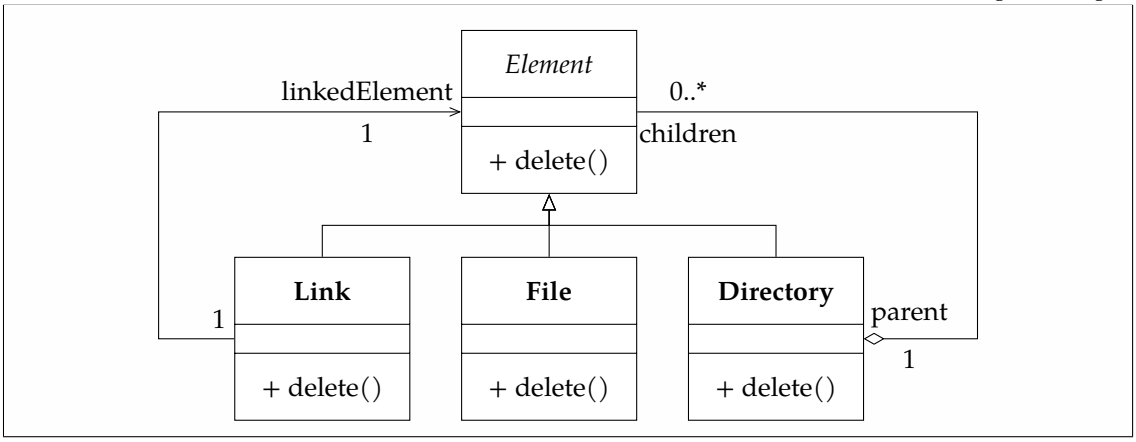
Examensaufgabe „Dateisystem: Implementierung durch Kompositum“ (66116-2019-H.T2-TA1-A1)

Entwurfsmuster
Kompositum (Composite)
Klassendiagramm
Implementierung in Java

Hierarchische Dateisysteme bestehen aus den FileSystemElements Ordner, Dateien und Verweise. Ein Ordner kann seinerseits Ordner, Dateien und Verweise beinhalten; jedem Ordner ist bekannt, welche Elemente (children) er enthält. Mit Ausnahme des Root-Ordners auf der obersten Hierarchieebene ist jeder Ordner, jede Datei und jeder Verweis Element eines Elternordners. Jedem Element ist bekannt, was sein Elternordner ist (parent). Ein Verweis verweist auf einen Verweis, eine Datei oder einen Ordner (link). Wenn ein Ordner gelöscht wird, werden alle seine Bestandteile ggf. rekursiv ebenfalls gelöscht. Sie dürfen die Lösungen für Aufgabenteil b) und c) in einem gemeinsamen Code kombinieren.

(a) Modellieren Sie diesen Sachverhalt mit einem UML-Klassendiagramm. Benennen Sie die Rollen von Assoziationen und geben Sie alle Kardinalitäten an. Ihre Lösung soll mindestens eine sinnvolle Spezialisierungsbeziehung enthalten.

Lösungsvorschlag



(b) Implementieren Sie das Klassendiagramm als Java- oder C++-Programm. Jedes Element des Dateisystems soll mindestens über ein Attribut `name` verfügen. Übergeben Sie den Elternordner jedes Elements als Parameter an den Konstruktor; der Elternordner des Root-Ordners kann dabei als `null` implementiert werden. Dokumentieren Sie Ihren Code.

Lösungsvorschlag

a

```
public abstract class Element {

  protected String name;

  protected Element parent;

  protected Element(String name, Element parent) {
    this.name = name;
    this.parent = parent;
  }
```

```
  public String getName() {
    return name;
  }

  public abstract void delete();

  public abstract boolean isDirectory();

  public abstract void addChild(Element child);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Element.java`

```
import java.util.ArrayList;
import java.util.List;

public class Directory extends Element {
  private List<Element> children;

  public Directory(String name, Element parent) {
    super(name, parent);
    children = new ArrayList<Element>();
    if (parent != null)
      parent.addChild(this);
  }

  public void delete() {
    System.out.println("The directory "" + name +
↪   "" was deleted and it's children were also deleted.");
    for (int i = 0; i < children.size(); i++) {
      Element child = children.get(i);
      child.delete();
    }
  }

  public void addChild(Element child) {
    children.add(child);
  }

  public boolean isDirectory() {
    return true;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Directory.java`

```
public class File extends Element {

  public File(String name, Element parent) {
    super(name, parent);
    parent.addChild(this);
  }
```

```
  public void delete() {
    System.out.println("The File "" + name + "" was deleted.");
  }

  public boolean isDirectory() {
    return false;
  }

  /**
   * Eine Datei kann keine Kinder haben. Deshalb eine Methode mit leerem
   * Methodenrumpf.
   */
  public void addChild(Element child) {
  }

}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/File.java

```
public class Link extends Element {

  private Element linkedElement;

  public Link(String name, Element parent, Element linkedElement) {
    super(name, parent);
    this.linkedElement = linkedElement;
    parent.addChild(this);
  }

  public void delete() {
    System.out.println("The Symbolic Link "" + name + "" was deleted.");
    linkedElement.delete();
    System.out.println("The linked element "" + name + "" was deleted too.");
  }

  public void addChild(Element child) {
    if (linkedElement.isDirectory())
      linkedElement.addChild(child);
  }

  public boolean isDirectory() {
    return linkedElement.isDirectory();
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Link.java

[a] TU Darmstadt: Dr. Michael Eichberg - Case Study Using the Composite and Proxy Design Patterns

(c) Ordnen Sie eine Methode `delete`, die Dateien, Ordner und Verweise rekursiv löscht, einer oder mehreren geeigneten Klassen zu und implementieren Sie sie. Zeigen Sie die Löschung jedes Elements durch eine Textausgabe von `name` an. `toString()` müssen Sie dabei nicht implementieren. Gehen Sie davon aus, dass

Verweis- und Ordnerstrukturen azyklisch sind und dass jedes Element des Dateisystems höchstens einmal existiert. Wenn ein Verweis gelöscht wird, wird sowohl der Verweis als auch das verwiesene Element bzw. transitiv die Kette der verwiesenen Elemente gelöscht. Bedenken Sie, dass die Löschung eines Elements immer auch Konsequenzen für den dieses Element beinhaltenden Ordner hat. Es gibt keinen Punktabzug, wenn Sie die Löschung des Root-Ordners nicht zulassen. Rekursion

Lösungsvorschlag

Siehe Antwort zu Aufgabe b)

(d) Was kann im Fall von `delete` passieren, wenn die Linkstruktur zyklisch ist oder die Ordnerstruktur zyklisch ist? Kann es zu diesen Problemen auch dann führen, wenn weder die Linkstruktur zyklisch ist, noch die Ordnerstruktur zyklisch ist? Wie kann man im Programm das Problem lösen, falls man Zyklizitäten zulassen möchte?

Lösungsvorschlag

Falls die Link- oder Ordnerstruktur zyklisch ist, kann es aufgrund der Rekursion zu einer Endlosschleife kommen. Diese Problem tritt bei azyklischen Strukturen nicht auf, weil der rekursive Löschvorgang beim letzten Element abgebrochen wird (Abbruchbedingung).

Das Problem kann zum Beispiel durch ein neues Attribut `gelöscht` in der Klasse `Link` oder `Directory` gelöst werden. Dieses Attribut wird auf `true` gesetzt, bevor es in die Rekursion einsteigt. Rufen sich die Klassen wegen der Zyklizität selbst wieder auf, kommt es durch entsprechende IF-Bedingungen zum Abbruch.

Außerdem ist ein Zähler denkbar, der sich bei jeder Rekursion hochsetzt und ab einem gewissen Grenzwert zum Abbruch führt.

(e) Was ist ein Design Pattern? Nennen Sie drei Beispiele und erläutern Sie sie kurz. Welches Design Pattern bietet sich für die Behandlung von hierarchischen Teil-Ganzes-Beziehungen an, wie sie im Beispiel des Dateisystems vorliegen?

Lösungsvorschlag

Design Pattern sind wiederkehrende, geprüfte, bewährte Lösungsschablonen für typische Probleme.

Drei Beispiele

Einzelstück (Singleton) Stellt sicher, dass nur *genau eine Instanz einer Klasse* erzeugt wird.

Beobachter (Observer) Das Observer-Muster ermöglicht einem oder mehreren Objekten, automatisch auf die *Zustandsänderung* eines bestimmten Objekts *zu reagieren*, um den eigenen Zustand anzupassen.

Stellvertreter (Proxy) Ein Proxy stellt einen Platzhalter für eine andere Komponente (Objekt) dar und kontrolliert den Zugang zum echten Objekt.

Für hierarchischen Teil-Ganzes-Beziehungen eigent sich das Kompositum (Composite). Es ermöglicht die Gleichbehandlung von Einzelelementen und Elementgruppierungen in einer verschachtelten Struktur (z. B. Baum), sodass aus Sicht des Clients keine explizite Unterscheidung notwendig ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Übungsaufgabe „Feld-Invertierer“ (Rekursion, Implementierung in Java)

Rekursion
Implementierung in Java

(a) Erstellen Sie eine neue Klasse `ArrayInvertierer` mit einer rekursiven Methode, die den Inhalt eines ihr übergebenen 1D-Arrays gefüllt mit Strings invertiert. Auf diese Weise kann z. B. ein deutscher Satz im Array gespeichert und dann verkehrt herum ausgegeben werden.

Wichtig: Nicht das übergebene Array soll verändert werden, sondern ein Neues erstellt und von der Methode zurückgegeben werden.

Tipp: Sie dürfen dafür gerne auch rekursive Hilfsmethoden benutzen.

(b) Implementieren Sie dann eine `main`-Methode, in der Sie zwei verschieden lange `String`-Arrays erzeugen und die Wortreihenfolge umkehren lassen. Das Ergebnis soll auf der Konsole ausgegeben werden und könnte z. B. wie folgt aussehen.

```
Den Satz
Ich find dich einfach klasse!
wuerde Meister Yoda so aussprechen:
klasse! einfach dich find Ich


Den Satz
Das war super einfach/schwer
wuerde Meister Yoda so aussprechen:
einfach/schwer super war Das
```

[optional] Wenn das ursprüngliche `String`-Array selbst verändert werden soll, braucht die rekursive Methode keine Rückgabe. Versuchen Sie, diese Aufgabe ohne das Nutzen einer Hilfsmethode zu lösen.

Lösungsvorschlag

```java
public class ArrayInvertierer {

  /**
   * Invertiert das übergebene String Feld rekursiv durch Aufruf der Methode
↪   in
   * {@link invertiereRekursiv} mit dem jeweils aktuellen Arrayindex als
   * Startwert.
   *
   * @param quelle Das Feld, dessen Inhalt invertiert werden soll.
   * @param ziel   Hilfs-Feld.
   * @param index  aktueller Index
   */
  private static void invertiereRekursiv(String[] quelle, String[] ziel, int
↪   index) {
    if (index < quelle.length) {
      ziel[quelle.length - index - 1] = quelle[index];
      invertiereRekursiv(quelle, ziel, ++index);
    }
  }
```

```
  /**
   * Invertiert das übergebene String Feld rekursiv durch Aufruf der Methode
↪   in
   * {@link invertiereRekursiv} mit dem Hilfsfeld und dem ersten Feldindex
↪   als
   * Startwert.
   *
   * @param quelle Das Feld, dessen Inhalt invertiert werden soll.
   *
   * @return Ein neues Feld, das den umgekehrten Inhalt besitzt.
   */
  private static String[] invertiereRekursiv(String[] quelle) {
    String[] ziel = new String[quelle.length];
    invertiereRekursiv(quelle, ziel, 0);
    return ziel;
  }

  /**
   * Die Lösung für die optionale Aufgaben. In situ bedeutet, dass kein neues
↪   Feld
   * erzeugt wird.
   *
   * @param quelle Ein Feld mit Wörtern.
   * @param index  Die Index-Nummer, die bearbeitet werden soll.
   */
  private static void invertiereRekursivInSitu(String[] quelle, int index) {
    if (index < quelle.length / 2) {
      int gespiegelterIndex = quelle.length - 1 - index;
      String tmp = quelle[gespiegelterIndex];
      quelle[gespiegelterIndex] = quelle[index];
      quelle[index] = tmp;
      invertiereRekursivInSitu(quelle, ++index);
    }
  }

  /**
   * Hilfsmethode zur Ausgabe des String-Arrays in einem Satz.
   *
   * @param feld Ein Feld mit Wörtern.
   */
  private static void gibFeldAus(String[] feld) {
    System.out.println(String.join(" ", feld));
  }

  /**
   * Lass Meister Yoda sprechen.
   *
   * @param satz   Ein Feld mit Wörtern.
   * @param inSitu Bei wahr wird die Methode {@link invertiereRekursivInSitu}
   *               verwendet. Achtung! Dadurch wird das Feld verändert.
   */
  public static void lassYodaSprechen(String[] satz, boolean inSitu) {
    System.out.println("\nDen Satz");
```

```
    System.out.print("  ");
    gibFeldAus(satz);
    System.out.println("würde Meister Yoda so aussprechen:");
    System.out.print("  ");
    if (!inSitu) {
      gibFeldAus(invertiereRekursiv(satz));
    } else {
      invertiereRekursivInSitu(satz, 0);
      gibFeldAus(satz);
    }
  }

  public static void main(String[] args) {
    lassYodaSprechen(new String[] { "Ich", "find", "dich", "einfach",
↪   "klasse!" }, false);
    lassYodaSprechen(new String[] { "Das", "war", "super", "einfach/schwer"
↪   }, true);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_2/ArrayInvertierer.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/10_Rekursion/Aufgabe_Feld-Invertierer.tex`

Übungsaufgabe „Fibonacci Fakultät“ (Rekursion)


```java
public class Rekursion {

  public static int fak(int n) {
    if (n == 1) {
      return 1;
    }
    return n * fak(n - 1);
  }

  public static int fib(int n) {
    if (n <= 1) {
      return n;
    }
    return fib(n - 1) + fib(n - 2);
  }

  public static void main(String[] args) {
    System.out.println(fak(6));
    System.out.println(fib(6));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/pu_1/Rekursion.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/10_Rekursion/Aufgabe_Fibonacci-Fakultaet.tex`

Übungsaufgabe „Potenz“ (Rekursion)

Gegeben ist folgende Methode.

```
public int function(int b, int e) {
  if (e == 1) {
    return b * 1;
  } else {
    e = e - 1;
    return b * function(b, e);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_1/Rekursion.java`

(a) Beschreiben Sie kurz, woran man erkennt, dass es sich bei der gegebenen Methode um eine rekursive Methode handelt. Gehen Sie dabei auf wichtige Bestandteile der rekursiven Methode ein.

Lösungsvorschlag

Die Methode mit dem Namen `function` ruft sich in der letzten Code-Zeile selbst auf. Außerdem gibt es eine Abbruchbedingung (`if (e == 1) { return b * 1; }`), womit verhindert wird, dass die Rekursion unendlich weiter läuft.

(b) Geben Sie die Rekursionsvorschrift für die Methode `function` an. Denken Sie dabei an die Angabe der Zahlenbereiche!

Lösungsvorschlag

$$\text{int function(int b, int e)} = \begin{cases} \text{return b*1,} & \text{falls e} = 1. \\ \text{return b*function(b,e-1),} & \text{falls e} > 1. \end{cases}$$

(c) Erklären Sie kurz, was die Methode `function` berechnet.

Lösungsvorschlag

Die Methode `function` berechnet die Potenz b^e.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/10_Rekursion/Aufgabe_Potenz.tex`

Übungsaufgabe „Rater“ (Rekursion)


```java
import java.util.Scanner;

public class Rater {

  public static int rateRekursiv(int anfang, int ende) {
    int mitte;
    int antwort;
    int ergebnis;
    if (anfang == ende) {
      ergebnis = anfang; // Abbruchfall
    } else {
      mitte = (anfang + ende) / 2; // halbiere Intervall
      System.out.println("Ist Ihre Zahl groesser als " + mitte + "? 0: ja, 1: nein");
      Scanner scanner = new Scanner(System.in); // Antwort einlesen
      antwort = scanner.nextInt();
      scanner.close();
      if (antwort == 0) { // suche rechts
        ergebnis = rateRekursiv(mitte + 1, ende);
      } else {
        ergebnis = rateRekursiv(anfang, mitte); // suche links
      }
    }
    return ergebnis;
  }

  public static int rateIterativ(int anfang, int ende) {
    int mitte;
    int antwort;
    while (anfang != ende) {
      mitte = (anfang + ende) / 2; // halbiere das Intervall
      System.out.println("Ist Ihre Zahl groesser als " + mitte + "? 0:ja, 1: nein");
      Scanner scanner = new Scanner(System.in); // Antwort einlesen
      antwort = scanner.nextInt();
      scanner.close();
      if (antwort == 0) {
        anfang = mitte + 1; // suche rechts
      } else {
        ende = mitte; // suche links
      }
    }
    return anfang;
  }

  public static void main(String[] args) {
    System.out.println("Ihre Zahl ist " + rateIterativ(1, 500));
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/pu_1/Rater.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/10_Rekursion/Aufgabe_Rater.tex`

Übungsaufgabe „iterativ-rekursiv“ (Iterative Realisation, Rekursion, Selectionsort)

Iterative Realisation
Rekursion
Selectionsort

In dieser Aufgabe soll ein gegebenes Integer Array mit Hilfe von **Selection Sort** sortiert werden. Es soll eine iterative und eine rekursive Methode geschrieben werden. Verwenden Sie zur Implementierung jeweils die Methodenköpfe `selectionSortIterativ()` und `selectionSortRekursiv()`. Eine `swap`-Methode, die für ein gegebenes Array und zwei Indizes die Einträge an den jeweiligen Indizes des Arrays vertauscht, ist gegeben und muss nicht implementiert werden. Es müssen keine weiteren Methoden geschrieben werden!

Lösungsvorschlag

iterativ

```java
public static void selectionSortIterativ(int[] arr) {
  for (int i = 0; i < arr.length - 1; i++) {
    int min = i;
    for (int j = i + 1; j < arr.length; j++) {
      if (arr[j] < arr[min]) {
        min = j;
      }
    }
    swap(arr, i, min);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/sortier/SelectionSort.java`

rekursiv

```java
public static void selectionSortRekursiv(int[] arr, int i) {
  if (i == arr.length - 1) {
    return;
  }
  int min = i;
  for (int j = i + 1; j < arr.length; j++) {
    if (arr[j] < arr[min]) {
      min = j;
    }
  }
  swap(arr, i, min);
  selectionSortRekursiv(arr, i + 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/sortier/SelectionSort.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/40_Sortieralgorithmen/20_Selectionsort/Aufgabe_iterativ-rekursiv.tex`

Übungsaufgabe „Methode „fill()““ (Backtracking, Rekursion)

Backtracking
Rekursion

Folgende Methode soll das Feld a (garantiert der Länge $2n$ und beim ersten Aufruf von außen mit 0 initialisiert) mittels rekursivem Backtracking so mit Zahlen $1 \leq x \leq n$ befüllen, dass jedes x genau zweimal in a vorkommt und der Abstand zwischen den Vorkommen genau x ist. Sie soll genau dann `true` zurückgeben, wenn es eine Lösung gibt.

Beispiele:

- `fill(2, [])` → `false`
- `fill(3, [])` → `[3; 1; 2; 1; 3; 2]`
- `fill(4, [])` → `[4; 1; 3; 1; 2; 4; 3; 2]`

```
boolean fill (int n , int[] a) {
  if (n <= 0) {
    return true;
  }
  // TODO
  return false;
}
```

Lösungsvorschlag

```
  public static boolean fill(int n, int[] a) {
    if (n <= 0) {
      return true;
    }
    for (int i = 0; i < a.length - n - 1; i++) {
      // Zwischen i und j müssen genau n andere Zahlen sein
      int j = i + n + 1;
      if (a[i] == 0 && a[j] == 0) {
        a[i] = a[j] = n;
        if (fill(n - 1, a)) {
          return true;
        }
        a[i] = a[j] = 0;
      }
    }
    return false;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/backtracking/RekursivesBacktracking.java`

```
fill(0, []):
fill(1, []): false
fill(2, []): false
fill(3, []): 3 1 2 1 3 2
fill(4, []): 4 1 3 1 2 4 3 2
fill(5, []): false
fill(6, []): false
fill(7, []): 7 3 6 2 5 3 2 4 7 6 5 1 4 1
fill(8, []): 8 3 7 2 6 3 2 4 5 8 7 6 4 1 5 1
fill(9, []): false
```

```
fill(10, []): false
fill(11, []): 11 6 10 2 9 3 2 8 6 3 7 5 11 10 9 4 8 5 7 1 4 1
```

Kompletter Code

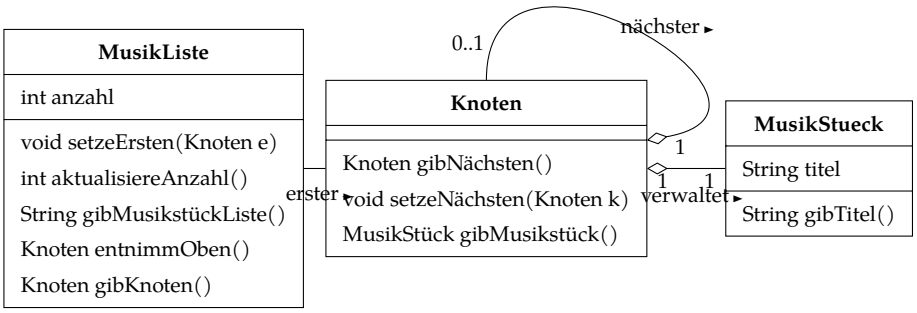
```
public class RekursivesBacktracking {

  public static boolean fill(int n, int[] a) {
    if (n <= 0) {
      return true;
    }
    for (int i = 0; i < a.length - n - 1; i++) {
      // Zwischen i und j müssen genau n andere Zahlen sein
      int j = i + n + 1;
      if (a[i] == 0 && a[j] == 0) {
        a[i] = a[j] = n;
        if (fill(n - 1, a)) {
          return true;
        }
        a[i] = a[j] = 0;
      }
    }
    return false;
  }

  public static void executeFill(int n) {
    int[] a = new int[n * 2];
    boolean result = fill(n, a);
    System.out.print("fill(" + n + ", []): ");
    if (result) {
      for (int i = 0; i < a.length; i++) {
        System.out.print(a[i] + " ");
      }
    } else {
      System.out.print("false");
    }

    System.out.println();
  }

  public static void main(String[] args) {
    executeFill(0);
    executeFill(1);
    executeFill(2);
    executeFill(3);
    executeFill(4);
    executeFill(5);
    executeFill(6);
    executeFill(7);
    executeFill(8);
    executeFill(9);
    executeFill(10);
    executeFill(11);
```

```
    }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/backtracking/RekursivesBacktracking.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/50_Backtracking/Aufgabe_Methode-fill.tex`

Übungsaufgabe „Playlist“ (Einfach-verkettete Liste, Implementierung in Java, Doppelt-verkettete Liste, Rekursion)



Das Klassendiagramm zeigt den Aufbau einer Playlist.

(a) Implementieren Sie das Klassendiagramm.

Lösungsvorschlag

Klasse „MusikListe“

```
public class MusikListe {
  private Knoten erster;
  private int anzahl;

  public MusikListe() {
    erster = null;
    anzahl = 0;
  }

  public void setzeErsten(Knoten knoten) {
    erster = knoten;
    aktualisiereAnzahl();
  }

  public int aktualisiereAnzahl() {
    if (erster == null) {
      anzahl = 0;
    } else {
      int zähler = 1;
      Knoten knoten = erster;
      while (!(knoten.gibNächsten() == null)) {
        knoten = knoten.gibNächsten();
        zähler = zähler + 1;
      }
      anzahl = zähler;
    }
    return anzahl;
  }

  public String gibMusikstückListe() {
```

```
    String ausgabe = " ";
    if (anzahl >= 1) {
      Knoten knoten = erster;
      ausgabe = knoten.gibMusikstück().gibTitel();
      for (int i = 1; i <= anzahl - 1; i++) {
        knoten = knoten.gibNächsten();
        ausgabe = ausgabe + " | " + knoten.gibMusikstück().gibTitel();
      }
    }
    return ausgabe;
  }

  public Knoten entnimmOben() {
    if (erster == null) {
      return erster;
    }
    Knoten alterKnoten = erster;
    erster = erster.gibNächsten();
    aktualisiereAnzahl();
    return alterKnoten;
  }

  public Knoten gibKnoten(int position) {
    if ((position < 1) || (position > anzahl)) {
      System.out.println(" FEHLER ! ");
      return null;
    }
    Knoten knoten = erster;
    for (int i = 1; i <= position - 1; i++) {
      knoten = knoten.gibNächsten();
    }
    return knoten;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java`

Klasse „Knoten“

```
public class Knoten {
  private Knoten nächster;
  private Knoten vorheriger;
  private MusikStueck lied;

  public Knoten(MusikStueck lied) {
    nächster = null;
    this.lied = lied;
    vorheriger = null;
  }

  public Knoten gibNächsten() {
    return nächster;
```

```
  }

  public void setzeNächsten(Knoten nächsterKnoten) {
    nächster = nächsterKnoten;
  }

  public MusikStueck gibMusikstück() {
    return lied;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java`

Klasse „MusikStueck“

```
public class MusikStueck {
  private String titel;

  public MusikStueck(String titel) {
    this.titel = titel;
  }

  public String gibTitel() {
    return titel;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikStueck.java`

(b) Schreiben Sie eine Testklasse, in der Sie eine Playlist mit mindestens vier Liedern erstellen.

Lösungsvorschlag

```
private MusikListe macheListe() {
  MusikListe liste = new MusikListe();

  MusikStueck stueck1 = new MusikStueck("Hangover");
  MusikStueck stueck2 = new MusikStueck("Roar");
  MusikStueck stueck3 = new MusikStueck("On the Floor");
  MusikStueck stueck4 = new MusikStueck("Whistle");

  Knoten platz1 = new Knoten(stueck1);
  Knoten platz2 = new Knoten(stueck2);
  Knoten platz3 = new Knoten(stueck3);
  Knoten platz4 = new Knoten(stueck4);

  liste.setzeErsten(platz1);
  platz1.setzeNächsten(platz2);
  platz2.setzenVorherigen(platz1);
  platz2.setzeNächsten(platz3);
  platz3.setzenVorherigen(platz2);
  platz3.setzeNächsten(platz4);
  platz4.setzenVorherigen(platz3);
  liste.aktualisiereAnzahl();
```

```
    return liste;
  }
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java`

Aufgabe 2

Die Playlist aus Aufgabe 1 soll nun erweitert werden. Aktualisieren Sie Ihren Code entsprechend!

(a) Bisher wurde das erste Element der Musikliste in einer öffentlich sichtbaren Variable gespeichert, dies ist jedoch nicht sinnvoll. Erstellen Sie eine Methode `setzeErsten()`, mit der anstatt dessen die Liste der erstellten Musikstücke angesprochen werden kann.

Lösungsvorschlag

```
public void setzeErsten(Knoten knoten) {
  erster = knoten;
  aktualisiereAnzahl();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java`

(b) Außerdem wird ein Attribut `anzahl` benötigt, dass mit Hilfe der Methode `aktualisiereAnzahl()` auf dem aktuellen Stand gehalten werden kann.

Lösungsvorschlag

```
private int anzahl;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java`

(c) Eine weitere Methode `gibMusikstückListe()` soll die Titel aller Lieder in der Liste als `String` zurückgeben.

Lösungsvorschlag

```
public String gibMusikstückListe() {
  String ausgabe = " ";
  if (anzahl >= 1) {
    Knoten knoten = erster;
    ausgabe = knoten.gibMusikstück().gibTitel();
    for (int i = 1; i <= anzahl - 1; i++) {
      knoten = knoten.gibNächsten();
      ausgabe = ausgabe + " | " + knoten.gibMusikstück().gibTitel();
    }
  }
  return ausgabe;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(d) Mit `entnimmOben()` soll der erste Titel aus der Liste entnommen werden können.

Lösungsvorschlag

```
public Knoten entnimmOben() {
  if (erster == null) {
    return erster;
  }
  Knoten alterKnoten = erster;
  erster = erster.gibNächsten();
  aktualisiereAnzahl();
  return alterKnoten;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(e) Es soll der Titel des Musikstücks ermittelt werden, das an einer bestimmten Position in der Musikliste abgespeichert ist. Implementieren Sie dazu die Methode `gibKnoten()`.

Lösungsvorschlag

```
public Knoten gibKnoten(int position) {
  if ((position < 1) || (position > anzahl)) {
    System.out.println(" FEHLER ! ");
    return null;
  }
  Knoten knoten = erster;
  for (int i = 1; i <= position - 1; i++) {
    knoten = knoten.gibNächsten();
  }
  return knoten;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

(f) Die Musikliste soll nun in eine doppelt verkettete Liste umgebaut werden. Fügen Sie entsprechende Attribute, getter- und setter-Methoden hinzu.

Lösungsvorschlag

```
private Knoten vorheriger;
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java

```
public MusikStueck gibMusikstück() {
  return lied;
}

public Knoten gibVorherigen() {
  return vorheriger;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java`

(g) Testen Sie die Funktionalität der neuen Methoden in Ihrer Testklasse.

Lösungsvorschlag

```
import static org.junit.Assert.*;
import org.junit.Test;

public class MusikListeTest {

  private MusikListe macheListe() {
    MusikListe liste = new MusikListe();

    MusikStueck stueck1 = new MusikStueck("Hangover");
    MusikStueck stueck2 = new MusikStueck("Roar");
    MusikStueck stueck3 = new MusikStueck("On the Floor");
    MusikStueck stueck4 = new MusikStueck("Whistle");

    Knoten platz1 = new Knoten(stueck1);
    Knoten platz2 = new Knoten(stueck2);
    Knoten platz3 = new Knoten(stueck3);
    Knoten platz4 = new Knoten(stueck4);

    liste.setzeErsten(platz1);
    platz1.setzeNächsten(platz2);
    platz2.setzenVorherigen(platz1);
    platz2.setzeNächsten(platz3);
    platz3.setzenVorherigen(platz2);
    platz3.setzeNächsten(platz4);
    platz4.setzenVorherigen(platz3);
    liste.aktualisiereAnzahl();
    return liste;
  }

  @Test
  public void methodeGibMusikstückListe() {
    MusikListe liste = macheListe();
    assertEquals("Hangover | Roar | On the Floor | Whistle",
↪   liste.gibMusikstückListe());
  }

  @Test
  public void methodeEntnimmOben() {
    MusikListe liste = macheListe();
    assertEquals("Hangover", liste.entnimmOben().gibMusikstück().gibTitel());
    assertEquals("Roar | On the Floor | Whistle",
↪   liste.gibMusikstückListe());
  }

  @Test
  public void methodeZähleEinträge() {
    MusikListe liste = macheListe();
    assertEquals(4, liste.zähleEinträge());
  }
}
```

Code-Beispiel auf Github ansehen: src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java

Rekursion

Die Anzahl der Titel in der Musikliste aus Aufgabe 1 kann auch unter Verwendung einer rekursiven Methode ermittelt werden. Implementieren Sie eine Methode `zaehleEintraege()`, die analog zu `aktualisiereAnzahl()` angibt, wie viele Titel in der Musikliste gespeichert sind, dies aber rekursiv ermittelt! Testen Sie diese Methode in der Testklasse. Hinweis: Um für die gesamte Musikliste aufgerufen werden zu können, muss diese Methode in der Musikliste selbst und auch in der Klasse Knoten existieren!

Lösungsvorschlag

Klasse „MusikListe“

```
  public int zähleEinträge() {
    if (erster == null) {
      return 0;
    }
    return erster.zähleEinträge();
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListe.java

Klasse „Knoten“

```
  public int zähleEinträge() {
    if (this.gibNächsten() == null) {
      return 1;
    } else {
      return this.gibNächsten().zähleEinträge() + 1;
    }
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/aud/listen/musikliste/Knoten.java

Klasse „TestKlasse“

```
  @Test
  public void methodeZähleEinträge() {
    MusikListe liste = macheListe();
    assertEquals(4, liste.zähleEinträge());
  }
}
```

Code-Beispiel auf Github ansehen: src/test/java/org/bschlangaul/aufgaben/aud/listen/musikliste/MusikListeTest.java

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/10_Listen/Aufgabe_Playlist.tex

Suche

Examensaufgabe „Unimodale Zahlenfolge“ (46115-2015-H.T1-A3)

Aufgabe 3

Eine Folge von Zahlen $a_1, \ldots, a_n$ heiße unimodal, wenn sie bis zu einem bestimmten Punkt echt ansteigt und dann echt fällt. Zum Beispiel ist die Folge $1, 3, 5, 6, 5, 2, 1$ unimodal, die Folgen $1, 3, 5, 4, 7, 2, 1$ und $1, 2, 3, 3, 4, 3, 2, 1$ aber nicht.

> **Exkurs: Unimodale Abbildung**
> Eine unimodale Abbildung oder unimodale Funktion ist in der Mathematik eine Funktion mit einem eindeutigen (lokalen und globalen) Maximum wie zum Beispiel $f(x) = -x^2$. [a]
>
> [a] `https://de.wikipedia.org/wiki/Unimodale_Abbildung`

(a) Entwerfen Sie einen Algorithmus, der zu (als Array) gegebener unimodaler Folge $a_1, \ldots, a_n$ in Zeit $\mathcal{O}(\log n)$ das Maximum $\max a_i$ berechnet. Ist die Folge nicht unimodal, so kann Ihr Algorithmus ein beliebiges Ergebnis liefern. Größenvergleiche, arithmetische Operationen und Arrayzugriffe können wie üblich in konstanter Zeit ($\mathcal{O}(1)$) getätigt werden. Hinweise: binäre Suche, divide-and-conquer.

Lösungsvorschlag

Wir wählen einen Wert in der Mitte der Folge aus. Ist der direkte linke und der direkte rechte Nachbar dieses Wertes kleiner, dann ist das Maximum gefunden. Ist nur linke Nachbar größer, setzen wir die Suche wie oben beschrieben in der linken Hälfte, sonst in der rechten Hälfte fort.

(b) Begründen Sie, dass Ihr Algorithmus tatsächlich in Zeit $\mathcal{O}(\log n)$ läuft.

Lösungsvorschlag

Da der beschriebene Algorithmus nach jedem Bearbeitungsschritt nur auf der Hälfte der Feld-Element zu arbeiten hat, muss im schlechtesten Fall nicht die gesamte Folge durchsucht werden. Nach dem ersten Teilen der Folge bleiben nur noch $\frac{n}{2}$ Elemente, nach dem zweiten Schritt $\frac{n}{4}$, nach dem dritten $\frac{n}{8}$ usw. Allgemein bedeutet dies, dass im i-ten Durchlauf maximal $\frac{n}{2^i}$ Elemente zu durchsuchen sind. Entsprechend werden $\log_2 n$ Schritte benötigt. Somit hat der Algorithmus zum Finden des Maximums in einer unimodalen Folge in der Landau-Notation ausgedrückt die Zeitkomplexität $\mathcal{O}(\log n)$.

(c) Schreiben Sie Ihren Algorithmus in Pseudocode oder in einer Programmiersprache Ihrer Wahl, z. B. Java, auf. Sie dürfen voraussetzen, dass die Eingabe in Form eines Arrays der Größe n vorliegt.

Lösungsvorschlag

Rekursiver Ansatz

```
public static int findeMaxRekursiv(int feld[], int links, int rechts) {
  if (links == rechts - 1) {
    return feld[links];
  }
  //    bedeutet aufrunden
  // https://stackoverflow.com/a/17149572
  int mitte = (int) Math.ceil((double) (links + rechts) / 2);
  if (feld[mitte - 1] < feld[mitte]) {
    return findeMaxRekursiv(feld, mitte, rechts);
  } else {
    return findeMaxRekursiv(feld, links, mitte);
  }
}

public static int findeMaxRekursiv(int feld[]) {
  return findeMaxRekursiv(feld, 0, feld.length - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java`

Iterativer Ansatz

```
public static int findeMaxIterativ(int[] feld) {
  int links = 0;
  int rechts = feld.length - 1;
  int mitte;

  while (links < rechts) {
    mitte = links + (rechts - links) / 2;
    if (feld[mitte] > feld[mitte - 1] && feld[mitte] > feld[mitte + 1]) {
      return feld[mitte];
    } else if (feld[mitte] > feld[mitte - 1]) {
      links = mitte + 1;
    } else {
      rechts = mitte - 1;
    }
  }
  return KEIN_MAX;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java`

(d) Beschreiben Sie in Worten ein Verfahren, welches in Zeit $\mathcal{O}(n)$ feststellt, ob eine vorgelegte Folge unimodal ist oder nicht.

Lösungsvorschlag

```
public static boolean testeUnimodalität(int[] feld) {
  if (feld.length < 2) {
```

```
      // Die Reihe braucht mindestens 3 Einträge
      return false;
    }

    if (feld[0] > feld[1]) {
      // Die Reihe muss zuerst ansteigen
      return false;
    }

    boolean maxErreicht = false;
    for (int i = 0; i < feld.length - 1; i++) {
      if (feld[i] > feld[i + 1] && !maxErreicht) {
        maxErreicht = true;
      }

      if (maxErreicht && feld[i] < feld[i + 1]) {
        // Das Maximum wurde bereichts erreicht und die nächste Zahl ist
↪ größer
        return false;
      }
    }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java`

(e) Begründen Sie, dass es kein solches Verfahren (Test auf Unimodalität) geben kann, welches in Zeit $\mathcal{O}(\log n)$ läuft.

Lösungsvorschlag

Da die Unimodalität nur durch einen Werte an einer beliebigen Stelle der Folge verletzt werden kann, müssen alle Elemente durchsucht und überprüft werden.

Lösungsvorschlag

Komplette Klasse

```
public static int findeMaxRekursiv(int feld[], int links, int rechts) {
  if (links == rechts - 1) {
    return feld[links];
  }
  //    bedeutet aufrunden
  // https://stackoverflow.com/a/17149572
  int mitte = (int) Math.ceil((double) (links + rechts) / 2);
  if (feld[mitte - 1] < feld[mitte]) {
    return findeMaxRekursiv(feld, mitte, rechts);
  } else {
    return findeMaxRekursiv(feld, links, mitte);
  }
}

public static int findeMaxRekursiv(int feld[]) {
  return findeMaxRekursiv(feld, 0, feld.length - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java`

Test

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class UnimodalFinderTest {

  private void testeMaxItertiv(int[] feld, int max) {
    assertEquals(max, UnimodalFinder.findeMaxIterativ(feld));
  }

  private void testeMaxRekursiv(int[] feld, int max) {
    assertEquals(max, UnimodalFinder.findeMaxRekursiv(feld));
  }

  private void testeMax(int[] feld, int max) {
    testeMaxItertiv(feld, max);
    testeMaxRekursiv(feld, max);
  }

  @Test
  public void findeMax() {
    testeMax(new int[] { 1, 2, 3, 1 }, 3);
  }

  @Test
  public void findeMaxLaengeresFeld() {
    testeMax(new int[] { 1, 3, 4, 6, 7, 8, 9, 11, 6, 5, 4, 3, 2 }, 11);
  }

  @Test
  public void keinMaxAufsteigend() {
    testeMaxItertiv(new int[] { 1, 2, 3 }, UnimodalFinder.KEIN_MAX);
  }

  @Test
  public void keinMaxAbsteigend() {
    testeMaxItertiv(new int[] { 3, 2, 1 }, UnimodalFinder.KEIN_MAX);
  }

  @Test
  public void maxNegativeZahlen() {
    testeMax(new int[] { -2, -1, 3, 1 }, 3);
  }

  private void testeUnimodalität(int[] feld, boolean wahr) {
    assertEquals(wahr, UnimodalFinder.testeUnimodalität(feld));
  }

  @Test
```

```
  public void unimodalität() {
    testeUnimodalität(new int[] { 1, 2, 3, 1 }, true);
  }

  @Test
  public void unimodalitätFalsch() {
    testeUnimodalität(new int[] { 1, -2, 3, 1, 2 }, false);
    testeUnimodalität(new int[] { 1, 2, 3, 1, 2 }, false);
    testeUnimodalität(new int[] { 3, 2, 1 }, false);
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinderTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2015/09/Thema-1/Aufgabe-3.tex`

Examensaufgabe „Bruchsicherheit von Smartphones“ (46115-2016-F.T2-A3)

Algorithmische Komplexität (O-Notation) Lineare Suche

Sie sollen mithilfe von Falltests eine neue Serie von Smartphones auf Bruchsicherheit testen.

Dazu wird eine Leiter mit n Sprossen verwendet; die höchste Sprosse, von der ein Smartphone heruntergeworfen werden kann ohne zu zerbrechen, heiße *„höchste sichere Sprosse“*. Das Ziel ist, die höchste sichere Sprosse zu ermitteln. Man kann davon ausgehen, dass die höchste sichere Sprosse nicht von der Art des Wurfs abhängt und dass alle verwendeten Smartphones sich gleich verhalten. Eine Möglichkeit, die höchste sichere Sprosse zu ermitteln, besteht darin, ein Gerät erst von Sprosse 1, dann von Sprosse 2, etc. abzuwerfen, bis es schließlich beim Wurf von Sprosse k beschädigt wird (oder Sie oben angelangt sind). Sprosse $k-1$ (bzw. n) ist dann die höchste sichere Sprosse. Bei diesem Verfahren wird maximal ein Smartphone zerstört, aber der Zeitaufwand ist ungünstig.

(a) Bestimmen Sie die Zahl der Würfe bei diesem Verfahren im schlechtesten Fall.

Lösungsvorschlag

> Die Zahl der Würfe im schlechtesten Fall ist $\mathcal{O}(k)$, wobei k die Anzahl der Sprossen ist. Geht das Smartphone erst bei der höchsten Sprosse kaputt, muss es k mal heruntergeworfen werden. Die Komplexität entspricht der der linearen Suche.

(b) Geben Sie nun ein Verfahren zur Ermittlung der höchsten sicheren Sprosse an, welches nur $\mathcal{O}(\log n)$ Würfe benötigt, dafür aber möglicherweise mehr Smartphones verbraucht.

Lösungsvorschlag

> Man startet bei Sprosse $\frac{n}{2}$. Wenn das Smartphone kaputt geht, macht man weiter mit der Sprosse in der Mitte der unteren Hälfte, ansonsten mit der Sprosse in der Mitte der oberen Hälfte. Das Ganze rekursiv.

(c) Es gibt eine Strategie zur Ermittlung der höchsten sicheren Sprosse mit $\mathcal{O}\left(\sqrt{n}\right)$ Würfen, bei dessen Anwendung höchstens zwei Smartphones kaputtgehen. Finden Sie diese Strategie und begründen Sie Ihre Funktionsweise und Wurfzahl.

Tipp: der erste Testwurf erfolgt von Sprosse $\left\lceil \sqrt{n} \right\rceil$.

> **Exkurs: Interpolationssuche**
>
> Die Interpolationssuche, auch Intervallsuche genannt, ist ein von der binären Suche abgeleitetes Suchverfahren, das auf Listen und Feldern zum Einsatz kommt.
>
> Während der Algorithmus der binären Suche stets das mittlere Element des Suchraums überprüft, versucht der Algorithmus der Interpolationssuche im Suchraum einen günstigeren Teilungspunkt als die Mitte zu erraten. Die Arbeitsweise ist mit der eines Menschen vergleichbar, der ein Wort in einem Wörterbuch sucht: Die Suche nach Zylinder wird üblicherweise am Ende des Wörterbuches begonnen, während die Suche nach Aal im vorderen

Bereich begonnen werden dürfte. [a]

[a] https://de.wikipedia.org/wiki/Quadratische_Binärsuche

Exkurs: Quadratische Binärsuche

Quadratische Binärsuche ist ein Suchalgorithmus ähnlich der Binärsuche oder Interpolationssuche. Es versucht durch Reduzierung des Intervalls in jedem Rekursionsschritt die Nachteile der Interpolationssuche zu vermeiden.

Nach dem Muster der Interpolationssuche wird zunächst in jedem rekursiven Schritt die vermutete Position k interpoliert. Anschließend wird – um die Nachteile der Interpolationssuche zu vermeiden – das Intervall der Länge $\sqrt{n}$ gesucht, in dem sich der gesuchte Wert befindet. Auf dieses Intervall wird der nächste rekursive Aufruf der Suche angewendet.

Auf diese Weise verkleinert sich der Suchraum bei gegebener Liste der Länge n bei jedem rekursiven Schritt auf eine Liste der Länge $n\ \sqrt{n}$. [a]

[a] https://de.wikipedia.org/wiki/Quadratische_Binärsuche

Lösungsvorschlag

Das Vorgehen ist folgendermaßen: Man beginnt auf Stufe 0 und falls das Handy nicht kaputt geht, addiert man jeweils Wurzel n. Falls das Handy kaputt geht, geht man linear in Einerschritten das Intervall von der unteren Grenze (ðvon der Stufe vor der letzten Addition) bis zur Kaputtstufe ab. [a]

[a] http://www.inf.fu-berlin.de/lehre/WS06/HA/skript/vorlesung6.pdf

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/03/Thema-2/Aufgabe-3.tex

Examensaufgabe „Minimum und Maximum“ (46115-2021-F.T1-TA2-A2)

(a) Argumentieren Sie, warum man das Maximum von n Zahlen nicht mit weniger als $n - 1$ Vergleichen bestimmen kann.

Lösungsvorschlag

Wenn die n Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu $n - 1$ und nicht n Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen $n - 1$ mit dem aktuellen Maximum vergleichen.

(b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge n mit genau $n - 1$ Vergleichen bestimmt.

Lösungsvorschlag

```java
public static int bestimmeMaximum(int[] a) {
  int max = a[0];
  for (int i = 1; i < a.length; i++) {
    if (a[i] > max) {
      max = a[i];
    }
  }
  return max;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java`

(c) Wenn man das Minimum und das Maximum von n Zahlen bestimmen will, dann kann das natürlich mit $2n - 2$ Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

Lösungsvorschlag

```java
/**
 * Diese Methode ist nicht optimiert. Es werden 2n - 2 Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
↪  gesucht
 *          werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
  int max = a[0];
  int min = a[0];
  for (int i = 1; i < a.length; i++) {
```

```
      if (a[i] > max) {
        max = a[i];
      }
      if (a[i] < min) {
        max = a[i];
      }
    }
    return new int[] { min, max };
  }

  /**
   * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
   * betrachtet. Die Anzahl der Vergleiche reduziert sich auf 3n/2 + 2 bzw.
   * 3(n-1)/2 + 4 bei einer ungeraden Anzahl an Zahlen im Feld.
   *
   * nach <a href=
   * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-
↪  minimum-comparisons/">techiedelight.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
↪  gesucht
   *          werden soll.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
   *         der zweite Eintrag das Maximum.
   */
  public static int[] minMaxIterativPaarweise(int[] a) {
    int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
    int n = a.length;

    boolean istUngerade = (n & 1) == 1;
    if (istUngerade) {
      n--;
    }

    for (int i = 0; i < n; i = i + 2) {
      int maximum, minimum;

      if (a[i] > a[i + 1]) {
        minimum = a[i + 1];
        maximum = a[i];
      } else {
        minimum = a[i];
        maximum = a[i + 1];
      }

      if (maximum > max) {
        max = maximum;
      }

      if (minimum < min) {
        min = minimum;
      }
```

```
    }

    if (istUngerade) {
      if (a[n] > max) {
        max = a[n];
      }

      if (a[n] < min) {
        min = a[n];
      }
    }
    return new int[] { min, max };
  }

  /**
   * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
   * funktioniert so ähnlich wie der Mergesort.
   *
   * nach <a href=
   * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-
↪  value-in-an-array">enjoyalgorithms.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
   *          gesucht werden soll.
   * @param l Die linke Grenze.
   * @param r Die rechts Grenze.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
   *         der zweite Eintrag das Maximum.
   */
  int[] minMaxRekursiv(int[] a, int l, int r) {
    int max, min;
    if (l == r) {
      max = a[l];
      min = a[l];
    } else if (l + 1 == r) {
      if (a[l] < a[r]) {
        max = a[r];
        min = a[l];
      } else {
        max = a[l];
        min = a[r];
      }
    } else {
      int mid = l + (r - l) / 2;
      int[] lErgebnis = minMaxRekursiv(a, l, mid);
      int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
      if (lErgebnis[0] > rErgebnis[0]) {
        max = lErgebnis[0];
      } else {
        max = rErgebnis[0];
      }
      if (lErgebnis[1] < rErgebnis[1]) {
```

```
        min = lErgebnis[1];
      } else {
        min = rErgebnis[1];
      }
    }
    int[] ergebnis = { max, min };
    return ergebnis;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Lineare und Binäre Suchverfahren“ (46115-2021-F.T2-TA2-A3)

Binäre Suche

Gegeben ist ein aufsteigend sortiertes Array A von n ganzen Zahlen und eine ganze Zahl x. Es wird der Algorithmus `BinarySearch` betrachtet, der A effizient nach dem Wert x absucht. Ergebnis ist der Index i mit $x = A[i]$ oder NIL, falls $x \notin A$.

Funktion BinarySearch(int A, int r)

```
l = 1;
r = A.length;
while r ≥ l do
    if x < A[m] then
        r = m - 1;
    else if x=A[m] then
        return m;
    else
        l = m + 1;
    end
    return NIL ;
end
```

(a) Durchsuchen Sie das folgende Feld jeweils nach den in (i) bis (iii) angegebenen Werten mittels binärer Suche. Geben Sie für jede Iteration die Werte /,r,m und den betretenen if-Zweig an. Geben Sie zudem den Ergebnis-Index bzw. NIL an.

Index

i]s] «| «| 2] 4] off

wen [ilsfol7] io] w]u]al ale!

(i) 10

(ii) 13

(iii) 22

(b) Betrachten Sie auf das Array aus Teilaufgabe a). Für welche Werte durchläuft der Algorith- mus nie den letzten else-Teil in Zeile 11? Hinweis: Unterscheiden Sie auch zwischen enthaltenen und nicht-enthaltenen Werten.

(c) Wie ändert sich das Ergebnis der binären Suche, wenn im sortierten Eingabefeld zwei aufeinanderfolgende, unterschiedliche Werte vertauscht wurden? Betrachten Sie hierbei die betroffenen Werte, die anderen Feldelemente und nicht enthaltene Werte in Abhängigkeit vom Ort der Vertauschung.

(d) Angenommen, das Eingabearray A für den Algorithmus für die binäre Suche enthält nur die Zahlen 0 und 1, aufsteigend sortiert. Zudem ist jede der beiden Zahlen

mindestens ein Mal vorhanden. Ändern Sie den Algorithmus für die binäre Suche so ab, dass er den bzw. einen Index k zurückgibt, für den gilt: Alk] =1 und Ak—1]=0.

(e) Betrachten Sie die folgende rekursive Variante von BinarySearch.

1 int RekBinarySearch(int[] A. int x. int £. int r)

| mi

3 | (rekursive Implementierung)

Der initiale Aufruf der rekursiven Variante lautet: RekBinarySearch (A, z, 1, A.length)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex

Examensaufgabe „Schnelle Suche von Schlüsseln: odd-ascending-even-descending-Folge“ (66115-2020-H.T2-TA2-A5)

Binäre Suche

Eine Folge von Zahlen ist eine *odd-ascending-even-descending*-Folge, wenn gilt:

> Zunächst enthält die Folge alle Schlüssel, die *ungerade* Zahlen sind, und diese Schlüssel sind aufsteigend sortiert angeordnet. Im Anschluss daran enthält die Folge alle Schlüssel, die *gerade* Zahlen sind, und diese Schlüssel sind absteigend sortiert angeordnet.

(a) Geben Sie die Zahlen 10, 3, 11, 20, 8, 4, 9 als *odd-ascending-even-descending*-Folge an.

Lösungsvorschlag

3, 9, 11, 20, 10, 8, 4

(b) Geben Sie einen Algorithmus (z. B. in Pseudocode oder Java) an, der für eine *odd-ascending-even-descending*-Folge F gegeben als Feld und einem Schlüsselwert S prüft, ob S in F vorkommt und `true` im Erfolgsfall und ansonsten `false` liefert. Dabei soll der Algorithmus im Worst-Case eine echt bessere Laufzeit als Linearzeit (in der Größe der Arrays) haben. Erläutern Sie Ihren Algorithmus und begründen Sie die Korrektheit.

Lösungsvorschlag

Bei dem Algorithmus handelt es sich um einen leicht abgewandelten Code, der eine „klassische“ binären Suche implementiert.

```
public static boolean suche(int[] feld, int schlüssel) {
  int links = 0, rechts = feld.length - 1;
  boolean istGerade = schlüssel % 2 == 0;
  while (links <= rechts) {
    int mitte = links + (rechts - links) / 2;
    if (feld[mitte] == schlüssel) {
      return true;
    }
    // Verschiebe die linke Grenze nach rechts, wenn die gesuchte
    // Zahl gerade ist und die Zahl in der Mitte größer als die
    // gesuchte Zahl ist oder wenn die gesuchte Zahl ungerade ist
    // und die Zahl in der Mitte kleiner.
    if ((istGerade && feld[mitte] > schlüssel) || (!istGerade &&
↪ feld[mitte] < schlüssel)) {
      // nach rechts verschieben
      links = mitte + 1;
    } else {
      // nach links verschieben
      rechts = mitte - 1;
    }
  }
  return false;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGerade.java`

(c) Erläutern Sie schrittweise den Ablauf Ihres Algorithmus für die Folge 1, 5, 11, 8, 4, 2 und den Suchschlüssel 4.

Lösungsvorschlag

Die erste Zeile der Methode `suche` initialisiert die Variable `links` mit 0 und `rechts` mit 5. Da `links` kleiner ist als `rechts`, wird die `while`-Schleife betreten und die Variable `mitte` auf 2 gesetzt. Da der gesuchte Schlüssel gerade ist und `feld[2]` 11 ist, also größer, wird in den `true`-Block der `if`-Bedingung besprungen und die Variable `links` aus 3 gesetzt.

Zu Beginn des 2. Durchlaufs der `while`-Schleife ergeben sich folgende Werte: `links`: 3 `mitte`: 4 `rechts`: 5.

In der anschließenden Bedingten Anweisung wird die `while`-Schleife verlassen und `true` zurückgegeben, da mit `feld[4]` der gewünschte Schlüssel gefunden wurde.

(d) Analysieren Sie die Laufzeit Ihres Algorithmus für den Worst-Case, geben Sie diese in $\mathcal{O}$-Notation an und begründen Sie diese.

Lösungsvorschlag

Die Laufzeit des Algorithmuses ist in der Ordnung $\mathcal{O}(\log_2 n)$.

Im schlechtesten Fall muss nicht die gesamte Folge durchsucht werden. Nach dem ersten Teilen der Folge bleiben nur noch $\frac{n}{2}$ Elemente, nach dem zweiten Schritt $\frac{n}{4}$, nach dem dritten $\frac{n}{8}$ usw. Allgemein bedeutet dies, dass im i-ten Durchlauf maximal $\frac{n}{2^i}$ Elemente zu durchsuchen sind. Entsprechend werden $\log_2 n$ Schritte benötigt.

Kompletter Code

```
public class UngeradeGerade {

  public static boolean suche(int[] feld, int schlüssel) {
    int links = 0, rechts = feld.length - 1;
    boolean istGerade = schlüssel % 2 == 0;
    while (links <= rechts) {
      int mitte = links + (rechts - links) / 2;
      if (feld[mitte] == schlüssel) {
        return true;
      }
      // Verschiebe die linke Grenze nach rechts, wenn die gesuchte
      // Zahl gerade ist und die Zahl in der Mitte größer als die
      // gesuchte Zahl ist oder wenn die gesuchte Zahl ungerade ist
      // und die Zahl in der Mitte kleiner.
      if ((istGerade && feld[mitte] > schlüssel) || (!istGerade && feld[mitte] <
↪  schlüssel)) {
        // nach rechts verschieben
        links = mitte + 1;
      } else {
        // nach links verschieben
```

```
        rechts = mitte - 1;
      }
    }
    return false;
  }

  public static void main(String[] args) {
    System.out.println(suche(new int[] { 1, 5, 11, 8, 4, 2 }, 4));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGerade.java`

Test-Code

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class UngeradeGeradeTest {

  private void assertSucheUnGerade(int[] feld, int suche, boolean ergebnis) {
    assertEquals(ergebnis, UngeradeGerade.suche(feld, suche));
  }

  @Test
  public void assertSucheUnGerade() {
    int[] feld = new int[] { 1, 3, 5, 7, 9, 10, 8, 6, 4, 2 };
    assertSucheUnGerade(feld, 4, true);
    assertSucheUnGerade(feld, 11, false);
    assertSucheUnGerade(feld, 0, false);
    assertSucheUnGerade(feld, 3, true);
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGeradeTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-5.tex`

Examensaufgabe „Minimum und Maximum“ (66115-2021-F.T1-TA2-A2)

(a) Argumentieren Sie, warum man das Maximum von n Zahlen nicht mit weniger als $n - 1$ Vergleichen bestimmen kann.

Lösungsvorschlag

Wenn die n Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu $n - 1$ und nicht n Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen $n - 1$ mit dem aktuellen Maximum vergleichen.

(b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge n mit genau $n - 1$ Vergleichen bestimmt.

Lösungsvorschlag

```java
public static int bestimmeMaximum(int[] a) {
  int max = a[0];
  for (int i = 1; i < a.length; i++) {
    if (a[i] > max) {
      max = a[i];
    }
  }
  return max;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java`

(c) Wenn man das Minimum und das Maximum von n Zahlen bestimmen will, dann kann das natürlich mit $2n - 2$ Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

Lösungsvorschlag

```java
/**
 * Diese Methode ist nicht optimiert. Es werden 2n - 2 Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
↪  gesucht
 *          werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
  int max = a[0];
  int min = a[0];
  for (int i = 1; i < a.length; i++) {
```

```
      if (a[i] > max) {
        max = a[i];
      }
      if (a[i] < min) {
        max = a[i];
      }
    }
    return new int[] { min, max };
  }

  /**
   * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
   * betrachtet. Die Anzahl der Vergleiche reduziert sich auf 3n/2 + 2 bzw.
   * 3(n-1)/2 + 4 bei einer ungeraden Anzahl an Zahlen im Feld.
   *
   * nach <a href=
   * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-
↪  minimum-comparisons/">techiedelight.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
↪  gesucht
   *          werden soll.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
   *         der zweite Eintrag das Maximum.
   */
  public static int[] minMaxIterativPaarweise(int[] a) {
    int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
    int n = a.length;

    boolean istUngerade = (n & 1) == 1;
    if (istUngerade) {
      n--;
    }

    for (int i = 0; i < n; i = i + 2) {
      int maximum, minimum;

      if (a[i] > a[i + 1]) {
        minimum = a[i + 1];
        maximum = a[i];
      } else {
        minimum = a[i];
        maximum = a[i + 1];
      }

      if (maximum > max) {
        max = maximum;
      }

      if (minimum < min) {
        min = minimum;
      }
```

```
    }

    if (istUngerade) {
      if (a[n] > max) {
        max = a[n];
      }

      if (a[n] < min) {
        min = a[n];
      }
    }
    return new int[] { min, max };
  }

  /**
   * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
   * funktioniert so ähnlich wie der Mergesort.
   *
   * nach <a href=
   * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-
↪  value-in-an-array">enjoyalgorithms.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
   *          gesucht werden soll.
   * @param l Die linke Grenze.
   * @param r Die rechts Grenze.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das
↪  Minimum,
   *         der zweite Eintrag das Maximum.
   */
  int[] minMaxRekursiv(int[] a, int l, int r) {
    int max, min;
    if (l == r) {
      max = a[l];
      min = a[l];
    } else if (l + 1 == r) {
      if (a[l] < a[r]) {
        max = a[r];
        min = a[l];
      } else {
        max = a[l];
        min = a[r];
      }
    } else {
      int mid = l + (r - l) / 2;
      int[] lErgebnis = minMaxRekursiv(a, l, mid);
      int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
      if (lErgebnis[0] > rErgebnis[0]) {
        max = lErgebnis[0];
      } else {
        max = rErgebnis[0];
      }
      if (lErgebnis[1] < rErgebnis[1]) {
```

```
        min = lErgebnis[1];
      } else {
        min = rErgebnis[1];
      }
    }
    int[] ergebnis = { max, min };
    return ergebnis;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Code-Inspection bei Binärer Suche“ (66116-2017-H.T1-TA2-A4)

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

RM1	(Dokumentation)	Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält.
RM2	(Dokumentation)	Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur so wie eine Design-by-Contract-Spezifikation.
RM3	(Dokumentation)	Deklarationen von Variablen werden kommentiert.
RM4	(Dokumentation)	Jede Kontrollstruktur wird kommentiert.
RM5	(Formatierung)	Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen.
RM6	(Formatierung)	Zwischen binären Operatoren und den Operanden stehen Leerzeichen.
RM7	(Programmierung)	Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden.
RM8	(Bezeichner)	Klassennamen werden groß geschrieben, Variablennamen klein.

```
/**
 * BinarySearch.java
 *
 * Eine Implementierung der "Binaere Suche"
 * mit einem iterativen Algorithmus
 */
class BinarySearch {

  /**
   * BinaereSuche
   * a: Eingabefeld
   * item: zusuchendesElement
   * returnValue: der Index des zu suchenden Elements oder -1
   *
   * Vorbedingung:
   * a.length > 0
   * a ist ein linear geordnetes Feld:
   * For all k: (1 <= k < a.length) ==> (a[k-1] <=a [k])
   *
```

```
   * Nachbedingung:
   * Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
   * Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
   * und a[k]==item.
   */
  public static int binarySearch(float a[], float item) {

    int End;  // exklusiver Index fuer das Ende des
              // zudurchsuchenden Teils des Arrays
    int start = 1; // inklusiver Index fuer den Anfang der Suche
    End = a.length;

    // Die Schleife wird verlassen, wenn keine der beiden Haelften das
    // Element enthaelt.
    while(start < End) {

      // Teilung des Arrays in zwei Haelften
      // untere Haelfte: [0,mid[
      // obere Haelfte: ]mid,End[
      int mid = (start + End) / 2;

      if (item > a[mid]) {
        // Ausschluss der oberen Haelfte
        start = mid + 1;
      } else if(item < a[mid]) {
        // Ausschluss der unteren Haelfte
        End = mid-1;
      } else {
        // Das gesuchte Element wird zurueckgegeben
        return (mid);
      }
    } // end of while

    // Bei Misserfolg der Suche wird -1 zurueckgegeben
    return (-1);
  }
}
```

(a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z. B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

Lösungsvorschlag

Zeile	Regel	Kommentar
3-8	RM1	Fehlen von Versionsinformationen, Datum und Urheberrechtsangaben
11-26	RM2	Fehlen der Invariante in der Design-by-Contract-Spezifikation
36,46	RM5	Fehlen des Leerzeichens vor der Klammer
48	RM6	Um einen binären (zweistellige) Operator handelt es sich im Code-Beispiel um den Subtraktionsoperator: `mid-1`. Hier fehlen die geforderten Leerzeichen.
32	RM7	Die Variable `End` wird in Zeile 32 deklariert, aber erst in Zeile initialisiert `End = a.length;`
32	RM8	Die Variable `End` muss klein geschrieben werden.

(b) Entspricht die Methode `binarySearch` ihrer Spezifikation, die durch Vor-und Nachbedingungen angeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Lösungsvorschlag

Korrektur der Vorbedingung

Die Vorbedingung ist nicht erfüllt, da weder die Länge des Feldes `a` noch die Reihenfolge der Feldeinträge geprüft wurden.

```
if (a.length <= 0) {
  return -1;
}

for (int i = 0; i < a.length; i ++) {
  if ( a[i] > a[i + 1]) {
    return -1;
  }
}
```

Korrektur der Nachbedingung

`int start` muss mit `0` initialisiert werden, da sonst `a[0]` vernachlässigt wird.

(c) Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

Lösungsvorschlag

Kontrollflussgraph
Vollständige Anweisungsüberdeckung

Zeile	Kommentar im Code	Korrektur
34-35	`// Die Schleife wird v⌋ erlassen, wenn keine der beiden Haelften das Elemen⌋ t enthaelt.`	`// Die Schleife wird v⌋ erlassen, wenn keine der beiden Haelften das El⌋ ement enthaelt oder das Element gefunden wurde.`
44	`// Ausschluss der oberen Haelfte`	`// Ausschluss der unteren Haelfte`
47	`// Ausschluss der unteren Haelfte`	`// Ausschluss der oberen Haelfte`
50	`// Das gesuchte Element wird zurueckgegeben`	`// Der Index des gesuchten Elements wird zurueckgegeb⌋ en`

(d) Geben Sie den Kontrollflussgraphen für die Methode `binarySearch` an.

(e) Geben Sie maximal drei Testfälle für die Methode `binarySearch` an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Lösungsvorschlag

Die gegebene Methode: `binarySearch(a[], item)`

Testfall

(i) Testfall: `a[] = {1, 2, 3}, item = 4`

(ii) Testfall: `a[] = {1, 2, 3}, item = 2`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/09/Thema-1/Teilaufgabe-2/Aufgabe-4.tex

Übungsaufgabe „Methode „sucheBinaer()““ (Binäre Suche)

Für diese Aufgabe wird die Vorlage Suchalgorithmen benötigt, die auf dem Beiblatt genauer erklärt wird.

Vervollständigen Sie die Methode `sucheBinaer()`. Die fertige Methode soll in der Lage sein, beliebige Werte in beliebigen sortierten Arrays zu suchen. Im gelben Textfeld des Eingabefensters soll dabei ausführlich und nachvollziehbar angezeigt werden, wie die Methode vorgeht. Beispielsweise so:

```
Führe die Methode sucheSequenziell() aus:
Suche in diesem Feld: 3, 5, 7, 17, 42, 23
Überprüfen des Werts an der Position 0
Überprüfen des Werts an der Position 1
Überprüfen des Werts an der Position 2
Überprüfen des Werts an der Position 3
Fertig :-)
```

```
Führe die Methode sucheBinaer() aus:
Suche in diesem Feld: 3, 5, 7, 17, 42, 23
Suchbereich: 0 bis 5
Mitte: 2, also neuer Bereich: 3 bis 5
Mitte: 4, also neuer Bereich: 3 bis 3
Mitte: 3, Treffer : -)
```

(a) Sequenzielle Suche

Lösungsvorschlag

```
}

/**
 * Sequenzielle Suche: Durchsucht das Array nach dem Wert und gibt dessen
 * Position als Ergebnis zurück.
 *
 * @param array Ein Feld mit Zahlen.
 * @param wert Die Zahl, die gesucht werden soll.
 *
 * @return Die Indexnummer der gesuchten Zahl.
 */
public int sucheSequenziell(int[] array, int wert) {
  fenster.schreibeZeile("\nFühre die Methode sucheSequenziell() aus:");
  fenster.schreibe("Suche in diesem Feld: ");
  fenster.schreibeArray(array);
  fenster.schreibeZeile("");
  // Wiederhole für alle Elemente des Arrays:
  for (int i = 0; i < array.length; i++) {
    fenster.schreibeZeile("Überprüfen des Werts an der Position " + i);
    // Wenn das Element an der Stelle i der gesuchte Wert ist:
    if (array[i] == wert) {
      fenster.schreibeZeile("Fertig :-)");
      // Gib die Position i als Ergebnis zurück:
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_2/Suchalgorithmen.java`

(b) Binäre Suche

Lösungsvorschlag

```
    }
  }
  fenster.schreibeZeile("Nichts gefunden :-(");
  // Gib den Sonderfalls -1 (nichts gefunden) als Ergebnis zurück:
  return -1;
}
```

```java
  /**
   * Binäre Suche: Durchsucht das Array nach dem Wert und gibt dessen
↪   Position als
   * Ergebnis zurück.
   *
   * @param array Ein Feld mit Zahlen.
   * @param wert Die Zahl, die gesucht werden soll.
   *
   * @return Die Indexnummer der gesuchten Zahl.
   */
  public int sucheBinaer(int[] array, int wert) {
    fenster.schreibeZeile("\nFühre die Methode sucheBinaer() aus:");
    fenster.schreibe("Suche in diesem Feld: ");
    fenster.schreibeArray(array);
    fenster.schreibeZeile("");
    int u = 0;
    int o = array.length - 1;
    fenster.schreibeZeile("Suchbereich: " + u + " bis " + o);
    while (u <= o) {
      int m = (u + o) / 2;
      if (array[m] == wert) {
        fenster.schreibe("Mitte: " + m);
        fenster.schreibeZeile(", Treffer : -) ");
        return m;
      } else if (array[m] > wert) {
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_2/Suchalgorithmen.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/30_Suchalgorithmen/Aufgabe_Methode-sucheBinaer.tex`

Sortieralgorithmen

Examensaufgabe „Schreibtischlauf Haldensortierung“ (46115-2013-F.T2-A6)

Aufgabe 6

(a) Vervollständigen Sie die folgende Sortierung mit MergeSort (Sortieren durch Mischen) — beginnen Sie dabei Ihren „rekursiven Abstieg“ immer im linken Teilfeld:

D | 40 5 89 95 85 84 || 14 25 20 52 7 71 |

Notation: Markieren Sie Zeilen mit D(ivide), in denen das Array zerlegt wird, und mit M(erge), in denen Teilarrays zusammengeführt werden. Beispiel:

D | 82 || 89 44 |

D 82 | 89 || 44 |

M 82 | 44 89 |

M | 44 82 89 |

Lösungsvorschlag

```
D | 40    5     89    95    85    84 || 14    25    20    52    7     71 |
D | 40    5     89 || 95    85    84 |
D | 40    5  || 89 |
D | 40 || 5  |
M | 5     40 |
M | 5     40    89 |
D                   |  95    85    84 |
D                   |  95    85 || 84 |
D                   |  95 || 85 |
M                   |  85    95 |
M                   |  84    85    95 |
M | 5     40    84    85    89    95 |
D                                     |  14    25    20 || 52    7     71 |
D                                     |  14    25 || 20 |
D                                     |  14 || 25 |
M                                     |  14    25 |
M                                     |  14    20    25  |
D                                                       |  52    7  || 71 |
D                                                       |  52 || 7  |
M                                                       |  7     52 |
M                                                       |  7     52    71 |
M                                     |  7     14    20     25    52    71 |
M                                     |  7     14    20     25    52    71 |
M | 5     7     14    20     25    40    52    71    84     85    89    95 |
```

(b) Sortieren Sie mittels HeapSort (Haldensortierung) die folgende Liste weiter: Notation: Markieren Sie die Zeilen wie folgt:

I: Initiale Heap-Eigenschaft hergestellt (größtes Element am Anfang der Liste).

R: Erstes und letztes Element getauscht und letztes „gedanklich entfernt“.

S: Erstes Element nach unten „versickert“ (Heap-Eigenschaft wiederhergestellt).

Lösungsvorschlag

```
I | 99 63 91 4 36 81 76 |

R | 76 63 91 4 36 81 || 99 |
S | 91 63 81 4 36 76 || 99 |

R | 76 63 81 4 36 || 91 99 |
S | 81 76 63 4 36 || 91 99 |

R | 36 76 63 4 || 81 91 99 |
S | 76 36 63 4 || 81 91 99 |

R | 4 36 63 || 76 81 91 99 |
S | 63 4 36 || 76 81 91 99 |

R | 4 36 || 63 76 81 91 99 |
S | 36 4 || 63 76 81 91 99 |

R | 4 || 36 63 76 81 91 99 |
S | 4 || 36 63 76 81 91 99 |

R | 4 36 63 76 81 91 99 |
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2013/03/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Bubble- und Quicksort bei 25,1,12,27,30,9,33,34,18,16“ (46115-2016-F.T1-A8)

Sortieralgorithmen
Bubblesort
Quicksort

(a) Sortieren Sie das Array mit den Integer Zahlen

25, 1, 12, 27, 30, 9, 33, 34, 18, 16

(i) mit *BubbleSort*

Lösungsvorschlag

```
  25   1   12   27   30   9   33   34   18   16  Eingabe
  25   1   12   27   30   9   33   34   18   16  Durchlauf Nr. 1
 >25   1<  12   27   30   9   33   34   18   16  vertausche (i 0<>1)
  1   >25  12<  27   30   9   33   34   18   16  vertausche (i 1<>2)
  1    12  25   27  >30   9<  33   34   18   16  vertausche (i 4<>5)
  1    12  25   27   9   30   33  >34   18<  16  vertausche (i 7<>8)
  1    12  25   27   9   30   33   18  >34   16< vertausche (i 8<>9)
  1    12  25   27   9   30   33   18   16   34  Durchlauf Nr. 2
  1    12  25  >27   9<  30   33   18   16   34  vertausche (i 3<>4)
  1    12  25   9   27   30  >33   18<  16   34  vertausche (i 6<>7)
  1    12  25   9   27   30   18  >33   16<  34  vertausche (i 7<>8)
  1    12  25   9   27   30   18   16   33   34  Durchlauf Nr. 3
  1    12 >25   9<  27   30   18   16   33   34  vertausche (i 2<>3)
  1    12   9   25   27  >30   18<  16   33   34  vertausche (i 5<>6)
  1    12   9   25   27   18  >30   16<  33   34  vertausche (i 6<>7)
  1    12   9   25   27   18   16   30   33   34  Durchlauf Nr. 4
  1   >12   9<  25   27   18   16   30   33   34  vertausche (i 1<>2)
  1    9    12   25  >27   18<  16   30   33   34  vertausche (i 4<>5)
  1    9    12   25   18  >27   16<  30   33   34  vertausche (i 5<>6)
  1    9    12   25   18   16   27   30   33   34  Durchlauf Nr. 5
  1    9    12  >25   18<  16   27   30   33   34  vertausche (i 3<>4)
  1    9    12   18  >25   16<  27   30   33   34  vertausche (i 4<>5)
  1    9    12   18   16   25   27   30   33   34  Durchlauf Nr. 6
  1    9    12  >18   16<  25   27   30   33   34  vertausche (i 3<>4)
  1    9    12   16   18   25   27   30   33   34  Durchlauf Nr. 7
  1    9    12   16   18   25   27   30   33   34  Ausgabe
```

(ii) mit *Quicksort*, wenn als Pivotelement das jeweils erste Element gewählt wird.

Beschreiben Sie die Abläufe der Sortierverfahren

(i) bei *BubbleSort* durch eine Angabe der Zwischenergebnisse nach jedem Durchlauf

(ii) bei *Quicksort* durch die Angabe der Zwischenergebnisse nach den rekursiven Aufrufen.

(b) Welche Laufzeit (asymptotisch, in O-Notation) hat BubbleSort bei beliebig großen Arrays mit n Elementen. Begründen Sie Ihre Antwort.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/03/Thema-1/Aufgabe-8.tex`

Examensaufgabe „händisch sortieren, implementieren, Komplexität“ (46115-2017-F.T2-A4)

Bubblesort

Bei Bubblesort wird eine unsortierte Folge von Elementen $a_1, a_2, \ldots, a_n$, von links nach rechts durchlaufen, wobei zwei benachbarte Elemente a_i und a_{i+1} getauscht werden, falls sie nicht in der richtigen Reihenfolge stehen. Dies wird so lange wiederholt, bis die Folge sortiert ist.

(a) Sortieren Sie die folgende Zahlenfolge mit Bubblesort. Geben Sie die neue Zahlenfolge nach jedem (Tausch-)Schritt an: 3, 2, 4, 1

Lösungsvorschlag

```
 3  2  4  1  Eingabe
 3  2  4  1  Durchlauf Nr. 1
>3  2< 4  1  vertausche (i 0<>1)
 2  3 >4  1< vertausche (i 2<>3)
 2  3  1  4  Durchlauf Nr. 2
 2 >3  1< 4  vertausche (i 1<>2)
 2  1  3  4  Durchlauf Nr. 3
>2  1< 3  4  vertausche (i 0<>1)
 1  2  3  4  Durchlauf Nr. 4
 1  2  3  4  Ausgabe
```

(b) Geben Sie den Bubblesort-Algorithmus für ein Array von natürlichen Zahlen in einer Programmiersprache Ihrer Wahl an. Die Funktion `swap (index1, index2)` kann verwendet werden, um zwei Elemente des Arrays zu vertauschen.

Lösungsvorschlag

```java
public class BubbleSort {

  public static void swap(int[] array, int index1, int index2) {
    int tmp = array[index1];
    array[index1] = array[index2];
    array[index2] = tmp;
  }

  public static void bubblesort(int[] array) {
    boolean swapped;
    do {
      swapped = false;
      for (int i = 0; i < array.length - 1; i++) {
        if (array[i] > array[i + 1]) {
          swap(array, i, i + 1);
          swapped = true;
        }
      }
    } while (swapped);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/fruehjahr/BubbleSort.java`

Test

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BubbleSortTest {

  @Test
  public void teste() {
    int[] array = new int[] { 3, 2, 4, 1 };
    BubbleSort.bubblesort(array);
    assertEquals(1, array[0]);
    assertEquals(2, array[1]);
    assertEquals(3, array[2]);
    assertEquals(4, array[3]);
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_46115/jahr_2017/fruehjahr/BubbleSortTest.java`

(c) Geben Sie eine obere Schranke für die Laufzeit an. Beschreiben Sie mögliche Eingabedaten, mit denen diese Schranke erreicht wird.

Lösungsvorschlag

$\mathcal{O}(n^2)$

Diese obere Schranke wird erreicht, wenn die Zahlenfolgen in der umgekehrten Reihenfolge bereits sortiert ist, z. B. 4, 3, 2, 1.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2017/03/Thema-2/Aufgabe-4.tex`

Examensaufgabe „(Sortierverfahren)“ (46115-2019-H.T1-A4)

Selectionsort

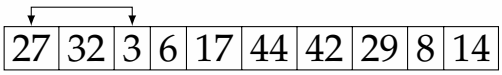
In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen *aufsteigend* sortiert werden. Das Feld habe n Elemente $A[1]$ bis $A[n]$. Der folgende Algorithmus sei gegeben:

```
var A : array[1..n] of integer;

procedure selection_sort
var i, j, smallest, tmp : integer;
begin
  for j := 1 to n-1 do begin
    smallest := j;
    for i := j + 1 to n do begin
      if A[i] < A[smallest] then
        smallest := i;
    end
    tmp = A[j];
    A[j] = A[smallest];
    A[smallest] = tmp;
  end
end
```

(a) Sortieren Sie das folgende Feld mittels des Algorithmus. Notieren Sie alle Werte, die die Variable *smallest* jeweils beim Durchlauf der inneren Schleife annimmt. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

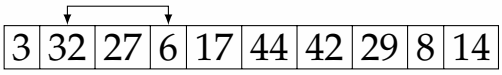
Lösungsvorschlag

Ausgang



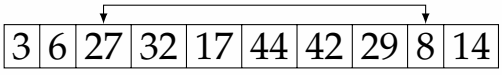
nach 1. Durchlauf ($j = 1$)

smallest: (1) 3



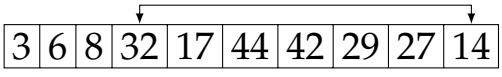
nach 2. Durchlauf ($j = 2$)

smallest: (2) 3 4



nach 3. Durchlauf ($j = 3$)

smallest: (3) 5 9



nach 4. Durchlauf ($j = 4$)

smallest: (4) 5 10

3	6	8	14	17	44	42	29	27	32

nach 5. Durchlauf ($j = 5$)

smallest: (5) -

3	6	8	14	17	44	42	29	27	32

nach 6. Durchlauf ($j = 6$)

smallest: (6) 7 8 9

3	6	8	14	17	27	42	29	44	32

nach 7. Durchlauf ($j = 7$)

smallest: (7) 8

3	6	8	14	17	27	29	42	44	32

nach 8. Durchlauf ($j = 8$)

smallest: (8) 10

3	6	8	14	17	27	29	32	44	42

nach 9. Durchlauf ($j = 9$)

smallest: (9) 10

3	6	8	14	17	27	29	32	44	42

fertig

3	6	8	14	17	27	29	32	42	44

(b) Der Wert der Variablen *smallest* wird bei jedem Durchlauf der äußeren Schleife mindestens ein Mal neu gesetzt. Wie muss das Feld A beschaffen sein, damit der Variablen *smallest* ansonsten niemals ein Wert zugewiesen wird? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Wenn das Feld bereits aufsteigend sortiert ist, dann nimmt die Variable *smallest* in der innneren Schleife niemals einen neuen Wert an.

(c) Welche Auswirkung auf die Sortierung oder auf die Zuweisungen an die Variable *smallest* hat es, wenn der Vergleich in Zeile 9 des Algorithmus statt $A[i] < A[\text{smallest}]$ lautet $A[i] \leq A[\text{smallest}]$? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Der Algorithmus sortiert dann nicht mehr *stabil*, ðdie Eingabereihenfolge von Elementen mit *gleichem Wert* wird beim Sortieren nicht mehr *bewahrt*.

(d) Betrachten Sie den Algorithmus unter der Maßgabe, dass Zeile 9 wie folgt geändert wurde:

```
if A[i] > A[smallest] then
```

Welches Ergebnis berechnet der Algorithmus nun?

Lösungsvorschlag

Der Algorithmus sortiert jetzt absteigend.

(e) Betrachten Sie die folgende *rekursive* Variante des Algorithmus. Der erste Parameter ist wieder das zu sortierende Feld, der Parameter n ist die Größe des Feldes und der Parameter *index* ist eine ganze Zahl. Die Funktion `min_index(A, x, y)` berechnet für $1 \leq x \leq y \leq n$ den Index des kleinsten Elements aus der Menge `{A[x], A[x+1],...,A[y]}`

```
procedure rek_selection_sort(A, n, index : integer)
var k, tmp : integer;
begin
if (Abbruchbedingung) then return;
  k = min_index(A, index, n);
  if k <> index then begin
  tmp := A[k];
  A[k] := A[index];
  A[index] := tmp;
  end
  (rekursiver Aufruf)
end
```

Der initiale Aufruf des Algorithmus lautet: `rek_selection_sort(A, n, 1)`

Vervollständigen Sie die fehlenden Angaben in der Beschreibung des Algorithmus für

- die Abbruchbedingung in Zeile 4 und

Lösungsvorschlag

> `n = index` bzw `n == index`
>
> Begründung: Wenn der aktuelle Index so groß ist wie die Anzahl der Elemente im Feld, dann muss / darf abgebrochen werden, denn dann ist das Feld soriert.

- den rekursiven Aufruf in Zeile 11.

Lösungsvorschlag

> `rek_selection_sort(A, n, index + 1)`
>
> Am Ende der Methode wurde an die Index-Position *index* das kleinste Element gesetzt, jetzt muss an die nächste Index-Position (*index* + *1*) der kleinste Wert, der noch nicht sortieren Zahlen, gesetzt werden.

Begründen Sie Ihre Antworten.

```
import static org.bschlangaul.helfer.Konsole.zeigeZahlenFeld;

public class SelectionSort {

  public static void selectionSort(int[] A) {
    int smallest, tmp;

    for (int j = 0; j < A.length - 1; j++) {
      System.out.println("\nj = " + (j + 1));
      smallest = j;
      for (int i = j + 1; i < A.length; i++) {
        if (A[i] < A[smallest]) {
          smallest = i;
          System.out.println(smallest + 1);
        }
      }
      tmp = A[j];
      A[j] = A[smallest];
      A[smallest] = tmp;
      zeigeZahlenFeld(A);
    }
  }

  public static void rekSelectionSort(int[] A, int n, int index) {
    int k, tmp;

    if (index == n - 1) {
      return;
    }
    k = minIndex(A, index, n);
    if (k != index) {
      tmp = A[k];
      A[k] = A[index];
      A[index] = tmp;
    }
    rekSelectionSort(A, n, index + 1);
```

```
  }

  public static int minIndex(int[] A, int x, int y) {
    int smallest = x;
    for (int i = x; i < y; i++) {
      if (A[i] < A[smallest]) {
        smallest = i;
      }
    }
    return smallest;
  }

  public static void main(String[] args) {
    int[] A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
    selectionSort(A);

    A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
    rekSelectionSort(A, A.length, 0);
    zeigeZahlenFeld(A);
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/SelectionSort.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Pseudo-Code Insertionsort, Bubblesort, Quicksort“ (46115-2021-F.T1-TA2-A1)

(a) Geben Sie für folgende Sortierverfahren jeweils zwei Felder A und B an, so dass das jeweilige Sortierverfahren angewendet auf A seine Best-Case-Laufzeit und angewendet auf B seine Worst-Case-Laufzeit erreicht. (Wir messen die Laufzeit durch die Anzahl der Vergleiche zwischen Elementen der Eingabe.) Dabei soll das Feld A die Zahlen 1,2,...,7 genau einmal enthalten; das Feld B ebenso. Sie bestimmen also nur die Reihenfolge der Zahlen.

Wenden Sie als Beleg für Ihre Aussagen das jeweilige Sortierverfahren auf die Felder A und B an und geben Sie nach jedem größeren Schritt des Algorithmus den Inhalt der Felder an.

Geben Sie außerdem für jedes Verfahren asymptotische Best- und Worst-Case-Laufzeit für ein Feld der Länge n an.

Für drei der Sortierverfahren ist der Pseudocode angegeben. Beachten Sie, dass die Feldindi- zes hier bei 1 beginnen. Die im Pseudocode verwendete Unterroutine Swap(A, :, j) vertauscht im Feld A die Elemente mit den Indizes i und j miteinander.

(i) Insertionsort

(ii) Bubblesort

(iii) Quicksort

Insertionsort(int[ ] A) for 7 = 2 to A.length do key = Alj] i=j-1 while i>0 and Ali] > key do Afi + 1] = Ali] t=t-1 Ali + 1] = key

Bubblesort(int|] A) n := length(A) repeat swapped = false fori=lton—1do if Ali — 1] > Alc] then Swap(A,i — 1,7)

swapped := true

until not swapped

Quicksort(int[ ] A, @ = 1, r = A.length) if 2<r then m = Partition(A, 2, r) | Quicksort(A, 2,m — 1) Quicksort(A,m + 1,r)

int Partition (int|] A, int 2, intr)

pivot = Alr]

i=

for j =¢€tor—1do

if A[j] < pivot then

Swap(A, i, 5) w=i+l

Swap(A,i,r)

return i

(b) Geben Sie die asymptotische Best- und Worst-Case-Laufzeit von Mergesort an.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex

Examensaufgabe „Qualitätssicherung, Testen bei Bubblesort“ (46116-2017-H.T1-TA1-A4)

Ein gängiger Ansatz zur Messung der Qualität von Software ist das automatisierte Testen von Programmen. Im Folgenden werden praktische Testmethoden anhand des nachstehend angegebenen Sortieralgorithmus diskutiert.

Algorithmus 1 Bubble Sort

```java
public class BubbleSort {
  void bubblesort(int[] array, int len) {
    for (int i = 0; i < len - 1; i++) {   // 1
      for (int j = 0; j < len - 1; j++) { // 2
        if (array[j] > array[j + 1]) {    // 3
          int temp = array[j];            // 4
          array[j] = array[j + 1];        // 5
          array[j + 1] = temp;            // 6
        }
      }
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2017/herbst/BubbleSort.java`

(a) Nennen Sie eine Art des Black-Box-Testens und beschreiben Sie deren Durchführung anhand des vorgegebenen Algorithmus.

Lösungsvorschlag

Beim Black-Box-Testen sind die Testfälle von Daten getrieben (Data-Driven) und beziehen sich auf die Anforderungen und das spezifizierte Verhalten.)

$\Rightarrow$ Aufruf der Methoden mit verschiedenen Eingangsparametern und Vergleich der erhaltenen Ergebnisse mit den erwarteten Ergebnissen.

Das Ziel ist dabei eine möglichst hohe Anforderungsüberdeckung, wobei man eine minimale Anzahl von Testfällen durch Äquivalenzklassenzerlegung (1) und Grenzwertanalyse (2) erhält.

zu (1): Man identifiziert Bereiche von Eingabewerten, die jeweils diesselben Ergebnisse liefern. Dies sind die sog. Äquivalenzklassen. Aus diesen wählt man nun je einen Repräsentanten und nutzen diesen für den Testfall.

zu (2): Bei der Grenzwertanalyse identifiziert man die Grenzbereiche der Eingabedaten und wählt Daten aus dem nahen Umfeld dieser für seine Testfälle.

Angewendet auf den gegebenen Bubblesort-Algorithmus würde die Grenzwertanalyse bedeuten, dass man ein bereits aufsteigend sortiertes Array und ein absteigend sortiertes Array übergibt.

(b) Zeichnen Sie ein mit Zeilennummem beschriftetes Kontrollflussdiagramm für den oben angegebenen Sortieralgorithmus.

Lösungsvorschlag

C2a Vollständige Pfadüberdeckung (Full Path Coverage)

Zur Erinnerung: Eine im Code enthaltene Wiederholung mit for muss wie folgt im Kontrollflussgraphen „zerlegt“ werden:

(c) Erklären Sie, ob eine vollständige Pfadüberdeckung für die gegebene Funktion möglich und sinnvoll ist.

Lösungsvorschlag

Eine vollständige Pfadüberdeckung (C_1-Test) kann nicht erreicht werden, da die Bedingung der inneren Wiederholung immer wahr ist, wenn die Bedingung der äußeren Wiederholung wahr ist. D. h., der Pfad S-1-1-2-2-1“ kann nie gegangen werden. Dies wäre aber auch nicht sinnvoll, weil jeder Eintrag mit jedem anderen verglichen werden soll und im Fall true $\rightarrow$ false ein Durchgang ausgelassen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/09/Thema-1/Teilaufgabe-1/Aufgabe-4.tex`

Examensaufgabe „AVL-Baum, Dijkstra, Tiefensuche“ (66115-2006-H.T1-A4)

AVL-Baum

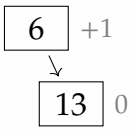
(a) Gegeben sei die folgende Folge ganzer Zahlen: 6, 13, 4, 8, 11, 9, 10.

(i) Fügen Sie obige Zahlen der Reihe nach in einen anfangs leeren AVL-Baum ein und stellen Sie den Baum nach jedem Einfügeschritt dar!

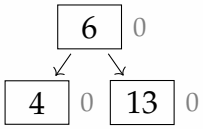
Lösungsvorschlag

Nach dem Einfügen von „6“:

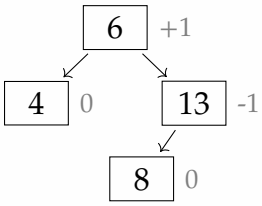
Nach dem Einfügen von „13“:



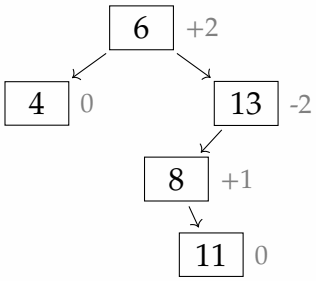
Nach dem Einfügen von „4“:



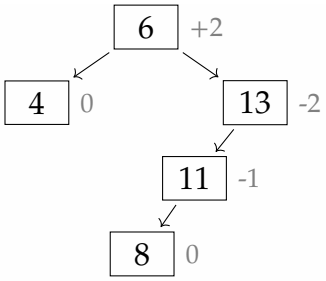
Nach dem Einfügen von „8“:



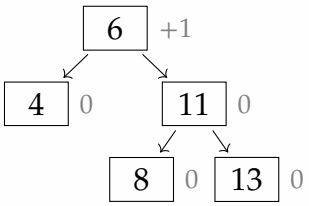
Nach dem Einfügen von „11“:



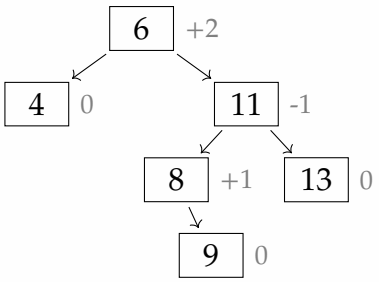
Nach der Linksrotation:



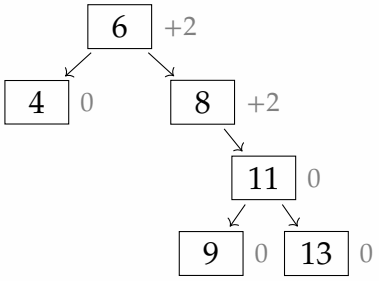
Nach der Rechtsrotation:



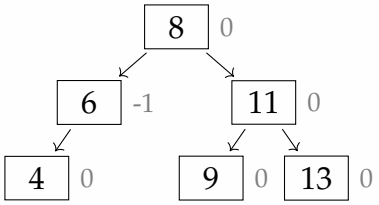
Nach dem Einfügen von „9“:



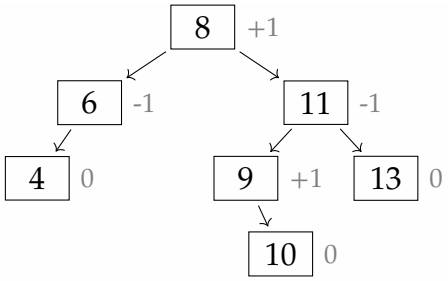
Nach der Rechtsrotation:



Nach der Linksrotation:



Nach dem Einfügen von „10“:

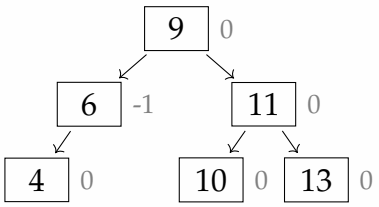


Algorithmus von Dijkstra
Tiefensuche
Quicksort

(ii) Löschen Sie das Wurzelelement des entstandenen AVL-Baums und stellen Sie die AVL-Eigenschaft wieder her!

Lösungsvorschlag

Nach dem Löschen von „8“:



(b) Gegeben sei der folgende gerichtete und gewichtete Graph:

- Bestimmen Sie mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten A zu allen anderen Knoten! Geben Sie dabei nach jedem Verarbeitungsschritt den Zustand der Hilfsdatenstruktur an!
- Skizzieren Sie einen Algorithmus für den Tiefendurchlauf von gerichteten Graphen, wobei jede Kante nur einmal verwendet werden darf!

(c) Ein wesentlicher Nachteil der Standardimplementierung des QUICKSORT Algorithmus ist dessen rekursiver Aufruf. Implementieren Sie den Algorithmus QUICKSORT ohne den rekursiven Prozeduraufruf!

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2006/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Selectionsort“ (66115-2014-H.T2-A6)

Gegeben sei ein einfacher Sortieralgorithmus, der ein gegebenes Feld A dadurch sortiert, dass er das *Minimum* m von A *findet*, dann das Minimum von A ohne das Element m usw.

(a) Geben Sie den Algorithmus in Java an. Implementieren Sie den Algorithmus *in situ*, ðso, dass er außer dem Eingabefeld nur konstanten Extraspeicher benötigt. Es steht eine Testklasse zur Verfügung.

Lösungsvorschlag

```java
public class SortierungDurchAuswaehlen {
  static void vertausche(int[] zahlen, int index1, int index2) {
    int tmp = zahlen[index1];
    zahlen[index1] = zahlen[index2];
    zahlen[index2] = tmp;
  }

  static void sortiereDurchAuswählen(int[] zahlen) {
    // Am Anfang ist die Markierung das erste Element im Zahlen-Array.
    int markierung = 0;
    while (markierung < zahlen.length) {
      // Bestimme das kleinste Element.
      // 'min' ist der Index des kleinsten Elements.
      // Am Anfang auf das letzte Element setzen.
      int min = zahlen.length - 1;
      // Wir müssen nicht bis letzten Index gehen, da wir 'min' auf das
↪   letzte Element
      // setzen.
      for (int i = markierung; i < zahlen.length - 1; i++) {
        if (zahlen[i] < zahlen[min]) {
          min = i;
        }
      }

      // Tausche zahlen[markierung] mit gefundenem Element.
      vertausche(zahlen, markierung, min);
      // Die Markierung um eins nach hinten verlegen.
      markierung++;
    }
  }

  public static void main(String[] args) {
    int[] zahlen = { 5, 2, 7, 1, 6, 3, 4 };
    sortiereDurchAuswählen(zahlen);
    for (int i = 0; i < zahlen.length; i++) {
      System.out.print(zahlen[i] + " ");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/SortierungDurchAuswaehlen.java`

(b) Analysieren Sie die Laufzeit Ihres Algorithmus.

Lösungsvorschlag Halde (Heap)

Beim ersten Durchlauf des *Selectionsort*-Algorithmus muss $n-1$ mal das Minimum durch Vergleich ermittel werden, beim zweiten Mal $n-2$. Mit Hilfe der *Gaußschen Summenformel* kann die Komplexität gerechnet werden:

$$(n-1)+(n-2)+\cdots+3+2+1=\frac{(n-1)\cdot n}{2}=\frac{n^2}{2}-\frac{n}{2}\approx\frac{n^2}{2}\approx n^2$$

Da es bei der Berechnung des Komplexität um die Berechnung der asymptotischen oberen Grenze geht, können Konstanten und die Addition, Subtraktion, Multiplikation und Division mit Konstanten z. b. $\frac{n^2}{2}$ vernachlässigt werden.

Der *Selectionsort*-Algorithmus hat deshalb die Komplexität $\mathcal{O}(n^2)$, er ist von der Ordnung $\mathcal{O}(n^2)$.

(c) Geben Sie eine Datenstruktur an, mit der Sie Ihren Algorithmus beschleunigen können.

Lösungsvorschlag

Der *Selectionsort*-Algorithmus kann mit einer Min- (in diesem Fall) bzw. einer Max-Heap beschleunigt werden. Mit Hilfe dieser Datenstruktur kann sehr schnell das Minimum gefunden werden. So kann auf die vielen Vergleiche verzichtet werden. Die Komplexität ist dann $\mathcal{O}(n \log n)$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/09/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Haldensortierung“ (66115-2015-H.T2-A2)

Heapsort

Gegeben sei folgende Klasse:

```
class W {
  int t;
  String f;
  // ...
```

Dazu gibt es verschiedene Comparatoren, zum Beispiel:

```
// ascending order for field W.t
class ComparatorAscByFieldT implements Comparator<W> {
  // Returns a negative integer, zero, or a positive integer as the
  // first argument is less than, equal to, or greater than the second.
  @Override
  public int compare(W o1, W o2) { // ...
```

Außerdem steht Ihnen die vorgegebene Methode swap zur Verfügung:

```
void swap(W[] w, int a, int b) { // ...
```

(a) Phase 1: Die Haldensortierung beginnt mit der Herstellung der Max-Heap-Eigenschaft von rechts nach links. Diese ist für alle Feldelemente im dunklen Bereich bereits erfüllt. Geben Sie die Positionen (IDs) derjenigen Elemente des Feldes an, die das Verfahren im ‚Versickerschritt* far das nächste Element mit Hilfe des `ComparatorAscByFieldT` miteinander vergleicht:

IDsangeben> 0 1 2 3 4 5 6 <iDs angeben

Nach dem Vergleichen werden gegebenenfalls Werte mit swap vertauscht. Geben Sie das Resultat (in obiger Array-Darstellung) nach diesem Schritt an.

(b) Phase 2: Das folgende Feld enthält den bereits vollständig aufgebauten MaxHeap:

Qo 1 2 3 4 5 6

71/6)/5;,3 7) 14,04 2

Die Haldensortierung verschiebt das maximale Element in den sortierten (dunklen) Bereich:

Q 1 2 3 4

2|6|/5/3/1/o

Geben Sie das Ergebnis des nachfolgenden „Versickerns“ (erneut in derselben Array-Darstellung) an, bei dem die Heap-Eigenschaft wiederhergestellt wird.

(c) Ergänzen Sie die rekursive Methode `reheap`, die die Max-Heap-Eigenschaft im Feld `w` zwischen den Indizes `i` und `k` (jeweils einschließlich) in O(log(k-i)) gemäß `Comparator<W> c` wiederherstellt, indem sie das Element `w[i]` „versickert“. `k` bezeichnet das Ende des unsortierten Bereichs.

```
// restores the max-heap property in w[i to k] using c
void reheap(W[] w, Comparator<W> c, int i, int k) {
  int leftId = 2 * i + 1;
  int rightId = leftId + 1;
  int kidId;
  // ToDo: Code hier ergaenzen
}
```

(d) Implementieren Sie nun die eigentliche Haldensortierung. Sie dürfen hier die Methode `reheap` verwenden.

```
// sorts w in-situ according to the order imposed by c
void heapSort(W[] w, Comparator<W> c) {
  int n = w.length;

  // Phase 1: Max-Heap-Eigenschaft herstellen
  // (siehe Teilaufgabe a)
  // ToDo: Code hier ergaenzen

  // Phase 2: jeweils Maximum entnehmen und sortierte Liste am Ende
↪  aufbauen
  // (siehe Teilaufgabe b)
  // ToDo: Code hier ergaenzen
}
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/09/Thema-2/Aufgabe-2.tex

Examensaufgabe „1 45 8 53 9 2 17 10“ (66115-2016-F.T1-A6)

Quicksort

Sortieren Sie die Werte

1 45 8 53 9 2 17 10

mit Quicksort.

Lösungsvorschlag

Sortieralgorithmus nach Saake

```
 1   45  8   53  9   2   17  10  zerlege
 1   45  8   53* 9   2   17  10  markiere (i 3)
 1   45  8  >53  9   2   17  10< vertausche (i 3<>7)
>1<  45  8   10  9   2   17  53  vertausche (i 0<>0)
 1  >45< 8   10  9   2   17  53  vertausche (i 1<>1)
 1   45 >8<  10  9   2   17  53  vertausche (i 2<>2)
 1   45  8  >10< 9   2   17  53  vertausche (i 3<>3)
 1   45  8   10 >9<  2   17  53  vertausche (i 4<>4)
 1   45  8   10  9  >2<  17  53  vertausche (i 5<>5)
 1   45  8   10  9   2  >17< 53  vertausche (i 6<>6)
 1   45  8   10  9   2   17 >53< vertausche (i 7<>7)
 1   45  8   10  9   2   17      zerlege
 1   45  8   10* 9   2   17      markiere (i 3)
 1   45  8  >10  9   2   17<     vertausche (i 3<>6)
>1<  45  8   17  9   2   10      vertausche (i 0<>0)
 1  >45  8<  17  9   2   10      vertausche (i 1<>2)
 1   8  >45  17  9<  2   10      vertausche (i 2<>4)
 1   8   9  >17  45  2<  10      vertausche (i 3<>5)
 1   8   9   2  >45  17  10<     vertausche (i 4<>6)
 1   8   9   2                  zerlege
 1   8*  9   2                  markiere (i 1)
 1  >8   9   2<                 vertausche (i 1<>3)
>1<  2   9   8                  vertausche (i 0<>0)
 1  >2<  9   8                  vertausche (i 1<>1)
 1   2  >9   8<                 vertausche (i 2<>3)
 1   2                          zerlege
 1*  2                          markiere (i 0)
>1   2<                         vertausche (i 0<>1)
>2   1<                         vertausche (i 0<>1)
                    17  45      zerlege
                    17* 45      markiere (i 5)
                   >17  45<     vertausche (i 5<>6)
                   >45  17<     vertausche (i 5<>6)
```

Sortieralgorithmus nach Horare

```
 1   45  8   53  9   2   17  10  zerlege
```

```
1   45  8   53* 9   2   17  10  markiere (i 3)
1   45  8  >53  9   2   17  10< vertausche (i 3<>7)
1   45  8   10  9   2   17      zerlege
1   45  8   10* 9   2   17      markiere (i 3)
1  >45  8   10  9   2<  17      vertausche (i 1<>5)
1   2   8  >10  9<  45  17      vertausche (i 3<>4)
1   2   8   9                  zerlege
1   2*  8   9                  markiere (i 1)
1   2                          zerlege
1*  2                          markiere (i 0)
        8   9                  zerlege
        8*  9                  markiere (i 2)
                10  45  17     zerlege
                10  45* 17     markiere (i 5)
                10 >45  17<    vertausche (i 5<>6)
                10  17         zerlege
                10* 17         markiere (i 4)
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Sortieren mit Quicksort“ (66115-2016-H.T2-A7)

Quicksort

(a) Gegeben ist die Ausgabe der Methode **Partition** (s. Pseudocode), rekonstruieren Sie die Eingabe.

Konkret sollen Sie das Array A = (_, _, 1, _, _) so vervollständigen, dass der Aufruf Partition(A, 1, 5) die Zahl 3 zurückgibt und nach dem Aufruf gilt, dass A = (1, 2, 3, 4, 5) ist.

Geben Sie A nach jedem Durchgang der for-Schleife in **Partition** an.

Lösungsvorschlag

```
 2  4  1  5  3  Eingabe
 2  4  1  5  3  zerlege
 2  4  1  5  3* markiere (i 4)
>2< 4  1  5  3  vertausche (i 0<>0)
 2 >4  1< 5  3  vertausche (i 1<>2)
 2  1 >4  5  3< vertausche (i 2<>4)
 2  1           zerlege
 2  1*          markiere (i 1)
>2  1<          vertausche (i 0<>1)
          5  4  zerlege
          5  4* markiere (i 4)
         >5  4< vertausche (i 3<>4)
 1  2  3  4  5  Ausgabe
```

(b) Beweisen Sie die Korrektheit von **Partition** (z. B. mittels einer Schleifeninvarianten)!

(c) Geben Sie für jede natürliche Zahl n eine Instanz I_n, der Länge n an, so dass QuickSort(I_n) $\Omega(n^2)$ Zeit benötigt. Begründen Sie Ihre Behauptung.

Lösungsvorschlag

$I_n = 1, 2, 3, \ldots, n$

Die Methode **Partition** wird n mal aufgerufen, weil bei jedem Aufruf der Methode nur eine Zahl, nämlich die größte Zahl, abgespalten wird.

- Partition(A, 1, n)
- Partition(A, 1, $n-1$)
- Partition(A, 1, $n-2$)
- Partition(A, 1, ...)
- Partition(A, 1, 1)

In der For-Schleife der Methode Partition wird bei jeder Wiederholung ein Vertauschvorgang durchgeführt (Die Zahlen werden mit sich selbst getauscht.)

```
 1  2  3  4  5  6  7  zerlege
```

```
 1  2  3  4  5  6  7* markiere (i 6)
>1< 2  3  4  5  6  7  vertausche (i 0<>0)
 1 >2< 3  4  5  6  7  vertausche (i 1<>1)
 1  2 >3< 4  5  6  7  vertausche (i 2<>2)
 1  2  3 >4< 5  6  7  vertausche (i 3<>3)
 1  2  3  4 >5< 6  7  vertausche (i 4<>4)
 1  2  3  4  5 >6< 7  vertausche (i 5<>5)
 1  2  3  4  5  6 >7< vertausche (i 6<>6)
 1  2  3  4  5  6     zerlege
 1  2  3  4  5  6*    markiere (i 5)
>1< 2  3  4  5  6     vertausche (i 0<>0)
 1 >2< 3  4  5  6     vertausche (i 1<>1)
 1  2 >3< 4  5  6     vertausche (i 2<>2)
 1  2  3 >4< 5  6     vertausche (i 3<>3)
 1  2  3  4 >5< 6     vertausche (i 4<>4)
 1  2  3  4  5 >6<    vertausche (i 5<>5)
 1  2  3  4  5        zerlege
 1  2  3  4  5*       markiere (i 4)
>1< 2  3  4  5        vertausche (i 0<>0)
 1 >2< 3  4  5        vertausche (i 1<>1)
 1  2 >3< 4  5        vertausche (i 2<>2)
 1  2  3 >4< 5        vertausche (i 3<>3)
 1  2  3  4 >5<       vertausche (i 4<>4)
 1  2  3  4           zerlege
 1  2  3  4*          markiere (i 3)
>1< 2  3  4           vertausche (i 0<>0)
 1 >2< 3  4           vertausche (i 1<>1)
 1  2 >3< 4           vertausche (i 2<>2)
 1  2  3 >4<          vertausche (i 3<>3)
 1  2  3              zerlege
 1  2  3*             markiere (i 2)
>1< 2  3              vertausche (i 0<>0)
 1 >2< 3              vertausche (i 1<>1)
 1  2 >3<             vertausche (i 2<>2)
 1  2                 zerlege
 1  2*                markiere (i 1)
>1< 2                 vertausche (i 0<>0)
 1 >2<                vertausche (i 1<>1)
```

(d) Was müsste Partition (in Linearzeit) leisten, damit QuickSort Instanzen der Länge n in $\mathcal{O}(n \cdot \log n)$ Zeit sortiert? Zeigen Sie, dass Partition mit der von Ihnen geforderten Eigenschaft zur gewünschten Laufzeit von QuickSort führt.

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ($a \geq 1$).

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ($b > 1$).

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von $T(n)$ unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Lösungsvorschlag

Die Methode **Partition** müsste die Instanzen der Länge n in zwei gleich große Teile spalten ($\frac{n-1}{2}$).

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

2

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

n

Ergibt folgende Rekursionsgleichung:

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

für $\varepsilon = 4$:
$f(n) = n \notin \mathcal{O}\left(n^{\log_2 2 - \varepsilon}\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) = n \in \Theta\left(n^{\log_2 2}\right) = \Theta(n)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$:

$$f(n) = n \notin \Omega\left(n^{\log_2 2+\varepsilon}\right)$$

$\Rightarrow T(n) \in \Theta\left(n^{\log_2 2} \cdot \log n\right) = \Theta(n \cdot \log n)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

Funktion Quicksort(A, $l = 1$, $r = A$.length)

```
if l < r then
    m = Partition(A, l, r);
    Quicksort(A, l, m − 1);
    Quicksort(A, m + 1, r);
end
```

Funktion Partition(A, int l, int r)

```
pivot = A[r];
i = l;
for j = l to r − 1 do
    if A[j] ≤ pivot then
        Swap(A, i, j);
        i = i + l;
    end
end
```

Funktion Swap(A, int l, int r)

```
temp = A[i];
A[i] = A[y];
A[j] = temp;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/09/Thema-2/Aufgabe-7.tex`

Examensaufgabe „Top-Level-Domains (TLD)“ (66115-2017-F.T1-A2)

In dieser Aufgabe sei vereinfachend angenommen, dass sich Top-Level-Domains (TLD) ausschließlich aus zwei oder drei der 26 Kleinbuchstaben des deutschen Alphabets ohne Umlaute zusammensetzen. Im Folgenden sollen TLDs lexikographisch aufsteigend sortiert werden, ðeine TLD (s_1, s_2) mit zwei Buchstaben (z. B. „co“ für Kolumbien) wird also vor einer TLD (t_1, t_2, t_3) der Länge drei (z. B. „com“) einsortiert, wenn $s_1 < t_1 \vee (s_1 = t_1 \wedge s_2 \leq t_2)$ gilt.

(a) Sortieren Sie zunächst die Reihung [„de“, „com“, „uk“, „org“, „co“, „net“, „fr“, „ee“] schrittweise unter Verwendung des Radix-Sortierverfahrens (Bucketsort). Erstellen Sie dazu eine Tabelle wie das folgende Muster und tragen Sie dabei in das Feld „Stelle“ die Position des Buchstabens ein, nach dem im jeweiligen Durchgang sortiert wird (das Zeichen am TLD-Anfang habe dabei die „Stelle“ 1).

> **Exkurs: Alphabet**
> abcdefghijklmnopqrstuvwxyz

Lösungsvorschlag

Stelle	Reihung							
	de_	com	uk_	org	co_	net	fr_	ee_
3	de_	uk_	co_	fr_	ee_	or**g**	co**m**	ne**t**
2	d**e**_	e**e**_	n**e**t	u**k**_	c**o**_	c**o**m	f**r**_	o**r**g
1	**c**o_	**c**om	**d**e_	**e**e_	**f**r_	**n**et	**o**rg	**u**k_

(b) Sortieren Sie nun die gleiche Reihung wieder schrittweise, diesmal jedoch unter Verwendung des Mergesort-Verfahrens (Sortieren durch Mischen). Erstellen Sie dazu eine Tabelle wie das folgende Muster und vermerken Sie in der ersten Spalte jeweils welche Operation durchgeführt wurde: Wenn Sie die Reihung geteilt haben, schreiben Sie in die linke Spalte ein T und markieren Sie die Stelle, an der Sie die Reihung geteilt haben, mit einem senkrechten Strich „|“. Wenn Sie zwei Teilreihungen durch Mischen zusammengeführt haben, schreiben Sie ein M in die linke Spalte und unterstreichen Sie die zusammengemischten Einträge. Beginnen Sie mit dem rekursiven Abstieg immer in der linken Hälfte einer (Teil-)Reihung.

```
O | Reihung
T | de_   com   uk_   org | co_   net   fr_   ee_
T | de_   com | uk_   org
T | de_ | com
M | com   de_
T |             uk_ | org
M |             org   uk_
```

```
M | com   de_   org   uk_
T |                         co_   net | fr_   ee_
T |                         co_ | net
M |                         co_   net
T |                                     fr_ | ee_
T |                                     ee_ | fr_
M |                         co_   ee_   fr_   net
M | co_   com   de_   ee_   fr_   net   org   uk_
```

(c) Implementieren Sie das Sortierverfahren Quicksort für String-TLDs in einer gängigen Programmiersprache Ihrer Wahl. Ihr Programm (Ihre Methode) wird mit drei Parametern gestartet: dem String-Array mit den zu sortierenden TLDs selbst sowie jeweils der Position des ersten und des letzten zu sortierenden Eintrags im Array.

Lösungsvorschlag

```
public class Quicksort {

  public static void swap(String[] array, int index1, int index2) {
    String tmp = array[index1];
    array[index1] = array[index2];
    array[index2] = tmp;
  }

  public static int partition(String[] array, int first, int last) {
    int pivotIndex = (last + first) / 2;
    String pivotValue = array[pivotIndex];
    int pivotIndexFinal = first;
    swap(array, pivotIndex, last);
    for (int i = first; i < last; i++) {
      if (array[i].compareTo(pivotValue) < 0) {
        swap(array, i, pivotIndexFinal);
        pivotIndexFinal++;
      }
    }
    swap(array, last, pivotIndexFinal);
    return pivotIndexFinal;
  }

  public static void sort(String[] array, int first, int last) {
    if (first < last) {
      int pivotIndex = partition(array, first, last);
      sort(array, first, pivotIndex - 1);
      sort(array, pivotIndex + 1, last);
    }
  }

  public static void main(String[] args) {
    String[] array = new String[] { "de", "com", "uk", "org", "co", "net",
↪   "fr", "ee" };
    sort(array, 0, array.length - 1);
    for (int i = 0; i < array.length; i++) {
```

```
        System.out.println(array[i]);
      }
    }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Quicksort.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Quicksort“ (66115-2018-F.T2-A7)

Quicksort

(a) Gegeben ist das folgende Array von Zahlen: $[23, 5, 4, 67, 30, 15, 25, 21]$.

Sortieren Sie das Array mittels Quicksort in-situ aufsteigend von links nach rechts. Geben Sie die (Teil-)Arrays nach jeder Swap-Operation (auch wenn Elemente mit sich selber getauscht werden) und am Anfang jedes Aufrufs der rekursiven Methode an. Verwenden Sie als Pivotelement jeweils das rechteste Element im Teilarray und markieren Sie dieses entsprechend. Teilarrays der Länge ≤ 2 dürfen im rekursiven Aufruf durch direkten Vergleich sortiert werden. Geben Sie am Ende das sortierte Array an.

(b) Welche Worst-Case-Laufzeit (O-Notation) hat Quicksort für n Elemente? Geben Sie ein Array mit fünf Elementen an, in welchem die Quicksort-Variante aus (a) diese Wort-Case-Laufzeit benötigt (ohne Begründung).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-7.tex`

Examensaufgabe „Sortieren von 15,4,10,7,1,8,10 mit Bubble- und Selectionsort“ (66115-2018-H.T2-A8)

Gegeben sei das folgende Feld A mit 7 Schlüsseln:

$$[15, 4, 10, 7, 1, 8, 10]$$

(a) Sortieren Sie das Feld mittels des Sortierverfahrens *Bubblesort*. Markieren Sie jeweils, welche zwei Feldwerte verglichen werden und geben Sie den Zustand des gesamten Feldes jeweils neu an, wenn Sie eine Vertauschung durchgeführt haben.

Lösungsvorschlag

```
 15  4   10  7   1   8   10  Eingabe
 15  4   10  7   1   8   10  Durchlauf Nr. 1
>15  4<  10  7   1   8   10  vertausche (i 0<>1)
 4  >15  10< 7   1   8   10  vertausche (i 1<>2)
 4   10 >15  7<  1   8   10  vertausche (i 2<>3)
 4   10  7  >15  1<  8   10  vertausche (i 3<>4)
 4   10  7   1  >15  8<  10  vertausche (i 4<>5)
 4   10  7   1   8  >15  10< vertausche (i 5<>6)
 4   10  7   1   8   10  15  Durchlauf Nr. 2
 4  >10  7<  1   8   10  15  vertausche (i 1<>2)
 4   7  >10  1<  8   10  15  vertausche (i 2<>3)
 4   7   1  >10  8<  10  15  vertausche (i 3<>4)
 4   7   1   8   10  10  15  Durchlauf Nr. 3
 4  >7   1<  8   10  10  15  vertausche (i 1<>2)
 4   1   7   8   10  10  15  Durchlauf Nr. 4
>4   1<  7   8   10  10  15  vertausche (i 0<>1)
 1   4   7   8   10  10  15  Durchlauf Nr. 5
 1   4   7   8   10  10  15  Ausgabe
```

(b) Sortieren Sie das Feld mittels des Sortierverfahrens *Selectionsort*. Markieren Sie jeweils, welche zwei Feldwerte verglichen werden und geben Sie den Zustand des gesamten Feldes jeweils neu an, wenn Sie eine Vertauschung durchgeführt haben.

Lösungsvorschlag

```
 15  4   10  7   1   8   10  Eingabe
 15  4   10  7   1   8   10* markiere (i 6)
>15  4   10  7   1   8   10< vertausche (i 0<>6)
 10  4   10  7   1   8*  15  markiere (i 5)
>10  4   10  7   1   8<  15  vertausche (i 0<>5)
 8   4   10  7   1*  10  15  markiere (i 4)
 8   4  >10  7   1<  10  15  vertausche (i 2<>4)
 8   4   1   7*  10  10  15  markiere (i 3)
>8   4   1   7<  10  10  15  vertausche (i 0<>3)
 7   4   1*  8   10  10  15  markiere (i 2)
```

```
>7   4   1<  8   10  10  15  vertausche (i 0<>2)
 1   4*  7   8   10  10  15  markiere (i 1)
 1  >4   7   8   10  10  15  vertausche (i 1<>1)
 1*  4   7   8   10  10  15  markiere (i 0)
>1   4   7   8   10  10  15  vertausche (i 0<>0)
 1   4   7   8   10  10  15  Ausgabe
```

(c) Vergleichen Sie beide Sortierverfahren hinsichtlich ihres Laufzeitverhaltens im *best case*. Welches Verfahren ist in dieser Hinsicht besser, wenn das zu sortierende Feld anfangs bereits sortiert ist? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Der Bubblesort-Algorithmus hat im *best case* eine Laufzeit von $\mathcal{O}(n)$, der Selectionsort-Algorithmus $\mathcal{O}(n^2)$.

Bubblesort steuert seine äußere bedingte Wiederholung in vielen Implementationen über eine boolsche Hilfvariable `getauscht`, die beim Betreten der Schleife erstmals auf falsch gesetzt wird. Erst wenn Vertauschungen vergenommen werden müssen, wird diese Varialbe auf wahr gesetzt und die äußere Schleife läuft ein weiteres Mal ab. Ist das zu sortierende Feld bereits sortiert, durchsucht der Algorithmus des Bubblesort das Feld einmal und terminiert dann.

Der Selectionsort-Algorithmus hingegen ist mit zwei ineinander verschränkten Schleifen umgesetzt, deren Wiederholungsanzahl sich starr nach der Anzahl der Elemente im Feld richtet.

Bubblesort

```java
int durchlaufNr = 0;
boolean getauscht;
do {
  durchlaufNr++;
  berichte.feld("Durchlauf Nr. " + durchlaufNr);
  getauscht = false;
  for (int i = 0; i < zahlen.length - 1; i++) {
    if (zahlen[i] > zahlen[i + 1]) {
      // Elemente vertauschen
      vertausche(i, i + 1);
      getauscht = true;
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/sortier/BubbleIterativ.java`

Selectionsort

```java
// Am Anfang ist die Markierung das letzte Element im Zahlen-Array.
int markierung = zahlen.length - 1;
```

```
    while (markierung >= 0) {
      berichte.feldMarkierung(markierung);
      // Bestimme das größtes Element.
      // max ist der Index des größten Elements.
      int max = 0;
      // Wir vergleichen zuerst die Zahlen mit der Index-Number
      // 0 und 1, dann 1 und 2, etc. bis zur Markierung
      for (int i = 1; i <= markierung; i++) {
        if (zahlen[i] > zahlen[max]) {
          max = i;
        }
      }

      // Tausche zahlen[markierung] mit dem gefundenem Element.
      vertausche(markierung, max);
      // Die Markierung um eins nach vorne verlegen.
      markierung--;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/sortier/SelectionRechtsIterativ.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/09/Thema-2/Aufgabe-8.tex`

Examensaufgabe „Notation des Informatik-Duden“ (66115-2019-H.T1-A5)

Quicksort

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente A[0] bis A[n-1]. Der folgende Algorithmus (in der Notation des Informatik-Duden) sei gegeben:

```
procedure quicksort(links, rechts : integer)
var i, j, x : integer;
begin
  i := links;
  j := rechts;
  if j > i then begin
    x := A[links];
    repeat
      while A[i] < x do i := i+1;
      while A[j] > x do j := j-1;
      if i < j then begin
        tmp := A[i]; A[i] := A[j]; A[j] := tmp;
        i := i+1; j := j-1;:
      end
    until i > j;
    quicksort(links, j);
    quicksort(i, rechts);
  end
end
```

Umsetzung in Java:

```
  public static void quicksort(int[] A, int links, int rechts) {
    System.out.println("quick");
    int i = links;
    int j = rechts;
    if (j > i) {
      int x = A[links];
      do {
        while (A[i] < x) {
          i = i + 1;
        }
        while (A[j] > x) {
          j = j - 1;
        }
        if (i <= j) {
          int tmp = A[i];
          A[i] = A[j];
          A[j] = tmp;
          i = i + 1;
          j = j - 1;
        }
        // Java verfügt über keine do-until Schleife.
        // Wir verwenden eine do-while-Schelfe mit einem umgedrehten Test
        // unit i > j -> while (i <= j)
      } while (i <= j);
      quicksort(A, links, j);
```

```
    quicksort(A, i, rechts);
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/herbst/QuickSort.java

Der initiale Aufruf der Prozedur lautet:

```
quicksort(0,n-1)
```

(a) Sortieren Sie das folgende Feld der Länge 7 mittels des Algorithmus. Notieren Sie jeweils alle Aufrufe der Prozedur quicksort mit den konkreten Parameterwerten. Geben Sie zudem für jeden Aufruf der Prozedur den Wert des in Zeile 7 gewählten Elements an.

27 13 21 3 6 17 44 42

Lösungsvorschlag

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
quicksort(1, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
x: 44
quicksort(5, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44
```

(b) Angenommen, die Bedingung $j > i$ in Zeile 6 des Algorithmus wird ersetzt durch die Bedingung $j \geq i$. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

Lösungsvorschlag

geht, dauert aber länger

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
quicksort(1, 1)
3 6 17 32 27 44 42
x: 6
quicksort(1, 0)
3 6 17 32 27 44 42
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
x: 17
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(3, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
x: 27
quicksort(3, 2)
3 6 17 27 32 44 42
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
x: 44
quicksort(5, 5)
3 6 17 27 32 42 44
x: 42
quicksort(5, 4)
3 6 17 27 32 42 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
```

```
3 6 17 27 32 42 44
x: 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(7, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44
```

(c) Angenommen, die Bedingung $i \leq j$ in Zeile 11 des Algorithmus wird ersetzt durch die Bedingung $i < j$. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

Lösungsvorschlag

bleibt hängen

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
```

(d) Wie muss das Feld A gestaltet sein, damit der Algorithmus mit der geringsten Anzahl von Schritten terminiert? Betrachten Sie dazu vor allem Zeile 7. Begründen Sie Ihre Antwort und geben Sie ein Beispiel.

Lösungsvorschlag

Im Worst Case (schlechtesten Fall) wird das Pivotelement stets so gewählt, dass es das größte oder das kleinste Element der Liste ist. Dies ist etwa der Fall, wenn als Pivotelement stets das Element am Ende der Liste gewählt wird und die zu sortierende Liste bereits sortiert vorliegt. Die zu untersuchende Liste wird dann in jedem Rekursionsschritt nur um eins kleiner und die Zeitkomplexität wird beschrieben durch $\mathcal{O}(n^2)$. Die Anzahl der Vergleiche ist in diesem Fall $\frac{n \cdot (n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1$.

Die Länge der jeweils längeren Teilliste beim rekursiven Aufrufe ist nämlich im Schnitt $\frac{2}{n} \sum_{i=\frac{n}{2}}^{n-1} i = \frac{3}{4}n - \frac{2}{4}$ und die Tiefe der Rekursion damit in $\mathcal{O}(\log(n))$.

Im Average Case ist die Anzahl der Vergleiche etwa $2 \cdot \log(2) \cdot (n+1) \cdot \log_2(n) \approx 1,39 \cdot (n+1) \cdot \log_2(n)$.

(e) Die rekursiven Aufrufe in den Zeilen 16 und 17 des Algorithmus werden zur Laufzeit des Computers auf dem Stack verwaltet. Die Anzahl der Aufrufe von quicksort auf dem Stack abhängig von der Eingabegröße n sei mit s(n) bezeichnet. Geben Sie die Komplexitätsklasse von s(n) für den schlimmsten möglichen Fall an. Begründen Sie Ihre Antwort.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-1/Aufgabe-5.tex

Examensaufgabe „Sortieren“ (66115-2021-F.T1-TA2-A1)

(a) Geben Sie für folgende Sortierverfahren jeweils zwei Felder A und B an, so dass das jeweilige Sortierverfahren angewendet auf A seine Best-Case-Laufzeit und angewendet auf B seine Worst-Case-Laufzeit erreicht. (Wir messen die Laufzeit durch die Anzahl der Vergleiche zwischen Elementen der Eingabe.) Dabei soll das Feld A die Zahlen $1, 2, \ldots, 7$ genau einmal enthalten; das Feld B ebenso. Sie bestimmen also nur die Reihenfolge der Zahlen.

Wenden Sie als Beleg für Ihre Aussagen das jeweilige Sortierverfahren auf die Felder A und B an und geben Sie nach jedem größeren Schritt des Algorithmus den Inhalt der Felder an.

Geben Sie außerdem für jedes Verfahren asymptotische Best- und Worst-Case-Laufzeit für ein Feld der Länge n an.

Die im Pseudocode verwendete Unterroutine Swap(A, i, j) vertauscht im Feld A die jeweiligen Elemente mit den Indizes i und j miteinander.

(i) **Insertionsort**

Lösungsvorschlag

Best-Case

1	2	3	4	5	6	7

```
 1  2  3  4  5  6  7  Eingabe
 1  2* 3  4  5  6  7  markiere (i 1)
 1  2  3* 4  5  6  7  markiere (i 2)
 1  2  3  4* 5  6  7  markiere (i 3)
 1  2  3  4  5* 6  7  markiere (i 4)
 1  2  3  4  5  6* 7  markiere (i 5)
 1  2  3  4  5  6  7* markiere (i 6)
 1  2  3  4  5  6  7  Ausgabe
```

Worst-Case

7	6	5	4	3	2	1

```
 7  6  5  4  3  2  1  Eingabe
 7  6* 5  4  3  2  1  markiere (i 1)
>7  7< 5  4  3  2  1  vertausche (i 0<>1)
 6  7  5* 4  3  2  1  markiere (i 2)
 6 >7  7< 4  3  2  1  vertausche (i 1<>2)
>6  6< 7  4  3  2  1  vertausche (i 0<>1)
 5  6  7  4* 3  2  1  markiere (i 3)
 5  6 >7  7< 3  2  1  vertausche (i 2<>3)
 5 >6  6< 7  3  2  1  vertausche (i 1<>2)
```

```
>5  5< 6  7  3  2  1  vertausche (i 0<>1)
 4  5  6  7  3* 2  1  markiere (i 4)
 4  5  6 >7  7< 2  1  vertausche (i 3<>4)
 4  5 >6  6< 7  2  1  vertausche (i 2<>3)
 4 >5  5< 6  7  2  1  vertausche (i 1<>2)
>4  4< 5  6  7  2  1  vertausche (i 0<>1)
 3  4  5  6  7  2* 1  markiere (i 5)
 3  4  5  6 >7  7< 1  vertausche (i 4<>5)
 3  4  5 >6  6< 7  1  vertausche (i 3<>4)
 3  4 >5  5< 6  7  1  vertausche (i 2<>3)
 3 >4  4< 5  6  7  1  vertausche (i 1<>2)
>3  3< 4  5  6  7  1  vertausche (i 0<>1)
 2  3  4  5  6  7  1* markiere (i 6)
 2  3  4  5  6 >7  7< vertausche (i 5<>6)
 2  3  4  5 >6  6< 7  vertausche (i 4<>5)
 2  3  4 >5  5< 6  7  vertausche (i 3<>4)
 2  3 >4  4< 5  6  7  vertausche (i 2<>3)
 2 >3  3< 4  5  6  7  vertausche (i 1<>2)
>2  2< 3  4  5  6  7  vertausche (i 0<>1)
 1  2  3  4  5  6  7  Ausgabe
```

(ii) Standardversion von **Quicksort** (Pseudocode s.u., Feldindizes beginnen bei 1), bei der das letzte Element eines Teilfeldes als Pivot-Element gewählt wird.

Funktion Quicksort(A, $l = 1, r = A$.length)

```
if l < r then
    m = Partition(A, l, r);
    Quicksort(A, l, m − 1);
    Quicksort(A, m + 1, r);
end
```

Funktion Partition(A, int l, int r)

```
pivot = A[r];
i = l;
for j = l to r − 1 do
    if A[j] < pivot then
        Swap(A, i, j);
        i = i + 1;
    end
end
```

Lösungsvorschlag

Best-Case

1	3	2	6	5	7	4

```
 1  3  2  6  5  7  4  zerlege
 1  3  2  6  5  7  4* markiere (i 6)
>1< 3  2  6  5  7  4  vertausche (i 0<>0)
 1 >3< 2  6  5  7  4  vertausche (i 1<>1)
 1  3 >2< 6  5  7  4  vertausche (i 2<>2)
 1  3  2 >6  5  7  4< vertausche (i 3<>6)
 1  3  2              zerlege
 1  3  2*             markiere (i 2)
>1< 3  2              vertausche (i 0<>0)
 1 >3  2<             vertausche (i 1<>2)
             5  7  6  zerlege
             5  7  6* markiere (i 6)
            >5< 7  6  vertausche (i 4<>4)
             5 >7  6< vertausche (i 5<>6)
```

Worst-Case

7	6	5	4	3	2	1

```
 1  2  3  4  5  6  7  zerlege
 1  2  3  4  5  6  7* markiere (i 6)
>1< 2  3  4  5  6  7  vertausche (i 0<>0)
 1 >2< 3  4  5  6  7  vertausche (i 1<>1)
 1  2 >3< 4  5  6  7  vertausche (i 2<>2)
 1  2  3 >4< 5  6  7  vertausche (i 3<>3)
 1  2  3  4 >5< 6  7  vertausche (i 4<>4)
 1  2  3  4  5 >6< 7  vertausche (i 5<>5)
 1  2  3  4  5  6 >7< vertausche (i 6<>6)
 1  2  3  4  5  6     zerlege
 1  2  3  4  5  6*    markiere (i 5)
>1< 2  3  4  5  6     vertausche (i 0<>0)
 1 >2< 3  4  5  6     vertausche (i 1<>1)
 1  2 >3< 4  5  6     vertausche (i 2<>2)
 1  2  3 >4< 5  6     vertausche (i 3<>3)
 1  2  3  4 >5< 6     vertausche (i 4<>4)
 1  2  3  4  5 >6<    vertausche (i 5<>5)
 1  2  3  4  5        zerlege
 1  2  3  4  5*       markiere (i 4)
>1< 2  3  4  5        vertausche (i 0<>0)
 1 >2< 3  4  5        vertausche (i 1<>1)
```

```
 1  2 >3< 4  5        vertausche (i 2<>2)
 1  2  3 >4< 5        vertausche (i 3<>3)
 1  2  3  4 >5<       vertausche (i 4<>4)
 1  2  3  4           zerlege
 1  2  3  4*          markiere (i 3)
>1< 2  3  4           vertausche (i 0<>0)
 1 >2< 3  4           vertausche (i 1<>1)
 1  2 >3< 4           vertausche (i 2<>2)
 1  2  3 >4<          vertausche (i 3<>3)
 1  2  3              zerlege
 1  2  3*             markiere (i 2)
>1< 2  3              vertausche (i 0<>0)
 1 >2< 3              vertausche (i 1<>1)
 1  2 >3<             vertausche (i 2<>2)
 1  2                 zerlege
 1  2*                markiere (i 1)
>1< 2                 vertausche (i 0<>0)
 1 >2<                vertausche (i 1<>1)
```

(iii) **QuicksortVar**: Variante von Quicksort, bei der immer das mittlere Element eines Teilfeldes als Pivot-Element gewählt wird (Pseudocode s.u., nur eine Zeile neu).
Bei einem Aufruf von PartitionVar auf ein Teilfeld $A[l \dots r]$ wird also erst mithilfe der Unterroutine Swap $A\left[\lfloor \frac{l+r-1}{2} \rfloor\right]$ mit $A[r]$ vertauscht.

Funktion QuicksortVar(A, $l = 1$, $r = A$.length)

```
if l < r then
    m = PartitionVar(A, l, r);
    QuicksortVar(A, l, m − 1);
    QuicksortVar(A, m + 1, r);
end
```

Funktion PartitionVar(A, int l, int r)

```
Swap(A, ⌊(l+r−1)/2⌋, r);
pivot = A[r];
i = l;
for j = l to r − 1 do
    if A[j] < pivot then
        Swap(A, i, j);
        i = i + 1;
    end
end
```

Lösungsvorschlag

Best-Case

1	2	3	4	5	6	7

```
 1  2  3  4  5  6  7  zerlege
 1  2  3  4* 5  6  7  markiere (i 3)
 1  2  3 >4  5  6  7< vertausche (i 3<>6)
>1< 2  3  7  5  6  4  vertausche (i 0<>0)
 1 >2< 3  7  5  6  4  vertausche (i 1<>1)
 1  2 >3< 7  5  6  4  vertausche (i 2<>2)
 1  2  3 >7  5  6  4< vertausche (i 3<>6)
 1  2  3              zerlege
 1  2* 3              markiere (i 1)
 1 >2  3<             vertausche (i 1<>2)
>1< 3  2              vertausche (i 0<>0)
 1 >3  2<             vertausche (i 1<>2)
             5  6  7  zerlege
             5  6* 7  markiere (i 5)
             5 >6  7< vertausche (i 5<>6)
            >5< 7  6  vertausche (i 4<>4)
             5 >7  6< vertausche (i 5<>6)
 1  2  3  4  5  6  7  Ausgabe
```

Worst-Case

2	4	6	7	1	5	3

```
 2  4  6  7  1  5  3  zerlege
 2  4  6  7* 1  5  3  markiere (i 3)
 2  4  6 >7  1  5  3< vertausche (i 3<>6)
>2< 4  6  3  1  5  7  vertausche (i 0<>0)
 2 >4< 6  3  1  5  7  vertausche (i 1<>1)
 2  4 >6< 3  1  5  7  vertausche (i 2<>2)
 2  4  6 >3< 1  5  7  vertausche (i 3<>3)
 2  4  6  3 >1< 5  7  vertausche (i 4<>4)
 2  4  6  3  1 >5< 7  vertausche (i 5<>5)
 2  4  6  3  1  5 >7< vertausche (i 6<>6)
 2  4  6  3  1  5     zerlege
 2  4  6* 3  1  5     markiere (i 2)
 2  4 >6  3  1  5<    vertausche (i 2<>5)
>2< 4  5  3  1  6     vertausche (i 0<>0)
 2 >4< 5  3  1  6     vertausche (i 1<>1)
 2  4 >5< 3  1  6     vertausche (i 2<>2)
 2  4  5 >3< 1  6     vertausche (i 3<>3)
```

Mergesort

```
 2  4  5  3 >1< 6     vertausche (i 4<>4)
 2  4  5  3  1 >6<    vertausche (i 5<>5)
 2  4  5  3  1        zerlege
 2  4  5* 3  1        markiere (i 2)
 2  4 >5  3  1<       vertausche (i 2<>4)
>2< 4  1  3  5        vertausche (i 0<>0)
 2 >4< 1  3  5        vertausche (i 1<>1)
 2  4 >1< 3  5        vertausche (i 2<>2)
 2  4  1 >3< 5        vertausche (i 3<>3)
 2  4  1  3 >5<       vertausche (i 4<>4)
 2  4  1  3           zerlege
 2  4* 1  3           markiere (i 1)
 2 >4  1  3<          vertausche (i 1<>3)
>2< 3  1  4           vertausche (i 0<>0)
 2 >3< 1  4           vertausche (i 1<>1)
 2  3 >1< 4           vertausche (i 2<>2)
 2  3  1 >4<          vertausche (i 3<>3)
 2  3  1              zerlege
 2  3* 1              markiere (i 1)
 2 >3  1<             vertausche (i 1<>2)
>2< 1  3              vertausche (i 0<>0)
 2 >1< 3              vertausche (i 1<>1)
 2  1 >3<             vertausche (i 2<>2)
 2  1                 zerlege
 2* 1                 markiere (i 0)
>2  1<                vertausche (i 0<>1)
>1< 2                 vertausche (i 0<>0)
 1 >2<                vertausche (i 1<>1)
```

(b) Geben Sie die asymptotische Best- und Worst-Case-Laufzeit von **Mergesort** an.

Lösungsvorschlag

Best-Case: $\mathcal{O}(n \cdot \log(n))$

Worst-Case: $\mathcal{O}(n^2)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex

Datenflussorientiertes Testen
Bubble Sort
Datenfluss-annotierter Kontrollflussgraph

Examensaufgabe „Sort-Methode und datenflussorientierte Überdeckungskriterien“ (66116-2016-H.T1-TA2-A3)

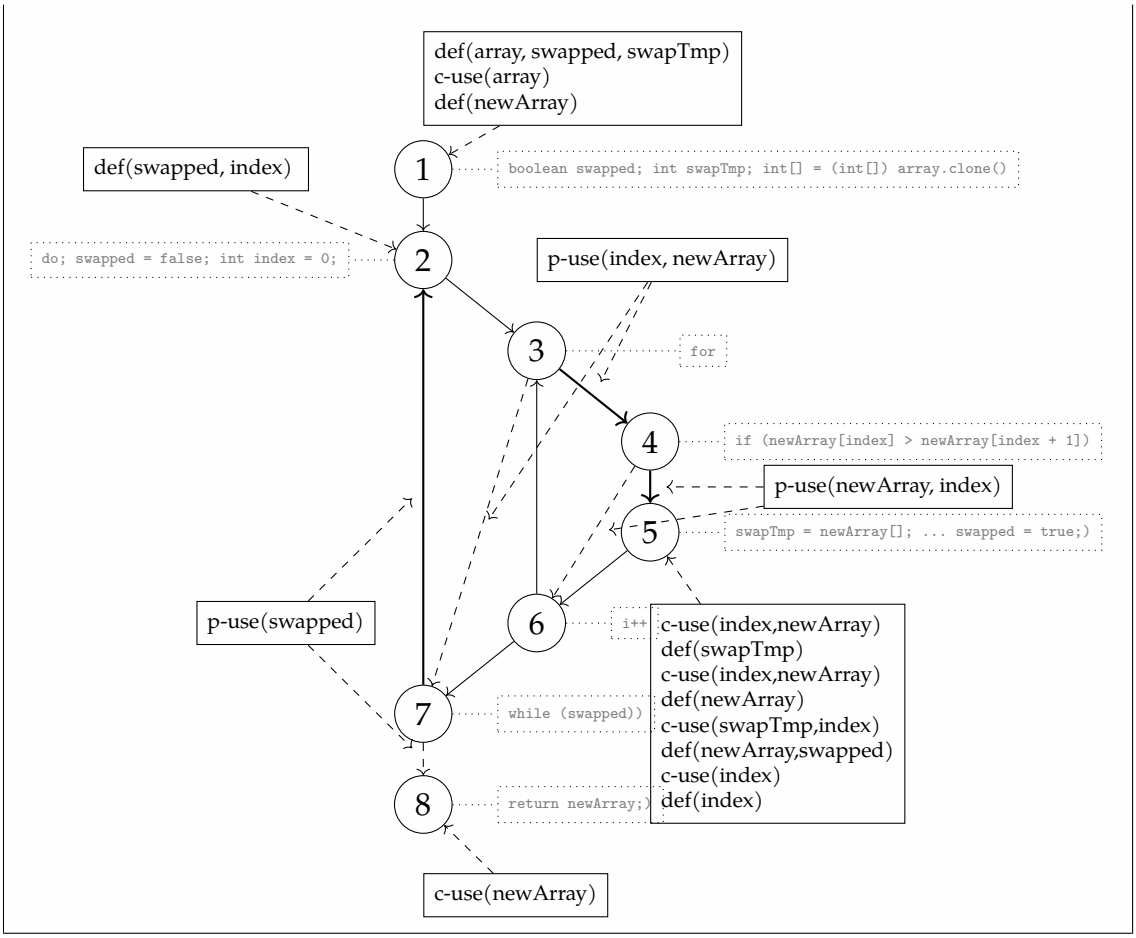
Gegeben Sei folgende Java-Methode `sort` zum Sortieren eines Feldes ganzer Zahlen:

```
  public static int[] sort(int[] array) {
    boolean swapped;
    int swapTmp;
    int[] newArray = (int[]) array.clone();
    do {
      swapped = false;
      for (int index = 0; index < newArray.length - 1; index++) {
        if (newArray[index] > newArray[index + 1]) {
          swapTmp = newArray[index];
          newArray[index] = newArray[index + 1];
          newArray[index + 1] = swapTmp;
          swapped = true;
        }
      }
    } while (swapped);
    return newArray;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/BubbleSort.java`

(a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments und annotieren Sie an den Knoten und Kanten die zugehörigen Datenflussinformationen (Definitionen bzw. berechnende oder prädikative Verwendung von Variablen).

Lösungsvorschlag



(b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

Der Graph hat 8 Knoten und 10 Kanten. Daher ist die zyklomatische Komplexität nach McCabe gegeben durch 10 - 8 + 2 = 4.

(c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

Lösungsvorschlag

Die Eingabe muss mindestens ein Feld der Länge 3 sein. Ansonsten wäre das Feld schon sortiert bzw. bräuchte nur eine Vertauschung und die innere if-Bedingung wäre nicht zu 100% überdeckt. Daher wählt man beispielsweise array = [1,3,2].

(d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

Lösungsvorschlag

Das Kriterium all-uses ist das Hauptkriterium des datenflussorientierten Testens, denn es testet den kompletten prädikativen und berechnenden Datenfluss. Konkret: von jedem Knoten mit einem globalen def(x) einer Variable x existiert ein definitions-freier Pfad bzgl. x (def-clear(x)) zu jedem erreichbaren Knoten mit einem c-use(x) oder p-use(x)).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Übungsaufgabe „Händisch Quick- und Mergesort“ (Mergesort, Quicksort)

Mergesort
Quicksort

Gegeben ist folgende Zahlenfolge:
35, 22, 5, 3, 28, 16, 8, 60, 17, 66, 4, 9, 82, 11, 10, 20

(a) Sortiere Sie händisch mit Mergesort. Orientieren Sie sich beim Aufschreiben der Zwischenschritte an dieser Darstellung:

(b) Sortiere Sie händisch mit Quicksort. Wählen Sie als Pivot-Element immer das Element in der Mitte - oder gegebenenfalls das Element direkt links neben der Mitte. Orientieren Sie sich beim Aufschreiben der Zwischenschritte an dieser Darstellung:

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/40_Sortieralgorithmen/Aufgabe_Haendisch-Quick-Mergesort.tex`

Übungsaufgabe „Händisches Sortieren“ (Bubblesort, Mergesort, Quicksort)

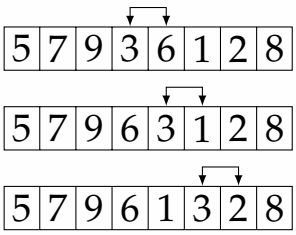
Bubblesort

Gegeben sei ein Array a, welches die Werte $5, 7, 9, 3, 6, 1, 2, 8$ enthält. Sortieren Sie das Array händisch mit:

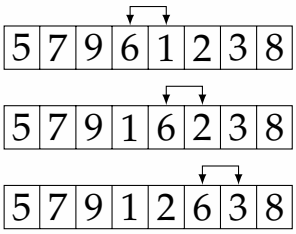
(a) Bubblesort

Lösungsvorschlag

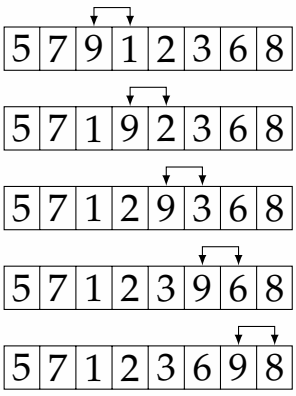
erster Durchgang



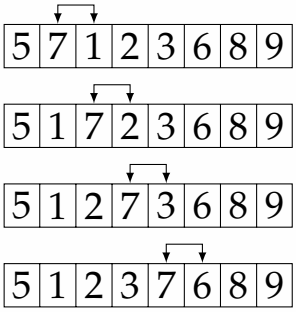
Zweiter Durchgang



Dritter Durchgang

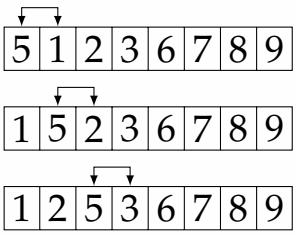


Vierter Durchgang

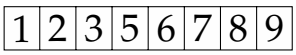


Mergesort

Fünfter Durchgang



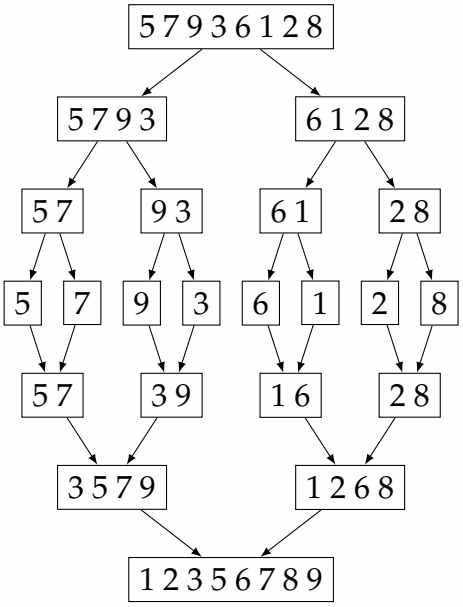
fertig



```
 5  7  9  3  6  1  2  8  Eingabe
 5  7  9  3  6  1  2  8  Durchlauf Nr. 1
 5  7 >9  3< 6  1  2  8  vertausche [2<>3]
 5  7  3 >9  6< 1  2  8  vertausche [3<>4]
 5  7  3  6 >9  1< 2  8  vertausche [4<>5]
 5  7  3  6  1 >9  2< 8  vertausche [5<>6]
 5  7  3  6  1  2 >9  8< vertausche [6<>7]
 5  7  3  6  1  2  8  9  Durchlauf Nr. 2
 5 >7  3< 6  1  2  8  9  vertausche [1<>2]
 5  3 >7  6< 1  2  8  9  vertausche [2<>3]
 5  3  6 >7  1< 2  8  9  vertausche [3<>4]
 5  3  6  1 >7  2< 8  9  vertausche [4<>5]
 5  3  6  1  2  7  8  9  Durchlauf Nr. 3
>5  3< 6  1  2  7  8  9  vertausche [0<>1]
 3  5 >6  1< 2  7  8  9  vertausche [2<>3]
 3  5  1 >6  2< 7  8  9  vertausche [3<>4]
 3  5  1  2  6  7  8  9  Durchlauf Nr. 4
 3 >5  1< 2  6  7  8  9  vertausche [1<>2]
 3  1 >5  2< 6  7  8  9  vertausche [2<>3]
 3  1  2  5  6  7  8  9  Durchlauf Nr. 5
>3  1< 2  5  6  7  8  9  vertausche [0<>1]
 1 >3  2< 5  6  7  8  9  vertausche [1<>2]
 1  2  3  5  6  7  8  9  Durchlauf Nr. 6
 1  2  3  5  6  7  8  9  Ausgabe
```

(b) Mergesort

Lösungsvorschlag




```
5  7  9  3  6  1  2  8  Eingabe
5  7  9  3||6  1  2  8  teile [3]
5  7||9  3              teile [1]
5||7                    teile [0]
5  7                    Nach dem Mischen [0-1]
      9||3              teile [2]
      3  9              Nach dem Mischen [2-3]
3  5  7  9              Nach dem Mischen [0-3]
            6  1||2  8  teile [5]
            6||1        teile [4]
            1  6        Nach dem Mischen [4-5]
                  2||8  teile [6]
                  2  8  Nach dem Mischen [6-7]
            1  2  6  8  Nach dem Mischen [4-7]
1  2  3  5  6  7  8  9  Nach dem Mischen
1  2  3  5  6  7  8  9  Ausgabe
```

(c) Quicksort

Lösungsvorschlag

```
 5  7  9  3  6  1  2  8  Eingabe
 5  7  9  3  6  1  2  8  zerlege
 5  7  9  3* 6  1  2  8  markiere [3]
 5  7  9 >3  6  1  2  8< vertausche [3<>7]
>5  7  9  8  6  1< 2  3  vertausche [0<>5]
 1 >7  9  8  6  5  2< 3  vertausche [1<>6]
 1  2 >9  8  6  5  7  3< vertausche [2<>7]
 1  2                    zerlege [0-1]
 1* 2                    markiere [0]
```

```
>1  2<                    vertausche [0<>1]
>2  1<                    vertausche [0<>1]
          8  6  5  7  9   zerlege [3-7]
          8  6  5* 7  9   markiere [5]
          8  6 >5  7  9<  vertausche [5<>7]
         >8  6  9  7  5<  vertausche [3<>7]
             6  9  7  8   zerlege [4-7]
             6  9* 7  8   markiere [5]
             6 >9  7  8<  vertausche [5<>7]
            >6< 8  7  9   vertausche [4<>4]
             6 >8< 7  9   vertausche [5<>5]
             6  8 >7< 9   vertausche [6<>6]
             6  8  7 >9<  vertausche [7<>7]
             6  8  7      zerlege [4-6]
             6  8* 7      markiere [5]
             6 >8  7<     vertausche [5<>6]
            >6< 7  8      vertausche [4<>4]
             6 >7< 8      vertausche [5<>5]
             6  7 >8<     vertausche [6<>6]
             6  7         zerlege [4-5]
             6* 7         markiere [4]
            >6  7<        vertausche [4<>5]
            >7  6<        vertausche [4<>5]
 1  2  3  5  6  7  8  9   Ausgabe
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/40_Sortieralgorithmen/Aufgabe_Haendisches-Sortieren.tex

Übungsaufgabe „Sortier-Vorlage“ (Selectionsort, Bubblesort)

Für diese Aufgabe wird die Vorlage Sortieralgorithmen benötigt, die auf dem Beiblatt genauer erklärt wird.

Die fertigen Methoden sollen in der Lage sein, beliebige Arrays zu sortieren. Im gelben Textfeld des Eingabefensters soll dabei wieder ausführlich und nachvollziehbar angezeigt werden, wie die jeweilige Methode vorgeht. Beispielsweise so:

```
Führe die Methode selectionSort() aus:
Sortiere dieses Feld: 5, 3, 17, 7, 42, 23
Der Marker liegt bei: 5
Das Maximum liegt bei: 4
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 5, 3, 17, 7, 23, 42
Der Marker liegt bei: 4
Das Maximum liegt bei: 4
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 5, 3, 17, 7, 23, 42
Der Marker liegt bei: 3
Das Maximum liegt bei: 2
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 5, 3, 7, 17, 23, 42
Der Marker liegt bei: 2
Das Maximum liegt bei: 2
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 5, 3, 7, 17, 23, 42
Der Marker liegt bei: 1
Das Maximum liegt bei: 0
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 3, 5, 7, 17, 23, 42
Der Marker liegt bei: 0
Das Maximum liegt bei: 0
Diese beiden Elemente werden nun vertauscht.
Ergebnis dieser Runde: 3, 5, 7, 17, 23, 42


Führe die Methode bubbleSort() aus:
Sortiere dieses Feld: 5, 3, 17, 7, 42, 23
Das Element an der Stelle 0 ist größer als sein Nachfolger.
Diese beiden werden nun vertauscht.
Ergebnis dieser Runde: 3, 5, 17, 7, 42, 23
Das Element an der Stelle 2 ist größer als sein Nachfolger.
Diese beiden werden nun vertauscht.
Ergebnis dieser Runde: 3, 5, 7, 17, 42, 23
Das Element an der Stelle 4 ist größer als sein Nachfolger.
Diese beiden werden nun vertauscht.
Ergebnis dieser Runde: 3, 5, 7, 17, 23, 42
```

(a) Vervollständige die Methode `selectionSort()`.

Lösungsvorschlag

```
/**
 * SelectionSort: Sortieren durch Selektion.
 *
 * @param array Ein Feld mit Zahlen.
 *
 * @return Ein sortiertes Feld mit Zahlen.
 */
public int[] selectionSort(int[] array) {
  fenster.schreibeZeile("\nFühre die Methode selectionSort() aus:");
  fenster.schreibe("Sortiere dieses Feld: ");
  fenster.schreibeArray(array);
  fenster.schreibeZeile("");
  int marker = array.length - 1;
  while (marker >= 0) {
    int max = 0;
    for (int i = 0; i <= marker; i++) {
      if (array[i] > array[max]) {
        max = i;
      }
    }
    fenster.schreibeZeile("Der Marker liegt bei: " + marker);
    fenster.schreibeZeile("Das Maximum liegt bei: " + max);
    fenster.schreibeZeile("Diese beiden Elemente werden nun vertauscht.");
    swap(array, marker, max);
    fenster.schreibe("Ergebnis dieser Runde: ");
    fenster.schreibeArray(array);
    fenster.schreibeZeile("");
    marker--;
  }
  return array;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_2/Sortieralgorithmen.java`

Bubblesort

(b) Vervollständige die Methode `bubbleSort()`.

Lösungsvorschlag

```java
/**
 * BubbleSort: Sortieren durch Vertauschen.
 *
 * @param array Ein Feld mit Zahlen.
 *
 * @return Ein sortiertes Feld mit Zahlen.
 */
public int[] bubbleSort(int[] array) {
  fenster.schreibeZeile("\nFühre die Methode bubbleSort() aus:");
  fenster.schreibe("Sortiere dieses Feld: ");
  fenster.schreibeArray(array);
  fenster.schreibeZeile("");
  boolean swapped;
  do {
    swapped = false;
    for (int i = 0; i < array.length - 1; i++) {
      if (array[i] > array[i + 1]) {
        fenster.schreibe("Das Element an der Stelle " + i);
        fenster.schreibeZeile(" ist größer als sein Nachfolger.");
        fenster.schreibeZeile("Diese beiden werden nun vertauscht.");
        swap(array, i, i + 1);
        fenster.schreibe("Ergebnis dieser Runde: ");
        fenster.schreibeArray(array);
        fenster.schreibeZeile(" ");
        swapped = true;
      }
    }
  } while (swapped);
  return array;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_2/Sortieralgorithmen.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/40_Sortieralgorithmen/Aufgabe_Sortier-Vorlage.tex`

Algorithmische Komplexität (O-Notation)

Examensaufgabe „Methoden „matrixSumme()“ und „find()““ (46115-2016-H.T2-A2)

Geben Sie jeweils die kleinste, gerade noch passende Laufzeitkomplexität folgender Java-Methoden im O-Kalkül (Landau-Notation) in Abhängigkeit von n und ggf. der Länge der Arrays an.

(a)

```java
int matrixSumme(int n, int[][] feld) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      sum += feld[i][j];
    }
  }
  return sum;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java`

Lösungsvorschlag

Die Laufzeit liegt in $\mathcal{O}(n^2)$.

Begründung (nicht verlangt): Die äußere Schleife wird n-mal durchlaufen. Die innere Schleife wird dann jeweils wieder n-mal durchlaufen. Die Größe des Arrays spielt hier übrigens keine Rolle, da die Schleifen ohnehin immer nur bis zum Wert n ausgeführt werden.

(b)

```java
int find(int key, int[][] keys) {
  int a = 0, o = keys.length;
  while (o - a > 1) {
    int m = (a + o) / 2;
    if (keys[m][0] > key)
      o = m;
    else
      a = m;
  }
  return a;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java`

Lösungsvorschlag

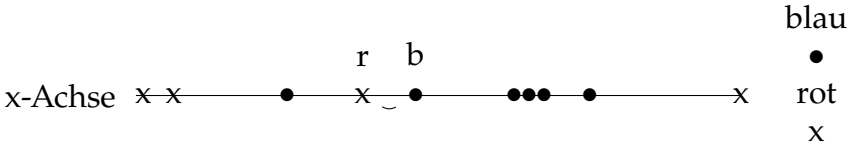
Die Laufzeit liegt in $\mathcal{O}(\log(\text{keys.length}))$. Dabei ist keys.length die Größe des Arrays bezüglich seiner ersten Dimension.

Begründung (nicht verlangt): Der Grund für diese Laufzeit ist derselbe wie bei der binären Suche. Die Größe des Arrays bezüglich seiner zweiten Dimension spielt hier übrigens keine Rolle, da diese Dimension hier ja nur einen einzigen festen Wert annimmt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/09/Thema-2/Aufgabe-2.tex`

Examensaufgabe „Nächstes rot-blaues Paar auf der x-Achse“ (46115-2020-F.T2-A6)

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x-Achse. Gesucht ist der minimale euklidische Abstand $d(r,b)$ über alle Punktepaare (r,b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstanz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x-Koordinate $A[i].x$ und eine Farbe $A[i].color \in \{\text{rot}, \text{blau}\}$. Das Feld A ist nach x-Koordinate sortiert, ðes gilt $A[1].x < A[2].x < \cdots < A[n].x$, wobei $n = |R| + |B|$.

(a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $\mathcal{O}(n)$ Zeit berechnet.

Lösungsvorschlag

Pseudo-Code

Algorithmus 1: Minimaler Euklidischer Abstand

$d_{min} := \max$; `// Setze` d_{min} `zuerst auf einen maximalen Wert.`
for *i in* $0 \ldots$ *vorletzter Index* **do** ; `// Iteriere über die Indizes des Punkte-Arrays` P `bis zum vorletzten Index` $P[n-1]$

  if $P[n].color \neq P[n+1].color$ **then** ; `// Berechne den Abstand nur, wenn die Punkte unterschiedliche Farben haben`

    $d = P[n+1].x - P[n].x$
    if $d < d_{min}$ **then**
      $d_{min} = d$
    end
  end
end

Java

```java
public double findMinimalDistance() {
  double distanceMin = Double.MAX_VALUE;
  for (int i = 0; i < latestIndex - 1; i++) {
    if (points[i].color != points[i + 1].color) {
      double distance = points[i + 1].x - points[i].x;
      if (distance < distanceMin) {
        distanceMin = distance;
      }
    }
```

```
    }
    return distanceMin;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java`

(b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge n nur einmal durchlaufen wird, ist die Laufzeit $\mathcal{O}(n)$ sichergestellt.

(c) Begründen Sie die Korrektheit Ihres Algorithmus.

Lösungsvorschlag

In d_{min} steht am Ende der gesuchte Wert (sofern nicht $d_{min} = Integer.MAX\_VALUE$ geblieben ist)

(d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Lösungsvorschlag

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von $\mathcal{O}(\log n)$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2020/03/Thema-2/Aufgabe-6.tex`

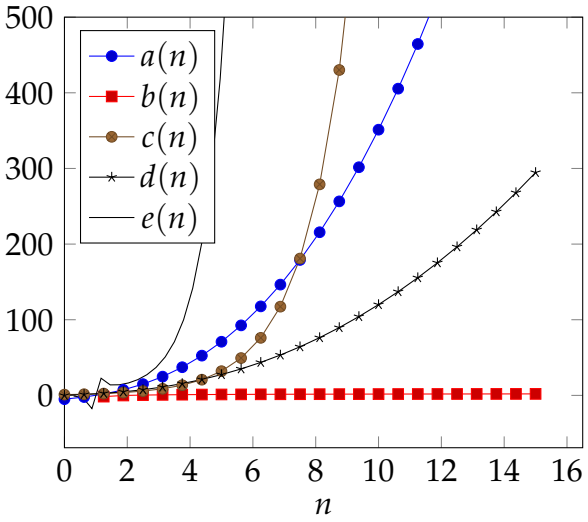
Examensaufgabe „O-Notation a(), b(), c(), d(), e(n)“ (46115-2021-F.T2-TA2-A1)

Sortieren Sie die unten angegebenen Funktionen der O-Klassen $\mathcal{O}(a)$, $\mathcal{O}(b)$, $\mathcal{O}(c)$, $\mathcal{O}(d)$ und $\mathcal{O}(e)$ bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge $\subsetneq$ sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_5. (Diese haben nichts mit den unten angegebenen Funktionen zu tun.)

$$\mathcal{O}(f_4) \subsetneq \mathcal{O}(f_3) = \mathcal{O}(f_5) \subsetneq \mathcal{O}(f_1) = \mathcal{O}(f_2)$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = \sqrt{n^5} + 4n - 5$
- $b(n) = \log_2(\log_2(n))$
- $c(n) = 2^n$
- $d(n) = n^2 \log(n) + 2n$
- $e(n) = \frac{4^n}{\log_2 n}$



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „4 Rekursionsgleichungen“ (66115-2011-F.T1-A1)

Master-Theorem

Bestimmen Sie mit Hilfe des Master-Theorems für die folgenden Rekursionsgleichungen möglichst scharfe asymptotische untere und obere Schranken, falls das Master-Theorem anwendbar ist! Geben Sie andernfalls eine kurze Begründung, warum das Master-Theorem nicht anwendbar ist!

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ($a \geq 1$).

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ($b > 1$).

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von $T(n)$ unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

(a) $T(n) = 16 \cdot T\left(\frac{n}{2}\right) + 40n - 6$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

16

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$40n - 6$

Ergibt folgende Rekursionsgleichung:

$T(n) = 16 \cdot T\left(\frac{n}{2}\right) + 40n - 6$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

für $\varepsilon = 14$:
$f(n) = 40n - 6 \in \mathcal{O}\left(n^{\log_2 16-14}\right) = \mathcal{O}\left(n^{\log_2 2}\right) = \mathcal{O}(n)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:
$f(n) = 40n - 6 \notin \Theta\left(n^{\log_2 16}\right) = \Theta\left(n^4\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$:
$f(n) = 40n - 6 \notin \Omega\left(n^{\log_2 16+\varepsilon}\right)$

$\Rightarrow T(n) \in \Theta\left(n^4\right)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

(b) $T(n) = 27 \cdot T\left(\frac{n}{3}\right) + 3n^2 \log n$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:
$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):
27

Anteil Verkleinerung des Problems (b):
um $\frac{1}{3}$ also $b = 3$

Laufzeit der rekursiven Funktion ($f(n)$):
$3n^2 \log n$

Ergibt folgende Rekursionsgleichung:
$T(n) = 27 \cdot T\left(\frac{n}{3}\right) + 3n^2 \log n$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a-\varepsilon}\right)$:
$f(n) = 3n^2 \log n = n \in \mathcal{O}\left(n^{\log_3 27-24}\right) = \mathcal{O}\left(n^{\log_3 3}\right) = \mathcal{O}(n)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:
$f(n) = 3n^2 \log n = n \notin \Theta\left(n^{\log_3 27}\right) = \Theta\left(n^3\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$:
$f(n) = 3n^2 \log n = n \notin \Omega\left(n^{\log_3 27+\varepsilon}\right)$

$\Rightarrow T(n) \in \Theta\left(n^3\right)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

(c) $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + 3n^2 + \log n$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:
$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

4

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$3n^2 + \log n$

Ergibt folgende Rekursionsgleichung:

$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + 3n^2 + \log n$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$f(n) = 3n^2 + \log n = n^2 = n \notin \mathcal{O}\left(n^{\log_2 4 - \varepsilon}\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) = 3n^2 + \log n = n^2 = n \in \Theta\left(n^{\log_2 4}\right) = \Theta\left(n^2\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$f(n) = 3n^2 + \log n = n^2 = n \notin \Omega\left(n^{\log_2 4 + \varepsilon}\right)$

$\Rightarrow T(n) \in \Theta\left(n^2 \log n\right)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

(d) $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + 100 \log n + \sqrt{2n} + n^{-2}$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

4

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$100 \log n + \sqrt{2n} + n^{-2}$

Ergibt folgende Rekursionsgleichung:

$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + 100 \log n + \sqrt{2n} + n^{-2}$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$f(n) = 100 \log n + \sqrt{2n} + n^{-2} = n \in \mathcal{O}\left(n^{\log_2 4 - 2}\right) = \mathcal{O}(n)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) = 100 \log n + \sqrt{2n} + n^{-2} = n \notin \Theta\left(n^{\log_2 4}\right) = \Theta\left(n^2\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$:

$$f(n) = 100\log n + \sqrt{2n} + n^{-2} = n \notin \Omega\left(n^{\log_2 4+\varepsilon}\right)$$

$\Rightarrow T(n) \in \Theta\left(n^2\right)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2011/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „limes“ (66115-2012-H.T2-A6)

Gegeben seien die Funktionen $f\colon \mathbb{N} \to \mathbb{N}$ und $g\colon \mathbb{N} \to \mathbb{N}$, wobei $f(n) = (n-1)^3$ und $g(n) = (2n+3)(3n+2)$. Geben Sie an, welche der folgenden Aussagen gelten. Beweisen Sie Ihre Angaben.

(a) $f(n) \in \mathcal{O}(g(n))$

(b) $g(n) \in \mathcal{O}(f(n))$

Exkurs: Regel von L'Hospital
Die Regel von de L'Hospital ist ein Hilfsmittel zum Berechnen von Grenzwerten bei Brüchen $\frac{f}{g}$ von Funktionen f und g, wenn Zähler und Nenner entweder beide gegen 0 oder beide gegen (+ oder -) unendlich gehen. Wenn in einem solchen Fall auch der Grenzwert des Bruches der Ableitungen existiert, so hat dieser denselben Wert wie der ursprüngliche Grenzwert: [a]

$$\lim_{x \to x_0} \frac{f(x)}{g(x)} = \lim_{x \to x_0} \frac{f'(x)}{g'(x)}$$

[a] `https://de.serlo.org/mathe/funktionen/grenzwerte-stetigkeit-differenzierbarkeit/grenzwert/regel-l-hospital`

Lösungsvorschlag

Es gilt Aussage (b), da $f(n) \in \mathcal{O}(n^3)$ und $g(n) \in \mathcal{O}(n^2)$ und der Grenzwert lim bei größer werdendem n gegen 0 geht. Damit wächst $f(n)$ stärker als $g(n)$, sodass nur Aussage (b) gilt und nicht (a). Dafür nutzen wir die formale Definition des $\mathcal{O}$-Kalküls, indem wir den Grenzwert $\frac{f}{g}$ bzw. $\frac{g}{f}$ bilden:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(n-1)^3}{(2n+3)(3n+2)} = \lim_{n\to\infty} \frac{3(n-1)^2}{(2n+3)\cdot 3 + 2\cdot(3n+2)} = \lim_{n\to\infty} \frac{6(n-1)}{12} = \infty$$

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{(2n+3)(3n+2)}{(n-1)^3} = \lim_{n\to\infty} \frac{(2n+3)\cdot 3 + 2\cdot(3n+2)}{3(n-1)^2} = \lim_{n\to\infty} \frac{12}{6(n-1)} = 0$$

Hinweis: Hierbei haben wir bei die Regel von L'Hospital angewendet.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/09/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Klasse „Stapel“ mit Methode „merge()““ (66115-2014-H.T2-A5)

Stapel (Stack)

Gegeben sei eine Standarddatenstruktur Stapel (Stack) mit den Operationen

- `void push(Element e)`
- `Element pop()`,
- `boolean isEmpty()`.

sowie dem Standardkonstruktor `Stapel()`, der einen leeren Stapel zur Verfügung stellt.

(a) Geben Sie eine Methode `Stapel merge(Stapel s, Stapel t)` an, die einen aufsteigend geordneten Stapel zurückgibt, unter der Bedingung, dass die beiden übergebenen Stapel aufsteigend sortiert sind, ð`S.pop()` liefert das größte Element in `s` zurück und `T.pop()` liefert das größte Element in `t` zurück. Als Hilfsdatenstruktur dürfen Sie nur Stapel verwenden, keine Felder oder Listen.

Hinweis: Nehmen Sie an, dass Objekte der Klasse `Element`, die auf dem Stapel liegen mit `compareTo()` vergleichen werden können. Zum Testen haben wir Ihnen eine Klasse `StapelTest` zur Verfügung gestellt, sie können Ihre Methode hier einfügen und testen, ob die Stapel korrekt sortiert werden. Überlegen Sie auch, was geschieht, wenn einer der Stapel (oder beide) leer ist!

Lösungsvorschlag

```
public static Stapel merge(Stapel s, Stapel t) {
  // Die beiden Stapel unsortiert aneinander hängen.
  Stapel mergedStack = new Stapel();
  while (!s.isEmpty()) {
    mergedStack.push(s.pop());
  }
  while (!t.isEmpty()) {
    mergedStack.push(t.pop());
  }
  // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
  Stapel tmpStack = new Stapel();
  while (!mergedStack.isEmpty()) {
    Element tmpElement = mergedStack.pop();
    while (!tmpStack.isEmpty() && tmpStack.top.getValue() >
↪ tmpElement.getValue()) {
      mergedStack.push(tmpStack.pop());
    }
    tmpStack.push(tmpElement);
  }
  return tmpStack;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java`

Komplette Klasse Stapel

```java
/**
 * https://www.studon.fau.de/file2860857_download.html
 */
public class Stapel {
  public Element top;

  public Stapel() {
    top = null;
  }

  /**
   * @param element Das Element, dass hinzugefügt werden soll zur Stapel.
   */
  public void push(Element element) {
    element.setNext(top);
    top = element;
  }

  /**
   * @return Das Element oder null, wenn der Stapel leer ist.
   */
  public Element pop() {
    if (top == null) {
      return null;
    }
    Element element = top;
    top = top.getNext();
    return element;
  }

  /**
   * @return Wahr wenn der Stapel leer ist.
   */
  public boolean isEmpty() {
    return top == null;
  }

  /**
   * @param s Stapel s
   * @param t Stapel t
   *
   * @return Ein neuer Stapel.
   */
  public static Stapel merge(Stapel s, Stapel t) {
    // Die beiden Stapel unsortiert aneinander hängen.
    Stapel mergedStack = new Stapel();
    while (!s.isEmpty()) {
      mergedStack.push(s.pop());
    }
    while (!t.isEmpty()) {
      mergedStack.push(t.pop());
    }
    // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
    Stapel tmpStack = new Stapel();
```

```
    while (!mergedStack.isEmpty()) {
      Element tmpElement = mergedStack.pop();
      while (!tmpStack.isEmpty() && tmpStack.top.getValue() >
↪   tmpElement.getValue()) {
        mergedStack.push(tmpStack.pop());
      }
      tmpStack.push(tmpElement);
    }
    return tmpStack;
  }

  public static void main(String[] args) {
    Stapel sa = new Stapel();
    sa.push(new Element(1));
    sa.push(new Element(2));
    sa.push(new Element(4));
    sa.push(new Element(5));
    sa.push(new Element(7));
    sa.push(new Element(8));
    Stapel sb = new Stapel();
    sb.push(new Element(2));
    sb.push(new Element(3));
    sb.push(new Element(6));
    sb.push(new Element(9));
    sb.push(new Element(10));

    Stapel sc = Stapel.merge(sa, sb);

    while (!sc.isEmpty()) {
      System.out.print(sc.pop().getValue() + ", ");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java`

Komplette Klasse Element

```
/**
 * https://www.studon.fau.de/file2860856_download.html
 */
public class Element {
  public int value;

  public Element next;

  public Element() {
    this.next = null;
  }

  public Element(int value, Element element) {
    this.value = value;
```

```
    this.next = element;
  }

  public Element(int value) {
    this.value = value;
    this.next = null;
  }

  public int getValue() {
    return value;
  }

  public Element getNext() {
    return next;
  }

  public void setNext(Element element) {
    next = element;
  }

  public int compareTo(Element element) {
    if (getValue() > element.getValue()) {
      return 1;
    } else if (element.getValue() == getValue()) {
      return 0;
    } else {
      return -1;
    }
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Element.java`

Test-Klasse

```
import static org.junit.Assert.*;
import org.junit.Test;

/**
 * https://www.studon.fau.de/file2860850_download.html
 */
public class StapelTest {

  @Test
  public void testeMethodenPushPop() {
    Stapel stapel = new Stapel();
    stapel.push(new Element(1));
    stapel.push(new Element(2));
    stapel.push(new Element(3));

    assertEquals(3, stapel.pop().value);
```

```
    assertEquals(2, stapel.pop().value);
    assertEquals(1, stapel.pop().value);
  }

  @Test
  public void testeMethodeMerge() {
    Stapel sa = new Stapel();
    sa.push(new Element(1));
    sa.push(new Element(3));
    sa.push(new Element(5));

    Stapel sb = new Stapel();
    sb.push(new Element(2));
    sb.push(new Element(4));

    Stapel sc = Stapel.merge(sa, sb);

    assertEquals(5, sc.pop().getValue());
    assertEquals(4, sc.pop().getValue());
    assertEquals(3, sc.pop().getValue());
    assertEquals(2, sc.pop().getValue());
    assertEquals(1, sc.pop().getValue());
  }

  @Test
  public void testeMethodeMergeMehrWerte() {
    Stapel sa = new Stapel();
    sa.push(new Element(1));
    sa.push(new Element(2));
    sa.push(new Element(4));
    sa.push(new Element(5));
    sa.push(new Element(7));
    sa.push(new Element(8));
    Stapel sb = new Stapel();
    sb.push(new Element(2));
    sb.push(new Element(3));
    sb.push(new Element(6));
    sb.push(new Element(9));
    sb.push(new Element(10));

    Stapel sc = Stapel.merge(sa, sb);

    assertEquals(10, sc.pop().getValue());
    assertEquals(9, sc.pop().getValue());
    assertEquals(8, sc.pop().getValue());
    assertEquals(7, sc.pop().getValue());
    assertEquals(6, sc.pop().getValue());
    assertEquals(5, sc.pop().getValue());
    assertEquals(4, sc.pop().getValue());
    assertEquals(3, sc.pop().getValue());
    assertEquals(2, sc.pop().getValue());
    assertEquals(2, sc.pop().getValue());
    assertEquals(1, sc.pop().getValue());
  }
```

```
  @Test
  public void testeMethodeMergeBLeer() {
    Stapel sa = new Stapel();
    sa.push(new Element(1));
    sa.push(new Element(3));
    sa.push(new Element(5));

    Stapel sb = new Stapel();

    Stapel sc = Stapel.merge(sa, sb);

    assertEquals(5, sc.pop().getValue());
    assertEquals(3, sc.pop().getValue());
    assertEquals(1, sc.pop().getValue());
  }

  @Test
  public void testeMethodeMergeALeer() {
    Stapel sa = new Stapel();

    Stapel sb = new Stapel();
    sb.push(new Element(2));
    sb.push(new Element(4));

    Stapel sc = Stapel.merge(sa, sb);

    assertEquals(4, sc.pop().getValue());
    assertEquals(2, sc.pop().getValue());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/StapelTest.java`

(b) Analysieren Sie die Laufzeit Ihrer Methode.

Lösungsvorschlag

Best case: $\mathcal{O}(1)$ Worst case: $\mathcal{O}(n^2)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/09/Thema-2/Aufgabe-5.tex`

Examensaufgabe „Sortieren mit Stapel“ (66115-2015-F.T2-A5)

Stapel (Stack)

Gegeben seien die Standardstrukturen Stapel (Stack) und Schlange (Queue) mit folgenden Standardoperationen:

Stapel	Schlange
`boolean isEmpty()`	`boolean isEmpty()`
`void push(int e)`	`enqueue(int e)`
`int pop()`	`int dequeue()`
`int top()`	`int head()`

Beim Stapel gibt die Operation `top()` das gleiche Element wie `pop()` zurück, bei der Schlange gibt `head()` das gleiche Element wie `dequeue()` zurück. Im Unterschied zu `pop()`, beziehungsweise `dequeue()`, wird das Element bei `top()` und `head()` nicht aus der Datenstruktur entfernt.

(a) Geben Sie in Pseudocode einen Algorithmus `sort(Stapel s)` an, der als Eingabe einen Stapel `s` mit `n` Zahlen erhält und die Zahlen in `s` sortiert. (Sie dürfen die Zahlen wahlweise entweder aufsteigend oder absteigend sortieren.) Verwenden Sie als Hilfsdatenstruktur ausschließlich eine Schlange `q`. Sie erhalten volle Punktzahl, wenn Sie außer `s` und `q` keine weiteren Variablen benutzen. Sie dürfen annehmen, dass alle Zahlen in `s` verschieden sind.

Lösungsvorschlag

```
q := neue Schlange
while s not empty:
    q.enqueue(S.pop())
while q not empty:
    while s not empty and s.top() < q.head():
        q.enqueue(s.pop())
    s.push(q.dequeue)
```

Als Java-Code

```
  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-
↪   stack/">Stapel-Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
```

```
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Sort

```
public class Sort {

  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-
↪   stack/">Stapel-Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }

  public static Stapel stapelBefüllen(int[] zahlen) {
    Stapel s = new Stapel();
    for (int i : zahlen) {
      s.push(i);
    }
    return s;
  }

  public static void zeigeStapel(Stapel s) {
    while (!s.isEmpty()) {
      System.out.print(s.pop() + " ");
    }
    System.out.println();
  }
```

```
  public static void main(String[] args) {
    Stapel s1 = stapelBefüllen(new int[] { 4, 2, 1, 5, 3 });
    sort(s1);
    zeigeStapel(s1);

    Stapel s2 = stapelBefüllen(new int[] { 1, 2, 6, 3, 9, 11, 4 });
    sort(s2);
    zeigeStapel(s2);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Schlange

```
public class Schlange {

  public Element head;

  public Schlange() {
    head = null;
  }

  public int head() {
    if (head.getNext() == null) {
      return head.getValue();
    }
    Element element = head;
    Element previous = head;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    return element.getValue();
  }

  /**
   * @param value Eine Zahl, die zur Schlange hinzugefügt werden soll.
   */
  public void enqueue(int value) {
    Element element = new Element(value);
    element.setNext(head);
    head = element;
  }

  /**
   * @return Das Element oder null, wenn der Schlange leer ist.
   */
  public int dequeue() {
    if (head.getNext() == null) {
      int result = head.getValue();
```

```
      head = null;
      return result;
    }
    Element element = head;
    Element previous = null;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    previous.setNext(null);
    return element.getValue();
  }

  /**
   * @return Wahr wenn der Schlange leer ist.
   */
  public boolean isEmpty() {
    return head == null;
  }

  public static void main(String[] args) {
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Schlange.java`

Klasse Element

```
public class Element {
  public int value;

  public Element next;

  public Element() {
    this.next = null;
  }

  public Element(int value, Element element) {
    this.value = value;
    this.next = element;
```

```
  }

  public Element(int value) {
    this.value = value;
    this.next = null;
  }

  public int getValue() {
    return value;
  }

  public Element getNext() {
    return next;
  }

  public void setNext(Element element) {
    next = element;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Element.java`

Test-Klasse

```
import static org.junit.Assert.*;

import org.junit.Test;

public class TestCase {

  @Test
  public void testeStapel() {
    Stapel s = new Stapel();
    s.push(1);
    s.push(2);
    s.push(3);

    assertEquals(false, s.isEmpty());

    assertEquals(3, s.top());
    assertEquals(3, s.pop());

    assertEquals(2, s.top());
    assertEquals(2, s.pop());

    assertEquals(1, s.top());
    assertEquals(1, s.pop());
    assertEquals(true, s.isEmpty());
  }

  @Test
```

```
  public void testeSchlange() {
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);

    assertEquals(false, s.isEmpty());

    assertEquals(1, s.head());
    assertEquals(1, s.dequeue());

    assertEquals(2, s.head());
    assertEquals(2, s.dequeue());

    assertEquals(3, s.head());
    assertEquals(3, s.dequeue());
    assertEquals(true, s.isEmpty());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/TestCase.java`

(b) Analysieren Sie die Laufzeit Ihrer Methode in Abhängigkeit von n.

Lösungsvorschlag

Zeitkomplexität: $\mathcal{O}(n^2)$, da es zwei ineinander verschachtelte `while` -Schleifen gibt, die von der Anzahl der Elemente im Stapel abhängen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/03/Thema-2/Aufgabe-5.tex`

Examensaufgabe „Methode „m()““ (66115-2018-F.T2-A6)

Master-Theorem

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ($a \geq 1$).

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ($b > 1$).

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von $T(n)$ unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

(a) Betrachten Sie die folgende Methode `m` in Java, die initial mit `m(r, 0, r.length)` für das Array `r` aufgerufen wird. Geben Sie dazu eine Rekursionsgleichung $T(n)$ an, welche die Anzahl an Rechenschritten von `m` in Abhängigkeit von der Länge `n = r.length` berechnet.

```java
public static int m(int[] r, int lo, int hi) {
  if (lo < 8 || hi <= 10 || lo >= r.length || hi > r.length) {
    throw new IllegalArgumentException();
  }

  if (hi - lo == 1) {
    return r[lo];
  } else if (hi - lo == 2) {
    return Math.max(r[lo], r[lo + 1]); // O(1)
  } else {
    int s = (hi - lo) / 3;
    int x = m(r, lo, lo + s);
    int y = m(r, lo + s, lo + 2 * s);
    int z = m(r, lo + 2 * s, hi);
    return Math.max(Math.max(x, y), 2); // O(1)
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2018/fruehjahr/MasterTheorem.java`

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

3

Anteil Verkleinerung des Problems (b):

um $\frac{1}{3}$ also $b = 3$

Laufzeit der rekursiven Funktion ($f(n)$):

$\mathcal{O}(1)$

Ergibt folgende Rekursionsgleichung:

$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \mathcal{O}(1)$

(b) Ordnen Sie die rekursive Funktion $T(n)$ aus (a) einem der drei Fälle des Mastertheorems zu und geben Sie die resultierende Zeitkomplexität an. Zeigen Sie dabei, dass die Voraussetzung des Falles erfüllt ist.

Lösungsvorschlag

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$f(n) \in \mathcal{O}\left(n^{\log_3 3 - \varepsilon}\right) = \mathcal{O}\left(n^{1-\varepsilon}\right) = \mathcal{O}(1)$ für $\varepsilon = 1$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) \notin \Theta\left(n^{\log_3 3}\right) = \Theta\left(n^1\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$f(n) \notin \Omega\left(n^{\log_3 3 + \varepsilon}\right) = \Omega\left(n^{1+\varepsilon}\right)$

Also: $T(n) \in \Theta\left(n^{\log_b a}\right)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Sortieren von O-Klassen“ (66115-2019-H.T1-A6)

(a) Sortieren Sie die unten angegebenen Funktionen der O-Klassen $\mathcal{O}(a(n))$, $\mathcal{O}(b(n))$, $\mathcal{O}(c(n))$, $\mathcal{O}(d(n))$ und $\mathcal{O}(e(n))$ bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge $\subset$ sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_5 (diese haben nichts mit den unten angegebenen Funktionen zu tun): [1]

$$\mathcal{O}(f_4(n)) \subset \mathcal{O}(f_3(n)) = \mathcal{O}(f_5(n)) \subset \mathcal{O}(f_1(n)) = \mathcal{O}(f_2(n))$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = n^2 \cdot \log_2(n) + 42$
- $b(n) = 2^n + n^4$
- $c(n) = 2^{2 \cdot n}$
- $d(n) = 2^{n+3}$
- $e(n) = \sqrt{n^5}$

Lösungsvorschlag

$$\begin{aligned} a(n) &= n^2 \cdot \log_2(n) + 42 && = n \\ b(n) &= 2^n + n^4 && = 2^n \\ c(n) &= 2^{2 \cdot n} && = 2^{2 \cdot n} \\ d(n) &= 2^{n+3} && = 2^n \\ e(n) &= \sqrt{n^5} \end{aligned}$$

$$\mathcal{O}(a(n)) \subset \mathcal{O}(e(n)) \subset \mathcal{O}(b(n)) = \mathcal{O}(d(n)) \subset \mathcal{O}(c(n))$$

$$\mathcal{O}(n^2 \cdot \log_2(n) + 42) \subset \mathcal{O}(\sqrt{n^5}) \subset \mathcal{O}(2^n + n^4) = \mathcal{O}(2^{n+3}) \subset \mathcal{O}(2^{2 \cdot n})$$

(b) Beweisen Sie die folgenden Aussagen formal nach den Definitionen der O-Notation oder widerlegen Sie sie.

(i) $\mathcal{O}(n \cdot \log_2 n) \subseteq \mathcal{O}(n \cdot (\log_2 n)^2)$

[1] `http://www.s-inf.de/Skripte/DaStru.2012-SS-Katoen.(KK).Klausur1MitLoesung.pdf`

Lösungsvorschlag Master-Theorem

Die Aussage gilt. Für $n \geq 16$ haben wir

$$(\log_2 n)^2 \leq n \Leftrightarrow \log_2 n \leq \sqrt{n}$$

und dies ist eine wahre Aussage für $n \geq 16$. Also gilt die Aussage mit $n_0 = 16$ und $c = 1$.

(ii) $2^{(n+1)} \in \mathcal{O}(n \cdot \log_2 n)$

(c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:

(i) $T(n) = 4 \cdot T(\frac{n}{2}) + n^2$

(ii) $T(n) = T(\frac{n}{2}) + \frac{n}{2}n^2 + n$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Mastertheorem“ (66115-2019-H.T2-A6)

Master-Theorem

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a-\varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$

Bestimmen und begründen Sie formal mit Hilfe dieses Satzes welche Komplexität folgende Laufzeitfunktionen haben.

(a) $T(n) = 8 \cdot T\left(\frac{n}{2}\right) + 5n^2$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

8

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$5n^2$

Ergibt folgende Rekursionsgleichung:

$T(n) = 8 \cdot T\left(\frac{n}{2}\right) + 5n^2$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a-\varepsilon}\right)$:

für $\varepsilon = 4$:

$f(n) = 5n^2 \in \mathcal{O}\left(n^{\log_2 8-4}\right) = \mathcal{O}\left(n^{\log_2 4}\right) = \mathcal{O}\left(n^2\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) = 5n^2 \notin \Theta\left(n^{\log_2 8}\right) = \Theta\left(n^3\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$:

$f(n) = 5n^2 \notin \mathcal{O}\left(n^{\log_2 8+\varepsilon}\right)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

(b) $T(n) = 9 \cdot T\left(\frac{n}{3}\right) + 5n^2$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

9

Anteil Verkleinerung des Problems (b):

um $\frac{1}{3}$ also $b = 3$

Laufzeit der rekursiven Funktion ($f(n)$):

$5n^2$

Ergibt folgende Rekursionsgleichung:

$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + 5n^2$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$f(n) = 5n^2 \notin \mathcal{O}\left(n^{\log_3 9 - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$f(n) = 5n^2 \in \Theta\left(n^{\log_3 9}\right) = \Theta\left(n^2\right)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$f(n) = 5n^2 \notin \mathcal{O}\left(n^{\log_3 9 + \varepsilon}\right)$ für $\varepsilon > 0$

$\Rightarrow T(n) \in \Theta\left(n^2 \cdot \log n\right)$

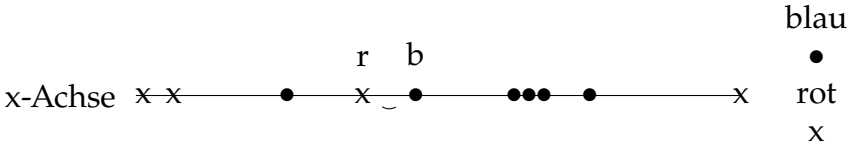
Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Nächstes rot-blaues Paar auf der x-Achse“ (66115-2020-F.T2-A8)

Algorithmische Komplexität (O-Notation)

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x-Achse. Gesucht ist der minimale euklidische Abstand $d(r,b)$ über alle Punktepaare (r,b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstanz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x-Koordinate $A[i].x$ und eine Farbe $A[i].color \in \{\text{rot}, \text{blau}\}$. Das Feld A ist nach x-Koordinate sortiert, ðes gilt $A[1].x < A[2].x < \cdots < A[n].x$, wobei $n = |R| + |B|$.

(a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $\mathcal{O}(n)$ Zeit berechnet.

Lösungsvorschlag

Pseudo-Code

Algorithmus 2: Minimaler Euklidischer Abstand

$d_{min} := \max$; `// Setze` d_{min} `zuerst auf einen maximalen Wert.`
for *i in* $0 \ldots$ *vorletzter Index* **do** ; `// Iteriere über die Indizes des Punkte-Arrays` P `bis zum vorletzten Index` $P[n-1]$

  if $P[n].color \neq P[n+1].color$ **then** ; `// Berechne den Abstand nur, wenn die Punkte unterschiedliche Farben haben`

    $d = P[n+1].x - P[n].x$
    if $d < d_{min}$ **then**
      $d_{min} = d$
    end
  end
end

Java

```java
public double findMinimalDistance() {
  double distanceMin = Double.MAX_VALUE;
  for (int i = 0; i < latestIndex - 1; i++) {
    if (points[i].color != points[i + 1].color) {
      double distance = points[i + 1].x - points[i].x;
      if (distance < distanceMin) {
        distanceMin = distance;
      }
    }
```

```
    }
    return distanceMin;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java`

(b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge n nur einmal durchlaufen wird, ist die Laufzeit $\mathcal{O}(n)$ sichergestellt.

(c) Begründen Sie die Korrektheit Ihres Algorithmus.

Lösungsvorschlag

In d_{min} steht am Ende der gesuchte Wert (sofern nicht $d_{min} = Integer.MAX\_VALUE$ geblieben ist)

(d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Lösungsvorschlag

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von $\mathcal{O}(\log n)$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-2/Aufgabe-8.tex`

Examensaufgabe „O-Notation“ (66115-2020-H.T1-TA2-A4)

Algorithmische Komplexität (O-Notation)

(a) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int a = 0, b = 0;
  int i = 0;
  while (i < n) {
    a = b + i;
    b = a;
    i = i + 1;
  }
  return a;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery1.java`

Bestimmen Sie die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

Lösungsvorschlag

Die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation ist $\mathcal{O}(n)$.

Die `while`-Schleife wird genau n mal ausgeführt. In der Schleife wird die Variable `i` in der Zeile `i = i + 1;` inkrementiert. `i` wird mit 0 initialisiert. Die `while`-Schleife endet, wenn `i` gleich groß ist als `n`.

(b) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int r = 0;
  while (n > 0) {
    int y = n;
    int x = n;
    for (int i = 0; i < y; i++) {
      for (int j = 0; j < i; j++) {
        r = r + 1;
      }
      r = r - 1;
    }
    n = n - 1;
  }
  return r;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery2.java`

Bestimmen Sie für das Code-Beispiel die asymptotische worst-case Laufzeit in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

Lösungsvorschlag Master-Theorem

`while`: n-mal

1. `for`: $n, n-1, \ldots, 2, 1$

2. `for`: $1, 2, \ldots, n-1, n$

$n \times n \times n = \mathcal{O}(n^3)$

(c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:

$$T(n) = T\left(\frac{n}{2}\right) + \tfrac{1}{2}n^2 + n$$

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ($a \geq 1$).

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ($b > 1$).

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von $T(n)$ unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

1

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$\frac{1}{2}n^2 + n$

Ergibt folgende Rekursionsgleichung:

$T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$

Nebenrechnung: $\log_b a = \log_2 1 = 0$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$\frac{1}{2}n^2 + n \notin \mathcal{O}\left(n^{-1}\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$\frac{1}{2}n^2 + n \notin \Theta(1)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$\varepsilon = 2$

$\frac{1}{2}n^2 + n \in \Omega\left(n^2\right)$

Für eine Abschätzung suchen wir eine Konstante, damit gilt:

$1 \cdot f(\frac{n}{2}) \leq c \cdot f(n)$

$\frac{1}{2} \cdot \frac{1}{4}n^2 + \frac{1}{2}n \leq c \cdot (\frac{1}{2} \cdot n^2 + n)$

Damit folgt $c = \frac{1}{4}$

und $0 < c < 1$

$\Rightarrow \Theta(\frac{1}{2}n^2 + n)$

$\Rightarrow \Theta(n^2)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-4.tex`

Übungsaufgabe „Algorithmen-Vergleich“ (Algorithmische Komplexität (O-Notation))

Seien **A** und **B** zwei Algorithmen, die dasselbe Problem lösen. Zur Lösung von Problemen der Eingabegröße n benötigt Algorithmus **A** $500 \cdot n^2 - 16 \cdot n$ Elementoperationen und **B** $\frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$ Elementoperationen.

$$A(n) = 500 \cdot n^2 - 16 \cdot n$$
$$B(n) = \frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$$

(a) Wenn Sie ein Problem für die Eingabegröße 256 lösen wollen, welchen Algorithmus würden Sie dann wählen?

Lösungsvorschlag

Wir können die Aufgabe durch Einsetzen lösen, ðwir berechnen explizit die Anzahl benötigter Elementoperationen und vergleichen. Algorithmus A benötigt $500 \cdot n2 - 16 \cdot n$ Operationen bei einer Eingabe der Größe n, also bei $n = 256$ genau

$$\begin{aligned} A(256) &= 500 \cdot 256^2 - 16 \cdot 256 \\ &= 32763904 \end{aligned}$$

In der gleichen Art können wir den Aufwand von Algorithmus B berechnen:

$$\begin{aligned} B(256) &= \frac{1}{2} \cdot 256^3 + \frac{11}{2} \cdot 256 + 7 \\ &= 8390023 \end{aligned}$$

In diesem Fall benötigt Algorithmus B also deutlich weniger Elementoperationen.

(b) Wenn Sie ein Problem lösen wollen, deren Eingabegröße immer mindestens 1024 ist, welchen Algorithmus würden Sie wählen? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Da in der Aufgabenstellung von Eingaben der Größe mindestens 1024 die Rede ist, stellen wir uns die Frage, für welche n Algorithmus A schneller als B ist, also für welche n

$$500 \cdot n^2 - 16 \cdot n < \frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$$

gilt. Dies lässt sich äquivalent umformen zu

$$\frac{1}{2} \cdot n^3 - 500 \cdot n^2 + \frac{43}{2} \cdot n + 7 > 0$$

Diese Ungleichung ist erfüllt, wenn allein $\frac{1}{2} \cdot n^3 - 500 \cdot n^2 > 0$ gilt, denn es gilt $\frac{43}{2} \cdot n + 7 > 0$. Das Problem reduziert sich also zu

$$\frac{1}{2} \cdot n^3 - 500 \cdot n^2 > 0 \Leftrightarrow n > 1000$$

Auf jeden Fall ist A schneller als B für $n > 1000$ (man erinnere sich, dass das bei n = 256 noch andersherum war). Wie ist dieses Verhalten zu erklären?

Obwohl der Aufwand von A im O-Kalkül $\mathcal{O}(n^2)$ und der von B $\mathcal{O}(n^3)$ ist, man also geneigt sein könnte zu sagen, A ist immer schneller als B, stimmt das nicht immer. Im Einzelfall können es durchaus große Konstanten (wie in diesem Fall die 500) sein, die dafür sorgen, dass n erst einmal sehr groß werden muss, damit sich die Laufzeiten tatsächlich so verhalten, wie erwartet. Wenn nur kleine Eingaben verarbeitet werden sollen, kann es manchmal also durchaus lohnenswert sein, einen $\mathcal{O}(n^3)$-Algorithmus anstatt eines $\mathcal{O}(n^3)$-Algorithmus zu verwenden, wenn die im O-Kalkül unterschlagenen Konstanten zuungunsten des eigentlich langsameren Algorithmus sprechen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_Algorithmen-Vergleich.tex`

Übungsaufgabe „Klasse-QueueElement“ (Algorithmische Komplexität (O-Notation))

Der Konstruktor `QueueElement(...)` und die Methode `setNext(...)` sowie `getNext(...)` haben $\mathcal{O}(1)$. Geben Sie die Zeitkomplexität der Methode `append(int content)` an, die einer Schlange ein neues Element anhängt.

```
public void append(int contents) {
QueueElement newElement = new QueueElement(contents) ;
  if (first == 0) {
    first = newElement;
    last = newElement;
  } else {
    // Ein neues Element hinten anhängen.
    last.setNext(newElement);
    // Das angehängte Element als Letztes setzen.
    last = last.getNext();
  }
}
```

Lösungsvorschlag

Das Anhängen eines neuen Elements in die gegebene Warteschlange hat die konstanten Rechenzeitbedarf von $\mathcal{O}(1)$, egal wie lange die Schlange ist, da wir das letzte Element direkt ansprechen können.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_Klasse-QueueElement.tex

Übungsaufgabe „Methode „magicStaff()““ (Algorithmische Komplexität (O-Notation))

Algorithmische Komplexität (O-Notation)

Welche Komplexität hat das Programmfragment?

```
public void magicStaff(int[] array) {
  for (int i = 0; i < array.length; i++) {
    int counter = 0;
    if (array[i] % 3 == 0) {
      break;
    }
    do {
      if (array[i] % 2 == 0) {
        array[i] += array[counter];
      }
    } while (counter++ < array.length);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/komplexitaet/Komplexitaet.java`

Bestimmen Sie in Abhängigkeit von n die Komplexität des Programmabschnitts im

(a) Best-Case.

Lösungsvorschlag

$\mathcal{O}(1)$: Wenn die erste Zahl im Feld `array` ohne Rest durch 3 teilbar ist, wird sofort aus der for-Schleife ausgestiegen (wegen der `break` Anweisung).

(b) Worst-Case.

Lösungsvorschlag

$\mathcal{O}(n^2)$: Wenn keine Zahl aus `array` ohne Rest durch 3 teilbar ist, werden zwei Schleifen (`for` und `do while`) über die Anzahl `n` der Elemente des Felds durchlaufen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_Methode-magicStaff.tex`

Übungsaufgabe „Polynome f(n) in g(n)“ (Algorithmische Komplexität (O-Notation))

Gegeben sind die zwei Funktionen. Gilt $f(n) \in \Theta(g(n))$? [2]

$$f(n) = 3n^5 + 4n^3 + 15$$
$$g(n) = n^5$$

Lösungsvorschlag

(a) Zu zeigen: $f(n) \in \mathcal{O}(g(n)) \Leftrightarrow (\exists c, n_0 > 0 \forall n_0 \geq n_0 : 3n^5 + 4n^3 + 15 \leq c \cdot n^5)$
Wähle z. B. $c = 3 + 4 + 15 = 22$,
dann gilt $\forall n \geq 1 : 3n^5 + 4n^3 + 15 \leq 3n^5 + 4n^5 + 15n^5 = \leq 22n^5$
$\Rightarrow f(n) \in \mathcal{O}(n^5)$

(b) Zu zeigen: $f(n) \in \Omega(g(n)) \Leftrightarrow (\exists c', n_0 > 0 \forall n_0 \geq n_0 : 3n^5 + 4n^3 + 15 \leq c' \cdot n^5)$
Wähle z. B. $c' = 3$,
dann gilt $\forall n \geq 1 : 3n^5 = +4n^3 + 15 \leq 3n^5$
$\Rightarrow f(n) \in \Omega(n^5)$

$\Rightarrow f(n) \in \Theta(g(n))$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_Polynome.tex`

[2] `https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/teaching/archive/SS17/ue_algodat/schaefer01.pdf`

Übungsaufgabe „mehrere Funktionen“ (Algorithmische Komplexität (O-Notation))

Geben Sie die Komplexität folgender Funktionen in der $\mathcal{O}$-Notation an!

(a) $x(n) = 4 \cdot n$

Lösungsvorschlag

$\mathcal{O}(n)$

(b) $a(n) = n^2$

Lösungsvorschlag

$\mathcal{O}(n^2)$

(c) $k(n) = 5 + n$

Lösungsvorschlag

$\mathcal{O}(n)$

(d) $p(n) = 4$

Lösungsvorschlag

$\mathcal{O}(1)$

(e) $j(n) = 4^n$

Lösungsvorschlag

$\mathcal{O}(4^n)$

(f) $b(n) = \frac{n}{8}$

Lösungsvorschlag

$\mathcal{O}(n)$

(g) $m(n) = \frac{1}{n}$

Lösungsvorschlag

$\mathcal{O}(1)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_mehrere-Funktionen.tex`

Master-Theorem

Algorithmenmuster

Examensaufgabe „Quicksort“ (46114-2008-H.T2-A3)

Quicksort ist ein Sortierungsverfahren, das nach dem Divide-and-Conquer-Prinzip (Teile und Herrsche) arbeitet. Wir betrachten im Folgenden die Anwendung dieses Verfahrens zum Sortieren von Integerzahlen. Die Sortierung soll in aufsteigender Reihenfolge der Werte erfolgen. Wir nehmen dabei an, dass die zu sortierenden Zahlen in einem Feld fester Länge abgelegt sind.

(a) Beschreiben Sie die Arbeitsweise des Divide-and-Conquer-Prinzips im allgemeinen Fall. Geben Sie dabei die Bedeutung der Schritte divide, conquer und combine an.

(b) Beschreiben Sie die Arbeitsweise des Algoritumus Quicksort. Geben Sie dabei an, worin die Schritte divide, conquer und combine im konkreten Fall bestehen.

(c) Geben Sie in C, C++ oder Java eine Implementierung des Algorithmus Quicksort an. Formulieren Sie die Implementierung als rekursive Funktion quicksort() und verwenden Sie das jeweils erste Element. des (Teil-)Feldes für die Aufteilung Verwenden Sie für Ihre Implementierung von quicksort() «lei Parameter:

   (i) das Feld, in dem die zu sortierenden Zahlen abgelogt sind:

   (ii) den Index des am weitesten links gelegenen Elementes des zu sortierenden Teilfeldes;

   (iii) den Index des am weitesten rechts gelegenen Elementes des zu sortierenden Teilfeldes.

   Erläutern Sie die Arbeitsweise Ihrer Implementierung. Kennzeichnen Sie die Schritte divide, conquer und combine des zugrundeliegenden Divide-and-Conquer-Prinzips.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46114/2008/09/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Methode „a()““ (46115-2016-H.T2-A4)

Mittels Dynamischer Programmierung (auch Memoization genannt) kann man insbesondere rekursive Lösungen auf Kosten des Speicherbedarf beschleunigen, indem man Zwischenergebnisse „abspeichert“ und bei (wiederkehrendem) Bedarf „abruft“, ohne sie erneut berechnen zu müssen.

Gegeben sei folgende geschachtelt-rekursive Funktion für $n, m \geq 0$:

$$a(n,m) = \begin{cases} n + \lfloor \frac{n}{2} \rfloor & \text{falls } m = 0 \\ a(1, m-1), & \text{falls } n = 0 \wedge m \neq 0 \\ a(n + \lfloor \sqrt{a(n-1,m)} \rfloor, m-1), & \textit{sonst} \end{cases}$$

(a) Implementieren Sie die obige Funktion `a(n,m)` zunächst ohne weitere Optimierungen als Prozedur/Methode in einer Programmiersprache Ihrer Wahl.

Lösungsvorschlag

```java
public static long a(int n, int m) {
  if (m == 0) {
    return n + (n / 2);
  } else if (n == 0 && m != 0) {
    return a(1, m - 1);
  } else {
    return a(n + ((int) Math.sqrt(a(n - 1, m))), m - 1);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java`

(b) Geben Sie nun eine DP-Implementierung der Funktion `a(n,m)` an, die `a(n,m)` für $0 \geq n \geq 100000$ und $0 \geq m \geq 25$ höchstens einmal gemäß obiger rekursiver Definition berechnet. Beachten Sie, dass Ihre Prozedur trotzdem auch weiterhin mit $n > 100000$ und $m > 25$ aufgerufen werden können soll.

Lösungsvorschlag

```
static long[][] tmp = new long[100001][26];

public static long aDp(int n, int m) {
  if (n <= 100000 && m <= 25 && tmp[n][m] != -1) {
    return tmp[n][m];
  } else {
    long merker;
    if (m == 0) {
      merker = n + (n / 2);
    } else if (n == 0 && m != 0) {
      merker = aDp(1, m - 1);
    } else {
      merker = aDp(n + ((int) Math.sqrt(aDp(n - 1, m))), m - 1);
    }
    if (n <= 100000 && m <= 25) {
      tmp[n][m] = merker;
    }
```

```
      return merker;
    }
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java

Lösungsvorschlag

Kompletter Code

```
public class DynamischeProgrammierung {
  public static long a(int n, int m) {
    if (m == 0) {
      return n + (n / 2);
    } else if (n == 0 && m != 0) {
      return a(1, m - 1);
    } else {
      return a(n + ((int) Math.sqrt(a(n - 1, m))), m - 1);
    }
  }

  static long[][] tmp = new long[100001][26];

  public static long aDp(int n, int m) {
    if (n <= 100000 && m <= 25 && tmp[n][m] != -1) {
      return tmp[n][m];
    } else {
      long merker;
      if (m == 0) {
        merker = n + (n / 2);
      } else if (n == 0 && m != 0) {
        merker = aDp(1, m - 1);
      } else {
        merker = aDp(n + ((int) Math.sqrt(aDp(n - 1, m))), m - 1);
      }
      if (n <= 100000 && m <= 25) {
        tmp[n][m] = merker;
      }
      return merker;
    }
  }

  public static void main(String[] args) {
    for (int i = 0; i < 100001; i++) {
      for (int j = 0; j < 26; j++) {
        tmp[i][j] = -1;
      }
    }
    System.out.println("schnell mit DP: " + aDp(7,7));
    System.out.println("langsam ohne DP: " + a(7,7));
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/09/Thema-2/Aufgabe-4.tex

Examensaufgabe „Primzahl“ (46115-2017-H.T2-A3)

Die Methode `pKR` berechnet die n-te Primzahl ($n \geq 1$) kaskadenartig rekursiv und äußerst ineffizient:

```
static long pKR(int n) {
  long p = 2;
  if (n >= 2) {
    p = pKR(n - 1); // beginne die Suche bei der vorhergehenden Primzahl
    int i = 0;
    do {
      p++; // pruefe, ob die jeweils naechste Zahl prim ist, d.h. ...
      for (i = 1; i < n && p % pKR(i) != 0; i++) {
      } // pruefe, ob unter den kleineren Primzahlen ein Teiler ist
    } while (i != n); // ... bis nur noch 1 und p Teiler von p sind
  }
  return p;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java`

Überführen Sie `pKR` mittels *dynamischer Programmierung* (hier also *Memoization*) und mit möglichst *wenigen Änderungen* so in die *linear* rekursive Methode `pLR`, dass `pLR(n, new long[n + 1])` ebenfalls die n-te Primzahl ermittelt:

```
private long pLR(int n, long[] ps) {
  ps[1] = 2;
  // ...
}
```

Lösungsvorschlag

Lösungsvorschlag

> **Exkurs: Kaskadenartig rekursiv**
> Kaskadenförmige Rekursion bezeichnet den Fall, in dem mehrere rekursive Aufrufe nebeneinander stehen.

Lösungsvorschlag

> **Exkurs: Linear rekursiv**
> Die häufigste Rekursionsform ist die lineare Rekursion, bei der in jedem Fall der rekursiven Definition höchstens ein rekursiver Aufruf vorkommen darf.

```
static long pLR(int n, long[] ps) {
  ps[1] = 2;
  long p = 2;
  if (ps[n] != 0) // Fall die Primzahl bereits gefunden / berechnet wurde,
    return ps[n]; // gib die berechnet Primzahl zurück.
  if (n >= 2) {
    // der einzige rekursive Aufruf steht hier, damit die Methode linear
↪   rekursiv
    // ist.
```

```
      p = pLR(n - 1, ps);
      int i = 0;
      do {
        p++;
        // Hier wird auf das gespeicherte Feld zurückgegriffen.
        for (i = 1; i < n && p % ps[i] != 0; i++) {
        }
      } while (i != n);
    }
    ps[n] = p; // Die gesuchte Primzahl im Feld speichern.
    return p;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java`

Der komplette Quellcode

```
/**
 * Berechne die n-te Primzahl.
 *
 * Eine Primzahl ist eine natürliche Zahl, die größer als 1 und ausschließlich
 * durch sich selbst und durch 1 teilbar ist.
 *
 * <ul>
 * <li>1. Primzahl: 2
 * <li>2. Primzahl: 3
 * <li>3. Primzahl: 5
 * <li>4. Primzahl: 7
 * <li>5. Primzahl: 11
 * <li>6. Primzahl: 13
 * <li>7. Primzahl: 17
 * <li>8. Primzahl: 19
 * <li>9. Primzahl: 23
 * <li>10. Primzahl: 29
 * </ul>
 */
public class PrimzahlDP {

  /**
   * Die Methode pKR berechnet die n-te Primzahl ({@code n >= 1}) Kaskadenartig
↪  Rekursiv.
   *
   * @param n Die Nummer (n-te) der gesuchten Primzahl. Die Primzahl 2 ist die
   *          erste Primzahl. Die Primzahl 3 ist die zweite Primzahl etc.
   *
   * @return Die gesuchte n-te Primzahl.
   */
  static long pKR(int n) {
    long p = 2;
    if (n >= 2) {
      p = pKR(n - 1); // beginne die Suche bei der vorhergehenden Primzahl
      int i = 0;
      do {
```

```
        p++; // pruefe, ob die jeweils naechste Zahl prim ist, d.h. ...
        for (i = 1; i < n && p % pKR(i) != 0; i++) {
        } // pruefe, ob unter den kleineren Primzahlen ein Teiler ist
      } while (i != n); // ... bis nur noch 1 und p Teiler von p sind
    }
    return p;
  }

  /**
   * Die Methode pLR berechnet die n-te Primzahl ({@code n >= 1}) Linear Rekursiv.
   *
   * @param n  Die Nummer (n-te) der gesuchten Primzahl. Die Primzahl 2 ist die
   *           erste Primzahl. Die Primzahl 3 ist die zweite Primzahl etc.
   * @param ps Primzahl Speicher. Muss mit n + 1 initialisert werden.
   *
   * @return Die gesuchte n-te Primzahl.
   */
  static long pLR(int n, long[] ps) {
    ps[1] = 2;
    long p = 2;
    if (ps[n] != 0) // Fall die Primzahl bereits gefunden / berechnet wurde,
      return ps[n]; // gib die berechnet Primzahl zurück.
    if (n >= 2) {
      // der einzige rekursive Aufruf steht hier, damit die Methode linear
↪   rekursiv
      // ist.
      p = pLR(n - 1, ps);
      int i = 0;
      do {
        p++;
        // Hier wird auf das gespeicherte Feld zurückgegriffen.
        for (i = 1; i < n && p % ps[i] != 0; i++) {
        }
      } while (i != n);
    }
    ps[n] = p; // Die gesuchte Primzahl im Feld speichern.
    return p;
  }

  static void debug(int n) {

↪   System.out.println(String.format("%d. Primzahl: %d (kaskadenartig rekursiv berechnet)",
↪   n, pKR(n)));

↪   System.out.println(String.format("%d. Primzahl: %d (linear rekursiv berechnet)",
↪   n, pLR(n, new long[n + 1])));
  }

  public static void main(String[] args) {
    System.out.println(pKR(10));
    System.out.println(pLR(10, new long[11]));

    for (int i = 1; i <= 10; i++) {
      debug(i);
```

```
      }
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java`

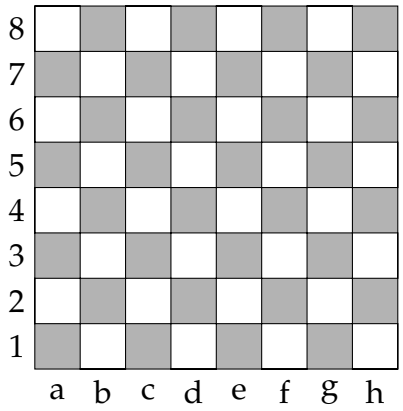
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2017/09/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Springerproblem beim Schach“ (46115-2018-H.T2-A5)

Backtracking

Das *Springerproblem* ist ein kombinatorisches Problem, das darin besteht, für einen Springer auf einem leeren Schachbrett eine Route von einem gegebenen Startfeld aus zu finden, auf der dieser jedes Feld des Schachbretts genau einmal besucht.

Ein Schachbrett besteht aus 8×8 Feldern. Ein Springer kann bei einem Zug von einem Ausgangsfeld aus eines von maximal 8 Folgefelder betreten, wie dies in der folgenden Abbildung dargestellt ist. Der Springer darf selbstverständlich nicht über den Rand des Schachbretts hinausspringen.



Eine Lösung des Springerproblems mit Startfeld `h1` sieht wie folgt aus. Die Felder sind in ihrer Besuchsreihenfolge durchnummeriert. Der Springer bewegt sich also von `h1` nach `f2`, dann von `f2` nach `h3` usw.

41	10	29	26	49	12	31	16
28	25	40	11	30	15	50	13
9	42	27	56	61	48	17	32
24	39	58	47	64	53	14	51
43	8	55	62	57	60	33	18
38	23	46	59	54	63	52	3
7	44	21	36	5	2	19	34
22	37	6	45	20	35	4	1

Formulieren Sie einen rekursiven Algorithmus zur Lösung des Springerproblems von einem vorgegebenen Startfeld aus. Es sollen dabei alle möglichen Lösungen des Springerproblems gefunden werden. Die Lösungen sollen durch Backtracking gefunden werden. Hierbei werden alle möglichen Teilrouten systematisch durchprobiert, und Teilrouten, die nicht zu einer Lösung des Springerproblems führen können, werden nicht weiterverfolgt. Dies ist durch rekursiven Aufruf einer Lösungsfunktion `huepf(z, y, z)` zu realisieren, wobei

- `x` und `y` die Koordinaten des als nächstes anzuspringenden Feldes sind, und
- `z` die aktuelle Rekursionstiefe enthält. Wenn die Rekursionstiefe 64 erreicht und das betreffende Feld noch unbesucht ist, ist eine Lösung des Springerproblems gefunden.

Der initiale Aufruf Ihres Algorithmus kann beispielsweise über den Aufruf

```
huepf(1, 8, 1)
```

erfolgen.

Wählen Sie geeignete Datenstrukturen zur Verwaltung der unbesuchten Felder und zum Speichern gefundener (Teil)Lösungen. Der Algorithmus soll eine gefundene Lösung in der oben angegebenen Form ausdrucken, also als Matrix mit der Besuchsreihenfolge pro Feld.

Lösungsvorschlag

```
/**
 * Nach <a href=
 * "https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-
↪  1">geeksforgeeks.org</a>
 */
public class Springerproblem {
  static int felderAnzahl = 8;

  static int lösung[][];

  static int xBewegungen[] = { 2, 1, -1, -2, -2, -1, 1, 2 };
  static int yBewegungen[] = { 1, 2, 2, 1, -1, -2, -2, -1 };

  static void druckeLösung() {
    for (int x = 0; x < felderAnzahl; x++) {
      for (int y = 0; y < felderAnzahl; y++) {
        System.out.print(lösung[x][y] + " ");
      }
      System.out.println();
    }
  }

  /**
   * Versuche zum angegeben Feld zu hüpfen.
   *
   * @param x Die x-Koordinate des als nächstes anzuspringenden Feldes.
   * @param y Die y-Koordinate des als nächstes anzuspringenden Feldes.
   * @param z Die aktuelle Rekursionstiefe.
   *
   * @return Wahr wenn das „angehüpfte" Feld in der Lösung ist, sonst falsch.
   */
  static boolean huepf(int x, int y, int z) {
    // nächste x-Koordinate
    int xN;
    // nächste y-Koordinate
    int yN;
    if (z == felderAnzahl * felderAnzahl) {
```

```
      return true;
    }

    for (int i = 0; i < 8; i++) {
      xN = x + xBewegungen[i];
      yN = y + yBewegungen[i];
      if (xN >= 0 && xN < felderAnzahl && yN >= 0 && yN < felderAnzahl &&
↪   lösung[xN][yN] == -1) {
        lösung[xN][yN] = z;
        if (huepf(xN, yN, z + 1)) {
          return true;
        } else {
          // backtracking
          lösung[xN][yN] = -1;
        }
      }
    }
    return false;
  }

  static boolean löseSpringerproblem(int x, int y) {
    lösung = new int[8][8];

    for (int i = 0; i < felderAnzahl; i++) {
      for (int j = 0; j < felderAnzahl; j++) {
        lösung[i][j] = -1;
      }
    }

    lösung[x][y] = 0;

    if (!huepf(x, y, 1)) {
      System.out.println("Es konnte keine Lösung gefunden werden.");
      return false;
    } else {
      druckeLösung();
    }

    return true;
  }

  public static void main(String args[]) {
    löseSpringerproblem(0, 0);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2018/herbst/Springerproblem.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2018/09/Thema-2/Aufgabe-5.tex`

Examensaufgabe „maximale Teilsumme“ (66115-2012-H.T1-A4)

Teile-und-Herrsche (Divide-and-Conquer)

Gegeben ist ein Array a von ganzen Zahlen der Länge n, z. B. :

i	0	1	2	3	4	5	6	7	8	9
a_i	5	-6	4	2	-5	7	-2	-7	3	5

Im Beispiel ist also $n = 10$. Es soll die maximale Teilsumme berechnet werden, also der Wert des Ausdrucks

$$\max_{i,j \leq n} \sum_{k=1}^{j-1} a_k$$

Im Beispiel ist dieser Wert 8 und wird für $i = 8, j = 10$ erreicht. Entwerfen Sie ein Divide-And-Conquer Verfahren, welches diese Aufgabenstellung in Zeit $\mathcal{O}(n \log n)$ löst. Skizzieren Sie Ihre Lösung hinreichend detailliert.

Tipp: Sie sollten ein geringfügig allgemeineres Problem lösen, welches neben der maximalen Teilsumme auch noch die beiden „maximalen Randsummen” berechnet. Die werden dann bei der Endausgabe verworfen.

```
/**
 * Klasse zur Berechnung der maximalen Teilsumme einer Zahlenfolge.
 *
 * nach Teilsumme.java Klasse mit Algorithmen für die Berechnung des größten
 * gemeinsamen Teilers zweier Ganzzahlen Algorithmen und Datenstrukturen,
 * Auflage 4, Kapitel 2.1
 *
 * nach Prof. Grude, Prof. Solymosi, (c) 2000-2008: 22. April 2008
 * <a href="http://public.beuth-hochschule.de/oo-
↪  plug/A&D/prog/kap21/Teilsumme.java">Teilsumme.java</a>
 */
public class Teilsumme {

  /**
   * Berechne die maximale Teilsumme an der rechten Grenze. Die Eingabeparameter
   * müssen diese Werte aufweisen: 0 &lt;= links &lt;= rechts &lt; folge.length.
   *
   * @param folge  Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
   *               werden soll.
   * @param links  Die Index-Nummer der linken Grenze.
   * @param rechts Die Index-Nummer der rechten Grenze.
   *
   * @return Die maximale Teilsumme.
   */
  private static int berechneRandRechts(int[] folge, int links, int rechts) {
    int max = 0;
    int sum = 0;
    for (int i = rechts; i >= links; i--) {
      sum += folge[i];
      max = Math.max(max, sum);
    }
    return max;
```

```
  }

  /**
   * Berechne die maximale Teilsumme an der linken Grenze. Die Eingabeparameter
   * müssen diese Werte aufweisen: 0 &lt;= links &lt;= rechts &lt; folge.length.
   *
   * @param folge  Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
   *               werden soll.
   * @param links  Die Index-Nummer der linken Grenze.
   * @param rechts Die Index-Nummer der rechten Grenze.
   *
   * @return Die maximale Teilsumme.
   */
  private static int berechneRandLinks(int[] folge, int links, int rechts) {
    int max = 0;
    int sum = 0;
    for (int i = links; i <= rechts; i++) {
      sum += folge[i];
      max = Math.max(max, sum);
    }
    return max;
  }

  /**
   * Berechne die maximale Teilsumme in der Zahlenfolge zwischen einer gegeben
   * linken und rechten Grenze. Die Eingabeparameter müssen diese Werte aufweisen:
   * 0 &lt;= links &lt;= rechts &lt; folge.length.
   *
   * @param folge  Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
   *               werden soll.
   * @param links  Die Index-Nummer der linken Grenze.
   * @param rechts Die Index-Nummer der rechten Grenze.
   *
   * @return Die maximale Teilsumme.
   */
  private static int berechne(int[] folge, int links, int rechts) {
    if (links == rechts) // nur ein Element
      return Math.max(0, folge[links]);
    else {
      final int mitte = (rechts + links) / 2;
      final int maxLinks = berechne(folge, links, mitte);
      final int maxRechts = berechne(folge, mitte + 1, rechts);
      final int maxGrenzeRechts = berechneRandRechts(folge, links, mitte);
      // linke Hälfte
      final int maxGrenzeLinks = berechneRandLinks(folge, mitte + 1, rechts);
      // rechte Hälfte
      return Math.max(maxRechts, Math.max(maxLinks, maxGrenzeRechts +
↪   maxGrenzeLinks));
    }
  }

  /**
   * Berechne die maximale Teilsumme einer Zahlenfolge rekursiv mit
   * logarithmischer Zeitkomplexität.
   *
```

```java
   * @param folge Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
   *              werden soll.
   *
   * @return Die maximale Teilsumme.
   */
  public static int berechne(int[] folge) {
    return berechne(folge, 0, folge.length - 1);
  }

  public static void main(String[] args) {
    int[] folge = { 5, -6, 4, 2, -5, 7, -2, -7, 3, 5 };
    int ergebnis = berechne(folge);
    System.out.println(ergebnis);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2012/herbst/Teilsumme.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Zahl der Inversionen von A“ (66115-2016-H.T1-A4)

Teile-und-Herrsche (Divide-and-Conquer)

Es sei $A[0 \ldots n-1]$ ein Array von paarweise verschiedenen ganzen Zahlen.

Wir interessieren uns für die Zahl der Inversionen von A; das sind Paare von Indices (i,j), sodass $i < j$ aber $A[i] > A[j]$. Die Inversionen im Array $[2,3,8,6,1]$ sind $(0,4)$, da $A[0] > A[4]$ und weiter $(1,4)$, $(2,3)$, $(2,4)$, $(3,4)$. Es gibt also 5 Inversionen.

(a) Wie viel Inversionen hat das Array $[3,7,1,4,5,9,2]$?

Lösungsvorschlag

- $(0,1)$: $3 > 1$
- $(0,6)$: $3 > 2$
- $(1,2)$: $7 > 1$
- $(1,3)$: $7 > 4$
- $(1,4)$: $7 > 5$
- $(1,6)$: $7 > 2$
- $(3,6)$: $4 > 2$
- $(4,6)$: $5 > 2$

(b) Welches Array mit den Einträgen $\{1, \ldots, n\}$ hat die meisten Inversionen, welches hat die wenigsten?

Lösungsvorschlag

Folgt nach der 1 eine absteigend sortiere Folge, so hat sie am meisten Inversionen, z. B. $\{1,7,6,5,4,3,2\}$. Eine aufsteigend sortierte Zahlenfolge hat keine Inversionen, z. B. $\{1,2,3,4,5,6,7\}$.

(c) Entwerfen Sie eine Prozedur `int merge(int[] a, int i, int h, int j);`

welche das Teilarray a[i.,j] sortiert und die Zahl der in ihm enthaltenen Inversionen zurückliefert, wobei die folgenden Vorbedingungen angenommen werden:

- $0 \leq i \leq h \leq j < n$, wobei n die Länge von a ist (`n = a.length`).
- $a[i \ldots h]$ und $a[h+1 \ldots j]$ sind aufsteigend sortiert.
- Die Einträge von $a[i \ldots j]$ sind paarweise verschieden.

Ihre Prozedur soll in linearer Zeit, also $\mathcal{O}(j-i)$ laufen. Orientieren Sie sich bei Ihrer Lösung an der Mischoperation des bekannten Mergesort-Verfahrens.

(d) Entwerfen Sie nun ein Divide-and-Conquer-Verfahren zur Bestimmung der Zahl der Inversionen, indem Sie angelehnt an das Mergesort-Verfahren einen Algorithmus `ZI` beschreiben, der ein gegebenes Array in sortierter Form liefert und gleichzeitig dessen Inversionsanzahl berechnet. Im Beispiel wäre also

$$ZI([2,3,8,6,1]) = ([1,2,3,6,8],5)$$

Die Laufzeit Ihres Algorithmus auf einem Array der Größe n soll $\mathcal{O}(n \log(n))$ sein.

Sie dürfen die Hilfsprozedur merge aus dem vorherigen Aufgabenteil verwenden, auch, wenn Sie diese nicht gelöst haben.

(e) Begründen Sie, dass Ihr Algorithmus die Laufzeit $\mathcal{O}(n \log(n))$ hat.

(f) Geben Sie die Lösungen folgender asymptotischer Rekurrenzen (in O-Notation) an:

(i) $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(\log n)$

(ii) $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$

(iii) $T(n) = 3 \cdot T(\frac{n}{2}) + \mathcal{O}(n)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Rucksackproblem“ (66115-2018-H.T2-A6)

Backtracking

Ein sehr bekanntes Optimierungsproblem ist das sogenannte Rucksackproblem: Gegeben ist ein Rucksack mit der Tragfähigkeit B. Weiterhin ist eine endliche Menge von Gegenständen mit Werten und Gewichten gegeben. Nun soll eine Teilmenge der Gegenstände so ausgewählt werden, dass ihr Gesamtwert maximal ist, aber ihr Gesamtgewicht die Tragfähigkeit des Rucksacks nicht überschreitet.

Mathematisch exakt kann das Rucksackproblem wie folgt formuliert werden:

Gegeben ist eine endliche Menge von Objekten U. Durch eine Gewichtsfunktion $w\colon U \to \mathbb{R}^+$ wird den Objekten ein Gewicht und durch eine Nutzenfunktion $v\colon U \to \mathbb{R}^+$ ein festgelegter Nutzwert zugeordnet.

Des Weiteren gibt es eine vorgegebene Gewichtsschranke $B \in \mathbb{R}^+$. Gesucht ist eine Teilmenge $K \subseteq U$, die die Bedingung $\sum_{u \in K} w(u) \leq B$ einhält und die Zielfunktion $\sum_{u \in K} v(u)$ maximiert.

Das Rucksackproblem ist NP-vollständig (Problemgröße ist die Anzahl der Objekte), sodass es an dieser Stelle wenig Sinn macht, über eine effiziente Lösung nachzudenken. Lösen Sie das Rucksackproblem daher mittels Backtracking und formulieren Sie einen entsprechenden Algorithmus. Gehen Sie davon aus, dass die Gewichtsschranke B sowie die Anzahl an Objekten N beliebig, aber fest vorgegeben sind.

Das Programm soll folgende Ausgaben liefern:

(a) Maximaler Nutzwert, der durch eine Objektauswahl unter Einhaltung der Gewichtsschranke B erreicht werden kann.

(b) Das durch die maximierende Objektmenge erreichte Gesamtgewicht.

(c) Diejenigen Objekte (Objektnummern) aus U, die zur Maximierung des Nutzwerts beigetragen haben.

Lösungsvorschlag

```java
/**
 * https://stackoverflow.com/a/14186622
 */
public class Rucksack {
  // static int[] werte = new int[] { 894, 260, 392, 281, 27 };
  // static int[] gewichte = new int[] { 8, 6, 4, 0, 21 };
  // static int[] werte = new int[] { 4, 2, 10, 1, 2 };
  // static int[] gewichte = new int[] { 12, 1, 4, 1, 2 };
  static int werte[] = new int[] { 60, 100, 120 };
  static int gewichte[] = new int[] { 10, 20, 30 };



  /**
   * Gewichtsschranke
   */
  //static int B = 30;
  //static int B = 15;
  static int B = 50;
```

```
  /**
   * Diejenigen Objekte aus U, die zur Maximierung des Nutzwerts beigetragen
   * haben.
   */
  static boolean[] auswahl = new boolean[werte.length];

  private static int berechne(int i, int W) {
    if (i < 0) {
      return 0;
    }
    int alt = berechne(i - 1, W);
    if (gewichte[i] > W) {
      // Backtracking!
      auswahl[i] = false;
      return alt;
    } else {

      int neu = berechne(i - 1, W - gewichte[i]) + werte[i];
      if (alt >= neu) {
        // Backtracking!
        auswahl[i] = false;
        return alt;
      } else {
        auswahl[i] = true;
        return neu;
      }
    }
  }

  static void werteAus() {
    System.out.println(berechne(werte.length - 1, B));

    int gesamtGewicht = 0;
    int gesamtWert = 0;

    for (int i = 0; i < auswahl.length; i++) {
      if (auswahl[i]) {
        gesamtGewicht += gewichte[i];
        gesamtWert += werte[i];
        System.out.println("Objekt-Nr. " + i + " Gewicht: " + gewichte[i] +
↪   " Wert: " + werte[i]);
      }
    }

    System.out.println("Gesamtgewicht: " + gesamtGewicht);
    System.out.println("Gesamtwert: " + gesamtWert);
  }

  public static void main(String[] args) {
    werteAus();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2018/herbst/Rucksack.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:

https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/09/Thema-2/Aufgabe-6.tex

Examensaufgabe „Muffinsorten“ (66115-2019-F.T1-A6)

Greedy-Algorithmus

Aus dem Känguru-Wettbewerb 2017 — Klassenstufen 3 und 4.

> Luna hat für den Kuchenbasar Muffins mitgebracht: 10 Apfelmuffins, 18 Nussmuffins, 12 Schokomuffins und 9 Blaubeermuffins. Sie nimmt immer 3 verschiedene Muffins und legt sie auf einen Teller. Welches ist die kleinste Zahl von Muffins, die dabei übrig bleiben können?
>
> A: 1, B: 3, C: 4, D: 7, E: 8

(a) Geben Sie die richtige Antwort auf die im Känguru-Wettbewerb gestellte Frage und begründen Sie sie.

Lösungsvorschlag

4 [a]

[a] https://www.youtube.com/watch?v=ceJW9kAplVY

(b) Lunas Freundin empfiehlt den jeweils nächsten Teller immer aus den drei aktuell häufigsten Muffinsorten zusammenzustellen. Leiten Sie aus dieser Idee einen effizienten GreedyAlgorithmus her, der die Fragestellung für beliebige Anzahlen von Muffins löst (nach wie vor soll es nur vier Sorten und je drei pro Teller geben). Skizzieren Sie in geeigneter Form, wie Ihr Algorithmus die Beispielinstanz von oben richtig löst.

Lösungsvorschlag

```java
public static int berechneRest4Sorten3ProTeller() {
  int[] muffins = new int[] { 10, 18, 12, 9 };
  int n = muffins.length;
  sortiere(muffins);

  // Wir nehmen uns 3 verschiedene Muffins solange, wie die dritthäufigste
  // Muffinsorte noch Muffins hat.
  while (muffins[n - 3] > 0) {
    muffins[n - 1]--;
    muffins[n - 2]--;
    muffins[n - 3]--;
    sortiere(muffins);
  }
  return berechneGesamtzahl(muffins);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java`

(c) Beschreiben Sie eine mögliche und sinnvolle Verallgemeinerung Ihrer Lösung auf n Muffinsorten und k Muffins pro Teller für $n > 4$ und $k > 3$.

Lösungsvorschlag

```java
public static int berechneRestAllgemein(int[] muffins, int k) {
  int n = muffins.length;
  sortiere(muffins);
  while (muffins[n - k] > 0) {
    for (int i = 1; i <= k; i++) {
      muffins[n - i]--;
    }
    sortiere(muffins);
  }
  return berechneGesamtzahl(muffins);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java`

(d) Diskutieren Sie, wie man die Korrektheit des Greedy-Algorithmus zeigen könnte, also dass er tatsächlich immer eine optimale Lösung findet. Ein kompletter, rigoroser Beweis ist nicht verlangt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/03/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Gutschein“ (66115-2020-H.T2-TA2-A4)

Das GUTSCHEIN-Problem ist gegeben durch eine Folge $w_1, \ldots, w_n$ von Warenwerten (wobei $w \in N_0$ für $i = 1, \ldots, n$) und einem Gutscheinbetrag $G \in \mathbb{N}_0$.

Da Gutscheine nicht in Bargeld ausgezahlt werden können, ist die Frage, ob man eine Teilfolge der Waren findet, sodass man genau den Gutschein ausnutzt. Formal ist dies die Frage, ob es eine Menge von Indizes I mit $I \subseteq \{1, \ldots, n\}$ gibt, sodass $\sum_{i \in I} w_i = G$

> **Exkurs: Teilsummenproblem**
> Das **Teilsummenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen $I = \{w_1, w_2, \ldots, w_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke c ist.

(a) Sei $w_1 = 10, w_2 = 30, w_3 = 40, w_4 = 20, w_5 = 15$ eine Folge von Warenwerten.

(i) Geben Sie einen Gutscheinbetrag $40 < G < 115$ an, sodass die GUTSCHEIN-Instanz eine Lösung hat. Geben Sie auch die lösende Menge $I \subseteq \{1, 2, 3, 4, 5\}$ von Indizes an.

Lösungsvorschlag

50
$I = \{1, 3\}$

(ii) Geben Sie einen Gutscheinbetrag G mit $40 < G < 115$ an, sodass die GUTSCHEIN-Instanz keine Lösung hat.

Lösungsvorschlag

51

(b) Sei *table* eine $(n \times (G + 1))$-Tabelle mit Einträgen table$[i, k]$, für $1 \leq i \leq n$ und $0 \leq k \leq G$, sodass

$$\text{table}[i,k] = \begin{cases} \textbf{true} & \text{falls es } I \subseteq \{1,\ldots,n\} \text{ mit } \sum_{i \in I} w_i = G \text{ gibt} \\ \textbf{false} & \text{sonst} \end{cases}$$

Geben Sie einen Algorithmus in Pseudo-Code oder Java an, der die Tabelle *table* mit *dynamischer Programmierung* in Worst-Case-Laufzeit $\mathcal{O}(n \times G))$ erzeugt. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus. Welcher Eintrag in *table* löst das GUTSCHEIN-Problem?

Lösungsvorschlag

Algorithmus 3: Gutschein-Problem

table = boolean array $n+1 \times (G+1)$; // Initialisiere ein boolsches Feld mit $n+1$ Zeilen für jeden Warenwert und 0 für keinen Warenwert und mit $G+1$ Spalten für alle Gutscheinbetrag bis G und 0 für keinen Gutscheinbetrag

for *k in* $1 \ldots G$ **do** ; // Wenn der Gutscheinbetrag größer als 0 ist und es keine Warenwerte gibt, ð$n = 0$, kann der Gutschein nicht eingelöst werden.

  table$[0][k] =$ false

end

for *i in* $0 \ldots n$ **do** ; // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst werden.

  table$[i][0] =$ true

end

for *i in* $1 \ldots n$ **do** ; // Durchlaufe jede Zeile der Warenwerte

  for *k in* $1 \ldots G$ **do** ; // Durchlaufe jede Spalte der Gutscheinbeträge in dieser Zeile

    table$[i][k] =$ table$[i-1][k]$; // Übernehme erstmals das Ergebnis der Zelle der vorhergehenden Zeile in der gleichen Spalte

    if $k \geq w_i$ *und table*$[i][k]$ *noch nicht markiert* **then** ; // Wenn der aktuelle Gutscheinbetrag größer als der aktuelle Warenwert und die aktuelle Zelle noch nicht als wahr markiert ist

      table$[i][k] =$ table$[i-1][k-w_i]$; // übernimmt das Ergebnis des aktuellen Gutscheinbetrags minus des aktuellen Warenwerts

      ;

    end

  end

end

```
/**
 * Nach <a href="https://www.geeksforgeeks.org/subset-sub-problem-dp-25">
 ↪  Subset
 * Sum Problem auf geeksforgeeks.org</a>
 */
public class Gutschein {
  /**
   * @param G Die Indizes der GUTSCHEIN-Beträge.
   *
   * @param W Das GUTSCHEIN-Problem ist gegeben durch eine Folge w1, ..., wn
 ↪  von
   *          Warenwerten.
```

```
   *
   * @return Wahr, wenn der Gutscheinbetrag vollständig in Warenwerten
↪  eingelöst
   *         werden kann, falsch wenn der Betrag nicht vollständig eingelöst
   *         werden kann.
   */
  public static boolean gutscheinDP(int G, int W[]) {
    // Der Eintrag in der Tabelle tabelle[i][k] ist wahr,
    // wenn es eine Teilsumme der
    // W[0..i-1] gibt, die gleich k ist.
    int n = W.length;
    boolean table[][] = new boolean[n + 1][G + 1];

    // Wenn der Gutschein-Betrag größer als 0 ist und es keine
    // Warenwerte (n = 0) gibt, kann der Gutschein nicht eingelöst
    // werden.
    for (int k = 1; k <= G; k++) {
      table[0][k] = false;
    }

    // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst werden.
    for (int i = 0; i <= n; i++) {
      table[i][0] = true;
    }

    for (int i = 1; i <= n; i++) {
      for (int k = 1; k <= G; k++) {
        table[i][k] = table[i - 1][k];
        // Warenwert
        int w = W[i - 1];
        if (k >= w && !table[i][k]) {
          table[i][k] = table[i - 1][k - w];
        }
      }
    }
    return table[n][G];
  }

  public static void main(String[] args) {
    System.out.println(gutscheinDP(50, new int[] { 10, 30, 40, 20, 15 }));
    System.out.println(gutscheinDP(41, new int[] { 10, 30, 40, 20, 15 }));

    System.out.println(gutscheinDP(3, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(5, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(6, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(2, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(1, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(7, new int[] { 1, 2, 3 }));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Gutschein.java`

Die äußere for-Schleife läuft n mal und die innere for-Schleife G mal.

Der letzte Eintrag in der Tabelle, also der Wert in der Zelle `table[W.length][G]`, löst das Gutscheinproblem. Steht hier `true`, dann gibt es eine Teilfolge der Waren, die den Gutscheinbetrag genau ausnutzt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-4.tex

Übungsaufgabe „Münzwechsler“ (Greedy-Algorithmus)

Greedy-Algorithmus

(a) Nehmen Sie an, es stehen beliebig viele 5-Cent, 2-Cent und 1-Cent-Münzen zur Verfügung. Die Aufgabe besteht darin, für einen gegebenen Cent-Betrag möglichst wenig Münzen zu verbrauchen. Entwerfen Sie eine Methode

```
public void wechselgeld (int n)
```

die diese Aufgabe mit einem Greedy-Algorithmus löst und für den Betrag von n Cent die Anzahl $c5$ der 5-Cent-Münzen, die Anzahl $c2$ der 2-Cent-Münzen und die Anzahl $c1$ der 1-Cent-Münzen berechnet und diese auf der Konsole ausgibt. Sie können dabei den Operator / für die ganzzahlige Division und den Operator % für den Rest bei der ganzzahligen Division verwenden. [1]

Lösungsvorschlag

```
public static void wechsle(int betrag) {
  int rest;
  int c5 = betrag / 5;
  rest = betrag % 5;
  int c2 = rest / 2;
  int c1 = rest % 2;

  System.out.println(String.format("Für den Betrag von %s Cent werden \n" +
↪ "%s Fünf-Cent-Münzen, \n"
      + "%s Zwei-Cent-Münzen und \n" + "%s Ein-Cent-Münzen ausgegeben.",
↪ betrag, c5, c2, c1));
 }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/greedy/Muenzwechsler.java`

(b) Es kann gezeigt werden, dass der Greedy-Algorithmus für den obigen Fall der Münzwerte 5, 2 und 1 optimal ist, ðdass er immer die Gesamtzahl der Münzen minimiert. Nehmen Sie nun an, es gibt die Münzwerte 5 und 1. Ist es dann möglich, einen dritten Münzwert so zu wählen, dass der Greedy-Algorithmus mit den drei Münzen nicht mehr optimal ist? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Falls der dritte Münzwert 4 ist, ist der Greedy-Algorithmus nicht mehr optimal. Der Greedy-Algorithmus benutzt zunächst so viele 5-Cent-Münzen wie möglich und dann so viele 4-Cent-Münzen wie möglich. Ein Betrag von 8 Cent wird also in eine 5-Cent und drei 1-Cent-Münzen aufgeteilt. Optimal ist aber die Aufteilung in zwei 4-Cent-Münzen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/20_Greedy-Algorithmen/Aufgabe_Muenzwechsler.tex`

[1] Quelle möglicherweise von `https://www.yumpu.com/de/document/read/17936760/ubungen-zum-prasenzmodul-algorithmen-und-datenstrukturen`

Übungsaufgabe „Wechselgeld“ (Greedy-Algorithmus)

Als Beispiel nehmen wir die Herausgabe von Wechselgeld auf Beträge unter 1€. Verfügbar sind die Münzen mit den Werten 50ct, 10ct, 5ct, 2ct, 1ct. Unser Ziel ist, so wenig Münzen wie möglich in das Portemonnaie zu bekommen. Ein Beispiel: $78ct = 50 + 2 \cdot 10 + 5 + 2 + 1$ Es wird jeweils immer die größte Münze unter dem Zielwert genommen und von diesem abgezogen. Das wird so lange durchgeführt, bis der Zielwert Null ist.

Formalisierung

Gesucht ist ein Algorithmus der folgende Eigenschaften beschreibt. Bei der *Eingabe* muss gelten:

(a) dass die eingegebene Zahl eine natürliche Zahl ist, also betrag > 0
(b) dass eine Menge von Münzwerten zur Verfügung steht münzen $= \{c_1, ..., c_n\}$ z. B. $\{1, 2, 5, 10, 20, 50\}$

Die *Ausgabe* besteht dann aus ganzen Zahlen wechselgeld$[1], \ldots,$ wechselgeld$[n]$. Dabei ist wechselgeld$[i]$ die Anzahl der Münzen des Münzwertes für c_i für $i = 1, \ldots, n$ und haben die Eigenschaften:

(a) wechselgeld$[1] \cdot c_1 + \ldots +$ wechselgeld$[n] \cdot c_n =$ betrag
(b) wechselgeld$[1] + \ldots +$ wechselgeld$[n]$ ist minimal unter allen Lösungen für 1.

Lösungsvorschlag

```java
/**
 * <a href=
 *
 ↪  "https://de.wikiversity.org/wiki/Kurs:Algorithmen_und_Datenstrukturen/Vorlesung/Greedyalgorithmen
 * und Datenstrukturen/Vorlesung/Greedyalgorithmen Wechselgeldalgorithmus</a>
 */
public class Wechselgeld {

  public static int[] berechneWechselgeld(int[] münzen, int betrag) {
    int[] wechselgeld = new int[münzen.length];
    int aktuelleMünze = münzen.length - 1;
    while (betrag > 0) {
      while (betrag < münzen[aktuelleMünze] && aktuelleMünze > 0)
        aktuelleMünze--;
      if (betrag >= münzen[aktuelleMünze] && aktuelleMünze >= 0) {
        betrag -= münzen[aktuelleMünze];
        wechselgeld[aktuelleMünze]++;
      } else
        return null;
    }
    return wechselgeld;
  }

  public static void main(String[] args) {
    int[] münzen = { 1, 2, 5, 10, 20, 50 };
    int betrag = 78;
```

```
    int[] wechselgeld = berechneWechselgeld(münzen, betrag);

    System.out.println(String.format("Der Betrag von %s Cent wird gewechselt in:",
↪   betrag));

    for (int i = 0; i < wechselgeld.length; i++) {
      System.out.println(String.format("%s x %s Cent", wechselgeld[i],
↪   münzen[i]));
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/muster/Wechselgeld.java`

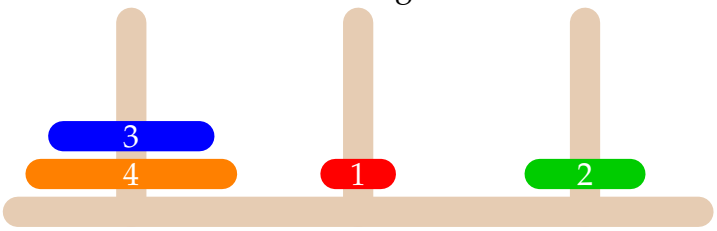
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/20_Greedy-Algorithmen/Aufgabe_Wechselgeld.tex`

Übungsaufgabe „Hanoi“ (Teile-und-Herrsche (Divide-and-Conquer))

Teile-und-Herrsche (Divide-and-Conquer)

Betrachten wir das folgende Spiel (Türme von Hanoi), das aus drei Stäben 1, 2 und 3 besteht, die senkrecht im Boden befestigt sind. Weiter gibt es n kreisförmige Scheiben mit einem Loch im Mittelpunkt, so dass man sie auf die Stäbe stecken kann. Dabei haben die Scheiben verschiedene Radien, alle sind unterschiedlich groß. Zu Beginn stecken alle Scheiben auf dem Stab 1, wobei immer eine kleinere auf einer größeren liegt. Das Ziel des Spiels ist es nun, die Scheiben so umzuordnen, dass sie in der gleichen Reihenfolge auf dem Stab 3 liegen. Dabei darf immer nur eine Scheibe bewegt werden und es darf nie eine größere auf einer kleineren Scheibe liegen. Stab 2 darf dabei als Hilfsstab verwendet werden.

Ein Beispiel für 4 Scheiben finden Sie in folgendem Bild:



Entwerfen Sie mit Hilfe der Vorlage eine varibale Simulation der Türme von Hanoi.

(a) Ein ELEMENT hat immer einen Wert (Integer) und kennt das Nachfolgende Element, wobei immer nur das jeweilige Element auf seinen Wert und seinen Nachfolger zugreifen darf

(b) Ein Turm ist einem Stack (Kellerspeicher) nachempfunden und kennt somit nur das erste Element. Hinweis: Beachten Sie, dass nur kleinere Elemente auf den bisherigen Stack gelegt werden können

(c) In der Klasse HANOI müssen Sie nur die Methode `public void hanoi (int n, TURM quelle, TURM ziel, TURM hilfe)` implementieren. Die anderen Methoden sind zur Veranschaulichung des Spiels! Entwerfen Sie eine rekursive Methode die einen Turm der Höhe n vom Stab quelle auf den Stab ziel transportiert und den Stab hilfe als Hilfsstab verwendet.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/30_Divide-and-Conquer/Aufgabe_Hanoi.tex`

Übungsaufgabe „Wegberechnung im Gitter“ (Dynamische Programmierung)

Betrachten Sie das folgende Gitter mit $m+1$ Zeilen und $n+1$ Spalten ($m \geq 1$ und $n \geq 1$): [2] geeksforgeeks [3]

Angenommen, Sie befinden sich zu Beginn am Punkt $(0,0)$ und wollen zum Punkt (m,n).

Für die Anzahl $A(i,j)$ aller verschiedenen Wege vom Punkt $(0,0)$ zum Punkt (i,j) lassen sich folgende drei Fälle unterscheiden (es geht jeweils um die kürzesten Wege ohne Umweg!):

- $1 \leq i \leq m$ und $j = 0$:

  Es gibt genau einen Weg von $(0,0)$ nach $(i,0)$ für $1 \leq i \leq m$.

- $i = 0$ und $1 \leq j \leq n$:

  Es gibt genau einen Weg von $(0,0)$ nach $(0,j)$ für $1 \leq j \leq n$.

- $1 \leq i \leq m$ und $1 \leq j \leq n$:

  auf dem Weg zu (i,j) muss als vorletzter Punkt entweder $(i-1,j)$ oder $(i,j-1)$ besucht worden sein.

Daraus ergibt sich folgende Rekursionsgleichung:

$$A(i,j) = \begin{cases} 1 & \text{falls } (1 \leq i \leq m \text{ und } j = 0) \text{ oder } (i = 0 \text{ und } 1 \leq j \leq n) \\ A(i-1,j) + A(i,j-1) & \text{falls } 1 \leq i \leq m \text{ und } 1 \leq j \leq n \end{cases}$$

Implementieren Sie die Java-Klasse `Gitter` mit der Methode

```
public int berechneAnzahlWege(),
```

die ausgehend von der Rekursionsgleichung durch dynamische Programmierung die Anzahl aller Wege vom Punkt $(0,0)$ zum Punkt (m,n) berechnet. Die Überprüfung, ob $m \leq 1$ und $n \leq 1$ gilt, können Sie der Einfachheit halber weglassen.

Lösungsvorschlag

```
public int berechneAnzahlWege() {
  int i, j;
  for (i = 1; i <= m; i++) {
    anzahlWege[i][0] = 1;
  }
  for (j = 1; j <= n; j++) {
    anzahlWege[0][j] = 1;
  }
  for (i = 1; i <= m; i++) {
```

[2] Quelle möglicherweise von https://www.yumpu.com/de/document/read/17936760/ubungen-zum-prasenzmodul-algorithmen-und-datenstrukturen

[3] https://www.geeksforgeeks.org/count-possible-paths-top-left-bottom-right-nxm-matrix/

```
      for (j = 1; j <= n; j++) {
        anzahlWege[i][j] = anzahlWege[i - 1][j] + anzahlWege[i][j - 1];
      }
    }
    return anzahlWege[m][n];
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/dp/Gitter.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/40_Dynamisches-Programmieren/Aufgabe_Wegberechnung.tex`

Übungsaufgabe „Damenproblem“ (Implementierung in Java, Backtracking)

Implementieren sie mittels Backtracking einen Algorithmus, der acht Damen auf einem Schachbrett so aufgestellt, dass keine zwei Damen einander gemäß ihren in den Schachregeln definierten Zugmöglichkeiten schlagen können. Für Damen heißt dies konkret: Es dürfen keine zwei Damen auf derselben Reihe, Linie oder Diagonale stehen. Es gibt 92 mögliche Lösungen für das 8×8 Feld.

Lösungsvorschlag

```
public class Damenproblem {
  static int n = 8;
  static int[][] spielBrett = new int[n][n];
  static int DAME = 1;
  static int LEER = 0;

  public static boolean istGültig(int zeile, int spalte) {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        if (spielBrett[i][j] == 1) {
          if (i == zeile || j == spalte) {
            return false;
          }
        }
      }
    }
    for (int i = 0; i < n; i++) {
      if (zeile + i < n && spalte + i < n && spielBrett[zeile + i][spalte + i] ==
↪   1)
        return false;
      if (zeile - i > -1 && spalte - i > -1 && spielBrett[zeile - i][spalte - i]
↪   == 1)
        return false;
      if (zeile + i < n && spalte - i > -1 && spielBrett[zeile + i][spalte - i] ==
↪   1)
        return false;
      if (zeile - i > -1 && spalte + i < n && spielBrett[zeile - i][spalte + i] ==
↪   1)
        return false;
    }
    return true;
  }

  public static boolean löse(int zeile) {
    if (zeile == n) {
      return true;
    }

    for (int i = 0; i < n; i++) {
      if (istGültig(zeile, i) == true) {
        spielBrett[zeile][i] = DAME;

        if (löse(zeile + 1) == true) {
```

```
            return true;
          }
          spielBrett[zeile][i] = LEER;
        }
      }

      return false;
    }
```

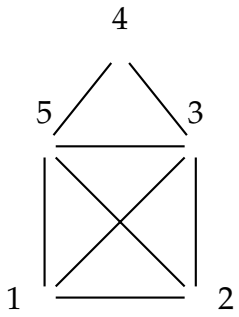
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/backtracking/damenproblem/Damenproblem.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/50_Backtracking/Aufgabe_Damenproblem.tex`

Übungsaufgabe „Nikolaus“ (Backtracking)

Backtracking

Hier ist das *„Haus des Nikolaus“* mit einer bestimmten Nummerierung der Eckpunkte vorgegeben. Es sollen alle Lösungen zum Zeichnen der Figur in einem Zug gefunden werden. Eine Lösung könnte dann in der Form 123451352 ausgegeben werden. Das Programm soll eine einfache Anpassung an andere Graphen ermöglichen. Der Ausschluss von gespiegelten Lösungen ist nicht gefordert.



Exkurs: Backtracking
Eine Lösung lässt sich nach dem Prinzip *Versuch und Testen* ermitteln. Eine vermutete Teillösung muss wieder verworfen werden, wenn ein Test ihre Ungültigkeit nachgewiesen hat. Man nennt diesen Ansatz deshalb auch *Rückverfolgen* oder *Backtracking*. Mit diesem Ansatz lassen sich eine ganze Reihe von Problemen in der Informatik sehr elegant formulieren und lösen. Hier eine kleine Auswahl (Genaueres dazu später):

Acht-Damen-Problem: Acht Damen sollen so auf ein Schachbrett gestellt werden, dass keine Dame eine andere bedroht.

Vier-Farben-Problem: Eine Landkarte soll mit vier Farben so gefärbt werden, dass benachbarte Länder immer unterschiedliche Farben bekommen.

Labyrinth-Problem: Ein Labyrinth mit Sackgassen und Verzweigungen ist zu durchlaufen, um den Ausgang zu finden.

Konkreter:

(a) Man versucht, eine Kante (Verbindungsstrecke) zu zeichnen, wenn sie zulässig ist oder noch nicht gezeichnet wurde.

(b) Ist das nicht möglich, muss die zuletzt gezeichnete Kante gelöscht werden.

(c) Ist es möglich, dann hat man das Problem um eine Stufe vereinfacht.

(d) Hat man durch dieses Verfahren insgesamt 8 Kanten zeichnen können, hat man eine Lösung gefunden. Jetzt löscht man wieder die zuletzt gezeichnete Kante und sucht nach weiteren Lösungen.

Realisierung des Programms

Datenstrukturen

Die folgende Tabelle gibt an, welche Verbindungslinien zulässig sind (durch X markiert). Die erste Zeile bedeutet also, dass von Punkt 1 zu den Punkten 2, 3 und 5 Strecken gezeichnet werden dürfen. Eine solche Tabelle heißt auch Adjazenzmatrix (von adjazieren; lat.: anwohnen, anliegen). Eine solche Tabelle lässt sich durch `boolean[] [] kanteZulaessig`; in einem zweidimensionalen Feld speichern. Eine entsprechende Tabelle `boolean[] [] kanteGezeichnet;` erfasst dann die schon gezeichneten Kanten. In einem weiteren eindimensionalen Feld wird jeweils eine Lösung erfasst.

Methoden

Es bieten sich folgende Methoden zur Strukturierung des Programmes an:

(a) `void initialsiereFelder()`

(b) `void zeichneKante(int von, int nach)`

(c) `void löscheKante(int von, int nach)`

(d) `void gibLösungAus()`

(e) `void versucheKanteZuZeichnen(int start)` : Die rekursive Methode soll vom Punkt start weitere Kanten zeichen.

(f) Das Hauptprogramm:

```
public static void main(String[] arg) {
  initialsiereFelder();
  for (int punktNr = 1; punktNr <= maxPunktAnzahl; punktNr++) {
    lösungsWeg[0] = punktNr; // Startpunkt eintragen
    versucheKanteZuZeichnen(punktNr);
  }
  System.out.println();
  System.out.println("Es ergaben sich " + lösungsAnzahl + " Loesungen.");
}
```

```
/**
 * Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen:
 * Aufgabenblatt 3: Algorithmenmuster.
 *
 * <a href="https://www.studon.fau.de/file2521908_download.html">Angabe: AB_3
 * Greedy_DP_Backtracking.pdf</a>
 * <a href="https://www.studon.fau.de/file2521907_download.html">Lösung: AB_3
 * Greedy_DP_Backtracking_Lsg.pdf</a>
 */
public class Nikolaus {
  static final int maxPunktAnzahl = 5;
  static final int maxKantenAnzahl = 8;
  static boolean[][] kanteZulässig;
  static boolean[][] kanteGezeichnet;
```

```
  static int[] lösungsWeg;
  static int aktuelleKantenAnzahl = 0;
  static int lösungsAnzahl = 0;

  /**
   * Zulässige Kanten für das „Haus des Nikolaus" eintragen. Der Nummerierung
   * liegt das Bild in main zu Grunde. Eine Anpassung an andere Graphen ist leicht
   * möglich.
   */
  static void initialsiereFelder() {
    kanteZulässig = new boolean[maxKantenAnzahl + 1][maxKantenAnzahl + 1];
    kanteGezeichnet = new boolean[maxKantenAnzahl + 1][maxKantenAnzahl + 1];
    lösungsWeg = new int[maxKantenAnzahl + 2]; // mit Startpunkt
    // Erst mal alles auf false ;
    for (int i = 1; i <= maxPunktAnzahl; i++) {
      for (int k = 1; k <= maxPunktAnzahl; k++) {
        kanteZulässig[i][k] = false;
        kanteGezeichnet[i][k] = false;
      }
    }

    kanteZulässig[1][2] = true; // von 1 nach 2 zulässig
    kanteZulässig[2][1] = true;

    kanteZulässig[1][3] = true;
    kanteZulässig[3][1] = true;

    kanteZulässig[1][5] = true;
    kanteZulässig[5][1] = true;

    kanteZulässig[2][3] = true;
    kanteZulässig[3][2] = true;

    kanteZulässig[2][5] = true;
    kanteZulässig[5][2] = true;

    kanteZulässig[3][4] = true;
    kanteZulässig[4][3] = true;

    kanteZulässig[3][5] = true;
    kanteZulässig[5][3] = true;

    kanteZulässig[4][5] = true;
    kanteZulässig[5][4] = true;
    for (int i = 0; i <= maxKantenAnzahl; i++) {
      lösungsWeg[i] = 0;
    }
  }

  static void zeichneKante(final int von, final int nach) {
    kanteGezeichnet[von][nach] = true;
    kanteGezeichnet[nach][von] = true;
    // Anzahl bereits gezeichneter Kanten erhöhen
    aktuelleKantenAnzahl++;
    // neuen Wegpunkt in Lösung aufnehmen
```

```
    lösungsWeg[aktuelleKantenAnzahl] = nach;
  }

  static void löscheKante(final int von, final int nach) {
    kanteGezeichnet[von][nach] = false;
    kanteGezeichnet[nach][von] = false;
    aktuelleKantenAnzahl--;
  }

  static boolean fertig() {
    return (aktuelleKantenAnzahl == maxKantenAnzahl);
  }

  static void gibLösungAus() {
    for (int i = 0; i <= maxKantenAnzahl; i++) {
      System.out.print(lösungsWeg[i]);
      System.out.print(" ");
      lösungsAnzahl++;
      if (lösungsAnzahl % 8 == 0) {
        System.out.println();
      }
    }
  }

  static void versucheKanteZuZeichnen(final int start) {
    for (int ziel = 1; ziel <= maxPunktAnzahl; ziel++) {
      if (kanteZulässig[start][ziel] && !kanteGezeichnet[start][ziel]) {
        zeichneKante(start, ziel);
        if (!fertig()) {
          versucheKanteZuZeichnen(ziel);
        } else {
          gibLösungAus();
        }
        löscheKante(start, ziel);
      }
    }
  }

  public static void main(final String[] arg) {
    initialsiereFelder();
    System.out

↪   .println("Das Programm bestimmt alle Lösungen des Problems, das Haus des Nikolaus in einem Zug zu z
    System.out.println("      4      ");
    System.out.println("     . .     ");
    System.out.println("    .   .    ");
    System.out.println("   5-----3   ");
    System.out.println("   |.   .|   ");
    System.out.println("   | . . |   ");
    System.out.println("   | . . |   ");
    System.out.println("   |.   .|   ");
    System.out.println("   1-----2   ");
    for (int punktNr = 1; punktNr <= maxPunktAnzahl; punktNr++) {
      lösungsWeg[0] = punktNr;
      versucheKanteZuZeichnen(punktNr);
```

```
    }
    System.out.println();
    System.out.println("Es ergaben sich " + lösungsAnzahl + " Lösungen.");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/backtracking/Nikolaus.java`

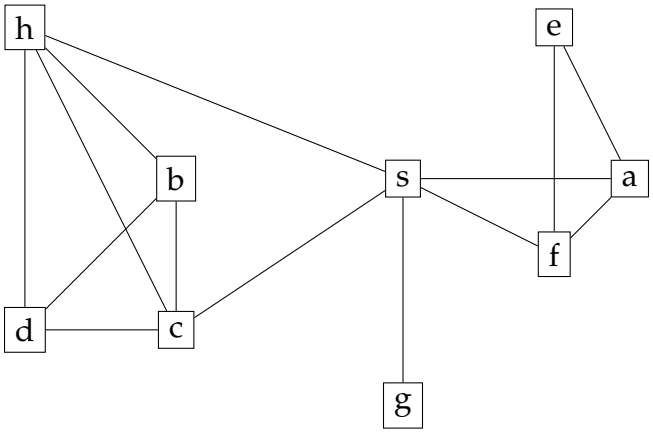
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/60_Algorithmenmuster/50_Backtracking/Aufgabe_Nikolaus.tex`

Listen

Examensaufgabe „dfs-number Graph s,a-h“ (46115-2014-H.T1-A8)

Aufgabe 8

(a) Führen Sie auf dem folgenden ungerichteten Graphen `G` eine Tiefensuche ab dem Knoten `s` aus (graphische Umsetzung). Unbesuchte Nachbarn eines Knotens sollen dabei in *alphabetischer Reihenfolge* abgearbeitet werden. Die Tiefensuche soll auf Basis eines *Stacks* umgesetzt werden. Geben Sie die Reihenfolge der besuchten Knoten, also die `dfs-number` der Knoten, und den Inhalt des Stacks in jedem Schritt an.



Lösungsvorschlag

In der Musterlösung auf Seite 3 lautet das Ergebnis s, a, e, f, c, b, d, h, g. Ich glaube jedoch diese Lösung ist richtig:

fett: Knoten, der entnommen wird.

kursiv: Knoten, die zum Stapel hinzugefügt werden.

Reihenfolge	Stapel	besucht
1	***s***	s
2	*a, c, f, g,* ***h***	h
3	a, c, f, g, *b,* ***d***	d
4	a, c, f, g, **b**	b
5	a, c, f, **g**	g
6	a, c, **f**	f
7	a, c, **e**	e
8	a, **c**	c
9	**a**	a

(b) Führen Sie nun eine Breitensuche auf dem gegebenen Graphen aus, diese soll mit einer Queue umgesetzt werden. Als Startknoten wird wieder *s* verwendet. Geben Sie auch hier die Reihenfolge der besuchten Knoten und den Inhalt der Queue bei jedem Schritt an.

Lösungsvorschlag

fett: Knoten, der entnommen wird.

kursiv: Knoten, die zur Warteschlange hinzugefügt werden.

Reihenfolge	Warteschlange	besucht
1	***s***	s
2	***a****, c, f, g, h*	a
3	**c**, f, g, h, *e*	c
4	**f**, g, h, e, *b, d*	f
5	**g**, h, e, b, d	g
6	**h**, e, b, d	h
7	**e**, b, d	e
8	**b**, d	b
9	**d**	d

(c) Geben Sie in Pseudocode den Ablauf von Tiefen- und Breitensuche an, wenn diese wie beschrieben mit einem Stack bzw. einer Queue implementiert werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2014/09/Thema-1/Aufgabe-8.tex`

Examensaufgabe „Mystery-Stacks“ (46115-2019-H.T1-A6)

Stapel (Stack)

Gegeben sei die Implementierung eines Stacks ganzer Zahlen mit folgender Schnittstelle:

```
import java.util.Stack;

/**
 * Um schnell einen lauffähigen Stack zu bekommen, verwenden wir den Stack aus
 * der Java Collection.
 */
public class IntStack {
  private Stack<Integer> stack = new Stack<Integer>();

  /**
   * Legt Element i auf den Stack.
   *
   * @param i Eine Zahl, die auf dem Stack gelegt werden soll.
   */
  public void push(int i) {
    stack.push(i);
  }

  /**
   * Gibt oberstes Element vom Stack.
   *
   * @return Das oberste Element auf dem Stapel.
   */
  public int pop() {
    return stack.pop();
  }

  /**
   * Fragt ab, ob Stack leer ist.
   *
   * @return Wahr, wenn der Stapel leer ist.
   */
  public boolean isEmpty() {
    return stack.empty();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/IntStack.java`

Betrachten Sie nun die Realisierung der folgenden Datenstruktur `Mystery`, die zwei Stacks benutzt.

```
public class Mystery {
  private IntStack a = new IntStack();
  private IntStack b = new IntStack();

  public void foo(int item) {
    a.push(item);
  }
```

```
public int bar() {
  if (b.isEmpty()) {
    while (!a.isEmpty()) {
      b.push(a.pop());
    }
  }
  return b.pop();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java`

(a) Skizzieren Sie nach jedem Methodenaufruf der im folgenden angegebenen Befehlssequenz den Zustand der beiden Stacks eines Objekts `m` der Klasse `Mystery`. Geben Sie zudem bei jedem Aufruf der Methode `bar` an, welchen Wert diese zurückliefert.

```
Mystery m = new Mystery();
m.foo(3);
m.foo(5);
m.foo(4);
m.bar();
m.foo(7);
m.bar();
m.foo(2);
m.bar();
m.bar();
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java`

Lösungsvorschlag

Code	Stack b	Stack b	Rückgabewert
`m.foo(3);`	{ 3 }	{ }	
`m.foo(5);`	{ 5, 3 }	{ }	
`m.foo(4);`	{ 4, 5, 3 }	{ }	
`m.bar();`	{ }	{ 5, 4 }	3
`m.foo(7);`	{ 7 }	{ 5, 4 }	
`m.bar();`	{ 7 }	{ 4 }	5
`m.foo(2);`	{ 2, 7 }	{ }	
`m.bar();`	{ 2, 7 }	{ }	4
`m.bar();`	{ }	{ 2 }	7

(b) Sei n die Anzahl der in einem Objekt der Klasse `Mystery` gespeicherten Werte. Im folgenden wird gefragt, wieviele Aufrufe von Operationen der Klasse `IntStack` einzelne Aufrufe von Methoden der Klasse `Mystery` verursachen. Begründen Sie jeweils Ihre Antwort.

(i) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im besten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

(ii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im schlechtesten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

(iii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im besten Fall?

Lösungsvorschlag

Wenn der Stack `b` nicht leer ist, dann werden zwei Aufrufe benötigt, nämlich `b.isEmpty()` und `b.pop()`

(iv) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im schlechtesten Fall?

Lösungsvorschlag

Wenn der Stack `b` leer ist, dann liegen all n Objekte im Stack `a`. Die Objekte im Stack `a` werden in der `while`-Schleife nach `b` verschoben. Pro Objekt sind drei Aufrufe nötig, also $3 \cdot n$. `b.isEmpty()` (erste Zeile in der Methode) und `b.pop()` (letzte Zeile in der Methode) wird immer aufgerufen. Wenn alle Objekt von `a` nach `b` verschoben wurden, wird zusätzlich noch einmal in der Bedingung der `while`-Schleife `a.isEmpty()` aufgerufen. Im schlechtesten Fall werden also $3 \cdot n + 3$ Operationen der Klasse `IntStack` aufgerufen.

(c) Welche allgemeinen Eigenschaften werden durch die Methoden `foo` und `bar` realisiert? Unter welchem Namen ist diese Datenstruktur allgemein bekannt?

Lösungsvorschlag

`foo()` Legt das Objekt auf den Stack `a`. Das Objekt wird in die Warteschlange eingereiht. Die Methode müsste eigentlich `enqueue()` heißen.

`bar()` Verschiebt alle Objekte vom Stack `a` in umgekehrter Reihenfolge in den Stack `b`, aber nur dann, wenn Stack `b` leer ist. Entfernt dann den obersten Wert aus dem Stack `b` und gibt ihn zurück. Das zuerst eingereihte Objekt wird aus der Warteschlange entnommen. Die Methode müsste eigentlich `dequeue()` heißen.

Die Datenstruktur ist unter dem Namen Warteschlange oder Queue bekannt

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Java-Klasse Stack“ (46115-2021-F.T2-TA2-A2)

Stapel (Stack)
Implementierung in Java

Gegeben sei die folgende Java-Implementierung eines Stacks.

```
class Stack {
  private Item head;

  public Stack() {
    head = null;
  }

  public void push(int val) {
    if (head == null) {
      head = new Item(val, null);
    } else {
      head = new Item(val, head);
    }
  }

  public int pop() {
    // ...
  }

  public int size() {
    // ...
  }

  public int min() {
    // ...
  }

  class Item {
    private int val;
    private Item next;

    public Item(int val, Item next) {
      this.val = val;
      this.next = next;
    }
  }
}
```

(a) Implementieren Sie die Methode `pop` in einer objektorientierten Programmiersprache Ihrer Wahl, die das erste Item des Stacks entfernt und seinen Wert zurückgibt. Ist kein Wert im Stack enthalten, so soll dies mit einer `IndexOutOfBoundsException` oder Ähnlichem gemeldet werden.

Beschreiben Sie nun jeweils die notwendigen Änderungen an den bisherigen Implementierungen, die für die Realisierung der folgenden Methoden notwendig sind.

Lösungsvorschlag

```
public int pop() {
  if (head != null) {
    int val = head.val;
```

```
      size--;
      head = head.next;
      return val;
    } else {
      throw new IndexOutOfBoundsException("The stack is empty");
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java`

(b) `size` gibt in Laufzeit $\mathcal{O}(1)$ die Anzahl der enthaltenen Items zurück.

Lösungsvorschlag

```
  public void push(int val) {
    if (head == null) {
      head = new Item(val, null);
    } else {
      head = new Item(val, head);
    }
    if (min > val) {
      min = val;
    }
    size++;
  }

  public int pop() {
    if (head != null) {
      int val = head.val;
      size--;
      head = head.next;
      return val;
    } else {
      throw new IndexOutOfBoundsException("The stack is empty");
    }
  }

  public int size() {
    return size;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java`

(c) `min` gibt (zu jedem Zeitpunkt) in Laufzeit $\mathcal{O}(1)$ den Wert des kleinsten Elements im Stack zurück.

Lösungsvorschlag

```
public void push(int val) {
  if (head == null) {
    head = new Item(val, null);
  } else {
    head = new Item(val, head);
  }
  if (min > val) {
    min = val;
```

```
    }
    size++;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java`

```
  public int min() {
    return min;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java`

Sie dürfen jeweils alle anderen angegebenen Methoden der Klasse verwenden, auch wenn Sie diese nicht implementiert haben. Sie können anstelle von objektorientiertem Quellcode auch eine informelle Beschreibung Ihrer Änderungen angeben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Reiseunternehmen“ (46116-2010-F.T1-A1)

Warteschlange (Queue)
Klassendiagramm

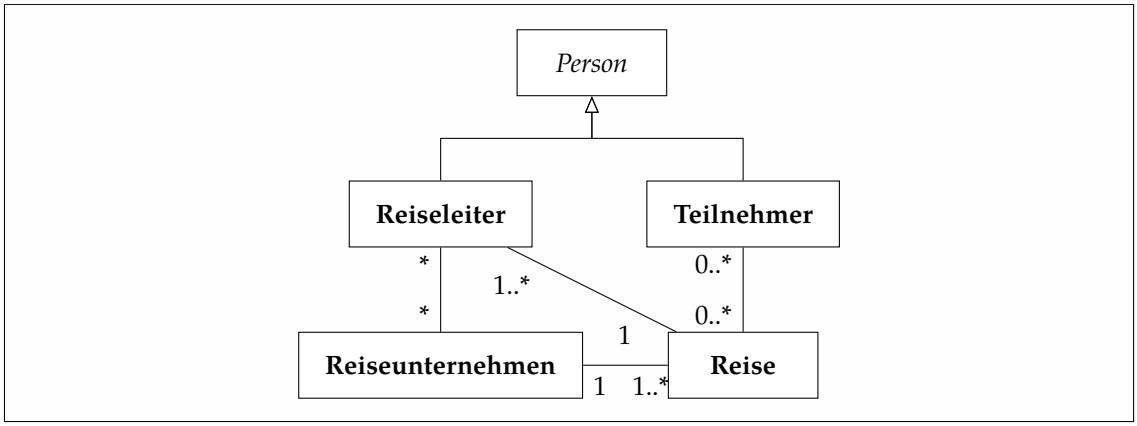
Es sei folgender Sachverhalt gegeben:

Ein Reiseunternehmen bietet verschiedene Reisen an. Dazu beschäftigt es eine Reihe von Reiseleitern, wobei eine Reise von mindestens einem Reiseleiter geleitet wird. Da Reiseleiter freiberuflich arbeiten, können sie bei mehreren Reiseunternehmen Reisen leiten.

An einer Reise können mehrere Teilnehmer teilnehmen, ein Teilnehmer kann auch an verschiedenen Reisen teilnehmen.

(a) Modellieren Sie diesen Sachverhalt in einem UML-Klassendiagramm. Für Teilnehmer und Reiseleiter sollen Sie dabei eine abstrakte Oberklasse definieren. Achten Sie dabei auf die Multiplizitäten der Assoziationen. Sie müssen keine Attribute bzw. Methoden angeben.

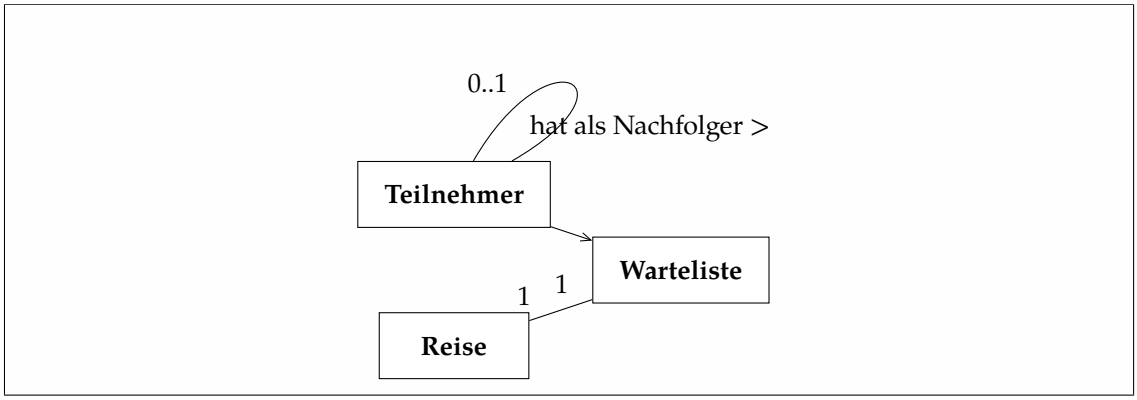
Lösungsvorschlag



(b) Eine Reise kann jedoch nur mit einer begrenzten Kapazität angeboten werden, das heißt, zu einer bestimmten Reise kann nur eine begrenzte Anzahl von Teilnehmern assoziiert werden. Als Ausgleich soll pro Reise eine Warteliste verwaltet werden.

Modellieren Sie diesen erweiterten Sachverhalt in einem neuen Diagramm. Nicht veränderte Klassen brauchen nicht noch einmal angegeben werden. Beachten Sie dabei, dass die Reihenfolge bei einer Warteliste eine Rolle spielt.

Lösungsvorschlag



(c) Implementieren Sie die in Aufgabenteil b) modellierten Klassen in Java. Fügen Sie eine Methode hinzu, die einen Teilnehmer von einer Reise entfernt. Dabei soll automatisch der erste Platz der Warteliste zu einem Reiseteilnehmer werden, wenn die Warteliste nicht leer ist. Achten Sie auf die Navigierbarkeit Ihrer Assoziationen. Sie können davon ausgehen, dass die Methode nur mit Teilnehmern aufgerufen wird, die in der Tat Teilnehmer der Reise sind.

Implementierung in Java

```
public class Teilnehmer {
  Teilnehmer nächster;

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2010/fruehjahr/reiseunternehmen/Teilnehmer.java`

```
/**
 * Diese Klasse ist eine Implementation einer einfach verketteten Liste. Sie
 * wird einerseit in der Klasse {@link Reise} genutzt, um die Reiseteilnehmer zu
 * speichern, andererseits um eine Warteliste darauf aufbauen zu könnten.
 */
public class TeilnehmerListe {

}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_46116/jahr_2010/fruehjahr/reiseunternehmen/TeilnehmerListe.java`

```
/**
 * Eine Reise kann jedoch nur mit einer begrenzten Kapazität angeboten
 * werden, das heißt, zu einer bestimmten Reise kann nur eine begrenzte
 * Anzahl von Teilnehmern assoziiert werden. Als Ausgleich soll pro
 * Reise eine Warteliste verwaltet werden.
 *
 * Modellieren Sie diesen erweiterten Sachverhalt in einem neuen
 * Diagramm. Nicht veränderte Klassen brauchen nicht noch einmal
 * angegeben werden. Beachten Sie dabei, dass die Reihenfolge bei einer
 * Warteliste eine Rolle spielt.
 *
 * Implementieren Sie die in Aufgabenteil b) modellierten Klassen in
 * Java. Fügen Sie eine Methode hinzu, die einen Teilnehmer von einer
 * Reise entfernt. Dabei soll automatisch der erste Platz der Warteliste
 * zu einem Reiseteilnehmer werden, wenn die Warteliste nicht leer ist.
 * Achten Sie auf die Navigierbarkeit Ihrer Assoziationen. Sie können
 * davon ausgehen, dass die Methode nur mit Teilnehmern aufgerufen wird,
 * die in der Tat Teilnehmer der Reise sind.
 */
public class Warteliste extends TeilnehmerListe {

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2010/fruehjahr/reiseunternehmen/Warteliste.java`

```
public class Reise {
```

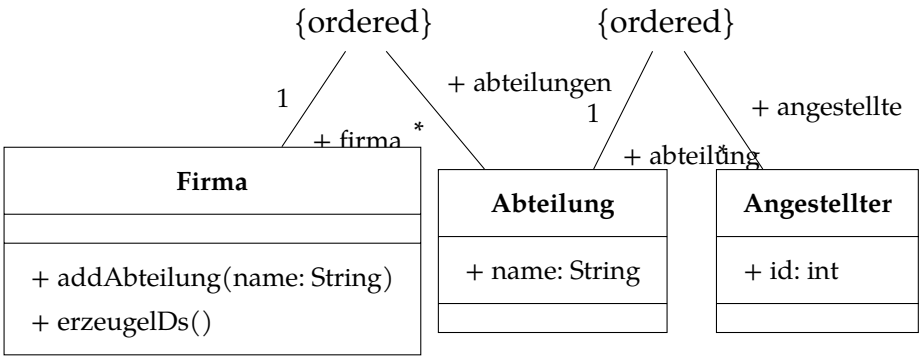
```
    Teilnehmer teilnehmer;

    Warteliste warteliste;

    void entferneTeilnehmer(Teilnehmer teilnehmer) {

    }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2010/fruehjahr/reiseunternehmen/Reise.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2010/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Firmenstruktur“ (46116-2011-F.T1-TA2-A1)

Firmenstruktur

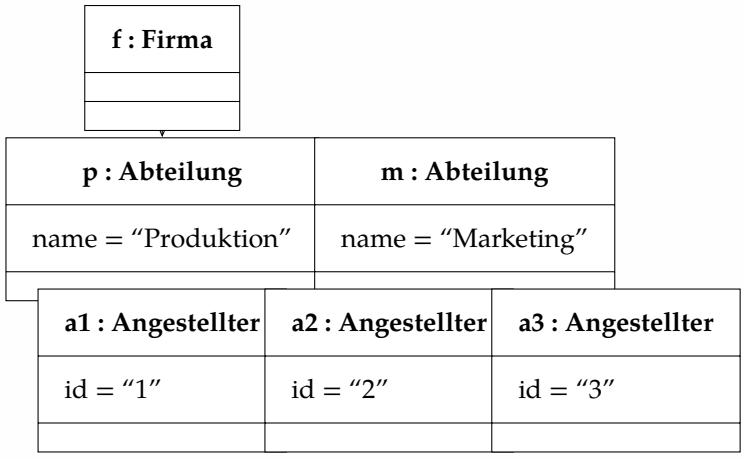


Eine Firma besteht aus `null` oder mehr Abteilungen, von denen jede `null` oder mehr Angestellte hat. Da sowohl die Abteilungen als auch deren Angestellten geordnet sind, sind Angestellte insgesamt geordnet. Sie haben durchgehende, ganzzahlige IDs, die bei 1 beginnen.

(a) Erstellen Sie exemplarisch ein Objektdiagramm: Stellen Sie eine Firma mit dem Instanznamen `f` und den zwei Abteilungen „Produktion“ (Name `p`) und „Marketing“ (Name `m`) dar. Die Produktion hat zwei Angestellte, Marketing hat einen Angestellten. Die Angestellten haben die Namen `a1`, `a2`, und `a3`.

Lösungsvorschlag

Die Instanzbezeichnung müsste noch unterstrichen werden. Das geht aber leider mit TikZ-UML nicht.



(b) Implementieren Sie das Klassendiagramm in Java oder in einer anderen geeigneten objektorientierten Programmiersprache Ihrer Wahl. Beachten Sie, dass die Assoziationen bidirektional und geordnet sind. Die beiden Methoden der Klasse `Firma` sollen dabei folgendes Verhalten haben:

Die Methode `erzeugeIDs` sorgt dafür, dass die IDs wieder korrekt zugewiesen sind. Die alten IDs können beliebig geändert werden, solange das Endergebnis wieder den obenstehenden Kriterien genügt.

Lösungsvorschlag

```
import java.util.ArrayList;
import java.util.List;

public class Firma {

  List<Abteilung> abteilungen;

  public Firma() {
    abteilungen = new ArrayList<Abteilung>();
  }

  public void addAbteilung(String name) {
    for (Abteilung abteilung : abteilungen) {
      if (abteilung.name.equals(name)) {

↪   System.out.println("Eine Abteilung mit diesem Namen gibt es bereits schon.");
        return;
      }
      abteilungen.add(new Abteilung(name));
    }
  }

  public void erzeugeIDs() {
    int idZähler = 1;
    for (Abteilung abteilung : abteilungen) {
      for (Angestellter angestellter : abteilung.angestellte) {
        angestellter.id = idZähler;
        idZähler++;
      }
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2011/fruehjahr/Firma.java`

```
import java.util.ArrayList;
import java.util.List;

public class Abteilung {
  public String name;

  public List<Angestellter> angestellte;

  public Abteilung(String name) {
    this.name = name;
    angestellte = new ArrayList<Angestellter>();
  }
}
```

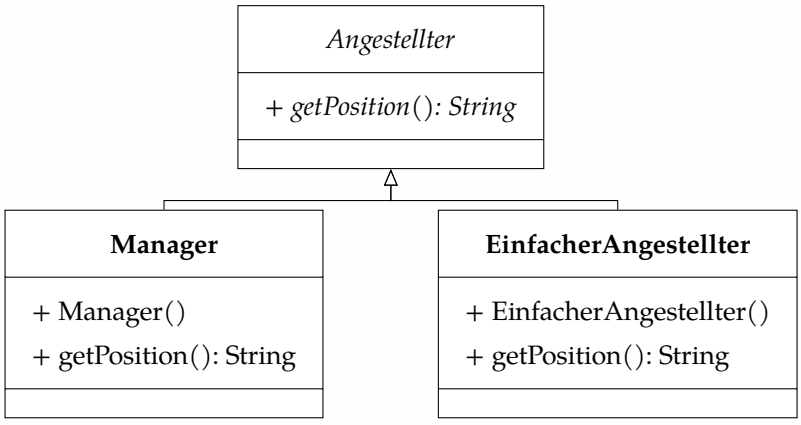
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2011/fruehjahr/Abteilung.java`

```java
import java.util.List;

public class Angestellter {
  public int id;

  public Firma firma;

  public List<Angestellter> angestellte;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2011/fruehjahr/Angestellter.java`

(c) Angestellte sollen in Manager und einfache Angestellte unterteilt werden. Zeichnen Sie ein Klassendiagramm mit der Oberklasse `Angestellter` und den zwei Unterklassen `Manager` und `EinfacherAngestellter`. Die Klasse `Angestellter` soll nicht instantiierbar sein und erzwingen, dass die Methode `getPosition()` (öffentlich, ohne Argumente, Rückgabewert `String`) von allen konkreten Unterklassen implementiert wird. `Manager` und `EinfacherAngestellter` sollen instantiierbar sein.

Lösungsvorschlag



(d) Wie lautet der Fachbegriff dafür, dass eine Methode in einer Klasse und in deren Unterklassen dieselbe Signatur hat, aber in den Unterklassen unterschiedlich implementiert ist?

Lösungsvorschlag

Abstrakte Methode

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2011/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Klassen „QueueElement“ und „Queue““ (66115-2007-F.T1-A7)

Implementieren Sie die angegebenen Methoden einer Klasse `Queue` für Warteschlangen. Eine Warteschlange soll eine unbeschränkte Anzahl von Elementen aufnehmen können. Elemente sollen am Ende der Warteschlange angefügt und am Anfang aus ihr entfernt werden. Sie können davon ausgehen, dass ein Klasse `QueueElement` mit der folgenden Schnittstelle bereits implementiert ist .

```
class QueueElement {

  private QueueElement next;
  private Object contents;

  QueueElement(Object contents) {
    this.contents = contents;
  }

  Object getContents() {
    return contents;
  }

  QueueElement getNext() {
    return next;
  }

  void setNext(QueueElement next) {
    this.next = next;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/QueueElement.java`

Von der Klasse `Queue` ist folgendes gegeben:

```
class Queue {
  QueueElement first;
  QueueElement last;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java`

(a) Schreiben Sie eine Methode `void append (Object contents)` , die ein neues Objekt in der Warteschlange einfügt.

Lösungsvorschlag

```
public void append(Object contents) {
  QueueElement newElement = new QueueElement(contents);
  if (first == null) {
    first = newElement;
    last = newElement;
  } else {
    // neues Element hinten anhängen
    last.setNext(newElement);
    // angehängtes Element ist Letztes
    last = last.getNext();
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java

(b) Schreiben Sie eine Methode `Object remove()`, die ein Element aus der Warteschlange entfernt und dessen Inhalt zurückliefert. Berücksichtigen Sie, dass die Warteschlange leer sein könnte.

Lösungsvorschlag

```
public Object remove() {
  Object tmp = null;
  if (first != null) {
    // Dein Inhalt des ersten Elements temporär speichern
    tmp = first.getContents();
    // Das erste Element aus der Schlange nehmen
    first = first.getNext();
  }
  // Den Inhalt des gelöschten Elements ausgeben bzw . null
  return tmp;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java

(c) Schreiben Sie eine Methode `boolean isEmpty()`, die überprüft, ob die Warteschlange leer ist.

Lösungsvorschlag

```
public boolean isEmpty() {
  return (first == null);
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java

Klasse Queue

```
class Queue {
  QueueElement first;
  QueueElement last;

  public void append(Object contents) {
    QueueElement newElement = new QueueElement(contents);
    if (first == null) {
      first = newElement;
      last = newElement;
    } else {
      // neues Element hinten anhängen
      last.setNext(newElement);
      // angehängtes Element ist Letztes
      last = last.getNext();
    }
  }

  public Object remove() {
```

```
    Object tmp = null;
    if (first != null) {
      // Dein Inhalt des ersten Elements temporär speichern
      tmp = first.getContents();
      // Das erste Element aus der Schlange nehmen
      first = first.getNext();
    }
    // Den Inhalt des gelöschten Elements ausgeben bzw . null
    return tmp;
  }

  public boolean isEmpty() {
    return (first == null);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java`

Tests

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class QueueTest {

  @Test
  public void methodAppend() {
    Queue queue = new Queue();
    assertEquals(true, queue.isEmpty());
    queue.append(1);
    assertEquals(false, queue.isEmpty());
  }

  @Test
  public void methodRemove() {
    Queue queue = new Queue();
    queue.append(1);
    queue.append(2);
    queue.append(3);

    assertEquals(1, queue.remove());
    assertEquals(2, queue.remove());
    assertEquals(3, queue.remove());
    assertEquals(null, queue.remove());
  }

  @Test
  public void methodIsEmpty() {
    Queue queue = new Queue();
    assertEquals(true, queue.isEmpty());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/QueueTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2007/03/Thema-1/Aufgabe-7.tex

Examensaufgabe „DoubleLinkedList“ (66115-2021-F.T2-TA2-A2)

Doppelt-verkettete Liste

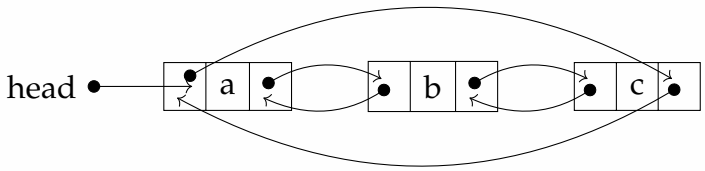
Gegeben sei die folgende Java-Implementierung einer doppelt-verketteten Liste.

```
class DoubleLinkedList {
  private Item head;

  public DoubleLinkedList() {
    head = null;
  }

  public Item append(Object val) {
    if (head == null) {
      head = new Item(val, null, null);
      head.prev = head;
      head.next = head;
    } else {
      Item item = new Item(val, head.prev, head);
      head.prev.next = item;
      head.prev = item;
    }
    return head.prev;
  }

  public Item search(Object val) {
    // ...
  }

  public void delete(Object val) {
    // ...
  }

  class Item {
    private Object val;
    private Item prev;
    private Item next;

    public Item(Object val, Item prev, Item next) {
      this.val = val;
      this.prev = prev;
      this.next = next;
    }
  }
}
```

(a) Skizzieren Sie den Zustand der Datenstruktur nach Aufruf der folgenden Befehlssequenz. Um Variablen mit Zeigern auf Objekte darzustellen, können Sie mit dem Variablennamen beschriftete Pfeile verwenden.

```
DoubleLinkedList list = new DoubleLinkedList();
list.append("a");
list.append("b");
list.append("c");
```

Lösungsvorschlag



(b) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `search`, die zu einem gegebenen Wert das Item der Liste mit dem entsprechenden Wert, oder `null` falls der Wert nicht in der Liste enthalten ist, zurückgibt.

Lösungsvorschlag

```
public Item search(Object val) {
  Item item = null;
  if (head != null) {
    item = head;
    do {
      if (item.val.equals(val)) {
        return item;
      }
      item = item.next;
    } while (!item.equals(head));
  }
  return null;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/DoubleLinkedList.java`

(c) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `delete`, die das erste Vorkommen eines Wertes aus der Liste entfernt. Ist der Wert nicht in der Liste enthalten, terminiert die Methode „stillschweigend“, ðohne Änderung der Liste und ohne Fehlermeldung. Sie dürfen die Methode `search` aus Teilaufgabe b) verwenden, auch wenn Sie sie nicht implementiert haben.

Lösungsvorschlag

```
public void delete(Object val) {
  Item item = search(val);
  if (item != null) {
    if (head.next.equals(head)) {
      head = null;
    } else {
      if (item.equals(head)) {
        head = item.next;
      }
      item.prev.next = item.next;
      item.next.prev = item.prev;
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/DoubleLinkedList.java`

(d) Beschreiben Sie die notwendigen Änderungen an der Datenstruktur und an den bisherigen Implementierungen, um eine Methode `size`, die die Anzahl der enthaltenen Items zurück gibt, mit Laufzeit $\mathcal{O}(1)$ zu realisieren.

Lösungsvorschlag

In der Klasse wird ein Zähler eingefügt, der bei jedem Aufruf der Methode `append` um eins nach oben gezählt wird und bei jedem erfolgreichen Löschen in der `delete`-Methode um eins nach unten gezählt wird. Mit `return` kann der Zählerstand in $\mathcal{O}(1)$ ausgegeben werden. Dazu müsste ein Getter zum Ausgeben implementiert werden. Die Datenstruktur bleibt unverändert.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Getränkelieferservice“ (66116-2012-H.T2-TA2-A1)

Ein Getränkelieferservice verwaltet die Bestellungen verschiedener Kunden. Die folgenden Teilaufgaben sind in einer objektorientierten Programmiersprache zu lösen (die verwendete Sprache ist vorab anzugeben.).

(a) Implementieren Sie eine Klasse `Kasten` zur Beschreibung eines Getränkekastens mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine Realisierung als Objekt- oder Klassenfeld sinnvoll ist.

- Es existiert ein einheitliches Kastenpfand in Höhe von 1,50 Euro.
- Für alle Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das jedoch von Kasten zu Kasten verschieden sein kann.
- Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist, sind die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen sollen in einem 2-dimensionalen Array abgelegt werden.

Geben Sie für die Klasse `Kasten` einen geeigneten Konstruktor an. Ergänzen Sie in der Klasse `Kasten` eine Objektmethode zur Berechnung des Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand.

(b) Schreiben Sie eine Klasse `Bestellung`. Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden, die über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert werden, sowie eine einfach verkettete Liste der bestellten Getränkekästen. Die Klasse Bestellung soll weiterhin eine Methode beinhalten, die den Gesamtpreis der Bestellung ermittelt.

(c) Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt, die zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1 Getränkekasten mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7 Euro auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das Kastenpfand 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf der Standardausgabe ausgegeben werden.

1,0	1,0	0,7
1,0	0,7	0
0,7	0	0

Lösungsvorschlag

```
/**
 * „Implementieren Sie eine Klasse Kasten zur Beschreibung eines
↪  Getränkekastens
 * mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine
 * Realisierung als Objekt- oder Klassenfeld sinnvoll ist."
```

```
 */
public class Kasten {
  /**
   * Wir verwenden static, also ein sogenanntes Klassenfeld, da das
↪  Kastenpfand
   * für alle Kästen gleich ist: „Es existiert ein einheitliches Kastenpfand
↪  in
   * Höhe von 1,50 Euro."
   */
  static double kastenPfad = 1.5;

  /**
   * Wir verwenden ein Objektfeld, d.h. ein nicht statisches Feld: „Für alle
   * Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das
↪  jedoch
   * von Kasten zu Kasten verschieden sein kann."
   */
  double flaschenPfad;

  /**
   * „Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist,
↪  sind
   * die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die
   * Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen
↪  sollen
   * in einem 2-dimensionalen Array abgelegt werden."
   */
  double[][] flaschen;

  /**
   * „sowie eine einfach verkettete Liste der bestellten Getränkekästen. "
   */
  Kasten nächsterKasten = null;

  /**
   * „Geben Sie für die Klasse Kasten einen geeigneten Konstruktor an."
   *
   * @param flaschen     Die Belegung des Kasten mit Flaschen als
   *                     zweidimensionales Feld der Flaschenpreise ohne
   *                     Flaschenpfand.
   * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle Flaschen
↪  in
   *                     diesem Kasten gleich ist.
   */
  public Kasten(double[][] flaschen, double flaschenPfad) {
    this.flaschen = flaschen;
    this.flaschenPfad = flaschenPfad;
  }

  /**
   * „Ergänzen Sie in der Klasse Kasten eine Objektmethode zur Berechnung des
   * Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand."
   *
   * @return Der Gesamtpreis des Getränkekastens inklusive Kasten- und
```

```
     *         Flaschenpfand.
     */
    double berechneGesamtPreis() {
      double gesamtPreis = kastenPfad;
      for (int i = 0; i < flaschen.length; i++) {
        double[] reihe = flaschen[i];
        for (int j = 0; j < reihe.length; j++) {
          double flaschenPreis = flaschen[i][j];
          // Nur im Kasten vorhandene Flaschen kosten auch Flaschenpfand.
          if (flaschenPreis > 0)
            gesamtPreis += flaschenPfad + flaschen[i][j];
        }
      }
      return gesamtPreis;
    }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2012/herbst/getraenke/Kasten.java`

```
import java.text.DecimalFormat;

/**
 * „Schreiben Sie eine Klasse Bestellung"
 */
public class Bestellung {
    /**
     * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden,
↪    die
     * über den Konstruktoraufruf erstellt wird."
     */
    int bestellNummer;

    /**
     * „Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert
↪    werden."
     */
    String kundenName;

    /**
     * „sowie eine einfach verkettete Liste der bestellten Getränkekästen. "
     */
    Kasten kästen = null;

    /**
     * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden,
↪    die
     * über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder
↪    Bestellung
     * der Name des Kunden gespeichert werden."
     *
     * @param bestellNummer Die Nummer der Getränkebestellung.
     * @param kundenName    Der Name des/der KundenIn.
     */
    public Bestellung(int bestellNummer, String kundenName) {
```

```
    this.bestellNummer = bestellNummer;
    this.kundenName = kundenName;
  }

  /**
   * „Die Klasse Bestellung soll weiterhin eine Methode beinhalten, die den
   * Gesamtpreis der Bestellung ermittelt."
   *
   * @return Der Gesamtpreis der Getränkebestellung.
   */
  double berechneGesamtPreis() {
    double gesamtPreis = 0;
    Kasten kasten = kästen;
    while (kasten != null) {
      gesamtPreis += kasten.berechneGesamtPreis();
      kasten = kasten.nächsterKasten;
    }
    return gesamtPreis;
  }

  /**
   * Nicht verlangt. Könnte auch in die Test-Methode geschrieben werden.
   *
   * @param flaschen     Die Belegung des Kasten mit Flaschen als
   *                     zweidimensionales Feld der Flaschenpreise ohne
   *                     Flaschenpfand.
   * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle Flaschen
↪   in
   *                     diesem Kasten gleich ist.
   */
  void bestelleKasten(double[][] flaschen, double flaschenPfad) {
    Kasten bestellterKasten = new Kasten(flaschen, flaschenPfad);
    if (kästen == null) {
      kästen = bestellterKasten;
      return;
    }
    Kasten kasten = kästen;

    Kasten letzterKasten = null;
    while (kasten != null) {
      letzterKasten = kasten;
      kasten = kasten.nächsterKasten;
    }
    letzterKasten.nächsterKasten = bestellterKasten;
  }

  /**
   * Kleines Schmankerl. Nicht verlangt. Damit wir nicht 9.899999999999999
↪   als
   * Aufgabe bekommen.
   *
   * @param preis Ein Preis als Gleitkommazahl.
   *
   * @return Der Preis als Text mit zwei Stellen nach dem Komma.
```

```
   */
  static String runde(double preis) {
    DecimalFormat df = new DecimalFormat("#.##");
    df.setMinimumFractionDigits(2);
    return df.format(preis);
  }

  /**
   * Die main-Methode soll hier als Testmethode verwendet werden:
   *
   * „Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt,
↪   die
   * zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1
↪   Getränkekasten
   * mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in
   * Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7
↪   Euro
   * auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das
   * Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das
↪   Kastenpfand
   * 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf
↪   der
   * Standardausgabe ausgegeben werden."
   *
   * @param args Kommandozeilenargumente, die uns nicht zu interessieren
↪   brauchen.
   */
  public static void main(String[] args) {
    Bestellung bestellung = new Bestellung(1, "Hermine Bschlangaul");

    // Müsste eigentlich nicht mehr gesetzt werden, da wir es schon in der
    // Klassendefinition gesetzt haben.
    Kasten.kastenPfad = 1.50;

    bestellung.bestelleKasten(new double[][] { { 0.75 } }, 0.15);
    bestellung.bestelleKasten(new double[][] { { 1.0, 1.0, 0.7 }, { 1.0, 0.7,
↪   0 }, { 0.7, 0, 0 } }, 0.15);

    // Oder kürzer
    // bestellung.bestelleKasten(new double[][] { { 1, 1, .7 }, { 1, .7 }, {
↪   .7 } }, .15);

    // Gegenrechnung:
    // 1 x 0.75 = 0.75
    // 3 x 1.00 = 3.00
    // 3 x 0.70 = 2.10
    // 7 x 0.15 = 1.05 (Flaschenpfad)
    // 3 x 1.50 = 3.00 (Kastenpfand)
    // ----
    // 9.90
    System.out.println("Der Gesamtpreis der Getränkebestellung beträgt: " +
↪   runde(bestellung.berechneGesamtPreis()) + " €");
  }
}
```

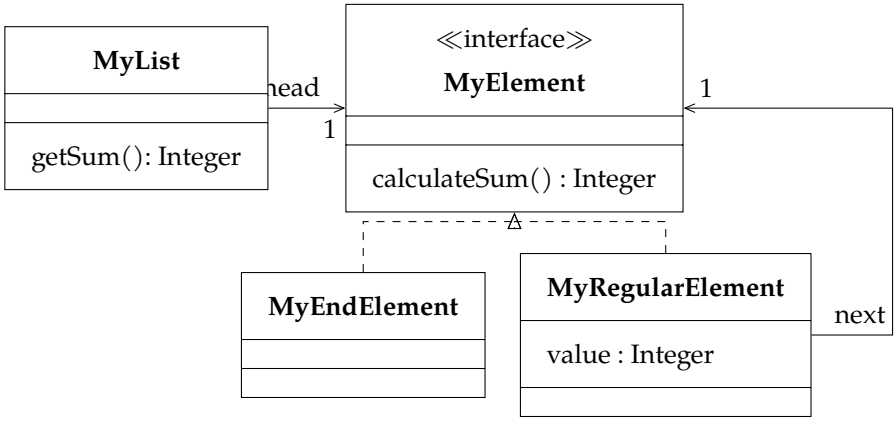
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2012/herbst/getraenke/Bestellung.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2012/09/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „MyList Kompositium“ (66116-2021-F.T1-TA1-A5)

Einfach-verkettete Liste

Die folgende Abbildung stellt den Entwurf der Implementierung einer verketteten Liste dar, welche Integer-Werte als Elemente enthalten kann.



Die Klassse `MyList` stellt die Methode `getSum()` zur Verfügung, welche die Summe über alle in einer Liste befindlichen Elemente berechnet. Ein Ausschnitt der Implementierung sieht folgendermaßen aus:

```
public class MyList {
  private MyElement head;

  public MyList() {
    this.head = new MyEndElement();
  }

  public int getSum() {
    // ..
  }
}
```

Gehen Sie im Folgenden davon aus, dass bereits Methoden existieren, welche Elemente in die Liste einfügen können.

(a) Implementieren Sie in einer objektorientierten Programmiersprache Ihrer Wahl, z. B. Java, die Methode `calculateSum()` der Klassen `MyEndElement` und `MyRegularElement`, so dass rekursiv die Summe der Elemente der Liste berechnet wird.

Als Abbruchbedingung darf hierbei nicht das Feld `MyRegluarElement.next` auf den Wert `null` überprüft werden.

Hinweis: Gehen Sie davon aus, die Implementierung von `MyList` garantiert, dass `MyRegluarElement.next` niemals den Wert `null` annimmt, sondern das letzte hinzugefügte `MyRegularElement` auf eine Instanz der Klasse `MyEndElement` verweist. Es gibt immer nur eine Instanz der Klasse `MyEndElement` in einer Liste.

Hinweis: Achten Sie auf die Angabe einer korrekten Methodensignatur.

Lösungsvorschlag

```java
int calculateSum() {
  return value + next.calculateSum();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyElement.java`

```java
int calculateSum() {
  return 0;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyEndElement.java`

(b) Nennen Sie den Namen des Entwurfsmusters, auf welchem das oben gegebene Klassendiagramm basiert, und ordnen Sie dieses in eine der Kategorien von Entwurfsmustern ein.

Hinweis: Es genügt die Angabe eines Musters, falls Sie mehrere Muster identifizieren sollten.

Lösungsvorschlag

Kompositium (Strukturmuster)

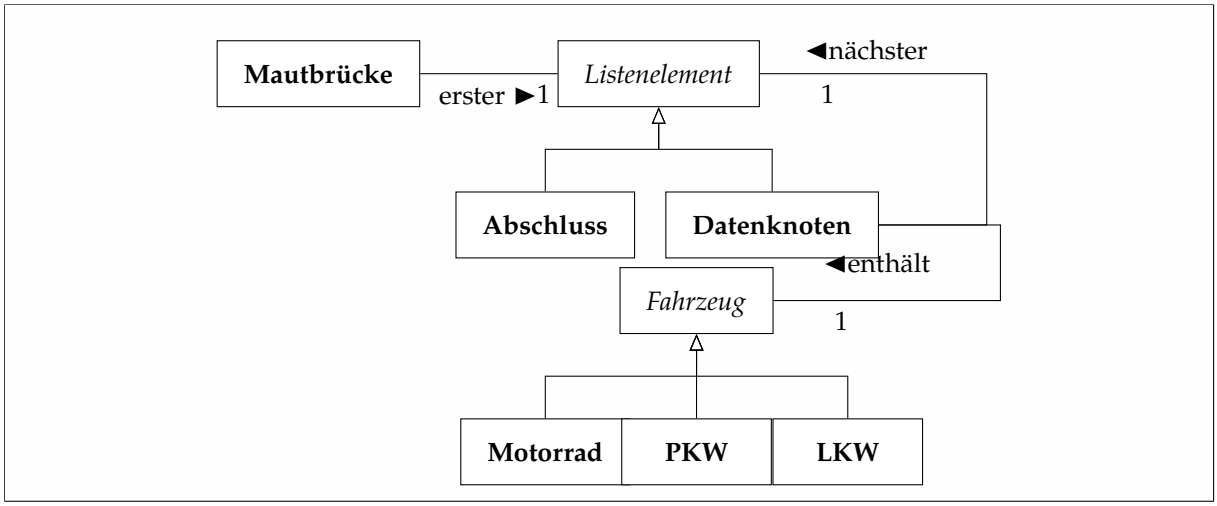
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-5.tex`

Übungsaufgabe „Maut“ (Einfach-verkettete Liste, Klassendiagramm, Kompositum (Composite))

Einfach-verkettete Liste
Klassendiagramm
Kompositum (Composite)

Für die Umsetzung der Maut auf deutschen Autobahnen soll eine Java-basierte Lösung entworfen werden. Dazu sollen alle Fahrzeuge, die von einer Mautbrücken erfasst werden, in einer *einfach verketteten Liste* abgelegt werden. Um einen besseren Überblick über die Einnahmen zu erhalten, soll zwischen *LKWs*, *PKWs* und *Motorrädern* unterschieden werden. Als Informatiker schlagen Sie eine *heterogene Liste* zur Realisierung vor. Notieren Sie unter Verwendung des *Entwurfsmusters Kompositum* ein entsprechendes *Klassendiagramm* zur Realisierung der Lösung für eine Mautbrücke. Auf die Angabe von Attributen und Methoden kann verzichtet werden. Kennzeichen Sie in Ihrem Klassendiagramm die *abstrakten Klassen* und benennen Sie die bestehenden *Beziehungen*.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/10_Listen/Aufgabe_Maut.tex`

Übungsaufgabe „Wörterbuch“ (Einfach-verkettete Liste, Kompositum (Composite))

Einfach-verkettete Liste
Kompositum (Composite)

Erstellen Sie ein Deutsch-Englisch Wörterbuch. Verwenden Sie dazu eine einfach verkettete Liste mit Kompositum. Identifizieren Sie die benötigten Klassen, legen Sie das Wörterbuch an und implementieren Sie anschließend die geforderten Methoden.

- Ein Listenelement, welches immer jeweils auf seinen Nachfolger verweisen kann, enthält jeweils einen Eintrag des Wörterbuchs. Ein Eintrag besteht aus dem deutschen und dem zugehörigen englischen Wort. Diese können natürlich jeweils zurückgegeben werden.

- Mit der Methode `einfuegen (String deutsch, String englisch)` soll ein neuer Eintrag in das Wörterbuch eingefügt werden können. Wie in jedem Wörterbuch müssen die (deutschen) Einträge jedoch alphabetisch sortiert sein, sodass nicht an einer beliebigen Stelle eingefügt werden kann. Um die korrekte Einfügeposition zu finden, ist das Vergleichen von Strings notwendig. Recherchieren Sie dazu, wie die Methode `compareTo()` in Java funktioniert!

- Der Aufruf der Methode `uebersetze(String deutsch)` auf der Liste soll nun für ein übergebenes deutsches Wort die englische Übersetzung ausgeben.

Lösungsvorschlag

Klasse `WörterbuchEintrag`

Die abstrakte Klasse im Kompositumentwurfsmuster von der sowohl die primitive Klasse als auch die Behälterklasse erben.

```
public abstract class WoerterbuchEintrag {
  protected WortPaar nächstes;

  protected WortPaar gibNächstes () {
    return nächstes;
  }

  protected void setzeNächstes (WortPaar wortPaar) {
    nächstes = wortPaar;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchEintrag.java`

Klasse `WortPaar`

Das Listenelement (die primitive Klasse im Kompositumentwurfsmuster).

```
public class WortPaar extends WoerterbuchEintrag {
  private final String deutsch;

  private final String englisch;
```

```
  public WortPaar(String deutsch, String englisch) {
    this.deutsch = deutsch;
    this.englisch = englisch;
  }

  public String gibDeutschesWort() {
    return deutsch;
  }

  public String gibEnglischesWort() {
    return englisch;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WortPaar.java`

Klasse `Wörterbuch`

Die Behälterklasse im Kompositumentwurfsmuster.

```
 */
public class Woerterbuch extends WoerterbuchEintrag {

  public void einfügen(String deutsch, String englisch) {
    WortPaar wort = new WortPaar(deutsch, englisch);

    // Spezialbehandlung, wenn vor das erste Wortpaar des Wörterbuchs eingefügt
    // werden muss.
    WortPaar kopf = gibNächstes();
    if (kopf == null || kopf.gibDeutschesWort().compareTo(wort.gibDeutschesWort())
↪   >= 0) {
      wort.setzeNächstes(kopf);
      setzeNächstes(wort);
      return;
    }
    WortPaar vergleichsWort = gibNächstes();
    while (vergleichsWort.gibNächstes() != null
        &&
↪   vergleichsWort.gibNächstes().gibDeutschesWort().compareTo(wort.gibDeutschesWort())
↪   < 0) {
      vergleichsWort = vergleichsWort.gibNächstes();
    }
    wort.setzeNächstes(vergleichsWort.gibNächstes());
    vergleichsWort.setzeNächstes(wort);
  }

  public String übersetze(String deutsch) {
    if (gibNächstes() == null) {
      return "Noch keine Wörter im Wörterbuch.";
    }
    WortPaar wort = gibNächstes();
    while (wort != null) {
```

```
      if (wort.gibDeutschesWort().equals(deutsch)) {
        return wort.gibEnglischesWort();
      }
      wort = wort.gibNächstes();
    }
    return "Es konnte keine passende Übersetzung gefunden werden";
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/Woerterbuch.java`

Test

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class WoerterbuchTest {

  @Test
  public void methodeÜbersetze() {
    Woerterbuch wörterbuch = new Woerterbuch();
    assertEquals("Noch keine Wörter im Wörterbuch.",
↪   wörterbuch.übersetze("Wassermelone"));
  }


  @Test
  public void methodeEinfügen() {
    Woerterbuch wörterbuch = new Woerterbuch();
    wörterbuch.einfügen("Wassermelone", "Watermelon");
    assertEquals("Watermelon", wörterbuch.übersetze("Wassermelone"));
  }

  @Test
  public void sortierung() {
    Woerterbuch wörterbuch = new Woerterbuch();
    wörterbuch.einfügen("Wassermelone", "Watermelon");
    wörterbuch.einfügen("Apfel", "Apple");
    wörterbuch.einfügen("Zitrone", "Lemon");
    wörterbuch.einfügen("Birne", "Pear");
    wörterbuch.einfügen("Klementine", "Clementine");

    WortPaar paar;
    paar = wörterbuch.gibNächstes();
    assertEquals("Apfel", paar.gibDeutschesWort());

    paar = paar.gibNächstes();
    assertEquals("Birne", paar.gibDeutschesWort());

    paar = paar.gibNächstes();
    assertEquals("Klementine", paar.gibDeutschesWort());
```

```
    paar = paar.gibNächstes();
    assertEquals("Wassermelone", paar.gibDeutschesWort());

    paar = paar.gibNächstes();
    assertEquals("Zitrone", paar.gibDeutschesWort());
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/10_Listen/Aufgabe_Woerterbuch.tex`

Übungsaufgabe „Tellerstapel-Biberschlagen“ (Warteschlange (Queue), Stapel (Stack))

Warteschlange (Queue)
Stapel (Stack)

Im Restaurant der Biberschule gibt es normalerweise zwei Warteschlangen: In der einen holen sich die kleinen Biber ihre hohen grünen Teller, in der anderen holen sich die großen Biber ihre flachen braunen Teller. Wegen Bauarbeiten kann es heute nur eine Warteschlange für alle Biber geben. Die Küchenbiber müssen deshalb einen Tellerstapel vorbereiten, der zur Schlange passt: Sie müssen die grünen und braunen Teller so stapeln, dass jeder Biber in der Schlange den passenden Teller bekommt. Schau dir zum Beispiel diese Warteschlange an. Für diese Warteschlange müssen die Teller so gestapelt sein.

Lösungsvorschlag

Daten, die mit Computerprogrammen verarbeitet werden sollen, müssen passend organisiert sein. Informatiker beschäftigen sich deshalb intensiv mit Datenstrukturen. Zwei einfache Datenstrukturen sind „Schlange” (Queue) und „Stapel” (Stack). Bei einer „Schlange” kann man nur auf die zuerst eingereihten Daten zugreifen (nach dem Prinzip FIFO: „first in, first out”). Bei einem „Stapel” kann man nur auf die zuletzt eingereihten Daten zugreifen (nach dem Prinzip LIFO: „last in, first out”). Die Datenstruktur der wartenden Biber ist eine „Schlange”. Die Datenstruktur der Teller ist ein „Stapel”.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/20_Warteschlange/Aufgabe_Tellerstapel-Biberschlagen.tex`

Übungsaufgabe „Informatik-Biber“ (Stapel (Stack))

Stapel (Stack)

Der Güterzug der Biberbahn wurde in der Wagenreihung `D-E-B-C-A` abgestellt: Die Lok kann vorwärts und rückwärts fahren und dabei beliebig viele Waggons ziehen und schieben. Jedes Mal, wenn ein Waggon angekoppelt oder ein Waggon abgekoppelt wird, zählt das als eine Rangieroperation. Wie viele Rangieroperationen sind mindestens nötig, um die Wagenreihung `A-B-C-D-E` herzustellen?

Lösungsvorschlag

Die Anzahl 8 ist richtig: Um einen Zug mit nur zwei Waggons umzuordnen, muss jeder der beiden Waggons einmal an- und einmal abgekoppelt werden, das sind vier Operationen. Bei dieser Aufgabe kann man die bereits geordneten Zugteile `D-E` und `B-C` als einzelne Waggons behandeln. Die ersten beiden umzuordnen, etwa `D-E` und `B-C`, erfordert also vier Operationen. Den so gewonnenen Zugteil `B-C-D-E` und den verbleibenden Waggon `A` umzuordnen erfordert weitere vier Operationen. Die Reihenfolge der Schritte mag varüeren, aber nur mit mehr Gleisen könnten Operationen eingespart werden.

Die zwei Abstellgleise können als Stapelspeicher (stacks) angesehen werden. Man kann Objekte hineintun und wieder herausholen – aber nicht in beliebiger Reihenfolge. Was zuletzt hineinkam (push), muss zuerst wieder heraus (pop). Stapelspeicher, manchmal auch Kellerspeicher genannt, werden von der Informatik in Programmen und Hardwareschaltungen für vielfältige Zwecke eingesetzt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/30_Stapel/Aufgabe_Informatik-Biber.tex`

Übungsaufgabe „Sackbahnhof“ (Stapel (Stack))

Stapel (Stack)

In einem Sackbahnhof mit drei Gleisen befinden sich in den Gleisen `S1` und `S2` zwei Züge jeweils mit Waggons für Zielbahnhof `A` und `B`. Gleis `S3` ist leer. Stellen Sie die Züge zusammen, die nur Waggons für einen Zielbahnhof enthalten! Betrachten Sie `S1`, `S2` und `S3` als Stapel und entwerfen Sie einen Algorithmus (Pseudocode genügt), der die Züge so umordnet, dass anschließend alle Waggons für `A` in `S1` und alle Waggons für `B` in `S2` stehen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/30_Stapel/Aufgabe_Sackbahnhof.tex`

Bäume

Examensaufgabe „Binäre Suchbäume, AVL-Bäume, Implementierung“ (46115-2010-F.T1-A5)

5. Datenstrukturen und Algorithmen: Binäre Suchbäume und AVL-Bäume

(a) Geben Sie jeweils eine Definition für binäre Suchbäume und AVL-Bäume an.

Lösungsvorschlag

Binärer Suchbaum Für jeden Knoten gilt: Ein Knoten enthält einen Schlüsselwert. Ein Knoten hat zwei Kindknoten: einen linken und einen rechten Kindknoten. Alle Schlüsselwerte des linken Teilbaums sind kleiner, alle Schlüsselwerte des rechten Teilbaums sind größer als der Wert des Knotens.

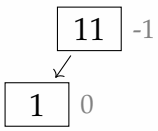
AVL-Baum Alle Eigenschaften des oben definierten binären Suchbaums gelten auch im AVL-Baum. Zusätzlich gilt: Die Differenz der Höhen des rechten und linken Teilbaums darf sich nie mehr als um eins unterscheiden.

(b) Zeichnen Sie einen AVL-Baum, der die folgenden Schlüssel enthält: 11, 1, 5, 37, 17, 29, 31, 3.

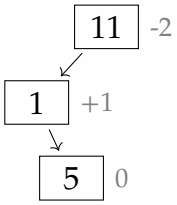
Lösungsvorschlag

Nach dem Einfügen von „11“:

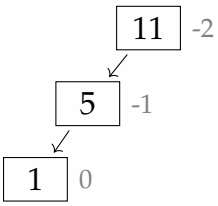
Nach dem Einfügen von „1“:



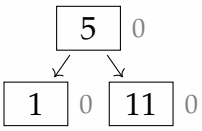
Nach dem Einfügen von „5“:



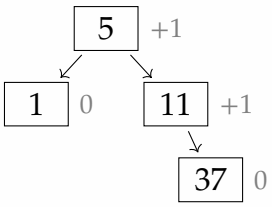
Nach der Linksrotation:



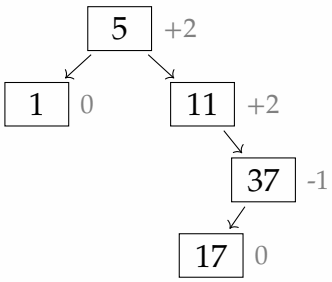
Nach der Rechtsrotation:



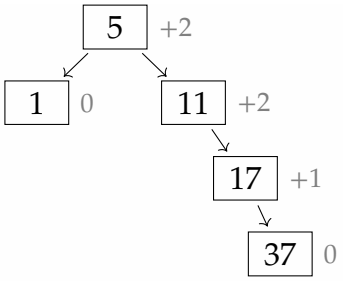
Nach dem Einfügen von „37“:



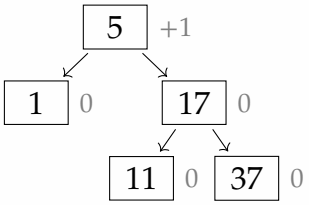
Nach dem Einfügen von „17“:



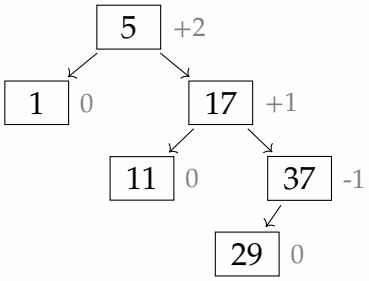
Nach der Rechtsrotation:



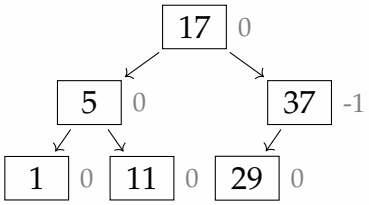
Nach der Linksrotation:



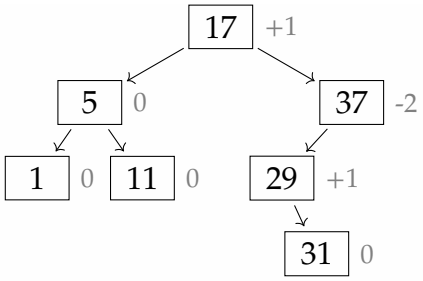
Nach dem Einfügen von „29“:



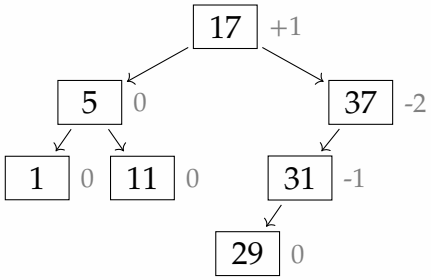
Nach der Linksrotation:



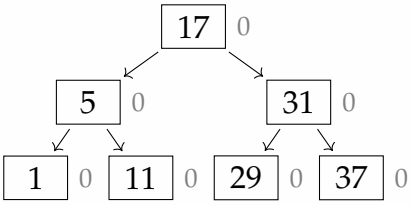
Nach dem Einfügen von „31“:



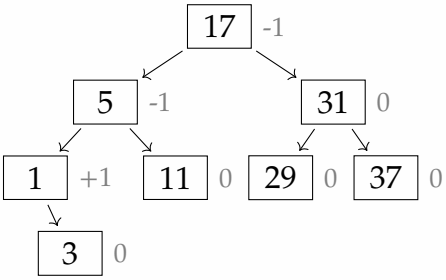
Nach der Linksrotation:



Nach der Rechtsrotation:



Nach dem Einfügen von „3“:



(c) Welche Zeitkomplexität haben die Operationen *Einfügen*, *Löschen* und *Suchen* auf AVL-Bäumen? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Für alle Operationen wird jeweils nur ein Pfad von der Wurzel bis zum gewünschten Knoten beschritten. Für alle Operation ist deshalb die maximale Höhe des Baums entscheidend. Da AVL-Bäume höhenbalanciert sind.

(d) Implementieren Sie die Datenstruktur AVL-Baum mit Schlüsseln vom Typ int (für Java oder entsprechende Datentypen für die anderen Programmiersprachen) in entweder Java, C++, Smalltalk oder Eiffel. Ihre Implementierung muss als einzige Operation die Methode suche bereitstellen, die einen Schlüssel als Parameter bekommt und

- true zurückliefert, wenn der gesuchte Schlüssel im Baum enthalten ist,
- ansonsten false.

Ihre Implementierung muss keine Konstruktoren oder andere Methoden zur Initialisierung bereitstellen. Desweiteren soll die Implementierung nur Schlüsselknoten berücksichtigen.

Kommentieren Sie Ihre Implementierung ausführlich. Geben Sie an, welche Programmiersprache Sie gewählt haben.

```
public class AVLBaum {

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2010/fruehjahr/AVLBaum.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2010/03/Thema-1/Aufgabe-5.tex`

Examensaufgabe „2-3-4-Baum“ (46115-2011-F.T1-A3)

B-Baum
AVL-Baum
B-Baum
AVL-Baum

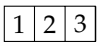
(a) Fügen Sie in einen anfangs leeren 2-3-4-Baum (B-Baum der Ordnung 4)[1] der Reihe nach die folgenden Schlüssel ein:

1, 2, 3, 5, 7, 8, 9, 4, 11, 12, 13, 6.

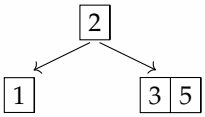
Dokumentieren Sie die Zwischenschritte so, dass die Entstehung des Baumes und nicht nur das Endergebnis nachvollziehbar ist.

Lösungsvorschlag

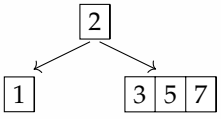
(i) 1, 2, 3 (Einfaches Einfügen):



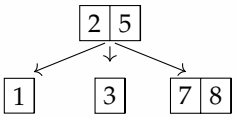
(ii) 5 (Split):



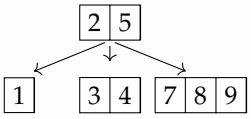
(iii) 7 (Einfaches Einfügen):



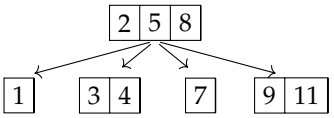
(iv) 8 (Split):



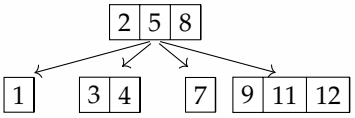
(v) 9, 4 (Einfaches Einfügen):



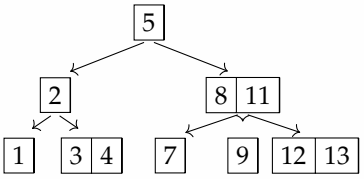
(vi) 11 (Split):



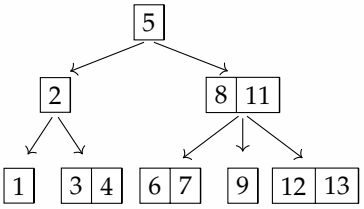
(vii) 12 (Einfaches Einfügen):



(viii) 13 (zwei Splits):



(ix) 6 (Einfaches Einfügen):



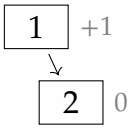
(b) Zeichnen Sie einen Rot-Schwarz-Baum oder einen AVL-Baum, der dieselben Einträge enthält.

Lösungsvorschlag

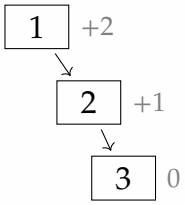
Nach dem Einfügen von „1“:

Nach dem Einfügen von „2“:

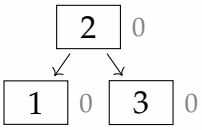
[1] ein Baum, für den folgendes gilt: Er besitzt in einem Knoten max. 3 Schlüssel-Einträge und 4 Kindknoten und minimal einen Schlüssel und 2 Nachfolger



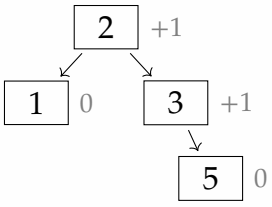
Nach dem Einfügen von „3“:



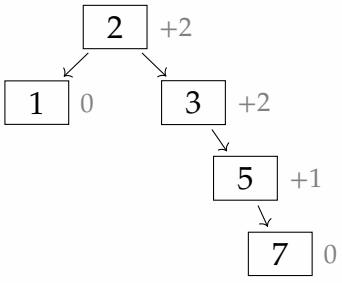
Nach der Linksrotation:



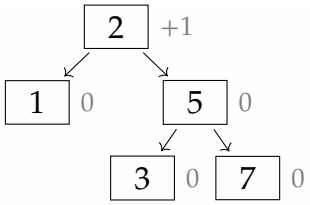
Nach dem Einfügen von „5“:



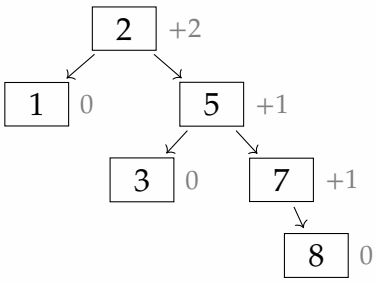
Nach dem Einfügen von „7“:



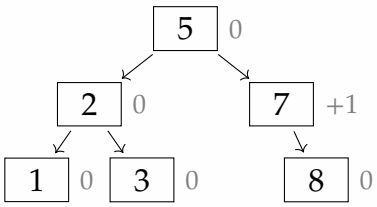
Nach der Linksrotation:



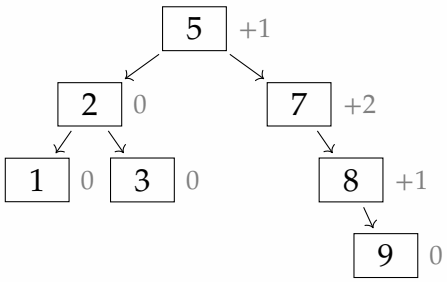
Nach dem Einfügen von „8“:



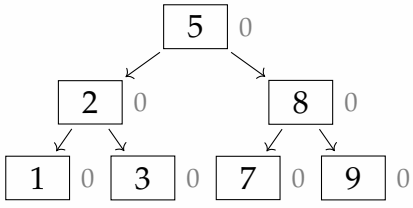
Nach der Linksrotation:



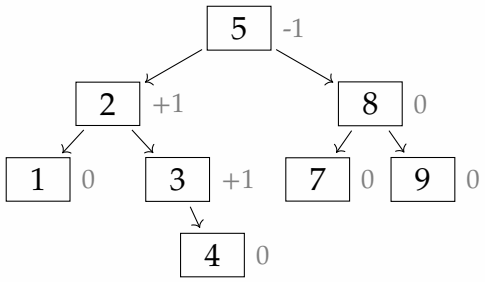
Nach dem Einfügen von „9“:



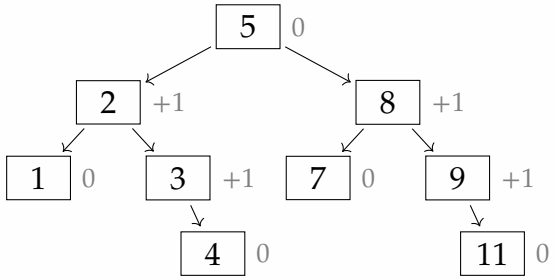
Nach der Linksrotation:



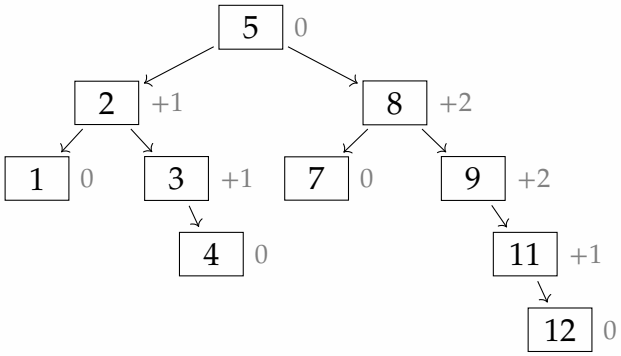
Nach dem Einfügen von „4“:



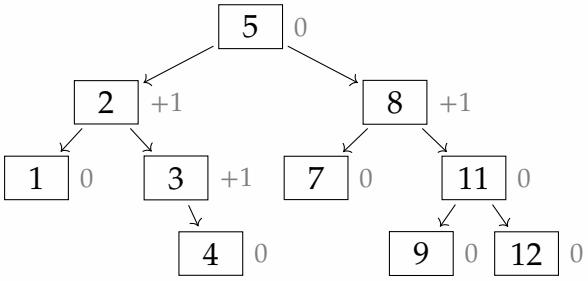
Nach dem Einfügen von „11“:



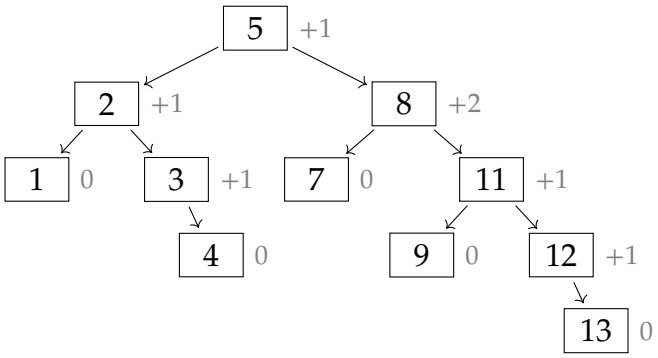
Nach dem Einfügen von „12“:



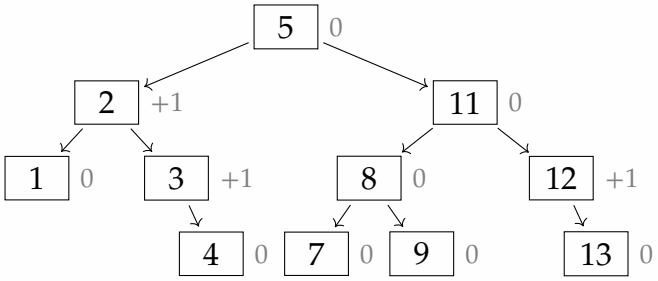
Nach der Linksrotation:



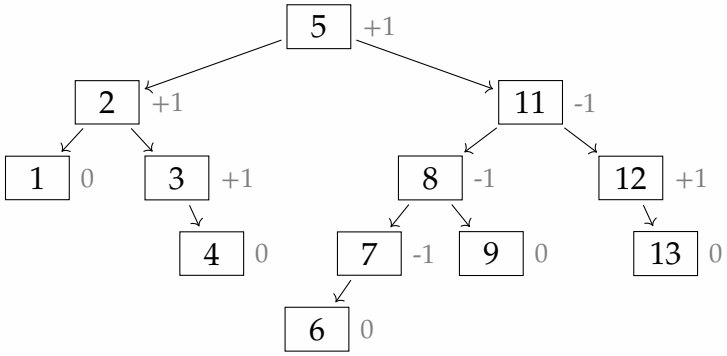
Nach dem Einfügen von „13“:



Nach der Linksrotation:



Nach dem Einfügen von „6“:



(c) Geben Sie eine möglichst gute untere Schranke (in Ω-Notation) für die Anzahl der Schlüssel in einem 2-3-4-Baum der Höhe h an.

Hinweis: Überlegen Sie sich, wie ein 2-3-4-Baum mit Höhe h und möglichst wenigen Schlüsseln aussieht.

Lösungsvorschlag

Ein 2-3-4-Baum mit möglichst wenigen Schlüsseln sieht aus wie ein Binärbaum:

- Ein Baum der Höhe 1 hat 1 Schlüssel.

- Ein Baum der Höhe 2 hat 3 Schlüssel.
- Ein Baum der Höhe 3 hat 7 Schlüssel.
- ...
- Ein Baum der Höhe h hat $2^h - 1$ Schlüssel.

Also liegt die Untergrenze für die Anzahl der Schlüssel in $\Omega(2^h)$.

(d) Geben Sie eine möglichst gute obere Schranke (in $\mathcal{O}$-Notation) für die Anzahl der Schlüssel in einem 2-3-4-Baum der Höhe h an.

Lösungsvorschlag

Ein 2-3-4-Baum mit möglichst vielen Schlüsseln hat in jedem Knoten drei Schlüssel. Und jeder Knoten, der kein Blatt ist, hat vier Kinder:

- Ein Baum der Höhe 1 hat 3 Schlüssel.
- Ein Baum der Höhe 2 hat 15 Schlüssel.
- Ein Baum der Höhe 3 hat 63 Schlüssel.
- ...
- Ein Baum der Höhe h hat $4^h - 1$ Schlüssel.

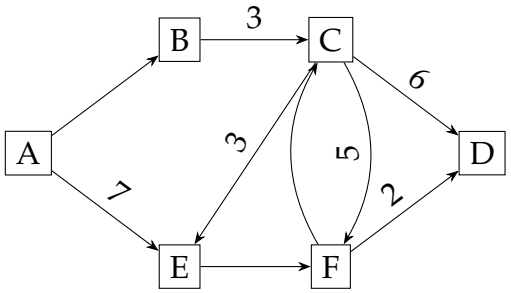
Also liegt die Obergrenze für die Anzahl der Schlüssel in $\mathcal{O}(4^h)$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2011/03/Thema-1/Aufgabe-3.tex`

Examensaufgabe „Graph A-F“ (46115-2012-F.T1-A6)

Aufgabe 6

Gegeben sei der folgende gerichtete Graph $G = (V, E, d)$ mit den angegebenen Kantengewichten.



(a) Geben Sie eine formale Beschreibung des abgebildeten Graphen G durch Auflistung von V, E und d an.

Lösungsvorschlag

$G = (V, E, d)$

mit

$V = \{A, B, C, D, EF\}$

und

$E = \{(A, B), (A, E), (B, C), (C, D), (C, E), (C, F), (E, F), (F, C), (F, D), \}$

und

$d = \{1, 7, 3, 6, 3, 5, 1, 1, 2\}$

Als Adjazenzliste

A: $\rightarrow$ B $\quad \xrightarrow{7}$ E

B: $\xrightarrow{3}$ C

C: $\xrightarrow{6}$ D $\quad \xrightarrow{3}$ E $\quad \xrightarrow{5}$ F

D:

E: $\rightarrow$ F

F: $\rightarrow$ C $\quad \xrightarrow{2}$ D

(b) Erstellen Sie die Adjazenzmatrix A zum Graphen G.

Lösungsvorschlag

Halde (Heap)

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} * & 1 & - & - & 7 & - \\ - & * & 3 & - & - & - \\ - & - & * & 6 & 3 & 5 \\ - & - & - & * & - & - \\ - & - & - & - & * & 1 \\ - & - & 1 & 2 & - & * \end{pmatrix} \end{array}$$

(c) Berechnen Sie unter Verwendung des Algorithmus nach Dijkstra - vom Knoten A beginnend - den kürzesten Weg, um alle Knoten zu besuchen. Die Restknoten werden in einer Halde (engl. Heap) gespeichert. Geben Sie zu jedem Arbeitsschritt den Inhalt dieser Halde an.

Lösungsvorschlag

Nr.	**besucht**	**A**	**B**	**C**	**D**	**E**	**F**
0		0	∞	∞	∞	∞	∞
1	A	**0**	1	∞	∞	7	∞
2	B	|	**1**	4	∞	7	∞
3	C	|	|	**4**	10	7	9
4	E	|	|	|	10	**7**	8
5	F	|	|	|	10	|	**8**
6	D	|	|	|	**10**	|	|

nach	**Entfernung**	**Reihenfolge**	**Pfad**
A → A	0	1	
A → B	1	2	A → B
A → C	4	3	A → B → C
A → D	10	6	A → B → C → D
A → E	7	4	A → E
A → F	8	5	A → E → F

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2012/03/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Hashing mit Modulo 11“ (46115-2013-F.T2-A4)

„Streuspeicherung“

Die Werte 7, 0, 9, 11, 18, 4, 5, 3, 13, 24, 2 sollen in eine Hashtabelle der Größe 11 (Fächer 0 bis 10) eingetragen werden. Die zur Hashfunktion $h(x) = (7 \cdot x)\%11$ gehörenden Schlüssel sind in der folgenden Tabelle bereits ausgerechnet:

x	7	0	9	11	18	4	5	3	13	24	2
$h(x)$	5	0	8	0	5	6	2	10	3	3	3

(a) Fügen Sie die oben genannten Schlüssel in der vorgegebenen Reihenfolge in einen Streuspeicher ein, welcher zur Kollisionsauflösung verkettete Listen verwendet, und stellen Sie die endgültige Streutabelle dar.

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0		5	13		7	4		8		3
	11			24		18					
				2							

(b) Fügen Sie die gleichen Schlüssel mit linearem Sondieren bei Schrittweite $+1$ zur Kollisionsauflösung in eine neue Hash-Tabelle ein. Geben Sie für jeden Schlüssel an, auf welche Felder beim Einfügen zugegriffen wird und ob Kollisionen auftreten. Geben Sie die gefüllte Streutabelle an.

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0	11_1	5	13	24_1	7	18_1	4_1	9	2_6	3

(c) Wie hoch ist der „Load“-Faktor (die Belegung) der Hashtabelle aus a) bzw. b) in Prozent? Können Sie weitere Schlüssel einfügen?

Lösungsvorschlag

Teilaufgabe a)

$\frac{11}{11} = 100\%$: Es können allerdings weitere Elemente eingefügt werden. Die Verkettung lässt einen Loadfaktor über 100% zu. Der Suchaufwand wird dann jedoch größer.

Teilaufgabe b)

$\frac{11}{11} = 100\%$: Es können keine weiteren Elemente eingefügt werden, da alle Buckets belegt sind.

(d) Würden Sie sich bei dieser Zahlensequenz für das Hashing-Verfahren nach a) oder nach b) entscheiden? Begründen Sie kurz Ihre Entscheidung.

Lösungsvorschlag

Das Verfahren a) scheint hier sinnvoller, da noch nicht zu viele Suchoperationen notwendig sind (max. 2), während bei Verfahren b) einmal bereits 6-mal sondiert werden muss.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2013/03/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Zahlenfolge in binärer Suchbaum, Min-Heap und AVL-Baum“ (46115-2014-F.T1-A7)

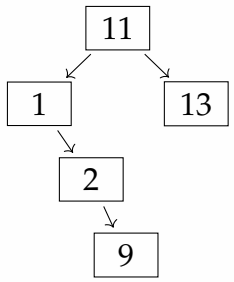
Halde (Heap)
Binärbaum
AVL-Baum
Min-Heap

Fügen Sie nacheinander die Zahlen 11, 1, 2, 13, 9, 10, 7, 5

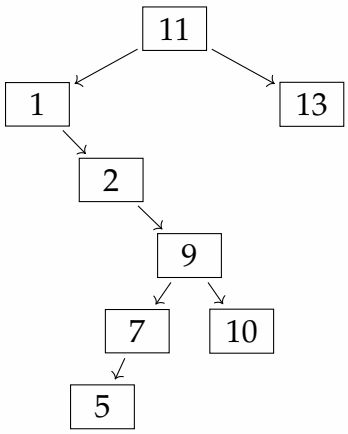
(a) in einen leeren binären Suchbaum und zeichnen Sie den Suchbaum jeweils nach dem Einfügen von „9“ und „5“

Lösungsvorschlag

Nach dem Einfügen von „9“:



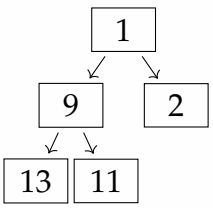
Nach dem Einfügen von „5“:



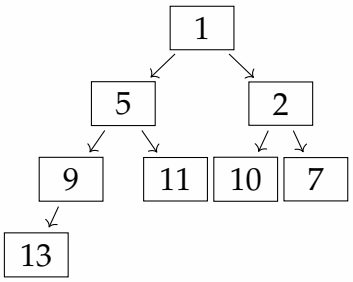
(b) in einen leeren Min-Heap ein, der bzgl. „$\leq$“ angeordnet ist und geben Sie den Heap nach „9“ und nach „5“ an

Lösungsvorschlag

Nach dem Einfügen von „9“:



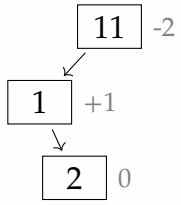
Nach dem Einfügen von „5“:



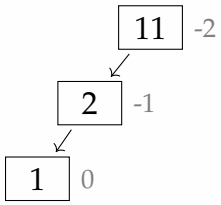
(c) in einen leeren AVL-Baum ein! Geben Sie den AVL Baum nach „2“ und „5“ an und beschreiben Sie die ggf. notwendigen Rotationen beim Einfügen dieser beiden Elemente!

Lösungsvorschlag

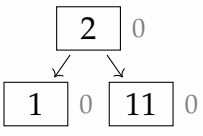
Nach dem Einfügen von „2“:



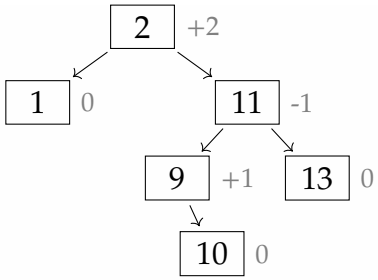
Nach der Linksrotation:



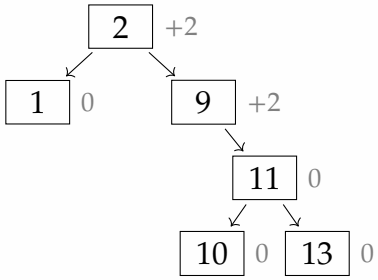
Nach der Rechtsrotation:



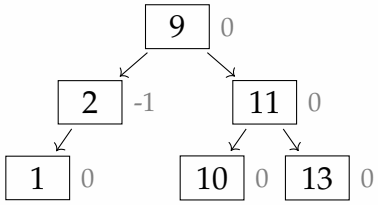
Nach dem Einfügen von „10“:



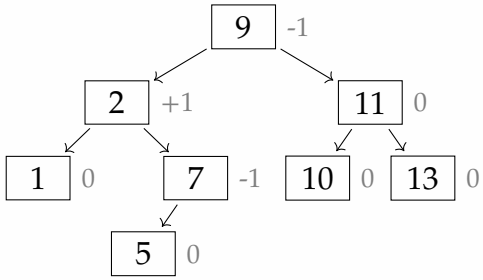
Nach der Rechtsrotation:



Nach der Linksrotation:



Nach dem Einfügen von „5“:



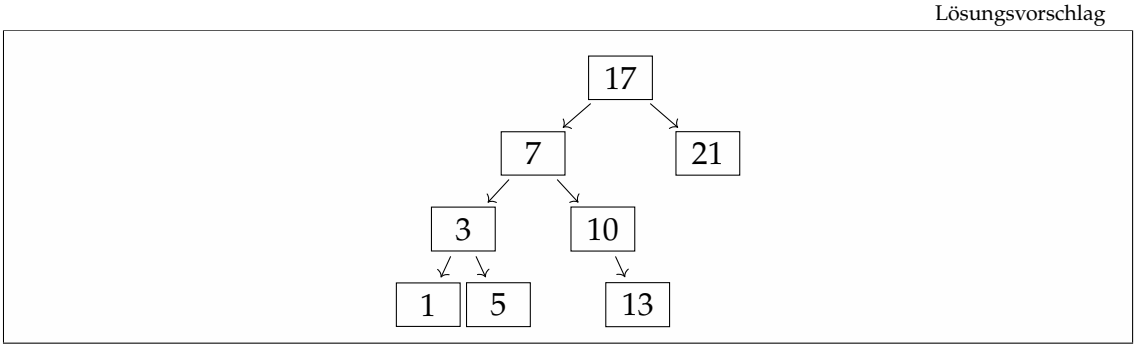
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2014/03/Thema-1/Aufgabe-7.tex`

Examensaufgabe „Binärer Suchbaum 17, 7, 21, 3, 10, 13, 1, 5“ (46115-2014-F.T2-A3)

Binärbaum

(a) Fügen Sie die Zahlen 17, 7, 21, 3, 10, 13, 1, 5 nacheinander in der vorgegebenen Reihenfolge in einen binären Suchbaum ein und zeichnen Sie das Ergebnis!

Lösungsvorschlag



(b) Implementieren Sie in einer objektorientierten Programmiersprache eine rekursiv festgelegte Datenstruktur, deren Gestaltung sich an folgender Definition eines binären Baumes orientiert!

Ein binärer Baum ist entweder ein leerer Baum oder besteht aus einem Wurzelelement, das einen binären Baum als linken und einen als rechten Teilbaum besitzt. Bei dieser Teilaufgabe können Sie auf die Implementierung von Methoden (außer ggf. notwendigen Konstruktoren) verzichten!

Lösungsvorschlag

Klasse „Knoten“

```
class Knoten {
  public int wert;
  public Knoten links;
  public Knoten rechts;
  public Knoten elternKnoten;

  Knoten(int wert) {
    this.wert = wert;
    links = null;
    rechts = null;
    elternKnoten = null;
  }

  public Knoten findeMiniumRechterTeilbaum() {
  }

  public void anhängen (Knoten knoten) {
  }
}
```

Klasse „BinärerSuchbaum“

```
public class BinärerSuchbaum {
  public Knoten wurzel;

  BinärerSuchbaum(Knoten wurzel) {
    this.wurzel = wurzel;
  }

  BinärerSuchbaum() {
    this.wurzel = null;
  }

  public void einfügen(Knoten knoten) {
  }

  public void einfügen(Knoten knoten, Knoten elternKnoten) {
  }

  public Knoten suchen(int wert) {
  }

  public Knoten suchen(int wert, Knoten knoten) {
  }

}
```

(c) Beschreiben Sie durch Implementierung in einer gängigen objektorientierten Programmiersprache, wie bei Verwendung der obigen Datenstruktur die Methode `loescheKnoten(w)` gestaltet sein muss, mit der der Knoten mit dem Eintrag `w` aus dem Baum entfernt werden kann, ohne die Suchbaumeigenschaft zu verletzen!

Lösungsvorschlag

```
public void loescheKnoten(int w) {
  Knoten knoten = suchen(w);
  if (knoten == null) return;
  // Der Knoten hat keine Teilbäume.
  if (knoten.links == null && knoten.rechts == null) {
    if (w < knoten.elternKnoten.wert) {
      knoten.elternKnoten.links = null;
    } else {
      knoten.elternKnoten.rechts = null;
    }
  }

  // Der Knoten besitzt einen Teilbaum.
  // links
  else if (knoten.links != null && knoten.rechts == null) {
    knoten.elternKnoten.anhängen(knoten.links);
  }
  // rechts
  else if (knoten.links == null) {
    knoten.elternKnoten.anhängen(knoten.rechts);
  }
```

```
    // Der Knoten besitzt zwei Teilbäume.
    else {
      Knoten minimumKnoten = knoten.findeMiniumRechterTeilbaum();
      minimumKnoten.links = knoten.links;
      minimumKnoten.rechts = knoten.rechts;
      knoten.elternKnoten.anhängen(minimumKnoten);
    }
  }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java

Lösungsvorschlag

Klasse „BinärerSuchbaum“

```
public class BinaererSuchbaum {
  public Knoten wurzel;

  BinaererSuchbaum(Knoten wurzel) {
    this.wurzel = wurzel;
  }

  BinaererSuchbaum() {
    this.wurzel = null;
  }

  public void einfügen(Knoten knoten) {
    if (wurzel != null) {
      einfügen(knoten, wurzel);
    } else {
      wurzel = knoten;
      knoten.elternKnoten = wurzel;
    }
  }

  public void einfügen(int wert) {
    einfügen(new Knoten(wert));
  }

  public void einfügen(Knoten knoten, Knoten elternKnoten) {
    if (knoten.wert <= elternKnoten.wert) {
      if (elternKnoten.links != null) {
        einfügen(knoten, elternKnoten.links);
      } else {
        elternKnoten.links = knoten;
        knoten.elternKnoten = elternKnoten;
      }
    } else {
      if (elternKnoten.rechts != null) {
        einfügen(knoten, elternKnoten.rechts);
      } else {
        elternKnoten.rechts = knoten;
        knoten.elternKnoten = elternKnoten;
```

```
      }
    }
  }

  public Knoten suchen(int wert) {
    if (wurzel == null || wurzel.wert == wert) {
      return wurzel;
    } else {
      return suchen(wert, wurzel);
    }
  }

  public Knoten suchen(int wert, Knoten knoten) {
    if (knoten.wert == wert) {
      return knoten;
    } else if (wert < knoten.wert && knoten.links != null) {
      return suchen(wert, knoten.links);
    } else if (wert > knoten.wert && knoten.rechts != null) {
      return suchen(wert, knoten.rechts);
    }
    return null;
  }

  public void loescheKnoten(int w) {
    Knoten knoten = suchen(w);
    if (knoten == null) return;
    // Der Knoten hat keine Teilbäume.
    if (knoten.links == null && knoten.rechts == null) {
      if (w < knoten.elternKnoten.wert) {
        knoten.elternKnoten.links = null;
      } else {
        knoten.elternKnoten.rechts = null;
      }
    }

    // Der Knoten besitzt einen Teilbaum.
    // links
    else if (knoten.links != null && knoten.rechts == null) {
      knoten.elternKnoten.anhängen(knoten.links);
    }
    // rechts
    else if (knoten.links == null) {
      knoten.elternKnoten.anhängen(knoten.rechts);
    }

    // Der Knoten besitzt zwei Teilbäume.
    else {
      Knoten minimumKnoten = knoten.findeMiniumRechterTeilbaum();
      minimumKnoten.links = knoten.links;
      minimumKnoten.rechts = knoten.rechts;
      knoten.elternKnoten.anhängen(minimumKnoten);
    }
  }
```

```
  // Der Baum aus dem Foliensatz
  public BinaererSuchbaum erzeugeTestBaum() {
    BinaererSuchbaum binärerSuchbaum = new BinaererSuchbaum();
    binärerSuchbaum.einfügen(new Knoten(7));
    binärerSuchbaum.einfügen(new Knoten(3));
    binärerSuchbaum.einfügen(new Knoten(11));
    binärerSuchbaum.einfügen(new Knoten(2));
    binärerSuchbaum.einfügen(new Knoten(6));
    binärerSuchbaum.einfügen(new Knoten(9));
    binärerSuchbaum.einfügen(new Knoten(1));
    binärerSuchbaum.einfügen(new Knoten(5));
    return binärerSuchbaum;
  }

  public void ausgebenInOrder() {

  }

  public static void main(String[] args) {
    BinaererSuchbaum binärerSuchbaum = new BinaererSuchbaum();
    BinaererSuchbaum testBaum = binärerSuchbaum.erzeugeTestBaum();

    // Teste das Einfügen

    System.out.println(testBaum.wurzel.wert); // 7
    System.out.println(testBaum.wurzel.links.wert); // 3
    System.out.println(testBaum.wurzel.links.links.wert); // 2
    System.out.println(testBaum.wurzel.links.rechts.wert); // 6
    System.out.println(testBaum.wurzel.rechts.wert); // 11

    // Teste das Suchen

    System.out.println("Gesucht nach 5 und gefunden: " + testBaum.suchen(5).wert);
    System.out.println("Gesucht nach 9 und gefunden: " + testBaum.suchen(9).wert);
    System.out.println("Gesucht nach 7 und gefunden: " + testBaum.suchen(7).wert);
    System.out.println("Gesucht nach 10 und gefunden: " + testBaum.suchen(10));

    // Teste das Löschen

    // Der Knoten hat keine Teilbäume.
    System.out.println("Noch nicht gelöschter Knoten 9: " +
↪   testBaum.suchen(9).wert);
    testBaum.loescheKnoten(9);
    System.out.println("Gelöschter Knoten 9: " + testBaum.suchen(9));

    // Der Knoten hat einen Teilbaum.
    // fristen Testbaum erzeugen.
    testBaum = binärerSuchbaum.erzeugeTestBaum();
    Knoten elternKnoten = testBaum.suchen(3);
    System.out.println("Rechts Kind von 3 vor dem Löschen: " +
↪   elternKnoten.rechts.wert);
    testBaum.loescheKnoten(6);
    System.out.println("Rechts Kind von 3Nach dem Löschen: " +
↪   elternKnoten.rechts.wert);
```

```
    // Der Knoten hat zwei Teilbäume.
    // fristen Testbaum erzeugen.
    testBaum = binärerSuchbaum.erzeugeTestBaum();
    Knoten wurzel = testBaum.wurzel;
    System.out.println("Linkes Kind der Wurzel vor dem Löschen: " +
↪   wurzel.links.wert); // 5
    testBaum.loescheKnoten(3);
    System.out.println("Linkes Kind der WurzelNach dem Löschen: " +
↪   wurzel.links.wert); // 3
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java`

Klasse „Knoten“

```
class Knoten {
  public int wert;
  public Knoten links;
  public Knoten rechts;
  public Knoten elternKnoten;

  Knoten(int wert) {
    this.wert = wert;
    links = null;
    rechts = null;
    elternKnoten = null;
  }

  public Knoten findeMiniumRechterTeilbaum() {
    if (rechts != null) {
      Knoten minimumKonten = rechts;
      while (minimumKonten.links != null) {
        minimumKonten = minimumKonten.links;
      }
      return minimumKonten;
    }
    return null;
  }

  public void anhängen (Knoten knoten) {
    if (knoten.wert < wert) {
      links = knoten;
    } else {
      rechts = knoten;
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/Knoten.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2014/03/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Hashing mit Modulo 8“ (46115-2015-H.T2-A1)

Fügen Sie die folgenden Werte in der gegebenen Reihenfolge in eine Streutabelle der Größe 8 (mit den Indizes 0 bis 7) und der Streufunktion $h(x) = x \mod 8$ ein. Verwenden Sie die jeweils angegebene Hash-Variante bzw. Kollisionsauflösung: 15, 3, 9, 23, 1, 8, 17, 4

(a) Offenes Hashing

Zur Kollisionsauflösung wird Verkettung verwendet.

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	$\cdots$
Inhalt	8			
	16			

Lösungsvorschlag

$$
\begin{aligned}
h(15) &= 15 \mod 8 &&= 7\\
h(3) &= 3 \mod 8 &&= 3\\
h(9) &= 9 \mod 8 &&= 1\\
h(23) &= 23 \mod 8 &&= 7\\
h(1) &= 1 \mod 8 &&= 1\\
h(8) &= 8 \mod 8 &&= 0\\
h(17) &= 17 \mod 8 &&= 1\\
h(4) &= 4 \mod 8 &&= 4
\end{aligned}
$$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	4			15
		1						23
		17						

(b) Geschlossenes Hashing

Zur Kollisionsauflösung wird lineares Sondieren (nur hochzählend) mit Schrittweite $+5$ verwendet.

Treten beim Einfügen Kollisionen auf, dann notieren Sie die Anzahl der Versuche zum Ablegen des Wertes im Subskript (z. B. das Einfügen des Wertes 8 gelingt im 5. Versuch: 8_5).

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	3	4	5	⋯
Inhalt	8					16_1	

Lösungsvorschlag

$h'(x) = x \mod 8$
$h(x,i) = (h'(x) + i \cdot 5) \mod 8$

17 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23_2		1_2	15

1. Versuch: $h(17,0) = (h'(17) + 0 \cdot 5) \mod 8 = (1+0) \mod 8 = 1 \mod 8 = 1$ (belegt von 9)
2. Versuch: $h(17,1) = (h'(17) + 1 \cdot 5) \mod 8 = (1+5) \mod 8 = 6 \mod 8 = 6$ (belegt von 1)
3. Versuch: $h(17,2) = (h'(17) + 2 \cdot 5) \mod 8 = (1+10) \mod 8 = 11 \mod 8 = 3$ (belegt von 3)
4. Versuch: $h(17,3) = (h'(17) + 3 \cdot 5) \mod 8 = (1+15) \mod 8 = 16 \mod 8 = 0$ (belegt von 8)
5. Versuch: $h(17,4) = (h'(17) + 4 \cdot 5) \mod 8 = (1+20) \mod 8 = 21 \mod 8 = 5$

4 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23_2	17_5	1_2	15

1. Versuch: $h(4,0) = (h'(4) + 0 \cdot 5) \mod 8 = (4+0) \mod 8 = 4$ (belegt von 23)
2. Versuch: $h(4,1) = (h'(4) + 1 \cdot 5) \mod 8 = (4+5) \mod 8 = 1$ (belegt von 9)
3. Versuch: $h(4,2) = (h'(4) + 2 \cdot 5) \mod 8 = (4+10) \mod 8 = 6$ (belegt von 1)
4. Versuch: $h(4,3) = (h'(4) + 3 \cdot 5) \mod 8 = (4+15) \mod 8 = 3$ (belegt von 3)
5. Versuch: $h(4,4) = (h'(4) + 4 \cdot 5) \mod 8 = (4+20) \mod 8 = 0$ (belegt von 8)
6. Versuch: $h(4,5) = (h'(4) + 5 \cdot 5) \mod 8 = (4+25) \mod 8 = 5$ (belegt von 17)
7. Versuch: $h(4,6) = (h'(4) + 6 \cdot 5) \mod 8 = (4+30) \mod 8 = 2$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9	4_7	3	23_2	17_5	1_2	15

(c) Welches Problem tritt auf, wenn zur Kollisionsauflösung lineares Sondieren mit Schrittweite 4 verwendet wird? Warum ist 5 eine bessere Wahl?

Lösungsvorschlag

Beim linearen Sondieren mit der Schrittweite 4 werden nur zwei verschiedene Buckets erreicht, beispielsweise: 1, 5, 1, 5, etc.

> Beim linearen Sondieren mit der Schrittweite 5 werden nacheinander alle möglichen Buckets erreicht, beispielsweise: 1, 6, 3, 0, 5, 2, 7, 4.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2015/09/Thema-2/Aufgabe-1.tex`

Examensaufgabe „Halden - Heaps“ (46115-2017-H.T2-A6)

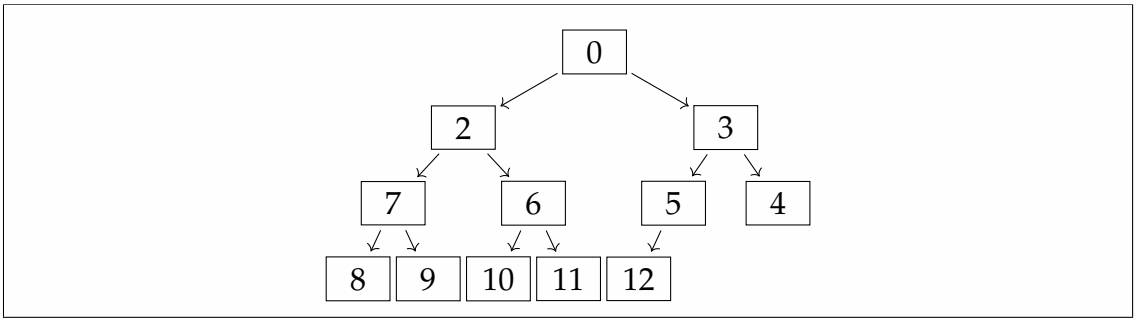
Halde (Heap)

Gegeben sei folgende Feld-Einbettung (Array-Darstellung) einer Min-Halde:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	2	3	7	6	5	4	8	9	10	11	12

(a) Stellen Sie die Halde graphisch als (links-vollständigen) Baum dar.

Lösungsvorschlag



(b) Entfernen Sie das kleinste Element (die Wurzel 0) aus der obigen initialen Halde, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	6	3	7	10	5	4	8	9	12	11

(c) Fügen Sie nun den Wert 1 in die obige initiale Halde ein, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	2	1	7	6	3	4	8	9	10	11	12	5

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2017/09/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Aussage widerlegen“ (46115-2019-F.T2-A3)

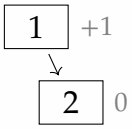
AVL-Baum

(a) Zeigen oder widerlegen Sie die folgende Aussage: Wird ein Element in einen AVL-Baum eingefügt und unmittelbar danach wieder gelöscht, so befindet sich der AVL-Baum wieder in seinem Ursprungszustand.

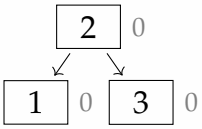
Lösungsvorschlag

Die Aussage ist falsch. Wir widerlegen die Aussage durch ein konkretes Beispiel:

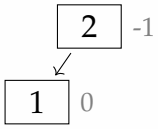
Unser Ausgangs-AVL-Baum:



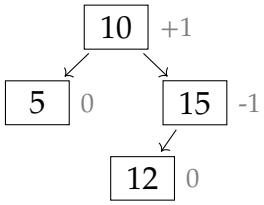
Nach dem Einfügen von „3“:



Nach dem Löschen von „3“:



(b) Fügen Sie in den gegebenen Baum den Schlüssel 11 ein.

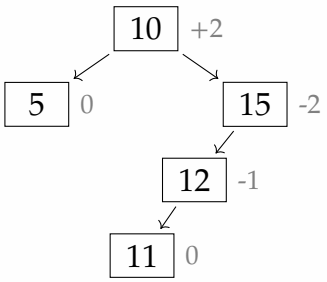


Rebalancieren Sie anschließend den Baum so, dass die AVL-Eigenschaft wieder erreicht wird. Zeichnen Sie den Baum nach jeder Einfach- und Doppelrotation und benennen Sie die Art der Rotation (Links-, Rechts-, Links-Rechts-, oder Rechts-Links-Rotation). Argumentieren Sie jeweils über die Höhenbalancen der Teilbäume.

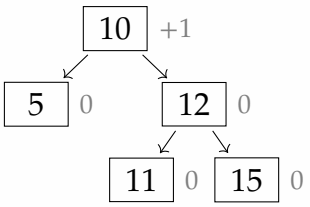
Tipp: Zeichnen Sie nach jedem Schritt die Höhenbalancen in den Baum ein.

Lösungsvorschlag

Nach dem Einfügen von „11“:



Nach der Rechtsrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/03/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Heapify“ (46115-2019-H.T2-A7)

Halde (Heap)

Schreiben Sie in Pseudocode eine Methode `heapify(int[] a)`, welche im übergebenen Array der Länge n die Heapeigenschaft in $\mathcal{O}(n)$ Schritten herstellt. D. h. als Ergebnis soll in a gelten, dass $a[i] \leq a[2i+1]$ und $a[i] \leq a[i+2]$.

Lösungsvorschlag

```java
import org.bschlangaul.helfer.Konsole;

/**
 * Nach Pseudocode nach
 * https://www.oreilly.com/library/view/algorithms-in-a/9780596516246/ch04s06.html
 */
public class Heapify {

  public static void buildHeap(int a[]) {
    int n = a.length;
    for (int i = n / 2 - 1; i >= 0; i--) {
      heapify(a, i, n);
    }
  }

  public static void heapify(int a[], int index, int max) {
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    int smallest;

    if (left < max && a[left] < a[index]) {
      smallest = left;
    } else {
      smallest = index;
    }

    if (right < max && a[right] < a[smallest]) {
      smallest = right;
    }

    if (smallest != index) {
      int tmp = a[index];
      a[index] = a[smallest];
      a[smallest] = tmp;
      heapify(a, smallest, max);
    }
  }

  public static void main(String[] args) {
    int[] a = new int[] { 5, 3, 16, 2, 10, 14 };
    buildHeap(a);
    Konsole.zeigeZahlenFeld(a); // 2 3 14 5 10 16
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/Heapify.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-2/Aufgabe-7.tex

Streutabellen (Hashing)

Examensaufgabe „Sondierfolgen für Hashing mit offener Adressierung“ (46115-2021-F.T1-TA2-A4)

Eine Sondierfolge s(k,i) liefert für einen Schlüssel k aus einem Universum U und Versuchsnummerni = 0,1,...,m-leine Folge von Indizes für eine Hashtabelle $T[0 \dots m-1]$. Mithilfe einer Son- dierfolge wird beim Hashing mit offener Adressierung z. B. beim Einfügen eines neuen Schlüssels k nach einem noch nicht benützten Tabelleneintrag gesucht. Seien h und h′ zwei verschiedene Hash- funktionen, die U auf 0,1,...,m—1 abbilden. Beantworten Sie die folgenden Fragen und geben Sie an, um welche Art von Sondieren es sich jeweils handelt.

(a) Was ist problematisch an der Sondierfolge $s(k,i) = (h(k) + 2i) \mod m$, wobei $m = 1023$ die Größe der Hashtabelle ist?

Lösungsvorschlag

Art Es handelt sich um lineares Sondieren.

Problematisch Es wird für einen großen Bereich an Sondierfolgen (512 (0-511,512-1023)) nur in jeden zweiten Bucket (z. B. geradzahlig) sondiert, erst dann wird in den bisher ausgelassenen Buckets (z. b. ungeradzahlig) sondiert.

(b) Was ist problematisch an der Sondierfolge $s(k,i) = (h(k) + i(i+1)) \mod m$, wobei $m = 1024$ die Größe der Hashtabelle ist?

Lösungsvorschlag

Art Es handelt sich um quadratisches Sondieren

Problematisch $i(i+1)$ gibt immer eine gerade Zahl. Eine gerade Zahl Modulo 1024 gibt auch immer eine grade Zahl. Es wird nie in den ungeraden Buckets sondiert.

(c) Was ist vorteilhaft an der Sondierfolge $s(k,i) = (h(k) + i \cdot h'(k)) \mod m$, wobei m die Größe der Hashtabelle ist?

Lösungsvorschlag

Auch die Sondierfolge ist abhängig von dem Schlüsselwert. Die Entstehung von Ballungen ist unwahrscheinlicher bei gut gewählten Hashfunktionen, eine gleichmäßige Verteilung wahrscheinlicher.

(d) Sei $h(k) = k \mod 6$ und $h(k) = k^2 \mod 6$

Fügen Sie die Schlüssel $14, 9, 8, 3, 2$ in eine Hashtabelle der Größe 7 ein. Verwenden Sie die Sondierfolge $s(k,i) = (h(k) + i \cdot h(k)) \mod 7$ und offene Adressierung. Notieren Sie die Indizes der Tabellenfelder und vermerken Sie neben jedem Feld die erfolglosen Sondierungen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-4.tex

Examensaufgabe „Klasse „BinBaum““ (66112-2003-H.T2-A8)

(a) Implementieren Sie in einer objektorientierten Sprache einen binären Suchbaum für ganze Zahlen! Dazu gehören Methoden zum Setzen und Ausgeben der Attribute `zahl`, `linker_teilbaum` und `rechter_teilbaum`. Design: eine Klasse `Knoten` und eine Klasse `BinBaum`. Ein Knoten hat einen linken und einen rechten Nachfolger. Ein Baum verwaltet die Wurzel. Er hängt neue Knoten an und löscht Knoten.

Lösungsvorschlag

```java
public class BinBaum {

  private Knoten wurzel = null;

  public void setzeWurzel(Knoten knoten) {
    wurzel = knoten;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

```java
public class Knoten {
  private int zahl;
  private Knoten links = null;
  private Knoten rechts = null;

  public Knoten() {
  }

  public Knoten(int zahl) {
    this.zahl = zahl;
  }

  public void setzeZahl(int zahl) {
    this.zahl = zahl;
  }

  public int gibZahl() {
    return zahl;
  }

  public void setzeLinks(Knoten k) {
    links = k;
  }

  public Knoten gibLinks() {
    return links;
  }

  public void setzeRechts(Knoten k) {
    rechts = k;
  }

  public Knoten gibRechts() {
    return rechts;
```

```
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java`

(b) Schreiben Sie eine Methode `fügeEin(...)`, die eine Zahl in den Baum einfügt!

Lösungsvorschlag

```
  public void fügeEin(int zahl) {
    Knoten aktueller = wurzel;
    Knoten neuerKnoten = new Knoten(zahl);
    if (wurzel == null) {
      wurzel = neuerKnoten;
      return;
    }
    while (aktueller != null) {
      // suche links
      if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
        aktueller = aktueller.gibLinks();
        // fuege ein
      } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null)
↪   {
        aktueller.setzeLinks(neuerKnoten);
        break;
      }
      // suche rechts
      if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
        aktueller = aktueller.gibRechts();
        // fuege ein
      } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null)
↪   {
        aktueller.setzeRechts(neuerKnoten);
        break;
      }
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

(c) Schreiben Sie eine Methode `void besuchePostOrder(...)`, die die Zahlen in der Reihenfolge postorder ausgibt!

Lösungsvorschlag

```
public static void besuchePostOrder(Knoten knoten) {
  // Sonderfall leerer (Teil-)Baum
  if (knoten == null) {
    System.out.println("Leerer Baum");
  } else {
    // Linker
    if (knoten.gibLinks() != null) {
      besuchePostOrder(knoten.gibLinks());
    }
    // Rechter
    if (knoten.gibRechts() != null) {
```

```
        besuchePostOrder(knoten.gibRechts());
      }
      System.out.println(knoten.gibZahl());
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

(d) Ergänzen Sie Ihr Programm um die rekursiv implementierte Methode `int berechneSumme(...)`, die die Summe der Zahlen des Unterbaums, dessen Wurzel der Knoten ist, zurückgibt! Falls der Unterbaum leer ist, ist der Rückgabewert 0!

```
int summe (Knoten x)...
```

Lösungsvorschlag

```
public int berechneSumme(Knoten knoten) {
  int ergebnis = 0;

  // Sonderfall: leerer Unterbaum
  if (knoten == null) {
    return 0;
  }
  // linker
  if (knoten.gibLinks() != null) {
    ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
  }
  // rechter
  if (knoten.gibRechts() != null) {
    ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
  }
  // Wurzel
  ergebnis = ergebnis + knoten.gibZahl();
  return ergebnis;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

(e) Schreiben Sie eine Folge von Anweisungen, die einen Baum mit Namen BinBaum erzeugt und nacheinander die Zahlen 5 und 7 einfügt! In den binären Suchbaum werden noch die Zahlen 4, 11, 6 und 2 eingefügt. Zeichnen Sie den Baum, den Sie danach erhalten haben, und schreiben Sie die eingefügten Zahlen in der Reihenfolge der Traversierungsmöglichkeit `postorder` auf!

(f) Implementieren Sie eine Operation `isSorted(...)`, die für einen (Teil-)baum feststellt, ob er sortiert ist.

Lösungsvorschlag

```
public boolean istSortiert(Knoten knoten) {
  // Baum leer
  if (knoten == null) {
    return true;
  }
```

```
    // linker Nachfolger nicht okay
    if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
↪   knoten.gibZahl()) {
      return false;
    }

    // rechter Nachfolger nicht okay
    if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
↪   knoten.gibZahl()) {
      return false;
    }
    // sonst prüfe Teilbaeume
    return (istSortiert(knoten.gibRechts()) &&
↪   istSortiert(knoten.gibLinks()));
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

Lösungsvorschlag

```
public class BinBaum {

  private Knoten wurzel = null;

  public void setzeWurzel(Knoten knoten) {
    wurzel = knoten;
  }

  public void fügeEin(int zahl) {
    Knoten aktueller = wurzel;
    Knoten neuerKnoten = new Knoten(zahl);
    if (wurzel == null) {
      wurzel = neuerKnoten;
      return;
    }
    while (aktueller != null) {
      // suche links
      if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
        aktueller = aktueller.gibLinks();
        // fuege ein
      } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null) {
        aktueller.setzeLinks(neuerKnoten);
        break;
      }
      // suche rechts
      if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
        aktueller = aktueller.gibRechts();
        // fuege ein
      } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null) {
        aktueller.setzeRechts(neuerKnoten);
        break;
      }
    }
```

```
  }

  public static void besuchePostOrder(Knoten knoten) {
    // Sonderfall leerer (Teil-)Baum
    if (knoten == null) {
      System.out.println("Leerer Baum");
    } else {
      // Linker
      if (knoten.gibLinks() != null) {
        besuchePostOrder(knoten.gibLinks());
      }
      // Rechter
      if (knoten.gibRechts() != null) {
        besuchePostOrder(knoten.gibRechts());
      }
      System.out.println(knoten.gibZahl());
    }
  }

  public int berechneSumme(Knoten knoten) {
    int ergebnis = 0;

    // Sonderfall: leerer Unterbaum
    if (knoten == null) {
      return 0;
    }
    // linker
    if (knoten.gibLinks() != null) {
      ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
    }
    // rechter
    if (knoten.gibRechts() != null) {
      ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
    }
    // Wurzel
    ergebnis = ergebnis + knoten.gibZahl();
    return ergebnis;
  }

  public boolean istSortiert(Knoten knoten) {
    // Baum leer
    if (knoten == null) {
      return true;
    }

    // linker Nachfolger nicht okay
    if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
↪   knoten.gibZahl()) {
      return false;
    }

    // rechter Nachfolger nicht okay
    if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
↪   knoten.gibZahl()) {
```

```
      return false;
    }
    // sonst prüfe Teilbaeume
    return (istSortiert(knoten.gibRechts()) && istSortiert(knoten.gibLinks()));
  }

  public static void main(String[] args) {
    BinBaum baum = new BinBaum();

    baum.fügeEin(5);
    baum.fügeEin(7);
    baum.fügeEin(4);
    baum.fügeEin(11);
    baum.fügeEin(6);
    baum.fügeEin(2);

    besuchePostOrder(baum.wurzel);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

```
public class Knoten {
  private int zahl;
  private Knoten links = null;
  private Knoten rechts = null;

  public Knoten() {
  }

  public Knoten(int zahl) {
    this.zahl = zahl;
  }

  public void setzeZahl(int zahl) {
    this.zahl = zahl;
  }

  public int gibZahl() {
    return zahl;
  }

  public void setzeLinks(Knoten k) {
    links = k;
  }

  public Knoten gibLinks() {
    return links;
  }

  public void setzeRechts(Knoten k) {
    rechts = k;
  }
```

```
  public Knoten gibRechts() {
    return rechts;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2003/09/Thema-2/Aufgabe-8.tex`

Examensaufgabe „Hashing mit Modulo 7“ (66112-2005-F.T2-A8)

Gegeben seien die folgenden Zahlen: 7, 4, 3, 5, 0, 1

(a) Zeichnen Sie eine Hash-Tabelle mit 8 Zellen und tragen Sie diese Zahlen genau in der oben gegebenen Reihenfolge in Ihre Hash-Tabelle ein. Verwenden Sie dabei die Streufunktion $f(n) = n^2 \mod 7$ und eine Kollisionsauflösung durch lineares Sondieren.

Lösungsvorschlag

$f(7) = 7^2 \mod 7 = 49 \mod 7 = 0$

$f(4) = 4^2 \mod 7 = 16 \mod 7 = 2$

$f(3) = 3^2 \mod 7 = 9 \mod 7 = 2$ lineares Sondieren: $+1 = 3$

$f(5) = 5^2 \mod 7 = 25 \mod 7 = 4$

$f(0) = 0^2 \mod 7 = 0 \mod 7 = 0$ lineares Sondieren: $+1 = 1$

$f(1) = 1^2 \mod 7 = 1 \mod 7 = 1$ lineares Sondieren: $-1 = 0$, $-1 = 7$

0	1	2	3	4	5	6	7
7	0	4	3	5			1

(b) Welcher Belegungsfaktor ist für die Streutabelle und die Streufunktion aus Teilaufgabe a zu erwarten, wenn sehr viele Zahlen eingeordnet werden und eine Kollisionsauflösung durch Verkettung (verzeigerte Listen) verwendet wird? Begründen Sie Ihre Antwort kurz.

Lösungsvorschlag

Der Belegungsfaktor berechnet sich aus der Formel:

$$\text{Belegungsfaktor} = \frac{\text{Anzahl tatsächlich eingetragenen Schlüssel}}{\text{Anzahl Hashwerte}}$$

Der Belegungsfaktor steigt kontinuierlich, je mehr Zahlen in die Streutabelle gespeichert werden.

Die Streufunktion legt die Zahlen nur in die Buckets 0, 1, 2, 4.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2005/03/Thema-2/Aufgabe-8.tex`

Examensaufgabe „B-Baum m=2“ (66112-2005-H.T2-A6)

B-Baum

(a) Erzeugen Sie aus der gegebenen Folge einen B-Baum der Ordnung $m = 2$:

22, 10, 19, 20, 1, 13, 11, 12, 7, 8, 5, 42, 33, 21, 52, 48, 50

Fügen Sie dazu die einzelnen Elemente in gegebener Reihenfolge in einen anfangs leeren B-Baum ein. Stellen Sie für jeden Wert die entsprechenden Zwischenergebnisse und die angewendeten Operationen als Bäume dar!

Lösungsvorschlag

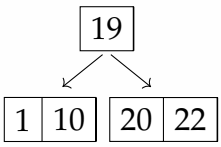
- +22 +10 +19 +20 Einfügen der ersten Zahlen bis zur kompletten Füllung der Wurzel:

10	19	20	22

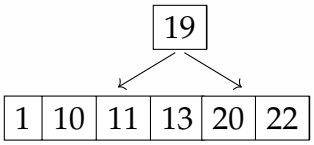
- +1 Einfügen der 1 führt zum Überlauf, deshalb Aufspaltung:

1	10	19	20	22

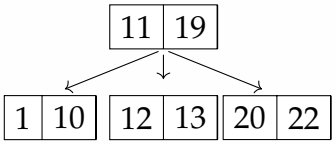
- Übernahme des mittleren Elements (19) in die Wurzel:



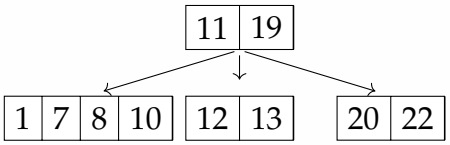
- +13 Einfügen der 13:



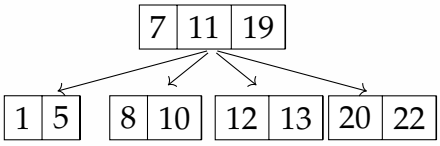
- +12 Einfügen der 12 nicht möglich, also wieder Aufspaltung. 11 als mittleres Element wird nach oben geschrieben:



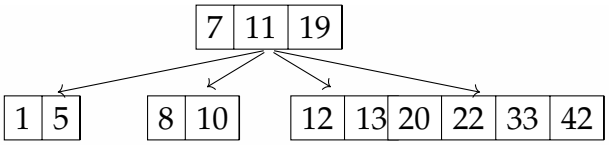
- +7 +8 Einfügen von 7 und 8:



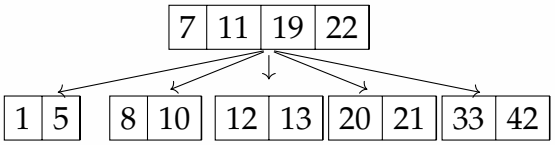
- +5 Einfügen von 5 nicht möglich, deshalb Aufspaltung, 7 als mittleres Element wird nach oben geschrieben:



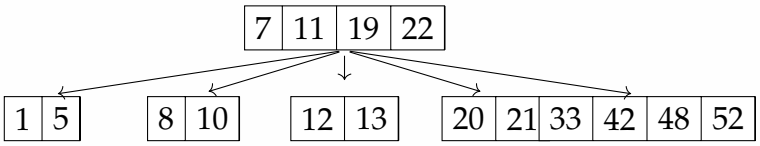
- +42 +33 Einfügen von 42 und 33:



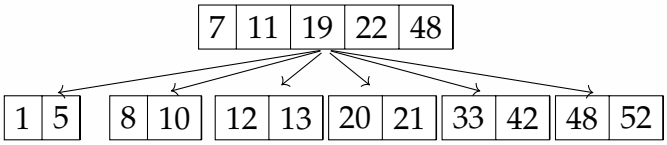
- +21 Einfügen von 21 nicht möglich, also Aufspaltung, 22 nach oben schieben



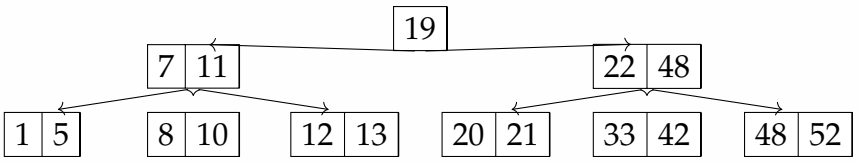
- +52 +48 Einfügen von 52 und 48



- +50 Einfügen von 50 nicht möglich, daher splitten und 48 eine Ebene nach oben schieben

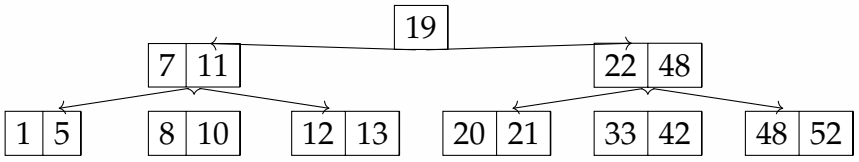


- Einfügen von 48 oben nicht möglich, da Knoten ebenfalls voll! -> weiterer Splitt notwendig, der neue Ebene erzeugt!



(b) In dem Ergebnisbaum suchen wir nun den Wert 17. Stellen Sie den Ablauf des Suchalgorithmus an einer Zeichnung graphisch dar!

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2005/09/Thema-2/Aufgabe-6.tex`

Examensaufgabe „3,5,1,2,4 in leerer Suchbaum und Heap“ (66115-2012-H.T2-A7)

Halde (Heap)

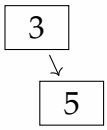
(a) Fügen Sie nacheinander die Zahlen 3, 5, 1, 2, 4

(i) in einen leeren binären Suchbaum ein

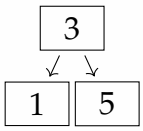
Lösungsvorschlag

Nach dem Einfügen von „3“:

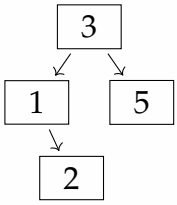
Nach dem Einfügen von „5“:



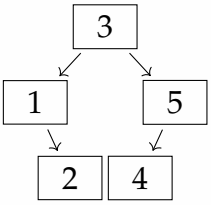
Nach dem Einfügen von „1“:



Nach dem Einfügen von „2“:



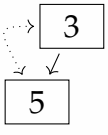
Nach dem Einfügen von „4“:



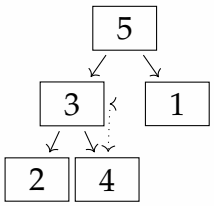
(ii) in einen leeren Heap ein

Lösungsvorschlag

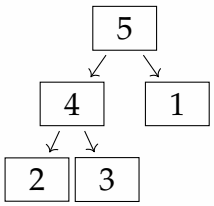
Erstellen einer Max.-Halde, einfügen von 3 und 5, Versickern notwendig:



Einfügen von 1 und 2 ohne Änderungen, Einfügen von 4, versickern notwendig:

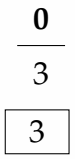


Fertiger Heap:

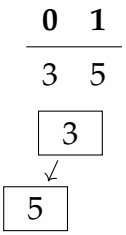


Ausführlicher als Max-Halde

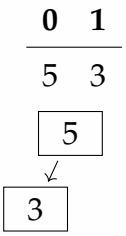
Nach dem Einfügen von „3“:



Nach dem Einfügen von „5“:

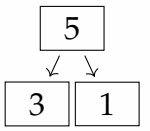


Nach dem Vertauschen von „5“ und „3“:



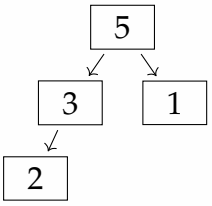
Nach dem Einfügen von „1“:

0	**1**	**2**
5	3	1



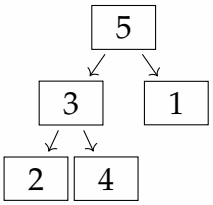
Nach dem Einfügen von „2“:

0	**1**	**2**	**3**
5	3	1	2



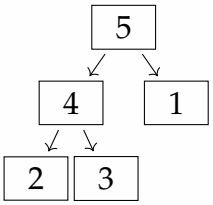
Nach dem Einfügen von „4“:

0	**1**	**2**	**3**	**4**
5	3	1	2	4



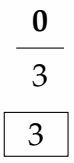
Nach dem Vertauschen von „4“ und „3“:

0	**1**	**2**	**3**	**4**
5	4	1	2	3

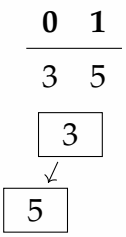


Ausführlicher als Min-Halde

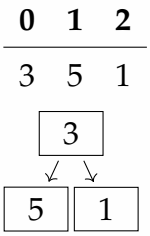
Nach dem Einfügen von „3“:



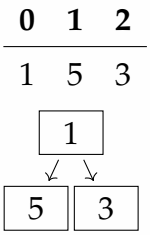
Nach dem Einfügen von „5“:



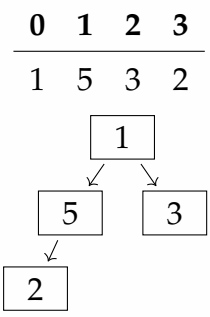
Nach dem Einfügen von „1“:



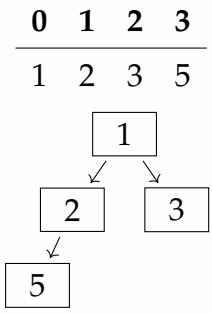
Nach dem Vertauschen von „1“ und „3“:



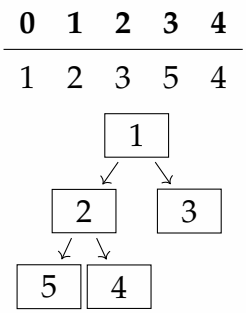
Nach dem Einfügen von „2“:



Nach dem Vertauschen von „2“ und „5“:



Nach dem Einfügen von „4“:



Geben Sie die Ergebnisse an (Zeichnung)

(b) Geben Sie zwei Merkmale an, bei denen sich Heaps und binäre Suchbäume wesentlich unterscheiden. Ein wesentlicher Unterschied zwischen Bubblesort und Mergesort ist z. B. die *worst case* Laufzeit mit $\mathcal{O}(n^2)$ für Bubblesort und $\mathcal{O}(n \log n)$ für Mergesort.

Lösungsvorschlag

	Binärer Suchbaum	Heap
Suchen beliebiger Wert (worst case)	$\mathcal{O}(\log(n))$	$\mathcal{O}(n)$
Suchen Min-Max (average case)	$\mathcal{O}(\log(n))$	$\mathcal{O}(1)$

a

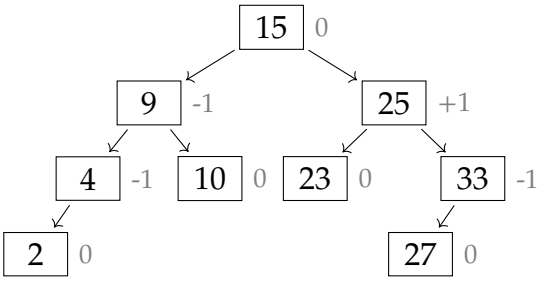
[a] `https://cs.stackexchange.com/q/27860`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/09/Thema-2/Aufgabe-7.tex`

Examensaufgabe „AVL 15,9,25,4,10,23,33,2,27; Einfüge 1,28; Löschen 15“ (66115-2012-H.T2-A8)

AVL-Baum

Gegeben sei der folgende AVL-Baum T. Führen Sie auf T folgende Operationen durch.



Lösungsvorschlag

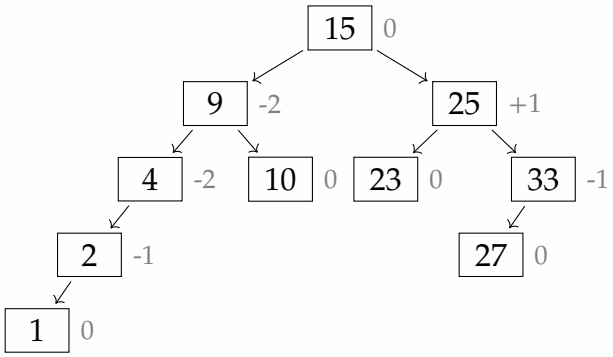
Wir führen alle Operationen am Ursprungsbaum T durch und nicht am veränderten Baum.

(a) Einfüge-Operationen:

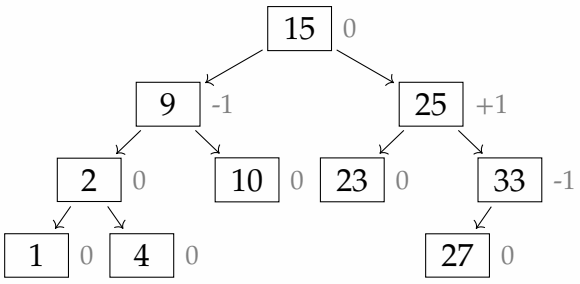
(i) Fügen Sie den Wert 1 in T ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

Nach dem Einfügen von „1“:



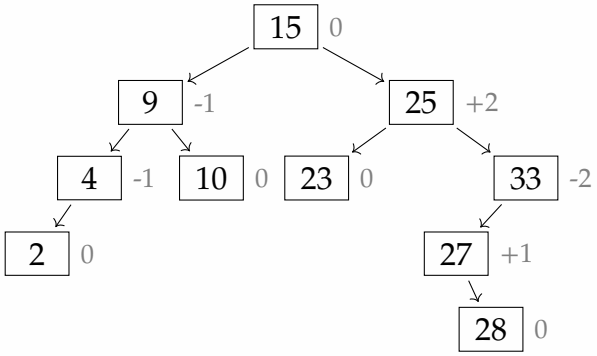
Nach der Rechtsrotation:



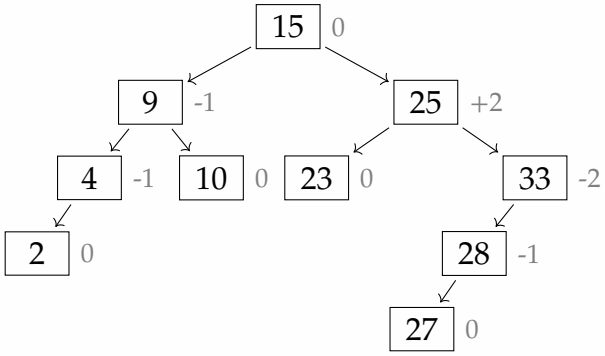
(ii) Fügen Sie nun den Wert 28 in T ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

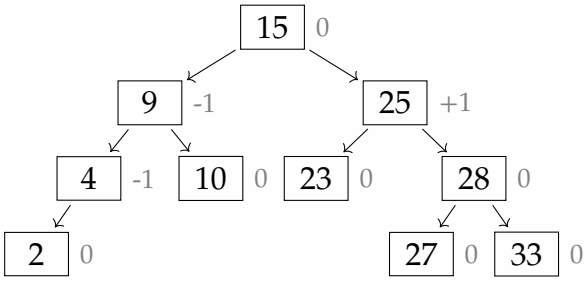
Nach dem Einfügen von „28“:



Nach der Linksrotation:



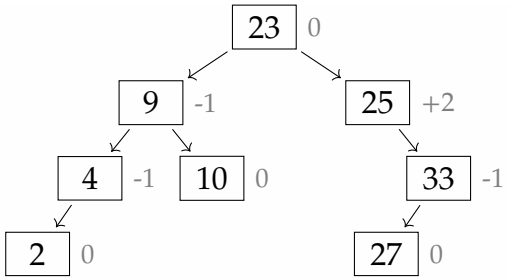
Nach der Rechtsrotation:



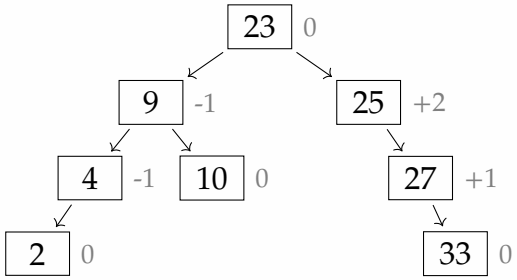
(b) Löschen Sie aus T den Wert 15. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

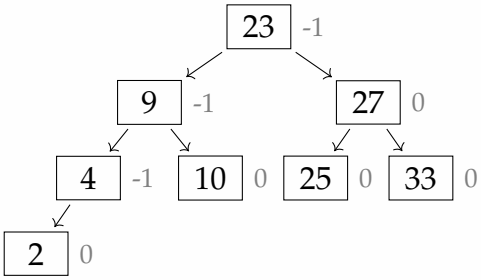
Nach dem Löschen von „15“:



Nach der Rechtsrotation:



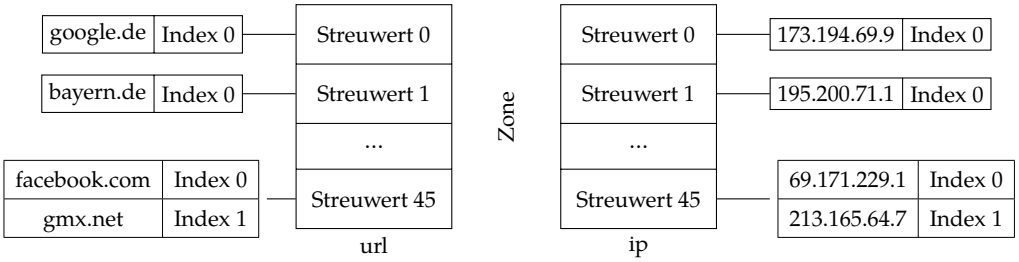
Nach der Linksrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/09/Thema-2/Aufgabe-8.tex`

Examensaufgabe „IP und ULR mit Hashes“ (66115-2013-F.T1-A6)

Um die URL (zum Beispiel google.de) und die zugehörige IP des Servers (hier 173.194.69.9) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion `hash()` den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

(a) Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.

(b) Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```
class Zone {
  private ArrayList<ArrayList<String>> urlList =
    new ArrayList<ArrayList<String>>();
  private ArrayList<ArrayList<String>> ipList =
    new ArrayList<ArrayList<String>>();
  public int hash(String url) {/* calculates hash-value h, >=0 */}
}
```

(i) Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `Zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

Lösungsvorschlag

```
boolean exists(int h) {
  if (urlList.size() - 1 < h || ipList.size() - 1 < h)
    return false;

  ArrayList<String> urlCollisionList = urlList.get(h);
  ArrayList<String> ipCollisionList = ipList.get(h);
  if (urlCollisionList.size() == 0 || ipCollisionList.size() == 0)
    return false;

  return true;
```

```
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java`

(ii) Die Methode `int getIndex (string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

Lösungsvorschlag

```
int getIndex(String url, ArrayList<String> urlList) {
  for (int i = 0; i < urlList.size(); i++) {
    if (urlList.get(i).equals(url))
      return i;
  }
  return -1;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java`

(iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup (String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

Lösungsvorschlag

```
String lookup(String url) {
  int h = hash(url);
  int collisionIndex = getIndex(url, urlList.get(h));
  if (collisionIndex == -1)
    return "Die URL kannte nicht in der Tabelle gefunden werden";
  return ipList.get(h).get(collisionIndex);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/03/Thema-1/Aufgabe-6.tex`

Examensaufgabe „Heap und binärer Suchbaum“ (66115-2013-H.T2-A7)

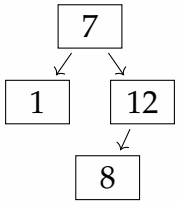
Binärbaum
Halde (Heap)

(a)

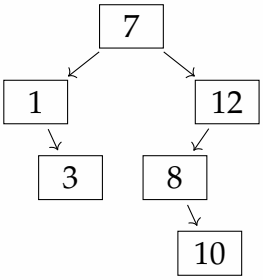
(i) Fügen Sie nacheinander die Zahlen 7, 1, 12, 8, 10, 3, 5 in einen leeren binären Suchbaum ein und zeichnen Sie den Suchbaum nach „8“ und nach „3“.

Lösungsvorschlag

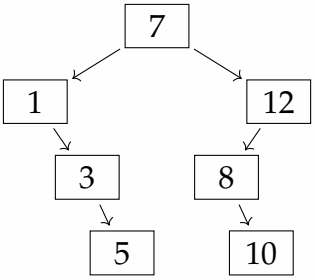
Nach dem Einfügen von „8“:



Nach dem Einfügen von „3“:



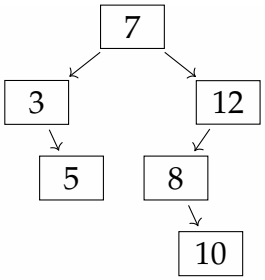
Nach dem Einfügen von „5“:



(ii) Löschen Sie die „1“ aus dem in (i) erstellten Suchbaum und zeichnen Sie den Suchbaum.

Lösungsvorschlag

Nach dem Löschen von „1“:



(iii) Fügen Sie 7, 1, 12, 8, 10, 3, 5 in einen leeren MIN-Heap ein, der bzgl. „$\leq$“ angeordnet ist. Geben Sie den Heap nach jedem Element an.

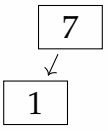
Lösungsvorschlag

Nach dem Einfügen von „7“:

0
7

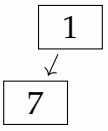
Nach dem Einfügen von „1“:

0	**1**
7	1



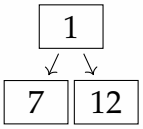
Nach dem Vertauschen von „1“ und „7“:

0	**1**
1	7



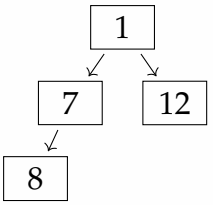
Nach dem Einfügen von „12“:

0	**1**	**2**
1	7	12



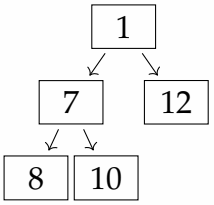
Nach dem Einfügen von „8“:

0	**1**	**2**	**3**
1	7	12	8



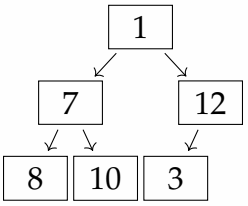
Nach dem Einfügen von „10“:

0	**1**	**2**	**3**	**4**
1	7	12	8	10



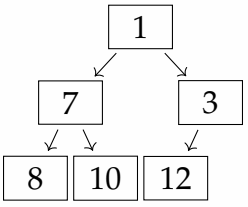
Nach dem Einfügen von „3“:

0	**1**	**2**	**3**	**4**	**5**
1	7	12	8	10	3

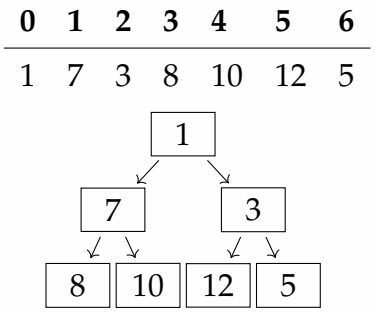


Nach dem Vertauschen von „3“ und „12“:

0	**1**	**2**	**3**	**4**	**5**
1	7	3	8	10	12



Nach dem Einfügen von „5“:



(b) Was ist die worst-case Laufzeit in O-Notation für das Einfügen eines Elements in einen Heap der Größe n? Begründen Sie ihre Antwort.

Lösungsvorschlag

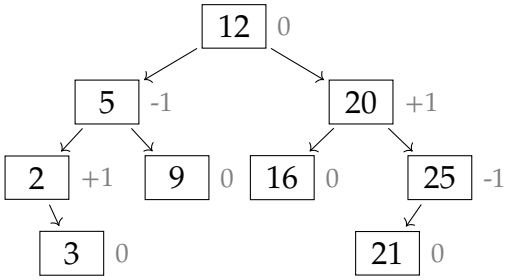
Die worst-case Laufzeit berechnet sich aus dem Aufwand für das Durchsickern eines eingefügten Elementes. Da das Durchsickern entlang eines Pfades im Baum erfolgt, entspricht der Aufwand im ungünstigsten Fall der Höhe des Baumes, ðlog$_2$ n. Insgesamt ergibt sich somit eine worst-case Laufzeit von $\mathcal{O}(\log n)$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/09/Thema-2/Aufgabe-7.tex`

Examensaufgabe „AVL-Baum 12,5,20,2,9,16,25,3,21“ (66115-2013-H.T2-A8)

AVL-Baum

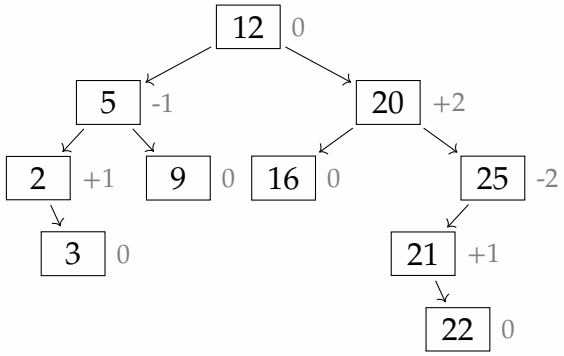
Gegeben sei der folgende AVL-Baum T. Führen Sie auf T folgende Operationen durch.



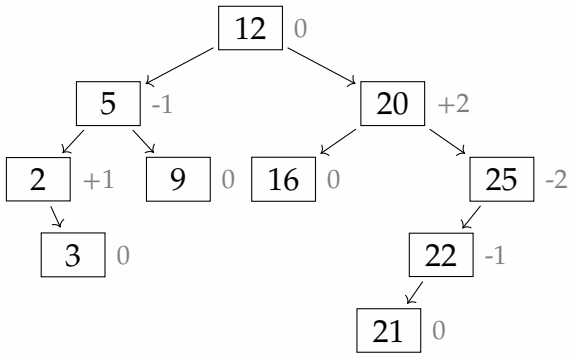
(a) Fügen Sie den Wert 22 in T ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

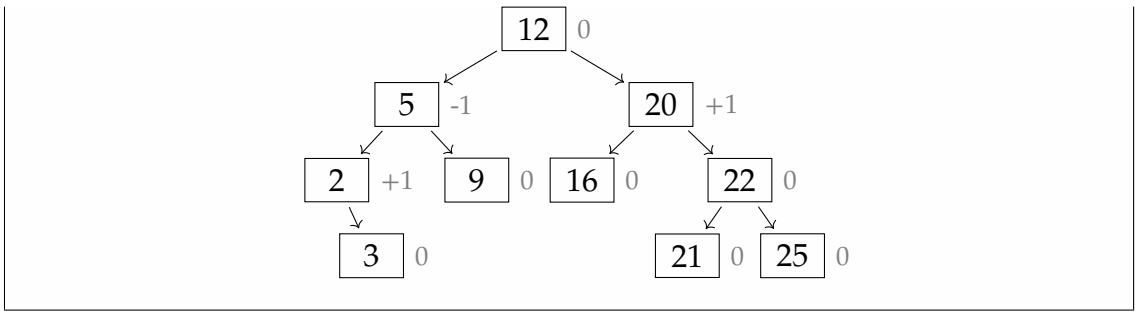
Nach dem Einfügen von „22“:



Nach der Linksrotation:



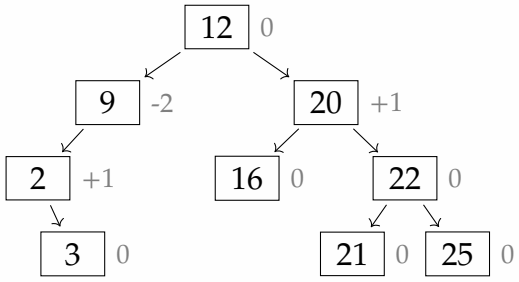
Nach der Rechtsrotation:



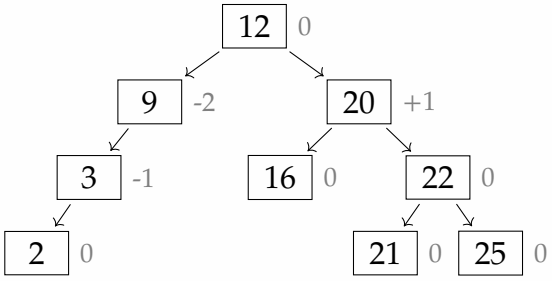
(b) Löschen Sie danach die 5. Balancieren Sie T falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

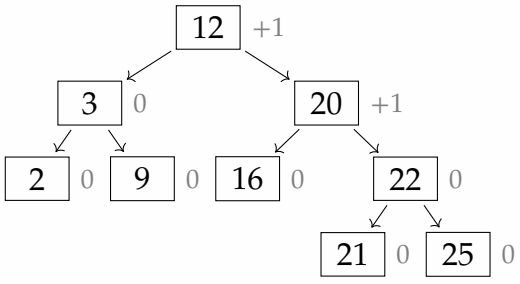
Nach dem Löschen von „5“:



Nach der Linksrotation:



Nach der Rechtsrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/09/Thema-2/Aufgabe-8.tex`

Examensaufgabe „Binäre Bäume“ (66115-2014-F.T1-A2)

Binärbaum

Implementieren Sie in einer objekt-orientierten Sprache Ihrer Wahl eine Klasse namens `BinBaum`, deren Instanzen binäre Bäume mit ganzzahligen Datenknoten darstellen, nach folgender Spezifikation:

(a) Beginnen Sie zunächst mit der Datenstruktur selbst:

- Mit Hilfe des Konstruktors soll ein neuer Baum erstellt werden, der aus einem einzelnen Knoten besteht, in dem der dem Konstruktor als Parameter übergebene Wert (ganzzahlig, in Java z.B. `int`) gespeichert ist.

Lösungsvorschlag

```
class Knoten {
  int value;

  Knoten left;
  Knoten right;

  public Knoten(int value) {
    this.value = value;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java`

- Die Methoden `setLeft(int value)` bzw. `setRight(int value)` sollen den linken bzw. rechten Teilbaum des aktuellen Knotens durch einen jeweils neuen Teilbaum ersetzen, der seinerseits aus einem einzelnen Knoten besteht, in dem der übergebene Wert `value` gespeichert ist. Sie haben keinen Rückgabewert.

Lösungsvorschlag

```
  public void setLeft(Knoten left) {
    this.left = left;
  }

  public void setRight(Knoten right) {
    this.right = right;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java`

- Die Methoden `getLeft()` bzw. `getRight()` sollen den linken bzw. rechten Teilbaum zurückgeben (bzw. `null`, wenn keiner vorhanden ist)

Lösungsvorschlag

```
  public Knoten getLeft() {
    return left;
  }

  public Knoten getRight() {
    return right;
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

- Die Methode `int getValue()` soll den Wert zurückgeben, der im aktuellen Wurzelknoten gespeichert ist.

Lösungsvorschlag

```
public int getValue() {
  return value;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

(b) Implementieren Sie nun die Methoden `preOrder()` bzw. `postOrder()`. Sie sollen die Knoten des Baumes mit Tiefensuche traversieren, die Werte dabei in pre-order bzw. post-order Reihenfolge in eine Liste (z.B. `List<Integer>`) speichern und diese Ergebnisliste zurückgeben. Die Tiefensuche soll dabei zuerst in den linken und dann in den rechten Teilbaum absteigen.

Lösungsvorschlag

```
void preOrder(Knoten knoten, List<Integer> list) {
  if (knoten != null) {
    list.add(knoten.getValue());
    preOrder(knoten.getLeft(), list);
    preOrder(knoten.getRight(), list);
  }
}

List<Integer> preOrder() {
  List<Integer> list = new ArrayList<>();
  preOrder(head, list);
  return list;
}

void postOrder(Knoten knoten, List<Integer> list) {
  if (knoten != null) {
    postOrder(knoten.getLeft(), list);
    postOrder(knoten.getRight(), list);
    list.add(knoten.getValue());
  }
}

List<Integer> postOrder() {
  List<Integer> list = new ArrayList<>();
  postOrder(head, list);
  return list;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

(c) Ergänzen Sie schließlich die Methode `isSearchTree ()`. Sie soll überprüfen, ob der Binärbaum die Eigenschaften eines binären Suchbaums erfüllt. Beachten Sie, dass die Laufzeit-Komplexität Ihrer Implementierung linear zur Anzahl der Knoten im Baum sein muss.

Lösungsvorschlag

```
boolean isSearchTree(Knoten knoten) {
  if (knoten == null) {
    return true;
  }

  if (knoten.getLeft() != null && knoten.getValue() <
↪ knoten.getLeft().getValue()) {
    return false;
  }

  if (knoten.getRight() != null && knoten.getValue() >
↪ knoten.getRight().getValue()) {
    return false;
  }

  return isSearchTree(knoten.getLeft()) && isSearchTree(knoten.getRight());
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Einfügen und dreimal einen Knoten löschen“ (66115-2014-F.T1-A3)

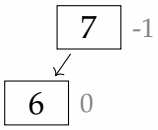
AVL-Baum

(a) Fügen Sie die Zahlen (7, 6, 2, 1, 5, 3, 8, 4) in dieser Reihenfolge in einen anfangs leeren AVL Baum ein. Stellen Sie die AVL Eigenschaft ggf. nach jedem Einfügen mit geeigneten Rotationen wieder her. Zeichnen Sie den AVL Baum einmal vor und einmal nach jeder einzelnen Rotation.

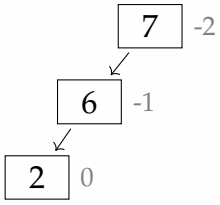
Lösungsvorschlag

Nach dem Einfügen von „7“:

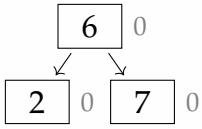
Nach dem Einfügen von „6“:



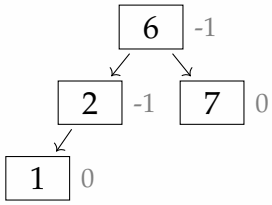
Nach dem Einfügen von „2“:



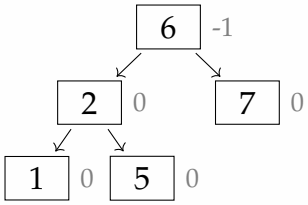
Nach der Rechtsrotation:



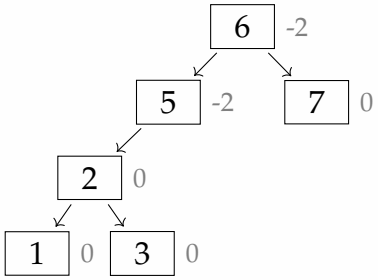
Nach dem Einfügen von „1“:



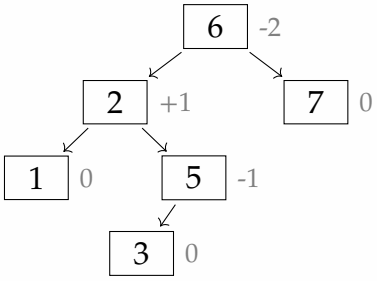
Nach dem Einfügen von „5“:



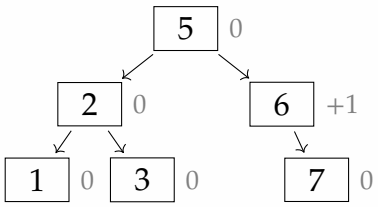
Nach der Linksrotation:



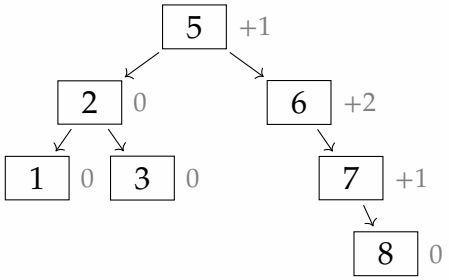
Nach dem Einfügen von „3“:



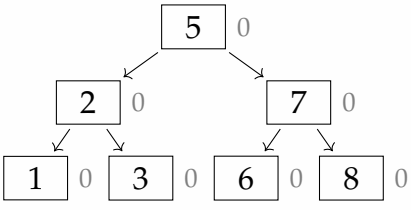
Nach der Rechtsrotation:



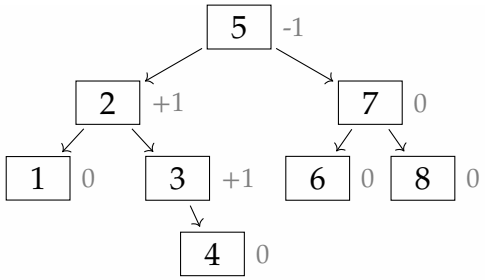
Nach dem Einfügen von „8“:



Nach der Linksrotation:

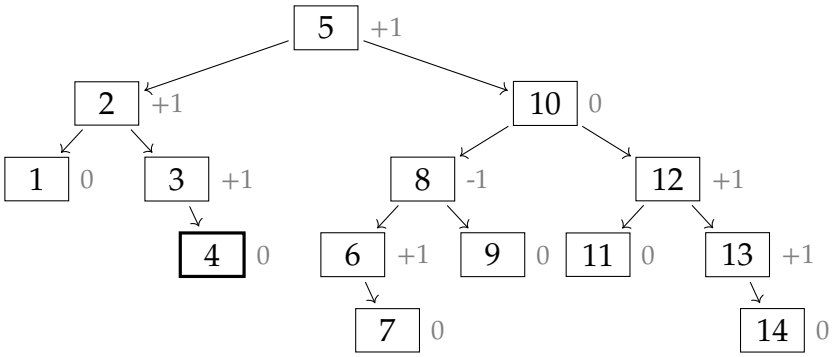


Nach dem Einfügen von „4“:



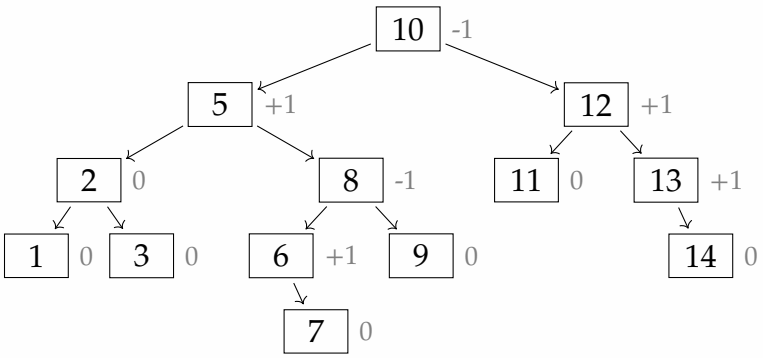
(b) Entfernen Sie den jeweils markierten Knoten aus den folgenden AVL-Bäumen. Stellen Sie die AVL-Eigenschaft ggf. durch geeignete Rotationen wieder her. Zeichnen Sie nur den resultierenden Baum.

(i) Baum 1:

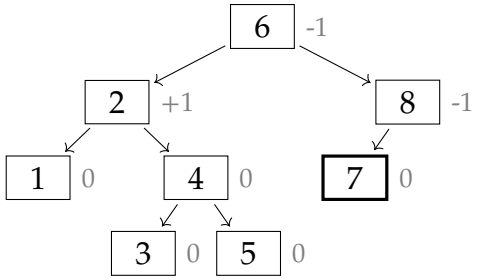


Lösungsvorschlag

Nach dem Löschen von „4“:

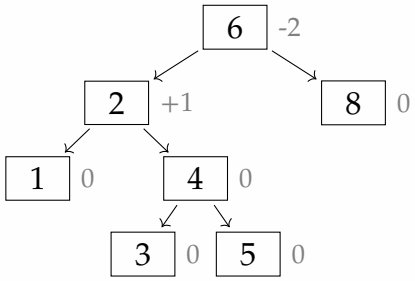


(ii) Baum 2:

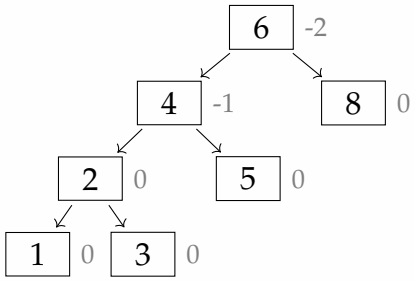


Lösungsvorschlag

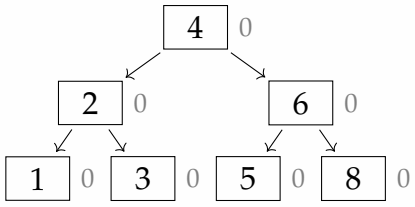
Nach dem Löschen von „7“:



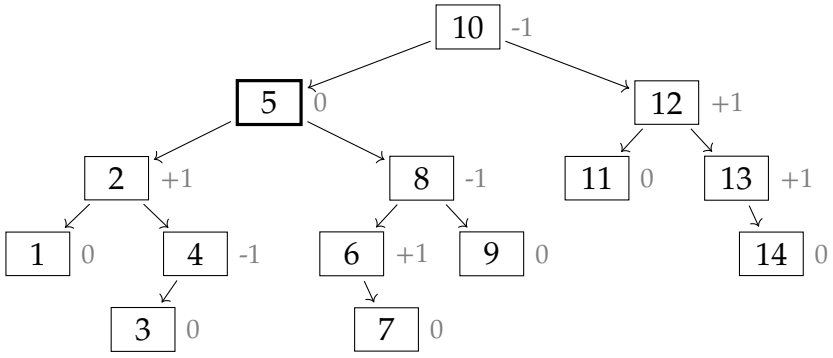
Nach der Linksrotation:



Nach der Rechtsrotation:

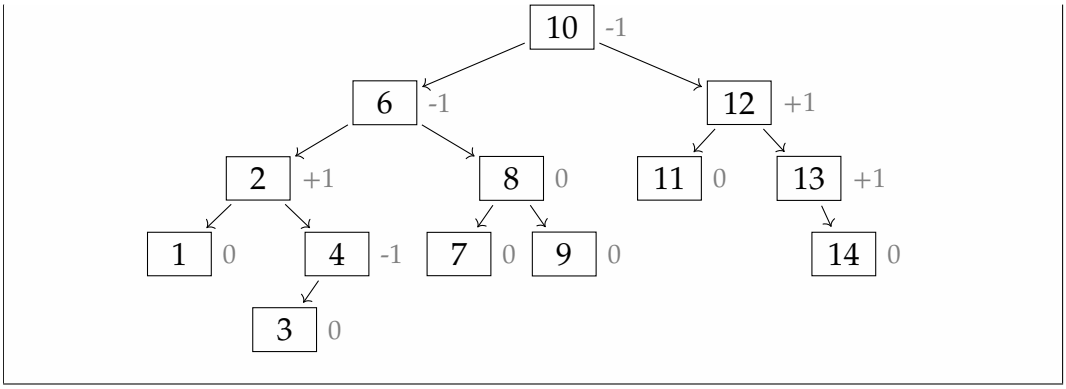


(iii) Baum 3:



Lösungsvorschlag

Nach dem Löschen von „5“:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/03/Thema-1/Aufgabe-3.tex`

Examensaufgabe „Hashing mit verketteten Listen und offener Adressierung“ (66115-2016-F.T2-A4)

Streutabellen (Hashing)
Separate Verkettung

Betrachte eine Hashtabelle der Größe $m = 10$.

(a) Welche der folgenden Hashfunktionen ist für Hashing mit verketteten Listen am besten geeignet? Begründen Sie Ihre Wahl!

(i) $h_1(x) = (4x + 3) \mod m$

Lösungsvorschlag

1 $h_1(1) = (4 \cdot 1 + 3) \mod 10 = 7$
2 $h_1(2) = (4 \cdot 2 + 3) \mod 10 = 1$
3 $h_1(3) = (4 \cdot 3 + 3) \mod 10 = 5$
4 $h_1(4) = (4 \cdot 4 + 3) \mod 10 = 9$
5 $h_1(5) = (4 \cdot 5 + 3) \mod 10 = 3$
6 $h_1(6) = (4 \cdot 6 + 3) \mod 10 = 7$
7 $h_1(7) = (4 \cdot 7 + 3) \mod 10 = 1$
8 $h_1(8) = (4 \cdot 8 + 3) \mod 10 = 5$
9 $h_1(9) = (4 \cdot 9 + 3) \mod 10 = 9$
10 $h_1(10) = (4 \cdot 10 + 3) \mod 10 = 3$

(ii) $h_2(x) = (3x + 3) \mod m$

Lösungsvorschlag

1 $h_2(1) = (3 \cdot 1 + 3) \mod 10 = 6$
2 $h_2(2) = (3 \cdot 2 + 3) \mod 10 = 9$
3 $h_2(3) = (3 \cdot 3 + 3) \mod 10 = 2$
4 $h_2(4) = (3 \cdot 4 + 3) \mod 10 = 5$
5 $h_2(5) = (3 \cdot 5 + 3) \mod 10 = 8$
6 $h_2(6) = (3 \cdot 6 + 3) \mod 10 = 1$
7 $h_2(7) = (3 \cdot 7 + 3) \mod 10 = 4$
8 $h_2(8) = (3 \cdot 8 + 3) \mod 10 = 7$
9 $h_2(9) = (3 \cdot 9 + 3) \mod 10 = 0$
10 $h_2(10) = (3 \cdot 10 + 3) \mod 10 = 3$

Lösungsvorschlag

Damit die verketteten Listen möglichst klein bleiben, ist eine möglichst gleichmäßige Verteilung der Schlüssel in die Buckets anzustreben. h_2 ist dafür besser geeignet als h_1, da h_2 in alle Buckets Schlüssel ablegt, h_1 jedoch nur in Buckets mit ungerader Zahl.

(b) Welche der folgenden Hashfunktionen ist für Hashing mit offener Adressierung am besten geeignet? Begründen Sie Ihre Wahl! Offene Adressierung

(i) $h_1(x,i) = (7 \cdot x + i \cdot m) \mod m$

(ii) $h_2(x,i) = (7 \cdot x + i \cdot (m-1)) \mod m$

Lösungsvorschlag

$h_2(x,i)$ ist besser geeignet. h_1 sondiert immer im selben Bucket, $(i \cdot m) \mod m$ heben sich gegenseitig auf, zum Beispiel ergibt:

- $h_1(3,0) = (7 \cdot 3 + 0 \cdot 10) \mod 10 = 1$
- $h_1(3,1) = (7 \cdot 3 + 1 \cdot 10) \mod 10 = 1$
- $h_1(3,2) = (7 \cdot 3 + 2 \cdot 10) \mod 10 = 1$

Während hingegen h_2 verschiedene Buckets belegt.

- $h_2(3,0) = (7 \cdot 3 + 0 \cdot 9) \mod 10 = 1$
- $h_2(3,1) = (7 \cdot 3 + 1 \cdot 9) \mod 10 = 0$
- $h_2(3,2) = (7 \cdot 3 + 2 \cdot 9) \mod 10 = 9$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-2/Aufgabe-4.tex

Examensaufgabe „Vergleich Suchbäume“ (66115-2016-F.T2-A7)

Bäume
Rot-Schwarz-Baum
AVL-Baum
Halde (Heap)

Wofür eignen sich die folgenden Baum-Datenstrukturen im Vergleich zu den anderen angeführten Baumstrukturen am besten, und warum. Sprechen Sie auch die Komplexität der wesentlichen Operationen und die Art der Speicherung an.

(a) Rot-Schwarz-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(b) AVL-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(c) Binärer-Heap

Lösungsvorschlag

Verwendungszweck zum effizienten Sortieren von Elementen. [a]

Einfügen (Zeitkomplexität)
$\mathcal{O}(1)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

B-Baum
R-Baum

Suchen (Zeitkomplexität)
$\mathcal{O}(n)$ (im Durchschnitt)
$\mathcal{O}(n)$ (im schlechtesten Fall) [b]

[a] deut. Wikipedia
[b] engl. Wikipedia

(d) B-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(e) R-Baum

Lösungsvorschlag

Verwendungszweck Ein R-Baum erlaubt die schnelle Suche in mehrdimensionalen ausgedehnten Objekten. [a]

Suchen (Zeitkomplexität)
$\mathcal{O}(\log_M n)$ (im Durchschnitt) [b]
$\mathcal{O}(n)$ (im schlechtesten Fall) [c]

[a] deut. Wikipedia
[b] eng. Wikipedia
[c] Simon Fraser University, Burnaby, Kanada

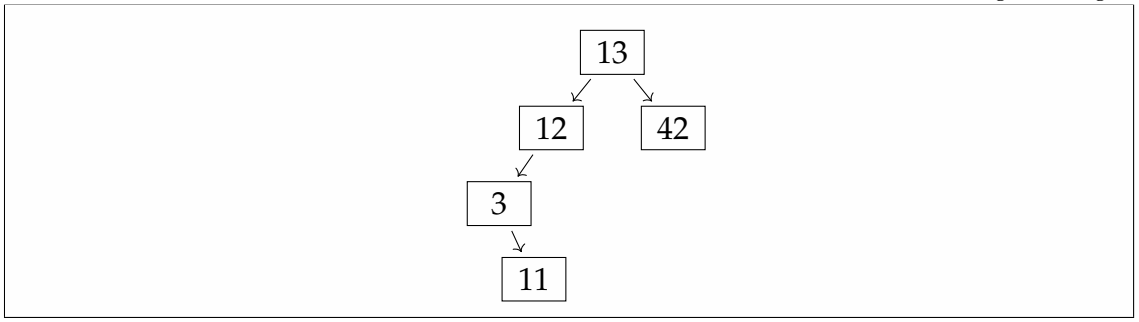
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-2/Aufgabe-7.tex`

Examensaufgabe „Binärbaum, Halde, AVL“ (66115-2017-H.T2-A8)

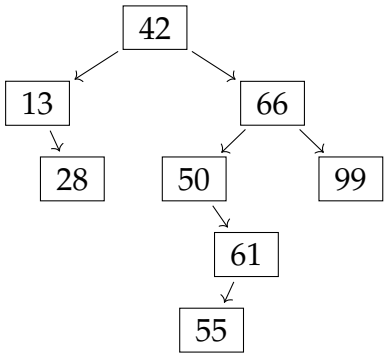
AVL-Baum
Halde (Heap)
Binärbaum

(a) Fügen Sie die Zahlen 13, 12, 42, 3, 11 in der gegebenen Reihenfolge in einen zunächst leeren binären Suchbaum mit aufsteigender Sortierung ein. Stellen Sie nur das Endergebnis dar.

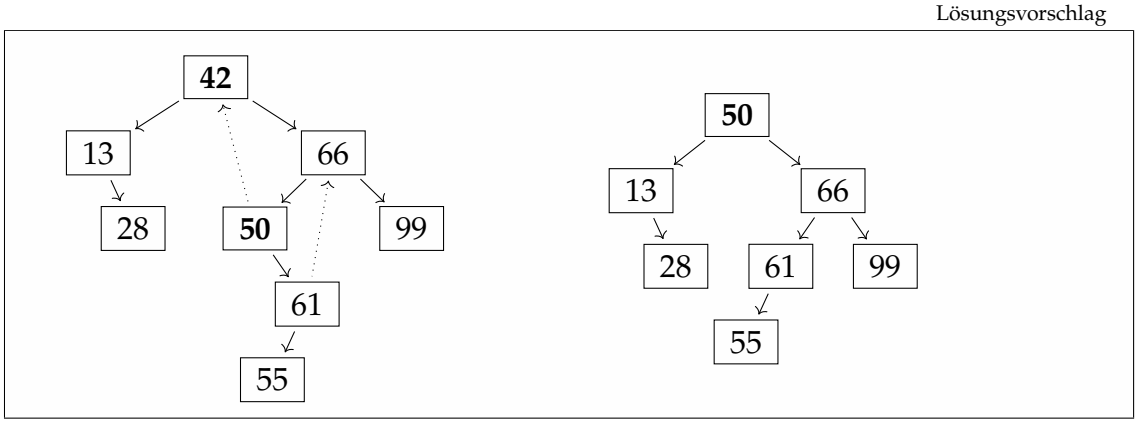
Lösungsvorschlag



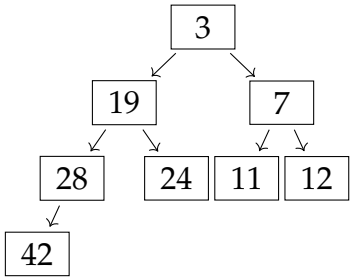
(b) Löschen Sie den Wurzelknoten mit Wert 42 aus dem folgenden *binären* Suchbaum mit aufsteigender Sortierung und ersetzen Sie ihn dabei durch einen geeigneten Wert aus dem *rechten* Teilbaum. Lassen Sie möglichst viele Teilbäume unverändert und erhalten Sie die Suchbaumeigenschaft.



Lösungsvorschlag

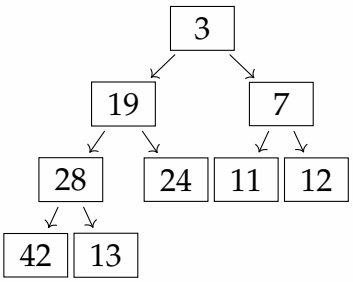


(c) Fügen Sie einen neuen Knoten mit dem Wert 13 in die folgende Min-Halde ein und stellen Sie anschließend die Halden-Eigenschaft vom neuen Blatt aus beginnend wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

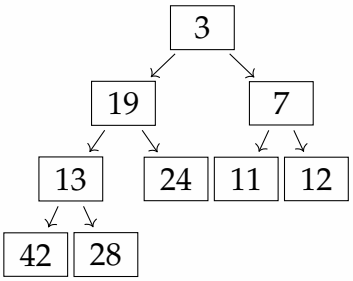


Lösungsvorschlag

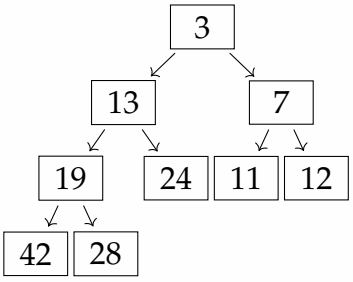
Nach dem Einfügen von „13“:



Nach dem Vertauschen von „13“ und „28“:



Nach dem Vertauschen von „13“ und „19“:

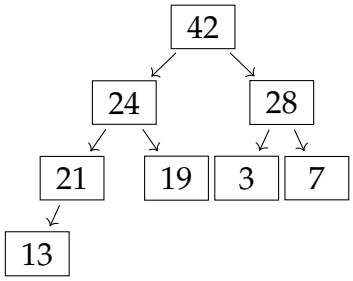


(d) Geben Sie für die ursprüngliche Min-Halde aus Teilaufgabe c)(ðohne den neu eingefügten Knoten mit dem Wert 13) die Feld-Einbettung (Array-Darstellung) an.

Lösungsvorschlag

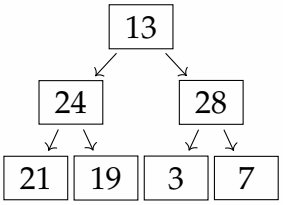
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	19	7	28	24	11	12	42

(e) Löschen Sie den Wurzelknoten mit Wert 42 aus der folgenden Max-Halde und stellen Sie anschließend die Halden-Eigenschaft ausgehend von einer neuen Wurzel wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

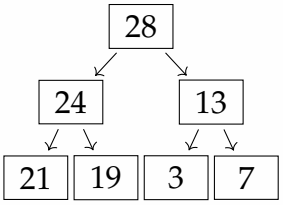


Lösungsvorschlag

Nach dem Ersetzen von „42“ mit „13“:

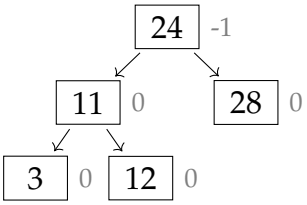


Nach dem Vertauschen von „13“ und „28“:



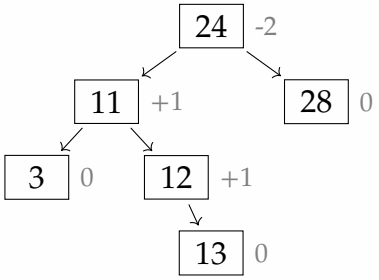
(f) Fügen Sie in jeden der folgenden AVL-Bäume mit aufsteigender Sortierung jeweils einen neuen Knoten mit dem Wert 13 ein und führen Sie anschließend bei Bedarf die erforderliche(n) Rotation(en) durch. Zeichnen Sie den Baum vor und nach den Rotationen.

(i) AVL-Baum A

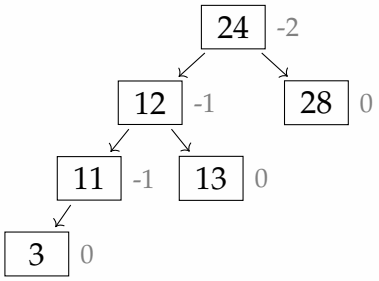


Lösungsvorschlag

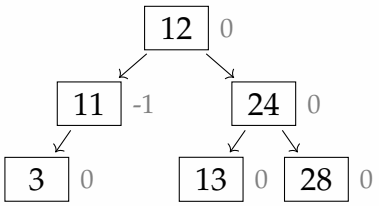
Nach dem Einfügen von „13“:



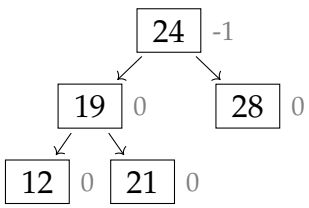
Nach der Linksrotation:



Nach der Rechtsrotation:

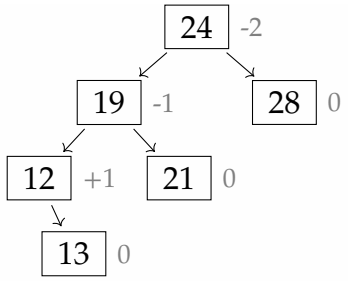


(ii) AVL-Baum B

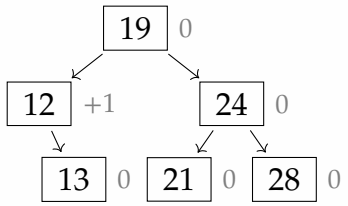


Lösungsvorschlag

Nach dem Einfügen von „13“:



Nach der Rechtsrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/09/Thema-2/Aufgabe-8.tex`

Examensaufgabe „AVL-Baum 5,14,28,10,3,12,13“ (66115-2018-F.T2-A8)

AVL-Baum

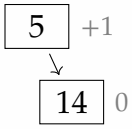
Bearbeiten Sie folgende Aufgaben zu AVL-Suchbäumen. Geben Sie jeweils bei jeder einzelnen Operation zum Einfügen, Löschen, sowie jeder elementaren Operation zum Wiederherstellen der AVL-Baumeigenschaften den entstehenden Baum als Baumzeichnung an. Geben Sie zur Darstellung der elementaren Operation auch vorübergehend ungültige AVL-Bäume an und stellen Sie Doppelrotationen in zwei Schritten dar. Dabei sollen die durchgeführten Operationen klar gekennzeichnet sein und die Baumknoten immer mit aktuellen Balancewerten versehen sein.

(a) Fügen Sie (manuell) nacheinander die Zahlen 5, 14, 28, 10, 3, 12, 13 in einen anfangs leeren AVL-Baum ein.

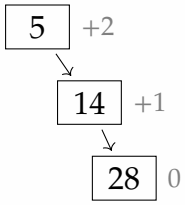
Lösungsvorschlag

Einfügen von „5“:

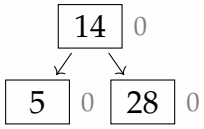
Einfügen von „14“:



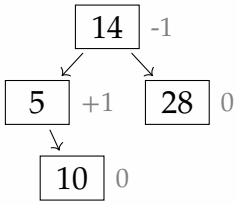
Einfügen von „28“:



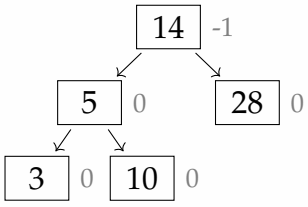
Linksrotation:



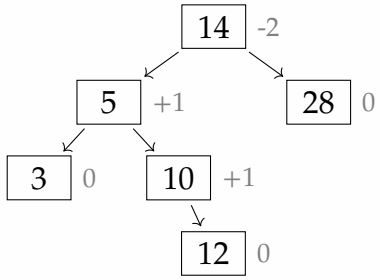
Einfügen von „10“:



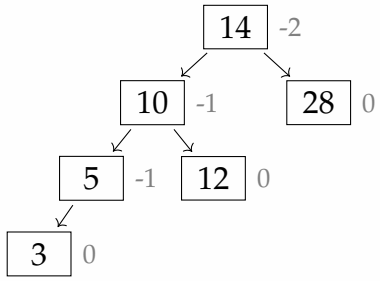
Einfügen von „3“:



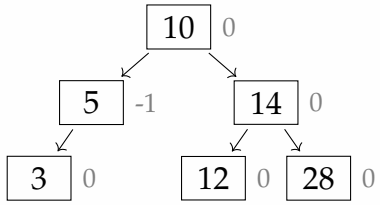
Einfügen von „12“:



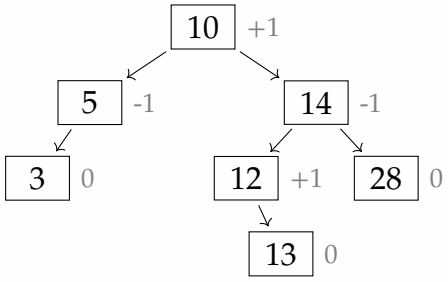
Linksrotation:



Rechtsrotation:

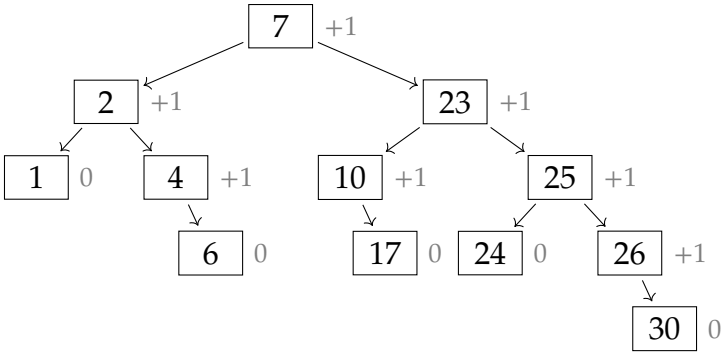


Einfügen von „13“:



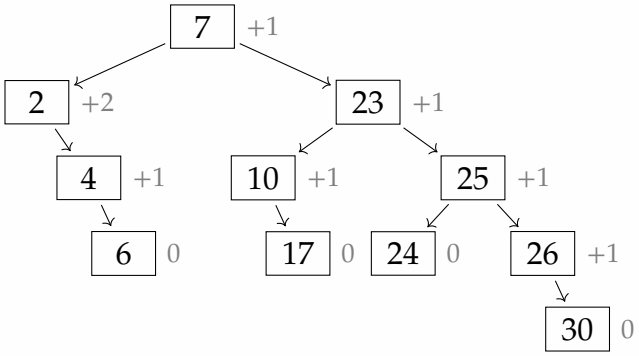
(b) Gegeben sei folgender AVL-Baum. Löschen Sie nacheinander die Knoten 1 und 23. Bei Wahlmöglichkeiten nehmen Sie jeweils den kleineren Wert anstatt eines

größeren.

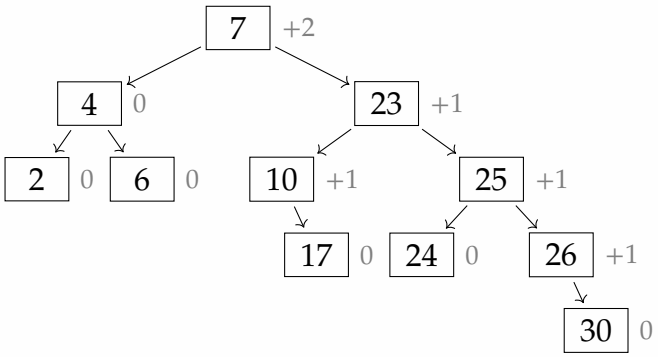


Lösungsvorschlag

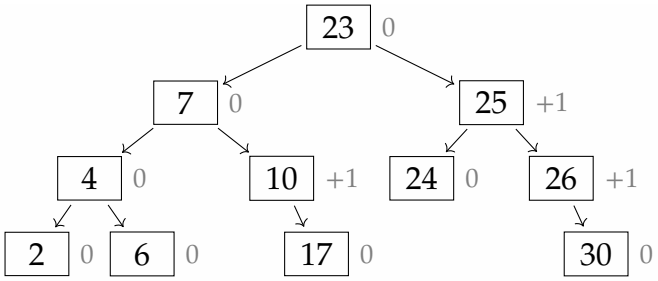
Löschen von „1“:



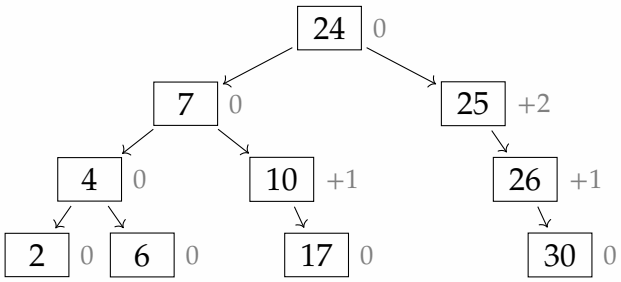
Linksrotation:



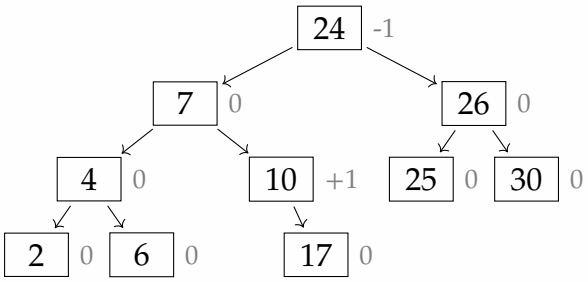
Linksrotation:



Löschen von „23“:



Linksrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-8.tex`

Examensaufgabe „Binärer Suchbaum“ (66115-2018-H.T1-A5)

AVL-Baum

Hinweis: Wir betrachten in dieser Aufgabe binäre Suchbäume, bei denen jeder innere Knoten genau zwei Kinder hat. Schlüssel werden nur in den inneren Knoten gespeichert - die Blätter speichern keinerlei Informationen.

(a) Welche Eigenschaften muss ein binärer Suchbaum haben, damit er ein AVL-Baum ist?

Lösungsvorschlag

Er muss die zusätzliche Eigenschaft haben, dass sich an jedem Knoten die Höhe der beiden Teilbäume um höchstens eins unterscheidet

(b) Mit $n(h)$ bezeichnen wir die minimale Anzahl innerer Knoten eines AVL-Baums der Höhe h.

(i) Begründen Sie, dass $n(1) = 1$ und $n(2) = 2$.

(ii) Begründen Sie, dass $n(h) = 1 + n(h-1) + n(h-2)$.

(iii) Folgern Sie, dass $n(h) > 2^{\frac{h}{2}-1}$.

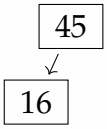
(c) Warum ist die Höhe jedes AVL-Baums mit n inneren Knoten $O(\log n)$?

(d) Fügen Sie die Elemente $(45, 16, 79, 31, 51, 87, 49, 61)$ in der angegebenen Reihenfolge in einen anfangs leeren binären Suchbaum ein (ohne Rebalancierungen). Zeichnen Sie den resultierenden Suchbaum nach jeder Einfügeoperation.

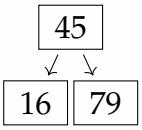
Lösungsvorschlag

Einfügen von „45“:

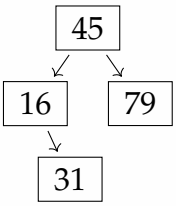
Einfügen von „16“:



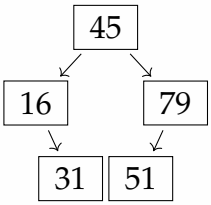
Einfügen von „79“:



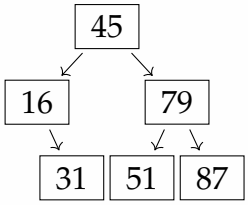
Einfügen von „31“:



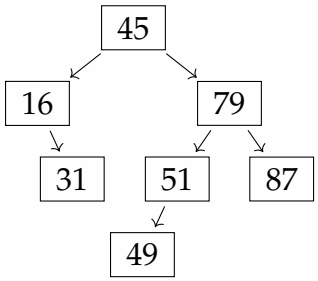
Einfügen von „51“:



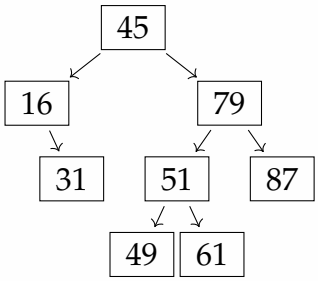
Einfügen von „87“:



Einfügen von „49“:



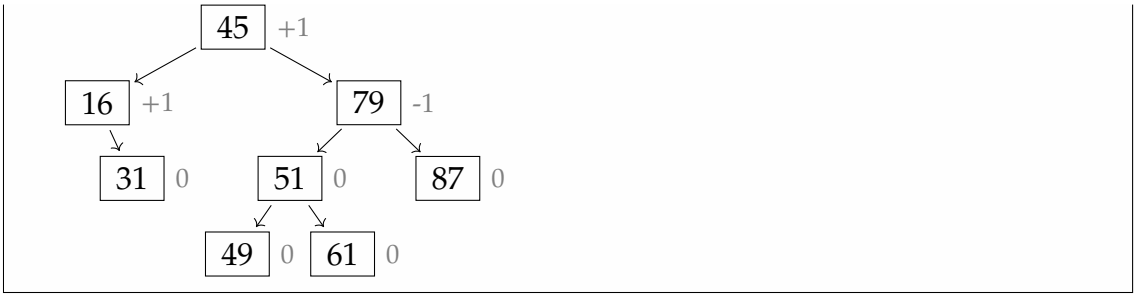
Einfügen von „61“:



(e) Ist der resultierende Suchbaum aus Teilaufgabe 5.4 ein AVL-Baum? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Ja, wie in der untenstehenden Grafik zu sehen ist, unterscheiden sich die Höhe der Teilbäume von allen Knoten nur um höchstens eins.



(f) Das Einfügen in einen AVL-Baum funktioniert (zunächst) wie beim binären Suchbaum durch Erweitern eines äußeren Knotens w:

vor dem Einfügen von 54 nach dem Einfügen von 54

Anschließend wird die AVL-Baum Eingenschaft (falls notwending) durch eine (Doppel-)Rotation wiederhergestellt: Wenn z der erste Knoten auf dem Pfad P von w zur Wurzel ist, der nicht balanciert ist, y das Kind von z auf P und r das Kind von y auf P, und wenn (a, b,c) die Inorder-Reihenfolge von x, y, 2 ist, dann führen wir die Rotation aus, die benötigt wird, um b zum obersten Knoten der drei zu machen.

Die folgende Illustration zeigt den Fall, dass key(y) < key(x) < key(z), ð(a,b,c) = (y,x,z), wobei w ein Knoten in Ty ist.

Sei Ah die Höhe des Teilbaums 73. Für i = 0,1,2 sei h; die Höhe des Teilbaums T; und für v = 2,y,z sei h, die Höhe des Teilbaums mit der Wurzel v vor der Restrukturierung. Begründen Sie, dass

(i) $h_0 = h$
(ii) $h_1 = h - 1$
(iii) $h_2 = h$
(iv) $h_x = h + 1$
(v) $h_y = h + 2$
(vi) $h_z = h + 3$

(g) Welche Höhe haben die Teilbäume mit den Wurzeln x, y, z nach der Restrukturierung? Begründen Sie Ihre Antworten.

(h) Begründen Sie, dass die oben gezeigte Doppelrotation die AVL-Baum-Eigenschaft wiederherstellt.

(i) Beschreiben Sie, wie ein binärer Baum der Höhe h in einem Array repräsentiert werden kann. Wie viel Speicherplatz ist für so eine Darstellung erforderlich?

(j) Warum verwendet man bei der Implementierung von AVL-Bäumen eine verzeigerte Struktur und nicht eine Array-basierte Repräsentation?

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/09/Thema-1/Aufgabe-5.tex`

Examensaufgabe „k-kleinste Elemente“ (66115-2019-F.T2-A1)

Halde (Heap)

Gegeben sei eine unsortierte Liste von n verschiedenen natürlichen Zahlen. Das k-kleinste Element ist das Element, das größer als genau $k-1$ Elemente der Liste ist.

(a) Geben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(n \cdot \log n)$ an, um das k-kleinste Element zu berechnen.

Lösungsvorschlag

a

[a] `https://en.wikipedia.org/wiki/Quickselect`

(b) Gegeben sei nun ein Algorithmus A, der den Median einer unsortierten Liste von n Zahlen in $\mathcal{O}(n)$ Schritten berechnet. Nutzen Sie Algorithmus A um einen Algorithmus B anzugeben, welcher das k-kleinste Element in $\mathcal{O}(n)$ Schritten berechnet.

Argumentieren Sie auch, dass der Algorithmus die gewünschte Laufzeit besitzt.

Lösungsvorschlag

a

[a] `https://en.wikipedia.org/wiki/Median_of_medians`

(c) Geben Sie einen Algorithmus an, der für alle $i = 1 \ldots, \lfloor n/k \rfloor$ das $i \cdot k$-kleinste Element berechnet. Die Laufzeit Ihres Algorithmus sollte $\mathcal{O}(n \cdot log(n/k))$ sein. Sie dürfen weiterhin Algorithmus A, wie in Teilaufgabe (b) beschrieben, nutzen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/03/Thema-2/Aufgabe-1.tex`

Examensaufgabe „Binärer Baum mit Methode foo“ (66115-2019-H.T1-A7)

Bäume

Gegeben sei die folgende Realisierung von binären Bäumen (in einer an Java angelehnten Notation):

```
class Node {
  Node left, right;
  int value;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/herbst/Node.java`

(a) Beschreiben Sie in möglichst wenigen Worten, was die folgende Methode `foo` auf einem nicht-leeren binären Baum berechnet.

```
int foo(Node node) {
  int b = node.value;
  if (b < 0) {
    b = -1 * b;
  }
  if (node.left != null) {
    b = b + foo(node.left);
  }
  if (node.right != null) {
    b = b + foo(node.right);
  }
  return b;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/herbst/Node.java`

Lösungsvorschlag

Die Methode `foo(Node node)` berechnet die Summe aller Knoten des Unterbaums des als Parameter übergebenen Knotens `node`. Der Schüsselwert des Knotens `node` selbst wird in die Summenberechnung mit einbezogen.

(b) Die Laufzeit der Methode `foo(tree)` ist linear in n, der Anzahl von Knoten im übergebenen Baum `tree`. Begründen Sie kurz, warum `foo(tree)` eine lineare Laufzeit hat.

Lösungsvorschlag

Die Methode `foo(Node node)` wird pro Knoten des Baums `tree` genau einmal aufgerufen. Es handelt sich um eine rekursive Methode, die auf den linken und rechten Kindknoten aufgerufen wird. Hat ein Knoten keine Kinder mehr, dann wir die Methode nicht aufgerufen.

(c) Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten die Wirkung der Methode `magic(tree)`. Welche Rolle spielt dabei die Methode `max`?

```java
void magic(Node node) {
  Node m = max(node);
  if (m.value > node.value) {
    // Werte von m und node vertauschen
    int tmp = m.value;
    m.value = node.value;
    node.value = tmp;
  }
  if (node.left != null)
    magic(node.left);
  if (node.right != null)
    magic(node.right);
}

Node max(Node node) {
  Node max = node;
  if (node.left != null) {
    Node tmp = max(node.left);
    if (tmp.value > max.value)
      max = tmp;
  }
  if (node.right != null) {
    Node tmp = max(node.right);
    if (tmp.value > max.value)
      max = tmp;
  }
  return max;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/herbst/Node.java`

Lösungsvorschlag

Methode `magic(tree)` vertauscht für jeden Knoten des Unterbaums den Wert mit dem Maximal-Knoten. Die Methode `max` liefert dabei den Knoten mit der größten Schlüsselwert im Unterbaum des übergebenen Knoten (sich selbst eingeschlossen).

(d) Geben Sie in Abhängigkeit von n, der Anzahl von Knoten im übergebenen Baum `tree`, jeweils eine Rekursionsgleichung für die asymptotische Best-Case-Laufzeit $(B(n))$ und Worst-Case-Laufzeit $(W(n))$ des Aufrufs `magic(tree)` sowie die entsprechende Komplexitätsklasse (Θ) an. Begründen Sie Ihre Antwort.

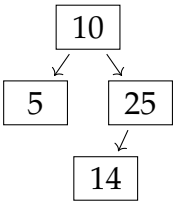
Hinweis: Überlegen Sie, ob die Struktur des übergebenen Baumes Einfluss auf die Laufzeit hat. Die lineare Laufzeit von `max(t)` in der Anzahl der Knoten des Baumes `t` darf vorausgesetzt werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-1/Aufgabe-7.tex`

Examensaufgabe „AVL-Baum 10,5,25,14 erweitern und Knoten entfernen“ (66115-2019-H.T2-A7)

AVL-Baum

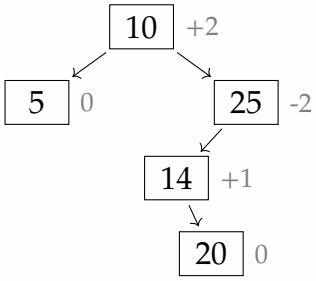
Fügen Sie (manuell) nacheinander die Zahlen $20, 31, 2, 17, 7$ in folgenden AVL-Baum ein. Löschen Sie anschließend aus dem entstandenen Baum nacheinander 14 und 25.



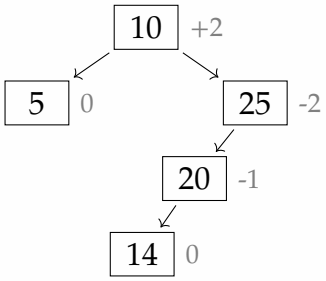
Zeichnen Sie jeweils direkt nach jeder einzelnen Operation zum Einfügen oder Löschen eines Knotens, sowie nach jeder elementaren Rotation den entstehenden Baum. Insbesondere sind evtl. anfallende Doppelrotationen in zwei Schritten darzustellen. Geben Sie zudem an jedem Knoten die Balancewerte an.

Lösungsvorschlag

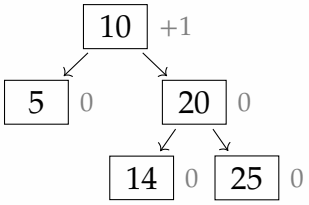
Nach dem Einfügen von „20“:



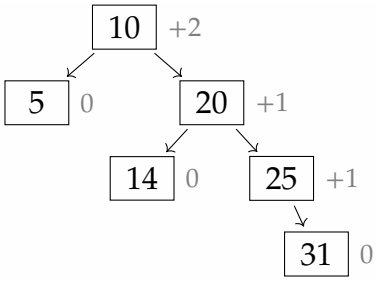
Nach der Linksrotation:



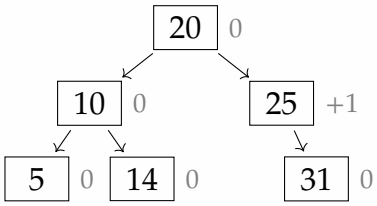
Nach der Rechtsrotation:



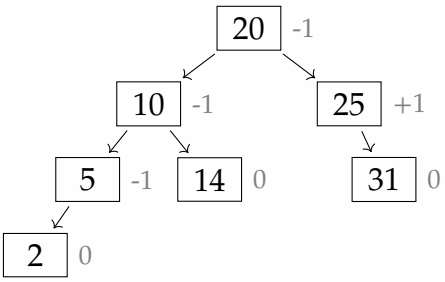
Nach dem Einfügen von „31“:



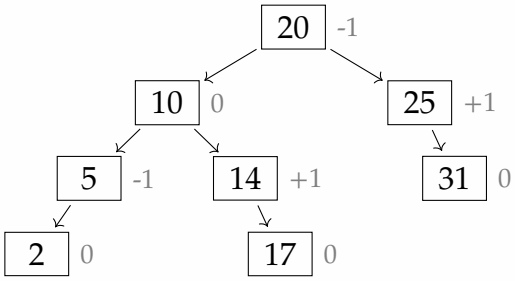
Nach der Linksrotation:



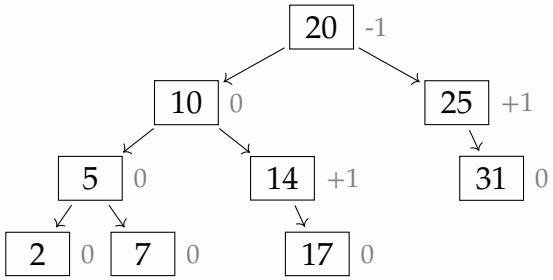
Nach dem Einfügen von „2“:



Nach dem Einfügen von „17“:

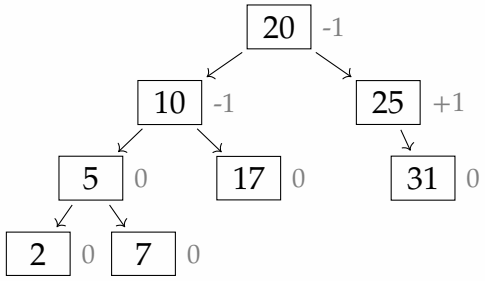


Nach dem Einfügen von „7“:

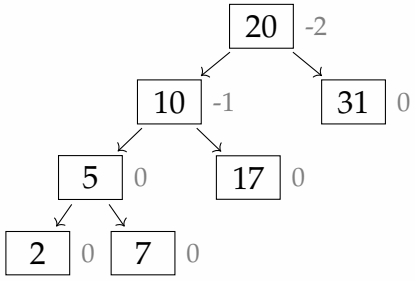


Löschen

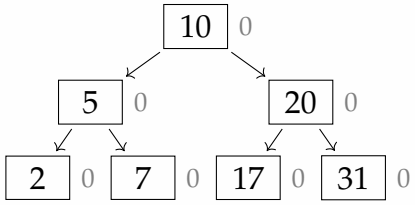
Nach dem Löschen von „14“:



Nach dem Löschen von „25“:



Nach der Rechtsrotation:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-2/Aufgabe-7.tex`

Examensaufgabe „Hashing mit mod 11 und 13“ (66115-2019-H.T2-A9)

Streutabellen (Hashing)

Verwenden Sie die Hashfunktion $h(k,i) = (h'(k) + i^2) \mod 11$ mit $h'(k) = k \mod 13$, um die Werte 12, 29 und 17 in die folgende Hashtabelle einzufügen. Geben Sie zudem jeweils an, auf welche Zellen der Hashtabelle zugegriffen wird.

0	1	2	3	4	5	6	7	8	9	10
			16		5				22	

Lösungsvorschlag

Einfügen des Wertes 12

$h'(12) = 12 \mod 13 = 12$

$h(12,0) = 12 + 0^2 \mod 11 = 1$

0	1	2	3	4	5	6	7	8	9	10
	12		16		5				22	

Einfügen des Wertes 29

$h'(29) = 29 \mod 13 = 3$

$h(29,0) = 3 + 0^2 \mod 11 = 3$ (belegt von 16)

$h(29,1) = 3 + 1^2 \mod 11 = 4$

0	1	2	3	4	5	6	7	8	9	10
	12		16	29	5				22	

Einfügen des Wertes 17

$h'(17) = 17 \mod 13 = 4$

$h(17,0) = 4 + 0^2 \mod 11 = 4$ (belegt von 29)

$h(17,1) = 4 + 1^2 \mod 11 = 5$ (belegt von 5)

$h(17,2) = 4 + 2^2 \mod 11 = 8$

0	1	2	3	4	5	6	7	8	9	10
	12		16	29	5			17	22	

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-2/Aufgabe-9.tex`

Examensaufgabe „Niedrigster gemeinsame Vorfahre“ (66115-2020-F.T2-A10) [Binärbaum]

Sei B ein binärer Suchbaum. In jedem Knoten v von B wird ein Schlüssel $v.\text{key} \in \mathbb{N}$ gespeichert sowie Zeiger $v.\text{left}$, $v.\text{right}$ und $v.\text{parent}$ auf sein linkes Kind, auf sein rechtes Kind und auf seinen Elternknoten. Die Zeiger sind *nil*, wenn der entsprechende Nachbar nicht existiert. Für zwei Knoten u und v ist wie üblich der *Abstand* die Anzahl der Kanten auf dem kürzesten Pfad von u nach v.

Für einen Knoten w von B sei $B(w)$ der Teilbaum von B mit Wurzel w. Für zwei Knoten u und v von B ist w ein *gemeinsamer* Vorfahre, wenn u und v in $B(w)$ liegen. Wir suchen den niedrigsten gemeinsamen Vorfahren $ngV(u,v)$ von u und v, also einen gemeinsamen Vorfahren w, so dass für jeden Vorfahren w von u und v gilt, dass w in $B(w)$ liegt. Wir betrachten verschiedene Szenarien, in denen Sie jeweils den niedrigsten gemeinsamen Vorfahren von u und v berechnen sollen.

> **Exkurs: Lowest Common Ancestor**
> Als Lowest Common Ancestor (LCA) oder „letzter gemeinsamer Vorfahre“ wird in der Informatik und Graphentheorie ein Ermittlungskonzept bezeichnet, das einen gegebenen gewurzelten Baum von Datenstrukturen effizient vorverarbeitet, sodass anschließend Anfragen nach dem letzten gemeinsamen Vorfahren für beliebige Knotenpaare in konstanter Zeit beantwortet werden können. [a]
>
> [a] https://de.wikipedia.org/wiki/Lowest_Common_Ancestor

(a) Wir bekommen u und v als Zeiger auf die entsprechenden Knoten in B geliefert. Beschreiben Sie in Worten und in Pseudocode einen Algorithmus, der den niedrigsten gemeinsamen Vorfahren von u und v berechnet. Analysieren Sie die Laufzeit Ihres Algorithmus.

```java
/**
 * NBV = Niedrigster gemeinsamer Vorfahre.
 *
 * https://afteracademy.com/blog/lowest-common-ancestor-of-a-binary-tree
 */
public class NGV {
  static class Knoten {
    int schlüssel;
    Knoten links;
    Knoten rechts;

    public Knoten(int schlüssel) {
      this.schlüssel = schlüssel;
    }
  }

  /**
   * ngV = niedrigster gemeinsamer Vorfahre
   *
   * @param wurzel  Der Wurzelkonten dies Binärbaums.
   * @param knoten1 Der erste Knoten, dessen niedrigster gemeinsamer Vorfahre
```

```
     *                gesucht werden soll.
     * @param knoten2 Der zweite Knoten, dessen niedrigster gemeinsamer Vorfahre
     *                gesucht werden soll.
     *
     * @return Der niedrigste gemeinsae Vorfahre der Knoten 1 und 2.
     */
    public static Knoten ngVRekursiv(Knoten wurzel, Knoten knoten1, Knoten
↪   knoten2) {
      if (wurzel == null)
        return null;
      if (wurzel.equals(knoten1) || wurzel.equals(knoten2))
        return wurzel;
      Knoten links = ngVRekursiv(wurzel.links, knoten1, knoten2);
      Knoten rechts = ngVRekursiv(wurzel.rechts, knoten1, knoten2);
      if (links == null)
        return rechts;
      else if (rechts == null)
        return links;
      else
        return wurzel;
    }

    /**
     * <pre>
     * {@code
     *     20
     *    / \
     *   8   22
     *  / \
     * 4   12
     *    / \
     *   10  14
     * }
     * </pre>
     *
     * Beispiele von
     * https://www.geeksforgeeks.org/lowest-common-ancestor-in-a-binary-search-
↪   tree/
     *
     * @param args Kommandozeilen-Argumente
     */
    public static void main(String[] args) {
      Knoten wurzel = new Knoten(20);

      Knoten knoten8 = new Knoten(8);
      Knoten knoten22 = new Knoten(22);
      Knoten knoten4 = new Knoten(4);
      Knoten knoten12 = new Knoten(12);
      Knoten knoten10 = new Knoten(10);
      Knoten knoten14 = new Knoten(14);

      wurzel.links = knoten8;
      wurzel.rechts = knoten22;
      wurzel.links.links = knoten4;
      wurzel.links.rechts = knoten12;
```

```
        wurzel.links.rechts.links = knoten10;
        wurzel.links.rechts.rechts = knoten14;

        System.out.println(ngVRekursiv(wurzel, knoten10, knoten14).schlüssel); // 12
        System.out.println(ngVRekursiv(wurzel, knoten14, knoten8).schlüssel); // 8
        System.out.println(ngVRekursiv(wurzel, knoten10, knoten22).schlüssel); // 20
    }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/NGV.java`

(b) Wir bekommen u und v wieder als Zeiger auf die entsprechenden Knoten in B geliefert. Seien d_u, und d_v, die Abstände von u bzw. v zum niedrigsten gemeinsamen Vorfahren von u und v. Die Laufzeit Ihres Algorithmus soll $O(\max\{d_u, d_v\})$ sein. Dabei kann Ihr Algorithmus in jedem Knoten v eine Information v.info speichern. Skizzieren Sie Ihren Algorithmus in Worten.

(c) Wir bekommen die Schlüssel u.key und v.key. Die Laufzeit Ihres Algorithmus soll proportional zum Abstand der Wurzel von B zum niedrigsten gemeinsamen Vorfahren von u und v sein. Skizzieren Sie Ihren Algorithmus in Worten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-2/Aufgabe-10.tex`

Examensaufgabe „Bäume“ (66115-2020-H.T1-TA2-A2)

Binärbaum

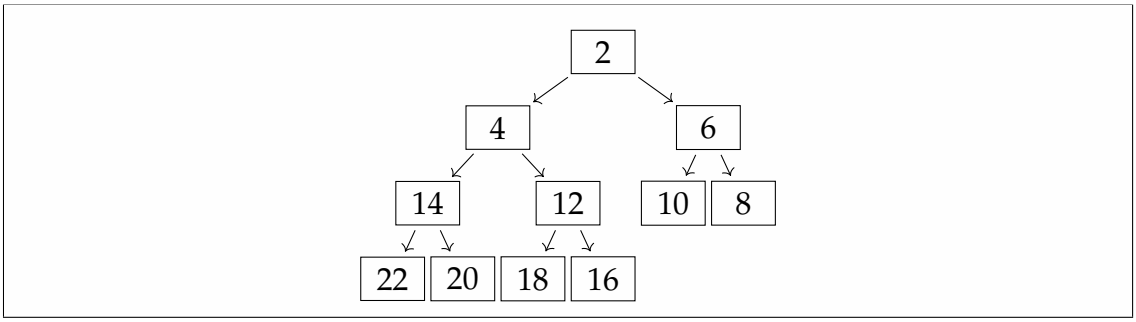
Wir betrachten ein Feld A von ganzen Zahlen mit `n` Elementen, die über die Indizes `A[0]` bis `A[n-1]` angesprochen werden können. In dieses Feld ist ein binärer Baum nach den folgenden Regeln eingebettet: Für das Feldelement mit Index i befindet sich

- der Elternknoten im Feldelement mit Index $\lfloor \frac{i-1}{2} \rfloor$,
- der linke Kindknoten im Feldelement mit Index $2 \cdot i + 1$, und
- der rechte Kindknoten im Feldelement mit Index $2 \cdot i + 2$.

(a) Zeichnen Sie den durch das folgende Feld repräsentierten binären Baum.

i	0	1	2	3	4	5	6	7	8	9	10
A[i]	2	4	6	14	12	10	8	22	20	18	16

Lösungsvorschlag



(b) Der folgende rekursive Algorithmus sei gegeben:

Pseudo-Code / Pascal

```
procedure magic(i, n : integer) : boolean
begin
  if (i > (n - 2) / 2) then
    return true;
  endif
  if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
      magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
    return true;
  endif
  return false;
end
```

Java-Implementation

```
  public static boolean magic(int i, int n) {
    System.out.println(String.format("i: %s n: %s", i, n));
    if (i > (n - 2) / 2) {
```

```
    return true;
  }
  if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) &&
↪ magic(2 * i + 2, n)) {
    return true;
  }
  return false;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java`

Gegeben sei folgendes Feld:

i	0	1	2	3
`A[i]`	2	4	6	14

Führen Sie `magic(0,3)` auf dem Feld aus. Welches Resultat liefert der Algorithmus zurück?

Lösungsvorschlag

true

(c) Wie nennt man die Eigenschaft, die der Algorithmus `magic` auf dem Feld A prüft? Wie lässt sich diese Eigenschaft formal beschreiben?

Lösungsvorschlag

Die sogenannte „Haldeneigenschaft“ bzw. „Heap-Eigenschaft“ einer Min-Halde. Der Schlüssel eines jeden Knotens ist kleiner (oder gleich) als die Schlüssel seiner Kinder.

Ein Baum erfüllt die Heap-Eigenschaft bezüglich einer Vergleichsrelation „$>$“ auf den Schlüsselwerten genau dann, wenn für jeden Knoten u des Baums gilt, dass $u_{\text{wert}} > v_{\text{wert}}$ für alle Knoten v aus den Unterbäumen von u.

(d) Welche Ausgaben sind durch den Algorithmus `magic` möglich, wenn das Eingabefeld aufsteigend sortiert ist? Begründen Sie Ihre Antwort.

Lösungsvorschlag

`true`. Eine sortierte aufsteigende Zahlenfolge entspricht den Haldeneigenschaften einer Min-Heap.

(e) Geben Sie zwei dreielementige Zahlenfolgen (bzw. Felder) an, eine für die `magic(0,2)` den Wert `true` liefert und eine, für die `magic(0,2)` den Wert `false` liefert.

Lösungsvorschlag

```
A = new int[] { 1, 2, 3 };
System.out.println(magic(0, 2)); // true

A = new int[] { 2, 1, 3 };
System.out.println(magic(0, 2)); // false
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java

(f) Betrachten Sie folgende Variante `almostmagic` der oben bereits erwähnten Prozedur `magic`, bei der die Anweisungen in Zeilen 3 bis 5 entfernt wurden:

Pseudo-Code / Pascal

```
procedure almostmagic(i, n : integer) : boolean
begin
  // leer
  // leer
  // leer
  if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
      magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
    return true;
  endif
  return false;
end
```

Java-Implementation

```
  public static boolean almostmagic(int i, int n) {
    System.out.println(String.format("i: %s n: %s", i, n));
    if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) &&
↪   magic(2 * i + 2, n)) {
      return true;
    }
    return false;
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java

Beschreiben Sie die Umstände, die auftreten können, wenn `almostmagic` auf einem Feld der Größe `n` aufgerufen wird. Welchen Zweck erfüllt die entfernte bedingte Anweisung?

Lösungsvorschlag

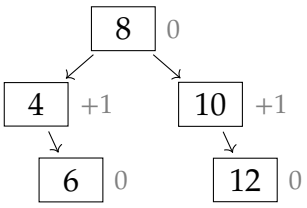
Wird die Prozedur zum Beispiel mit `almostmagic(0, n + 1)` aufgerufen, kommst es zu einem sogenannten „Array-Index-Ouf-of-Bounds“ Fehler, d. h. die Prozedur will auf Index des Feldes zugreifen, der im Feld gar nicht existiert. Die drei zusätzlichen Zeilen in der Methode `magic` bieten dafür einen Schutz, indem sie vor den Index-Zugriffen auf das Feld `true` zurückgeben.

```
    // A = new int[] { 1, 2, 3 };
    // System.out.println(almostmagic(0, 4)); // Exception in thread "main"
↪   java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for
↪   length 3
```

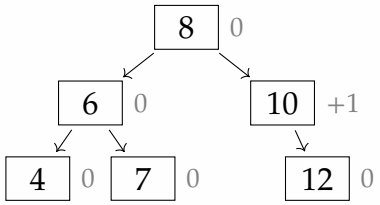
Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java

(g) Fügen Sie jeweils den angegebenen Wert in den jeweils angegebenen AVL-Baum mit aufsteigender Sortierung ein und zeichnen Sie den resultierenden Baum vor möglicherweise erforderlichen Rotationen. Führen Sie danach bei Bedarf die erforderliche(n) Rotation(en) aus und zeichnen Sie dann den resultierenden Baum. Sollten keine Rotationen erforderlich sein, so geben Sie dies durch einen Text wie „keine Rotationen nötig" an. AVL-Baum

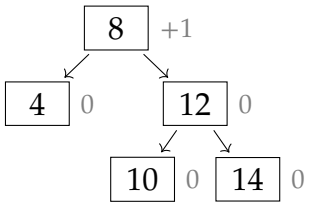
(i) Wert 7 einfügen



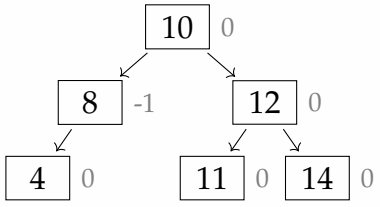
Lösungsvorschlag



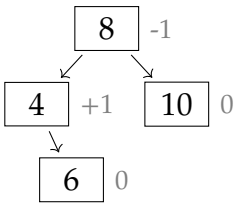
(ii) Wert 11 einfügen



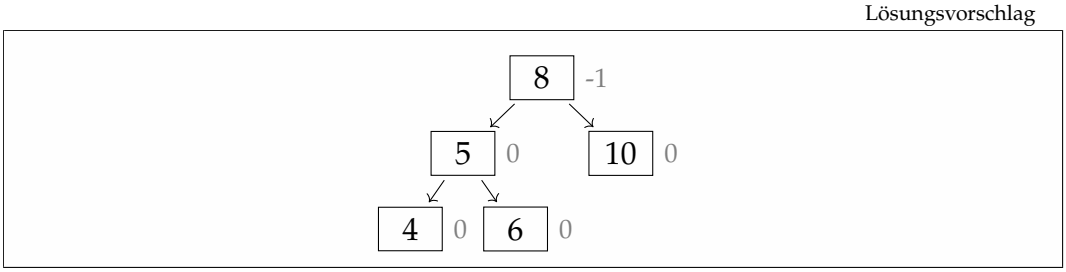
Lösungsvorschlag



(iii) Wert 5 einfügen



Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Streuspeicherung“ (66115-2020-H.T1-TA2-A5)

Gegeben seien die folgenden Schlüssel k zusammen mit ihren Streuwerten $h(k)$:

k	B	Y	E	!	A	U	D	?
$h(k)$	5	4	0	4	4	0	7	2

(a) Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein und lösen Sie Kollisionen durch verkettete Listen auf.

Stellen Sie die Streutabelle in folgender Art und Weise dar:

Lösungsvorschlag

Fach	Schlüssel k (verkettete Liste, zuletzt eingetragener Schlüssel rechts)
0	E, U,
1	
2	?
3	
4	Y, !, A
5	B,
6	
7	D

(b) Fügen Sie die gleichen Schlüssel in der gleichen Reihenfolge und mit der gleichen Streufunktion in eine neue Streutabelle der Größe 8 ein. Lösen Sie Kollisionen diesmal aber durch lineares Sondieren mit Schrittweite $+1$ auf.

Geben Sie für jeden Schlüssel jeweils an, welche Fächer Sie in welcher Reihenfolge sondiert haben und wo der Schlüssel schlussendlich gespeichert wird.

Lösungsvorschlag

Fach	Schlüssel k
0	E
1	U
2	D
3	?
4	Y
5	B
6	!
7	A

Schlüssel	Sondierung	Speicherung
B		5
Y		4
E		0
!	4, 5	6
A	4, 5, 6	7
U	0	1
D	7, 0, 1	2
?	2	3

(c) Bei der doppelten Streuadressierung verwendet man eine Funktionsschar h_i, die sich aus einer primären Streufunktion h_0 und einer Folge von sekundären Streufunktionen $h_1, h_2, \ldots$ zusammensetzt. Die folgenden Werte der Streufunktionen sind gegeben:

k	B	Y	E	!	A	U	D	?
$h_0(k)$	5	4	0	4	4	0	7	2
$h_1(k)$	6	3	3	3	1	2	6	0
$h_2(k)$	7	2	6	2	6	4	5	6
$h_3(k)$	0	1	1	1	3	6	4	4

Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein und geben Sie für jeden Schlüssel jeweils an, welche Streufunktion h_i zur letztendlichen Einsortierung verwendet wurde.

Lösungsvorschlag

Fach	Schlüssel k
0	E
1	A
2	U
3	!
4	Y
5	B
6	?
7	D

Schlüssel	Streufunktion
B	$h_0(k)$
Y	$h_0(k)$
E	$h_0(k)$
!	$h_1(k)$
A	$h_1(k)$
U	$h_1(k)$
D	$h_0(k)$
?	$h_2(k)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-5.tex`

Examensaufgabe „Vornamen“ (66115-2020-H.T2-TA2-A1)

Eine Hashfunktion h wird verwendet, um Vornamen auf die Buchstaben $\{A, \ldots, Z\}$ abzubilden. Dabei bildet h auf den ersten Buchstaben des Vornamens als Hashwert ab.

Sei H die folgende Hashtabelle (Ausgangszustand):

Schlüssel	Inhalt
A	
B	
C	
D	Dirk
E	
F	
G	
H	
I	Inge
J	
K	Kurt
L	
M	

Schlüssel	Inhalt
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	

(a) Fügen Sie der Hashtabelle H die Vornamen

Martin, Michael, Norbert, Matthias, Alfons, Bert, Christel, Adalbert, Edith, Emil

in dieser Reihenfolge hinzu, wobei Sie Kollisionen durch lineares Sondieren (mit Inkrementieren zum nächsten Buchstaben) behandeln.

Lösungsvorschlag

Schlüssel	Inhalt
A	Alfons
B	Bert
C	Christel
D	Dirk
E	Adalbert
F	Edith
G	Emil
H	
I	Inge
J	
K	Kurt
L	
M	Martin

Schlüssel	Inhalt
N	Michael
O	Norbert
P	Matthias
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	

(b) Fügen Sie der Hashtabelle H die Vornamen

Brigitte, Elmar, Thomas, Katrin, Diana, Nathan, Emanuel, Sebastian, Torsten, Karolin

in dieser Reihenfolge hinzu, wobei Sie Kollisionen durch Verkettung der Überläufer behandeln. (Hinweis: Verwenden Sie die Hashtabelle im Ausgangszustand.)

Lösungsvorschlag

Schlüssel	Inhalt
A	
B	Brigitte
C	
D	Dirk, Diana
E	Elmar, Emanuel
F	
G	
H	
I	Inge
J	
K	Kurt, Katrin, Karolin
L	
M	

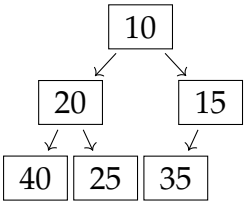
Schlüssel	Inhalt
N	Nathan
O	
P	
Q	
R	
S	Sebastian
T	Thomas, Torsten
U	
V	
W	
X	
Y	
Z	

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „DeleteMin“ (66115-2020-H.T2-TA2-A3)

Halde (Heap)

Es sei der folgende Min-Heap gegeben:



(a) Geben Sie obigen Min-Heap in der Darstellung eines Feldes an, wobei die Knoten in Level-Order abgelegt sind.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**
10	20	15	40	25	35

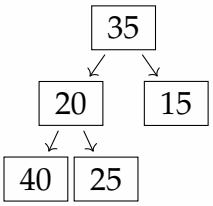
(b) Führen Sie wiederholt DeleteMin-Operationen auf dem gegebenen Heap aus, bis der Heap leer ist. Zeichnen Sie dafür den aktuellen Zustand des Heaps als Baum und als Feld nach jeder Änderung des Heaps, wobei Sie nur gültige Bäume zeichnen (d. h. solche die keine Lücken haben). Dokumentieren Sie, was in den einzelnen Schritten geschieht.

Lösungsvorschlag

Löschen von 10

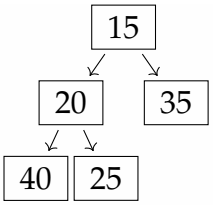
Nach dem Ersetzen von „10“ durch „35“:

0	**1**	**2**	**3**	**4**
35	20	15	40	25



Nach dem Vertauschen von „35“ und „15“:

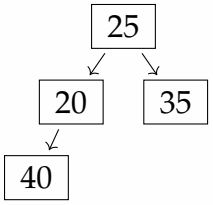
0	**1**	**2**	**3**	**4**
15	20	35	40	25



Löschen von 15

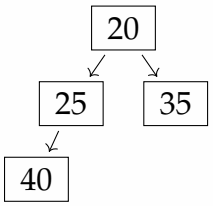
Nach dem Ersetzen von „15“ mit „25“:

0	**1**	**2**	**3**
25	20	35	40



Nach dem Vertauschen von „25“ und „20“:

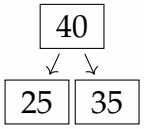
0	**1**	**2**	**3**
20	25	35	40



Löschen von 20

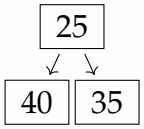
Nach dem Vertauschen von „20“ mit „40“:

0	**1**	**2**
40	25	35



Nach dem Vertauschen von „40“ und „25“:

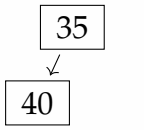
0	**1**	**2**
25	40	35



Löschen von 25

Nach dem Ersetzen von „25“ durch „35“:

0	**1**
35	40



Löschen von 35

Nach dem Ersetzen von „35“ mit „40“:

0
40

Löschen von 40

Nach dem Löschen von „40“:

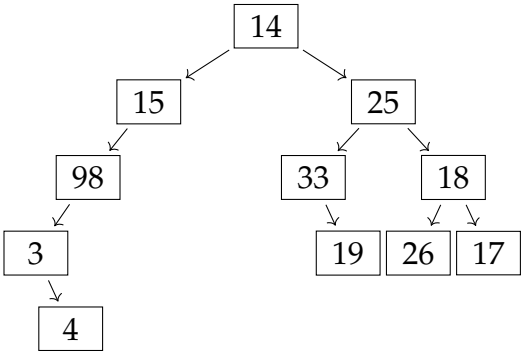
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Binärbäume“ (66115-2021-F.T2-TA2-A3)

Binärbaum

(a) Betrachten Sie folgenden Binärbaum T.

Geben Sie die Schlüssel der Knoten in der Reihenfolge an, wie sie von einem Preorder-Durchlauf (= TreeWalk) von T ausgegeben werden.



Exkurs: Preorder-Traversierung eines Baum

besuche die Wurzel, dann den linken Unterbaum, dann den rechten Unterbaum; auch: WLR

```
  private void besuchePreorder(BaumKnoten knoten, ArrayList<Comparable>
↪   schlüssel) {
    if (knoten != null) {
      schlüssel.add((Comparable) knoten.gibSchlüssel());
      besuchePreorder(knoten.gibLinks(), schlüssel);
      besuchePreorder(knoten.gibRechts(), schlüssel);
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/baum/BinaerBaum.java`

Lösungsvorschlag

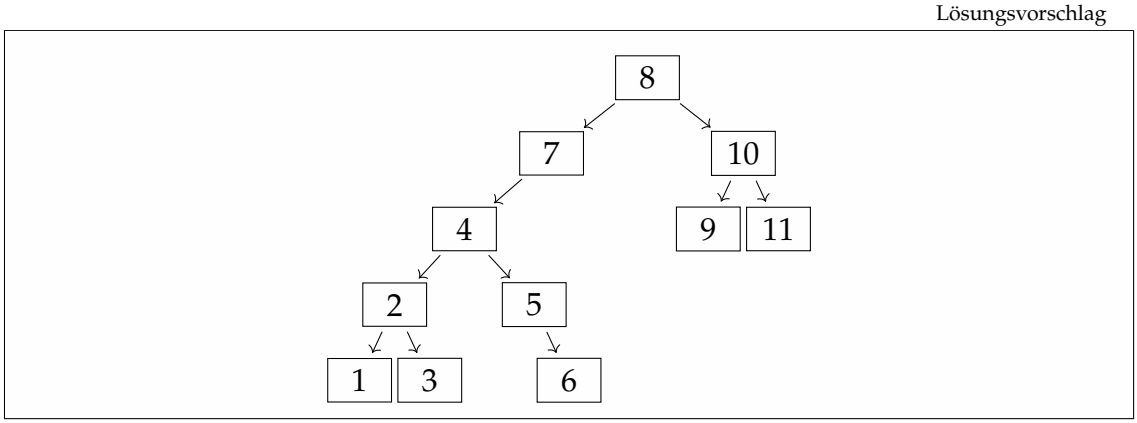
14, 15, 98, 3, 4, 25, 33, 19, 18, 26, 17

(b) Betrachten Sie folgende Sequenz als Ergebnis eines Preorder-Durchlaufs eines binären Suchbaumes T. Zeichnen Sie T und erklären Sie, wie Sie zu Ihrer Schlussfolgerung gelangen.

$$[8,7,4,2,1,3,5,6,10,9,11]$$

Hinweis: Welcher Schlüssel ist die Wurzel von T? Welche Knoten sind in seinem linken/rechten Teilbaum gespeichert? Welche Schlüssel sind die Wurzeln der jeweiligen Teilbäume?

Lösungsvorschlag



(c) Anstelle von sortierten Zahlen soll ein Baum nun verwendet werden, um relative Positionsangaben zu speichern. Jeder Baumknoten enthält eine Beschriftung und einen Wert (vgl. Abb. 1), der die ganzzahlige relative Verschiebung in horizontaler Richtung gegenüber seinem Elternknoten angibt. Die zu berechnenden Koordinaten für einen Knoten ergeben sich aus seiner Tiefe im Baum als y-Wert und aus der Summe aller Verschiebungen auf dem Pfad zur Wurzel als x-Wert. Das Ergebnis der Berechnung ist in Abb. 2 visualisiert. Geben Sie einen Algorithmus mit linearer Laufzeit in Pseudo-Code oder einer objektorientierten Programmiersprache Ihrer Wahl an. Der Algorithmus erhält den Zeiger auf die Wurzel eines Baumes als Eingabe und soll Tupel mit den berechneten Koordination aller Knoten des Baums in der Form (Beschriftung, x, y) zurück- oder ausgeben.

Lösungsvorschlag

```
public class Knoten {
  public Knoten links;
  public Knoten rechts;
  public String name;

  /**
   * Bewegung bezüglich des Vorknotens. Relative Lage.
   */
  public int xVerschiebung;

  public Knoten(String name, int xVerschiebung) {
    this.name = name;
    this.xVerschiebung = xVerschiebung;
  }

  public void durchlaufen() {
    durchlaufe(this, 0 + xVerschiebung, 0);
  }

  private void durchlaufe(Knoten knoten, int x, int y) {
    System.out.println("Beschriftung: " + knoten.name + " x: " + x + " y: " +
↪   y);

    if (links != null) {
      links.durchlaufe(links, x + links.xVerschiebung, y + 1);
    }
```

```
    if (rechts != null) {
      rechts.durchlaufe(rechts, x + rechts.xVerschiebung, y + 1);
    }
  }

  public static void main(String[] args) {
    Knoten a = new Knoten("a", 1);
    Knoten b = new Knoten("b", 1);
    Knoten c = new Knoten("c", -2);
    Knoten d = new Knoten("d", 2);
    Knoten e = new Knoten("e", 0);

    a.links = b;
    a.rechts = c;
    c.links = d;
    c.rechts = e;

    a.durchlaufen();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/Knoten.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Hashing“ (66115-2021-F.T2-TA2-A5)

(a) Nennen Sie zwei wünschenswerte Eigenschaften von Hashfunktionen.

Lösungsvorschlag

surjektiv Die Abbildung soll surjektiv sein, ðjeder Index soll berechnet werden können.

gleichverteilt Durch die Hashfunktion soll möglichst eine Gleichverteilung auf die Buckets (Indexliste) erfolgen.

effizient Zudem sollte die Verteilung mittels Hashfunktion möglichst effizient gewählt werden.

(b) Wie viele Elemente können bei Verkettung und wie viele Elemente können bei offener Adressierung in einer Hashtabelle mit m Zeilen gespeichert werden?

Lösungsvorschlag

Verkettung Es darf mehr als ein Element pro Bucket enthalten sein, deswegen können beliebig viele Element gespeichert werden.

offene Addressierung (normalerweise) ein Element pro Bucket, deshalb ist die Anzahl der speicherbaren Elemente höchstens m. Können in einem Bucket k Elemente gespeichert werden, dann beträgt die Anzahl der speicherbaren Elemente $k \cdot m$.

(c) Angenommen, in einer Hashtabelle der Größe m sind alle Einträge (mit mindestens einem Wert) belegt und insgesamt n Werte abgespeichert.

Geben Sie in Abhängigkeit von m und n an, wie viele Elemente bei der Suche nach einem nicht enthaltenen Wert besucht werden müssen. Sie dürfen annehmen, dass jeder Wert mit gleicher Wahrscheinlichkeit und unabhängig von anderen Werten auf jeden der m Plätze abgebildet wird (einfaches gleichmäßiges Hashing).

Lösungsvorschlag

$\frac{n}{m}$ Beispiel: 10 Buckets, 30 Elemente: $\frac{30}{10} = 3$ Elemente im Bucket, die man durchsuchen muss.

(d) Betrachten Sie die folgende Hashtabelle mit der Hashfunktion $h(x) = x \mod 11$. Hierbei steht $\varnothing$ für eine Zelle, in der kein Wert hinterlegt ist.

Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	$\varnothing$	$\varnothing$	3	$\varnothing$	16	28	18	$\varnothing$	$\varnothing$	32

Führen Sie nun die folgenden Operationen mit offener Adressierung mit linearem Sondieren aus und geben Sie den Zustand der Datenstruktur nach jedem Schritt an. Werden für eine Operation mehrere Zellen betrachtet, aber nicht modifiziert, so geben Sie deren Indizes in der betrachteten Reihenfolge an.

(i) Insert 7

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	∅	∅	3	∅	16	28	18	7_2	∅	32

(ii) Insert 20

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	∅	∅	3	∅	16	28	18	7_2	20_1	32

(iii) Delete 18

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	∅	∅	3	∅	16	28	del	7_2	20_1	32

del ist eine Marke, die anzeigt, dass gelöscht wurde und der Bucket nicht leer ist.

(iv) Search 7

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	∅	∅	3	∅	16	28	del	7_2	20_1	32

$h(7) = 7 \mod 11 = 7$

7 (Index) $\rightarrow$ del lineares sondieren $\rightarrow$ 8 (Index) $\rightarrow$ gefunden

(v) Insert 5

Lösungsvorschlag

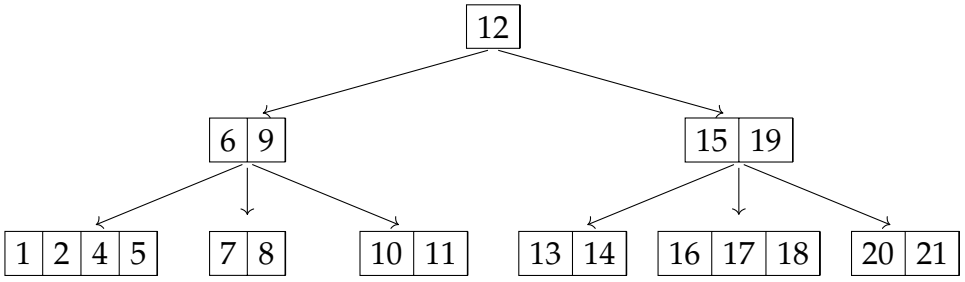
Index	0	1	2	3	4	5	6	7	8	9	10
Wert	11	∅	∅	3	∅	16	28	5_3	7_2	20_1	32

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-5.tex`

Examensaufgabe „B-Baum k=2“ (66116-2013-F.T2-TA1-A3)

B-Baum

Gegeben sei der folgende B-Baum:



(a) Was bedeutet k bei einem B-Baum mit Grad k? Geben Sie k für den obigen B-Baum an.

Lösungsvorschlag

Jeder Knoten außer der Wurzel hat mindestens k und höchstens $2k$ Einträge. Die Wurzel hat zwischen einem und $2k$ Einträgen. Die Einträge werden in allen Knoten sortiert gehalten. Alle Knoten mit n Einträgen, außer den Blättern, haben $n + 1$ Kinder.

Für den gegebenen Baum kann die Ordnung $k = 2$ angegeben werden.

(b) Was sind die Vorteile von B-Bäumen im Vergleich zu binären Baumen?

Lösungsvorschlag

B-Bäume sind immer höhenbalanciert. B-Bäume haben eine geringere Höhe, wodurch eine schnellere Suche möglich wird, da weniger Aufrufe nötig sind.[a]

[a] `http://wwwbayer.in.tum.de/lehre/WS2001/HSEM-bayer/BTreesAusarbeitung.pdf`

(c) Wozu werden B-Bäume in der Regel verwendet und wieso?

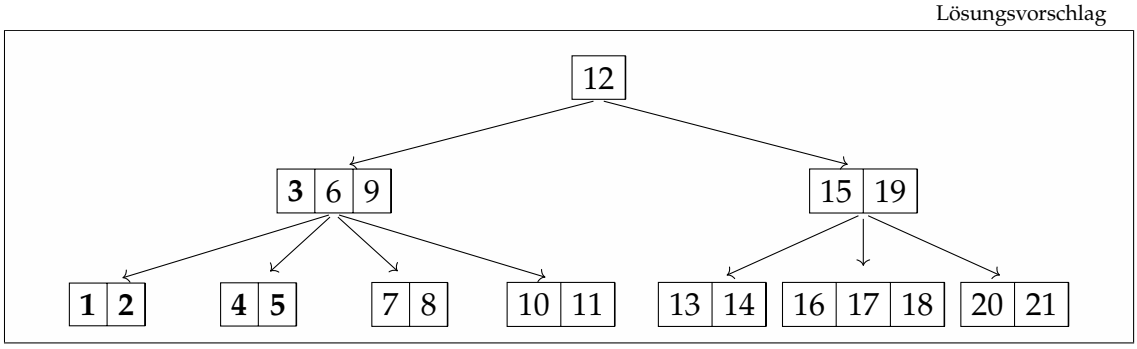
Lösungsvorschlag

B-Bäume werden für Hintergrundspeicherung (z. B. von Datenbanksystemen, Dateisystem) verwendet. Die Knotengrößen werden auf die Seitenkapazitäten abgestimmt.

B-Bäume sind eine daten- und Indexstruktur, die häufig in Datenbanken und Daeisystemen eingesetzt werden. Da ein B-Baum immer vollständig balanciert ist und die Schlüssel sortiert gespeichert werden, ist ein schnelles Auffinden von Inhalten gegeben.

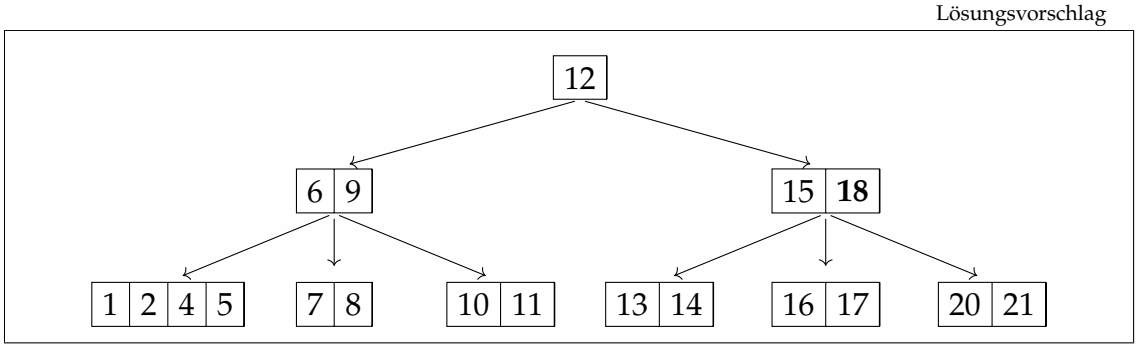
(d) Fügen Sie den Wert 3 in den B-Baum ein, und zeichnen Sie den vollständigen B-Baum nach dem Einfügen und möglichen darauf folgenden Operationen.

Lösungsvorschlag



(e) Entfernen Sie aus dem ursprünglichen B-Baum den Wert 19. Zeichnen Sie das vollständige Ergebnis nach dem Löschen und möglichen darauf folgenden Operationen. Sollte es mehrere richtige Lösungen geben, reicht es eine Lösung zu zeichnen.

Lösungsvorschlag

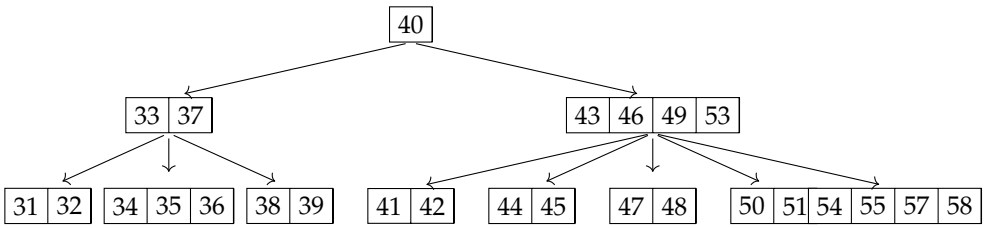


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2013/03/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Indexstrukturen“ (66116-2015-F.T2-TA1-A3)

B-Baum

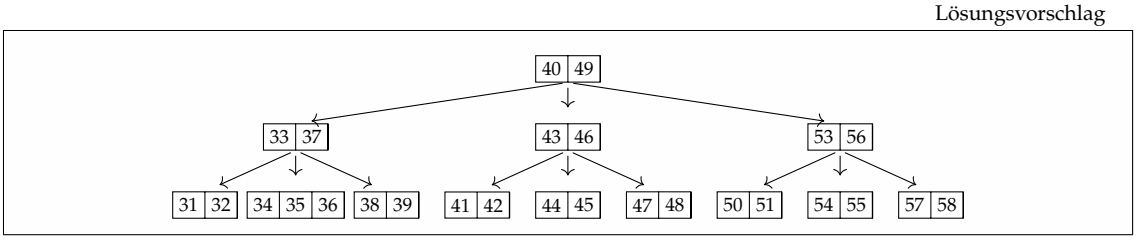
Als Indexstruktur einer Datenbank sei folgender B-Baum ($k = 2$) gegeben:



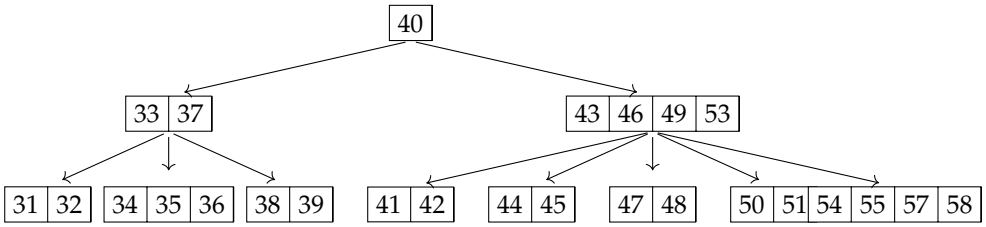
Führen Sie nacheinander die folgenden Operationen aus. Geben Sie die auftretenden Zwischenergebnisse an. Teilbäume, die sich in einem Schritt nicht verändern, müssen nicht erneut gezeichnet werden. Sollten Wahlmöglichkeiten auftreten, so sind größere Schlüsselwerte bzw. weiter rechts liegende Knoten zu bevorzugen.

(a) Einfügen des Wertes 56

Lösungsvorschlag



(b) Löschen des Wertes 37

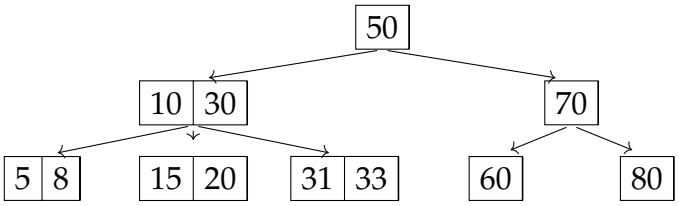


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „B-Baum der Ordnung 3“ (66116-2015-H.T2-TA1-A3)

B-Baum

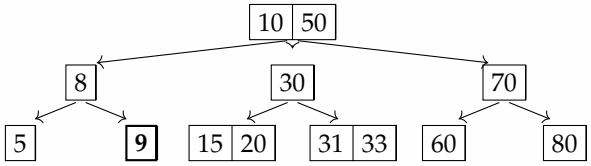
Gegeben ist der folgende B-Baum der Ordnung 3 (max. drei Kindknoten, max. zwei Schlüssel pro Knoten):



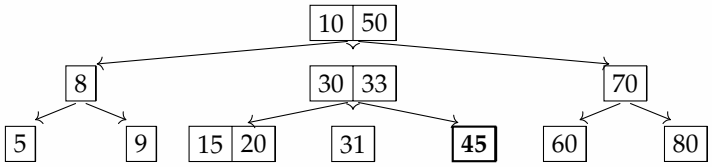
Fügen Sie die Werte 9 und 45 ein. Löschen Sie anschließend die Werte 30 und 70. Zeichnen Sie den Baum nach jeder Einfüge- bzw. Lösch-Operation.

Lösungsvorschlag

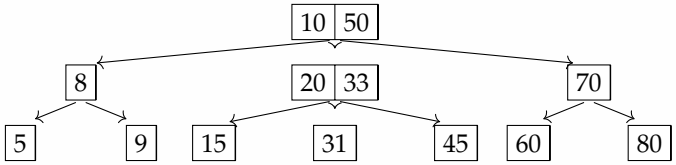
9 einfügen



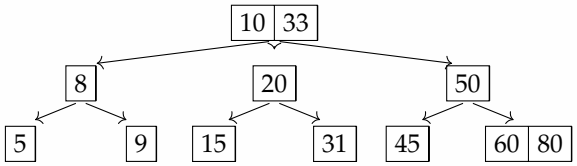
45 einfügen



30 löschen



70 löschen



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/09/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Aufbau eines B-Baums“ (66116-2017-F.T1-TA1-A2)

B-Baum

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüsssel einfügen:

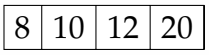
8, 10, 12, 20, 5, 30, 25, 11

Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

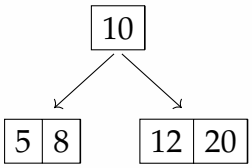
Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

Lösungsvorschlag

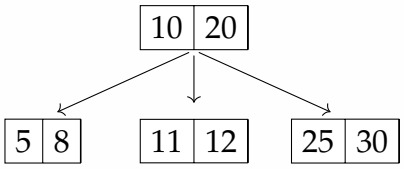
- Schlüsselwert 8 (einfaches Einfügen)
- Schlüsselwert 10 (einfaches Einfügen)
- Schlüsselwert 12 (einfaches Einfügen)
- Schlüsselwert 20 (einfaches Einfügen)



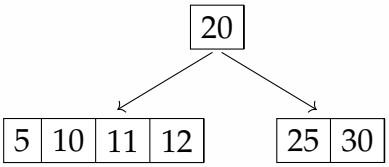
- Schlüsselwert 5 (Split)



- Schlüsselwert 30 (einfaches Einfügen)
- Schlüsselwert 25 (einfaches Einfügen)
- Schlüsselwert 11 (Split)



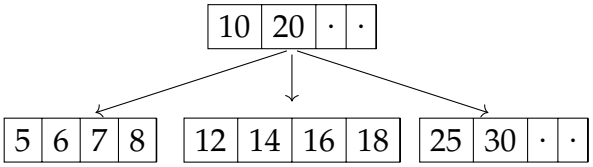
- Löschen des Schlüsselwerts 8 (Mischen/Verschmelzen)



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

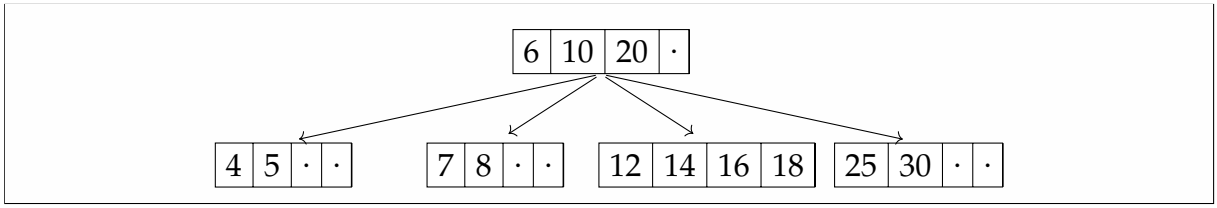
Examensaufgabe „Physische Datenstrukturen)“ (66116-2020-F.T1-TA2-A6) [B-Baum]

Fügen Sie die Zahl 4 in den folgenden B-Baum ein.



Zeichnen Sie den vollständigen, resultierenden Baum.

Lösungsvorschlag



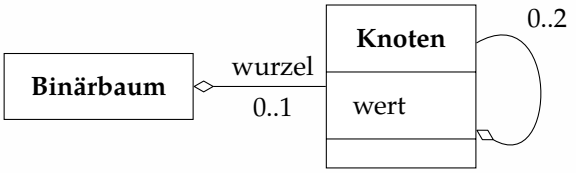
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-2/Aufgabe-6.tex`

Übungsaufgabe „Binärbaum: Klassendiagramm und Implementierung" (Binärbaum, Klassendiagramm, Implementierung in Java)

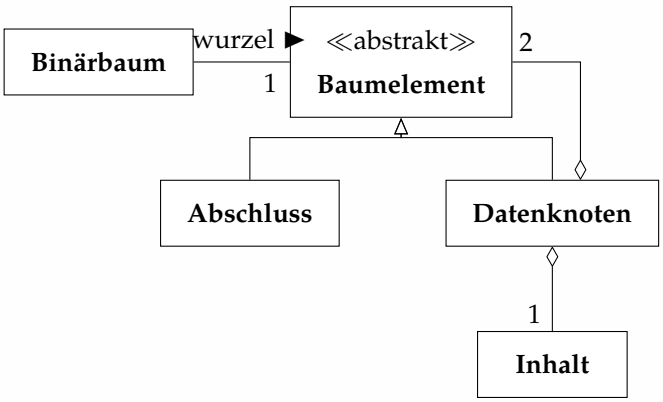
(a) Erstellen Sie ein Klassendiagramm für einen Binärbaum.

Lösungsvorschlag

einfacher Binärbaum



Binärbaum mit Kompositum



(b) Entwerfen Sie eine mögliche Implementierung zur Erzeugung eines binären Baumes in Java.

Lösungsvorschlag

einfacher Binärbaum

```
public class Binaerbaum {
  public Knoten wurzel;

  public void fügeEin(int wert) {
    if (wurzel == null) {
      wurzel = new Knoten(wert);
    } else {
      if (wert <= wurzel.gibWert()) {
        if (wurzel.gibLinks() != null) {
          wurzel.gibLinks().fügeEin(wert);
        } else {
          wurzel.setzeLinks(new Knoten(wert));
        }
      } else {
        if (wurzel.gibRechts() != null) {
```

```
            wurzel.gibRechts().fügeEin(wert);
          } else {
            wurzel.setzeRechts(new Knoten(wert));
          }
        }
      }
    }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/einfach/Binaerbaum.java`

```
public class Knoten {

  private Knoten links;
  private Knoten rechts;
  private int wert;

  public Knoten(int wert) {
    this.wert = wert;
  }

  public void setzeWert(int w) {
    wert = w;
  }

  public int gibWert() {
    return wert;
  }

  public void setzeLinks(Knoten l) {
    links = l;
  }

  public void setzeRechts(Knoten r) {
    rechts = r;
  }

  public Knoten gibLinks() {
    return links;
  }

  public Knoten gibRechts() {
    return rechts;
  }

  public void fügeEin(int wert) {
    if (wert <= this.gibWert()) {
      if (this.gibLinks() != null) {
        this.gibLinks().fügeEin(wert);
      } else {
        this.setzeLinks(new Knoten(wert));
      }
    } else {
```

```
      if (this.gibRechts() != null) {
        this.gibRechts().fügeEin(wert);
      } else {
        this.setzeRechts(new Knoten(wert));
      }
    }
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/einfach/Knoten.java`

Binärbaum mit Kompositum

```
class Binaerbaum {
  private Baumelement wurzel;

  public Binaerbaum() {
    wurzel = new Abschluss();
  }

  public void setzeWurzel(Baumelement wurzel) {
    this.wurzel = wurzel;
  }

  public Baumelement gibWurzel() {
    return wurzel;
  }

  public int gibAnzahl() {
    return wurzel.gibAnzahl();
  }

  public void gibAus() {
    wurzel.gibAus();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Binaerbaum.java`

```
abstract class Baumelement {
  public abstract void setzteLinks(Baumelement nl);

  public abstract void setzeRechts(Baumelement nr);

  public abstract Baumelement gibLinks();

  public abstract Baumelement gibRechts();

  public abstract Datenelement gibInhalt();

  public abstract int gibAnzahl();
```

```
  public abstract void gibAus();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Baumelement.java`

```
class Abschluss extends Baumelement {

  public void setzteLinks(Baumelement links) {
    System.out.println("Ein Abschluss hat kein linkes Element!");
  }

  public void setzeRechts(Baumelement rechts) {
    System.out.println("Ein Abschluss hat kein rechts Element!");
  }

  public Baumelement gibLinks() {
    System.out.println("Linkes Element nicht bekannt!");
    return this;
  }

  public Baumelement gibRechts() {
    System.out.println("Linkes Element nicht bekannt!");
    return this;
  }

  public Datenelement gibInhalt() {
    return null;
  }

  public int gibAnzahl() {
    return 0;
  }

  public void gibAus() {
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Abschluss.java`

```
class Datenknoten extends Baumelement {
  private Baumelement links, rechts;
  private Datenelement inhalt;

  public Datenknoten(Baumelement links, Baumelement rechts, Datenelement
↪   inhalt) {
    this.links = links;
    this.rechts = rechts;
    this.inhalt = inhalt;
  }

  public void setzteLinks(Baumelement links) {
    this.links = links;
```

```
  }

  public void setzeRechts(Baumelement rechts) {
    this.rechts = rechts;
  }

  public void inhaltSetzen(Datenelement inhalt) {
    this.inhalt = inhalt;
  }

  public Baumelement gibLinks() {
    return links;
  }

  public Baumelement gibRechts() {
    return rechts;
  }

  public Datenelement gibInhalt() {
    return inhalt;
  }

  public int gibAnzahl() {
    return 1 + links.gibAnzahl() + rechts.gibAnzahl();
  }

  public void gibAus() {
    System.out.print(" [");
    links.gibAus();
    inhalt.gibAus();
    rechts.gibAus();
    System.out.print("] ");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Datenknoten.java`

```
abstract class Datenelement {
  public abstract void gibAus();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Datenelement.java`

```
class Inhalt extends Datenelement {
  private String inhalt;

  public Inhalt(String inhalt) {
    this.inhalt = inhalt;
  }

  public String gibInhalt() {
    return inhalt;
  }
```

```
  public void gibAus() {
    System.out.print(inhalt);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Inhalt.java`

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BinaerbaumTest {

  @Test
  public void teste(){
    Binaerbaum baum = new Binaerbaum();
    Inhalt[] inhalte = new Inhalt[16];
    Datenknoten[] datenknoten = new Datenknoten[16];
    inhalte[0] = new Inhalt("Inhalt 1");

    inhalte[1] = new Inhalt("Inhalt 2");
    inhalte[2] = new Inhalt("Inhalt 3");

    inhalte[3] = new Inhalt("Inhalt 4");
    inhalte[4] = new Inhalt("Inhalt 5");

    for (int i = 0; i < 5; i++) {
      datenknoten[i] = new Datenknoten(new Abschluss(), new Abschluss(),
↪   inhalte[i]);
    }
    baum.setzeWurzel(datenknoten[0]);

    datenknoten[0].setzteLinks(datenknoten[1]);
    datenknoten[0].setzeRechts(datenknoten[2]);

    datenknoten[1].setzteLinks(datenknoten[3]);
    datenknoten[1].setzeRechts(datenknoten[4]);

    assertEquals(5, baum.gibAnzahl());

    Inhalt inhalt = (Inhalt)
↪   baum.gibWurzel().gibLinks().gibLinks().gibInhalt();
    assertEquals("Inhalt 4", inhalt.gibInhalt());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/baum/kompositum/BinaerbaumTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/20_Binaerer-Suchbaum/Aufgabe_Klassendiagramm-Implementierung.tex`

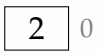
Übungsaufgabe „AVL-Baum 2, 8, 10, 1, 4, 5, 11“ (AVL-Baum)

AVL-Baum

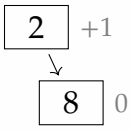
Fügen Sie die Zahlen 2, 8, 10, 1, 4, 5, 11 in der vorgegebenen Reihenfolge in einen AVL-Baum ein. Wie sieht der finale AVL-Baum aus?

Lösungsvorschlag

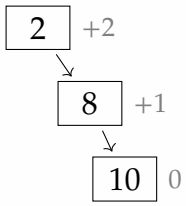
Nach dem Einfügen von „2“:



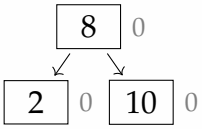
Nach dem Einfügen von „8“:



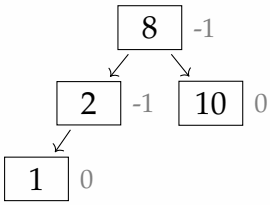
Nach dem Einfügen von „10“:



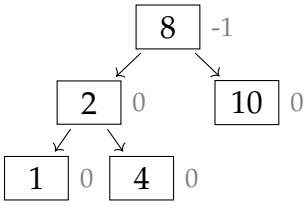
Nach der Linksrotation:



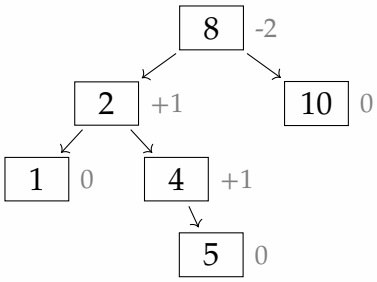
Nach dem Einfügen von „1“:



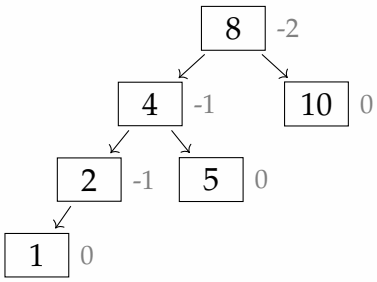
Nach dem Einfügen von „4“:



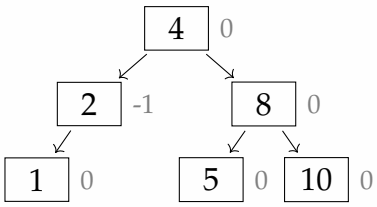
Nach dem Einfügen von „5“:



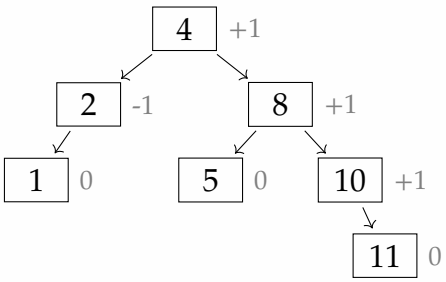
Nach der Linksrotation:



Nach der Rechtsrotation:



Nach dem Einfügen von „11“:

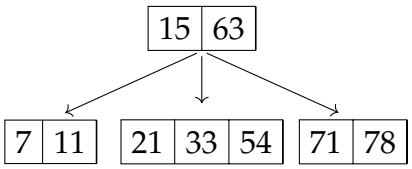


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/30_AVL-Baum/Aufgabe_AVL-Baum-2-8-10-1-4-5-11.tex`

Übungsaufgabe „Einfügen und Löschen in B-Bäumen“ (B-Baum)

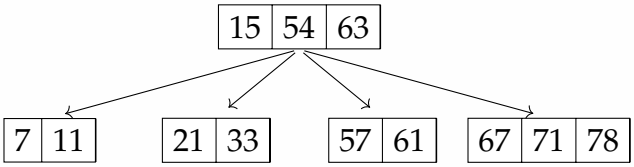
B-Baum

(a) Gegeben ist der unten vereinfacht dargestellte B-Baum der Klasse der Ordnung 2. Fügen Sie die Schlüsselwerte 67, 57, 61, 75, 5, 13, 2, 91, 9, 17, 10 und 8 ein. Geben Sie in jedem Einfügeschritt die verwendete Maßnahme (einfaches Einfügen in einen Knoten, Splitten) an und zeichnen Sie den Baum nach jedem Knotensplit neu.

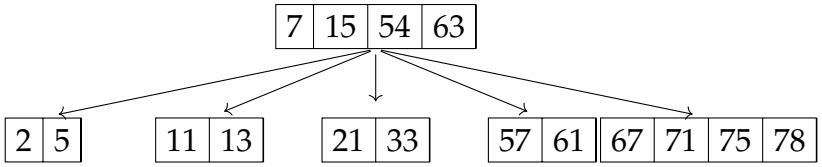


Lösungsvorschlag

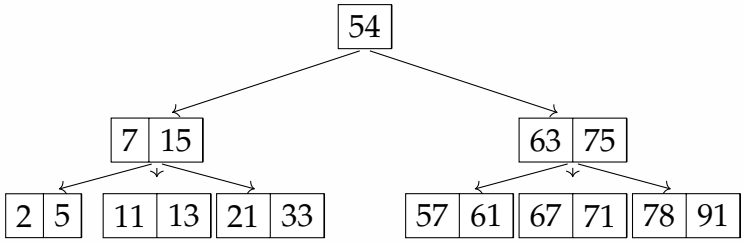
- Schlüsselwert 67 (einfaches Einfügen)
- Schlüsselwert 57 (einfaches Einfügen)
- Schlüsselwert 61 (Splitten)



- Schlüsselwert 75 (einfaches Einfügen)
- Schlüsselwert 5 (einfaches Einfügen)
- Schlüsselwert 13 (einfaches Einfügen)
- Schlüsselwert 2 (Splitten)

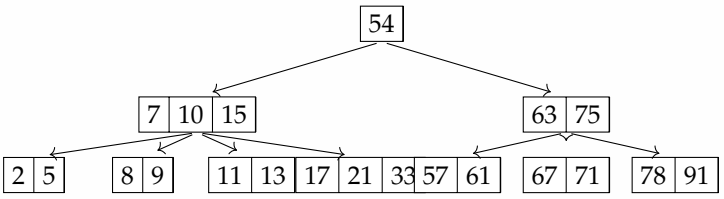


- Schlüsselwert 91 (Splitten)

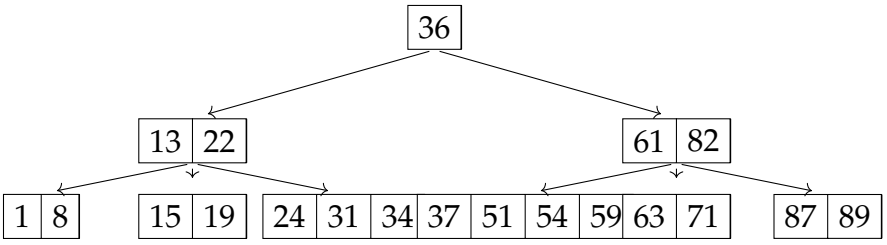


- Schlüsselwert 9 (einfaches Einfügen)
- Schlüsselwert 17 (einfaches Einfügen)
- Schlüsselwert 10 (einfaches Einfügen)

- Schlüsselwert 8 (Splitt)

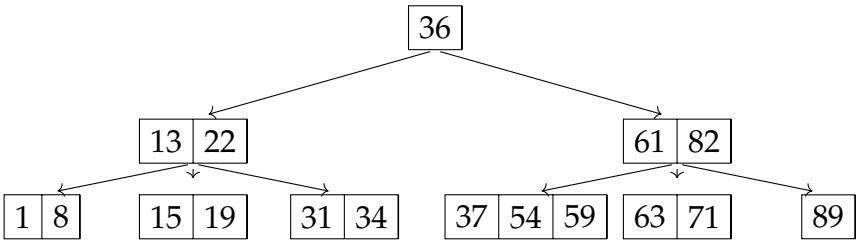


(b) Gegeben ist der unten dargestellte B-Baum der Ordnung 2. Löschen Sie die Schlüsselwerte 24, 51, 87, 34, 71, 19, 31 und 8. Geben Sie in jedem Löschschritt die verwendete Maßnahme (einfaches Löschen, Mischen, Ausgleichen) an und zeichnen Sie den Baum nach jeder Veränderung der Knotenstruktur (Mischen, Ausgleichen) neu. Für Ausgleichsoperationen sollen nur unmittelbare Nachbarknoten herangezogen werden.



Lösungsvorschlag

- Schlüsselwert 24 (einfaches Löschen)
- Schlüsselwert 51 (einfaches Löschen)

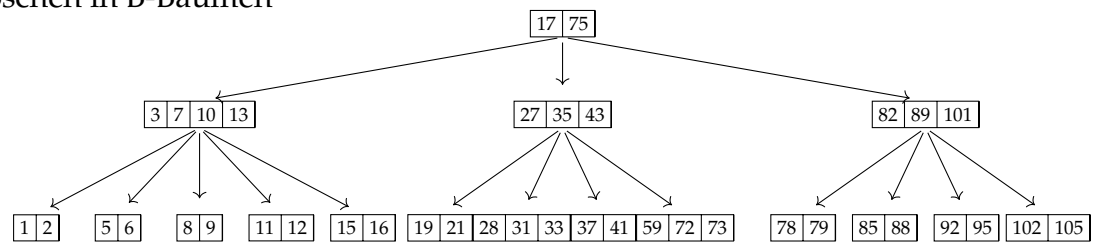


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/50_B-Baum/Aufgabe_Uni-Hamburg.tex`

Übungsaufgabe „Löschen in B-Bäumen“ (B-Baum)

B-Baum

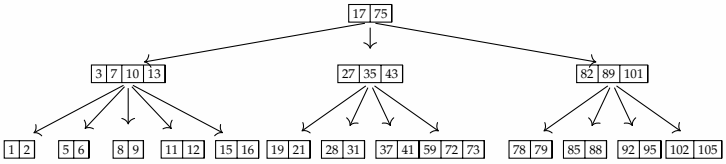
Löschen in B-Bäumen [2]



(a) Löschen 33

Lösungsvorschlag

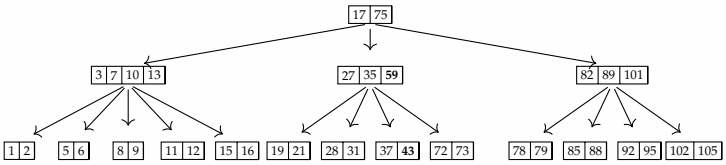
Löschen der 33 führt zu keinem Unterlauf.



(b) Löschen 41

Lösungsvorschlag

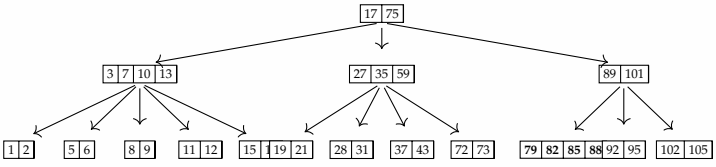
Ausgleichen Rotieren nach links



(c) Löschen 78

Lösungsvorschlag

Mischen der Zahlen 79 82 85 88



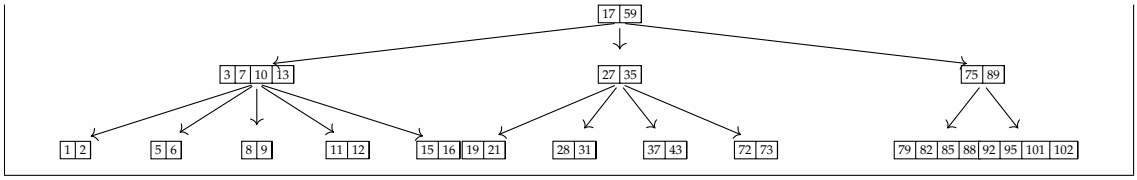
(d) Löschen 105

Lösungsvorschlag

Mischen der Zahlen 92 95 101 102

Rotieren 59 75

[2] `https://www.youtube.com/watch?v=in_JgH-XUhY`



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/50_B-Baum/Aufgabe_YouTube_Loeschen.tex`

Übungsaufgabe „Modulo 11 und 17“ (Hashfunktion, Streutabellen (Hashing), Separate Verkettung, Lineares Sondieren, Quadratisches Sondieren)

(a) Ist $h(k) = k^2 \mod 11$ eine gut gewählte Hashfunktion? Begründen Sie Ihre Antwort.

Tipp: Berechnen Sie zunächst $h(k)$ für $0 \leq k < 11$. Überlegen Sie dann, welche Werte $h(k')$ für $k' = a \cdot 11 + k$ mit $a > 0$ und $0 \leq k < 11$ annehmen kann.

Lösungsvorschlag

Nein, h ist keine gute Hashfunktion. Betrachten wir zunächst die Wertetabelle von h für $0 \leq k < 11$. Wir erhalten

k	Nebenrechnung	$h(k)$
0	$0^2 \mod 11 = 0 \mod 11$	0
1	$1^2 \mod 11 = 1 \mod 11$	1
2	$2^2 \mod 11 = 4 \mod 11$	4
3	$3^2 \mod 11 = 9 \mod 11$	9
4	$4^2 \mod 11 = 16 \mod 11$	5
5	$5^2 \mod 11 = 25 \mod 11$	3
6	$6^2 \mod 11 = 36 \mod 11$	3
7	$7^2 \mod 11 = 49 \mod 11$	5
8	$8^2 \mod 11 = 64 \mod 11$	9
9	$9^2 \mod 11 = 81 \mod 11$	4
10	$10^2 \mod 11 = 100 \mod 11$	1

Wir sehen, dass nie die Werte 2, 6, 7, 8 und 10 eingenommen werden. Man könnte nun noch hoffen, dass das vielleicht für irgendein größeres k der Fall ist, dem ist jedoch nicht so. Wir können uns leicht davon überzeugen, dass für ein beliebiges $k' = a \cdot 11 + k$ mit $a > 0$ und $0 \leq k < 11$ folgendes gilt:

$$\begin{aligned} h(k') &= (k')^2 \mod 11 \\ &= (a \cdot 11 + k)^2 \mod 11 \\ &= (a^2 \cdot 11^2 + 2ak \cdot 11 + k^2) \mod 11 \\ &= (k^2) \mod 11 \\ &= h(k) \end{aligned}$$

Separate Verkettung

Somit haben wir die Berechnung des Hashwertes für ein beliebiges k' auf die Berechnung des Hashwertes für ein $k < 11$ zurückgeführt, was impliziert, dass kein Schlüssel jemals auf etwas anderes als 0, 1, 3, 4, 5 oder 9 abgebildet werden kann.

(b) Die Schlüssel 23, 57, 26, 6, 77, 43, 74, 60, 9, 91 sollen in dieser Reihenfolge mit der Hashfunktion $h(k) = k \mod 17$ in eine Hashtabelle der Länge 17 eingefügt werden.

Lösungsvorschlag

Lösungsvorschlag

Exkurs: Sondieren

separate Verkettung Kollisionsauflösung durch Verkettung (separate chaining): Jedes Bucket speichert mit Hilfe einer dynamischen Datenstruktur (Liste, Baum, weitere Streutabelle, ...) alle Elemente mit dem entsprechenden Hashwert.

lineares Sondieren es wird um ein konstantes Intervall verschoben nach einer freien Stelle gesucht. Meistens wird die Intervallgröße auf 1 festgelegt.

quadratisches Sondieren Nach jedem erfolglosen Suchschritt wird das Intervall quadriert.

(i) Verwenden Sie separate Verkettung zur Kollisionsauflösung.

Lösungsvorschlag

Nebenrechnung:

$17 \cdot 1 = 17$
$17 \cdot 2 = 34$
$17 \cdot 3 = 51$
$17 \cdot 4 = 68$
$17 \cdot 5 = 85$

Modulo-Berechnung der gegebenen Zahlen:

$23 \mod 17 = \mathbf{6}$ da $23 : 17 = 1$, Rest 6 da $23 = 1 \cdot 17 + 6$
$57 \mod 17 = 57 - 3 \cdot 17 = 57 - 51 = \mathbf{6}$
$26 \mod 17 = 26 - 17 = \mathbf{9}$
$6 \mod 17 = 6 - 0 \cdot 17 = \mathbf{6}$
$77 \mod 17 = 77 - 4 \cdot 17 = 77 - 68 = \mathbf{9}$
$43 \mod 17 = \mathbf{9}$
$74 \mod 17 = \mathbf{6}$
$60 \mod 17 = \mathbf{9}$
$9 \mod 17 = \mathbf{9}$
$91 \mod 17 = \mathbf{6}$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Schlüssel							23			26							
							57			77							
							6			43							
							74			60							
							91			9							

(ii) Verwenden Sie lineares Sondieren zur Kollisionsauflösung.

Lösungsvorschlag

Die Hashfunkion lautet:
$h'(k) = k \mod 17$
Die verwendete Hashfunktion beim linearen Sondieren:
$h(k,i) = (h'(k) - i) \mod 17$

Mit Schrittweite -1 ergeben sich folgende Sondierungsfolgen:

Schlüssel	Index
23	6
57	6 5
26	9
6	6 5 4
77	9 8
43	9 8 7
74	6 5 4 3
60	9 8 7 6 5 4 3 2
9	9 8 7 6 5 4 3 2 1
91	6 5 4 3 2 1

Damit ergibt sich folgende Hashtabelle:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Schlüssel	91	9	60	74	6	57	23	43	77	26							

(iii) Verwenden Sie quadratisches Sondieren zur Kollisionsauflösung.

Lösungsvorschlag

Die Hashfunkion lautet:
$h'(k) = k \mod 17$
Die verwendete Hashfunktion beim quadratischen Sondieren:
$h(k,i) = (h'(k) + i^2) \mod 17$

Am Beispiel von zwei Schlüsseln werden die Sondierungsfolgen berechnet:

$h'(23) = 6$

i. Sondierungsfolge:
$h(23,0) = (h'(23) + 0^2) \mod 17 = (6+0) \mod 17 = 6 \mod 17 = \mathbf{6}$

ii. Sondierungsfolge:
$h(23,1) = (h'(23) + 1^2) \mod 17 = (6+1) \mod 17 = 7 \mod 17 = \mathbf{7}$

iii. Sondierungsfolge:
$h(23,2) = (h'(23) + 2^2) \mod 17 = (6+4) \mod 17 = 10 \mod 17 = \mathbf{10}$

iv. Sondierungsfolge:
$h(23,3) = (h'(23) + 3^2) \mod 17 = (6+9) \mod 17 = 15 \mod 17 = \mathbf{15}$

v. Sondierungsfolge:
$h(23,4) = (h'(23) + 4^2) \mod 17 = (6+16) \mod 17 = 22 \mod 17 = \mathbf{5}$

$h'(26) = 9$

i. Sondierungsfolge:
$h(26,0) = (h'(26) + 0^2) \mod 17 = (9+0) \mod 17 = 9 \mod 17 = \mathbf{9}$

ii. Sondierungsfolge:
$h(26,1) = (h'(26) + 1^2) \mod 17 = (9+1) \mod 17 = 10 \mod 17 = \mathbf{10}$

iii. Sondierungsfolge:
$h(26,2) = (h'(26) + 2^2) \mod 17 = (9+4) \mod 17 = 13 \mod 17 = \mathbf{13}$

iv. Sondierungsfolge:
$h(26,3) = (h'(26) + 3^2) \mod 17 = (9+9) \mod 17 = 18 \mod 17 = \mathbf{1}$

v. Sondierungsfolge:
$h(26,4) = (h'(26) + 4^2) \mod 17 = (9+16) \mod 17 = 25 \mod 17 = \mathbf{8}$

vi. Sondierungsfolge:
$h(26,5) = (h'(26) + 5^2) \mod 17 = (9+25) \mod 17 = 34 \mod 17 = \mathbf{0}$

Es ergeben sich folgende Sondierungsfolgen:

Schlüssel	Index
23	6
57	6 7
26	9
6	6 7 10
77	9 10 13
43	9 10 13 1
74	6 7 10 15
60	9 10 13 1 8
9	9 10 13 1 8 0
91	6 7 10 15 5

Damit ergibt sich folgende Hashtabelle:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Schlüssel	9	43				91	23	57	60	26	6			77		74	

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/60_Hashing/Aufgabe_Modulo-11-17.tex`

Übungsaufgabe „“ (Streutabellen (Hashing))

3

$$h'(k) = k \mod 11$$
$$h(k,i) = (h'(k) + i + 3i^2) \mod 11$$
$$h'(17) = 17 \mod 11 = 6$$

Sondierungsfolgen

$$h(17,0) = (17 + 0 + 3 \cdot 0^2) \mod 11 = 6$$
$$h(17,1) = (17 + 1 + 3 \cdot 1^2) \mod 11 = 21 \mod 11 = 10$$
$$h(17,2) = (17 + 2 + 3 \cdot 2^2) \mod 11 = 31 \mod 11 = 31 - 2 \cdot 11 = 9$$
$$h(17,3) = (17 + 3 + 3 \cdot 3^2) \mod 11 = 47 \mod 11 = 47 - 4 \cdot 11 = 3$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/60_Hashing/Aufgabe_Sondieren_i-3i-hoch-2.tex`

[3] nach Foliensatz der RWTH Aachen, Seite 19 `https://moves.rwth-aachen.de/wp-content/uploads/SS15/dsal/lec13.pdf`

Übungsaufgabe „plus und minus i hoch 2“ (Streutabellen (Hashing))

4

Formel

$$h(k,i) := h'(k) + (-1)^{i+1} \cdot \left\lfloor \frac{i+1}{2} \right\rfloor^2 \mod m$$

$$k, k+1^2, k-1^2, k+2^2, k-2^2, \ldots k+(\tfrac{m-1}{2})^2, k-(\tfrac{m-1}{2})^2 \bmod m$$

Werte

$m = 19$, d. h. das Feld (die Tabelle) hat die Index-Nummern 0 bis 18. $k = h(x) = 7$

Sondierungsfolgen

i	Rechnung	Ergebnis	Index in der Tabelle
0	$7+0^2$	7	7
1	$7+1^2$	8	8
1	$7-1^2$	6	6
2	$7+2^2$	11	11
2	$7-2^2$	3	2
3	$7+3^2 = 7+9$	16	16
3	$7-3^2 = 7-9$	-2	17 $(19-2=10)$ oder $(0 \to 0, -1 \to 18, -2 \to 17)$
4	$7+4^2 = 7+16$	23	4 $(23-19=4)$ oder $(19 \to 0, 20 \to 1, 21 \to 2, 22 \to 3, 23 \to 4)$
4	$7-4^2 = 7-16$	-9	10 $(19-9=10)$ oder $(0 \to 0, -1 \to 18, -2 \to 17, \cdots, -9 \to 10)$
5	$7+5^2 = 7+25$	32	13 $(32-19=13)$
5	$7-5^2 = 7-25$	-18	1 $(19-18=1)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/80_Baeume/60_Hashing/Aufgabe_Sondieren_plus-minus-i-hoch-2.tex`

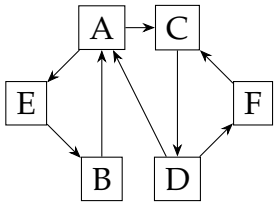
[4] nach Foliensatz der TU BraunschweigSeite 25`https://www.ibr.cs.tu-bs.de/courses/ws0708/aud/skript/hash.np.pdf`

Graphen

Examensaufgabe „Adjazenzmatrix und Adjazenzliste“ (46114-2006-F.T2-A6)

Aufgabe 6 (Graphrepräsentation)

Repräsentieren Sie den folgenden Graphen sowohl mit einer Adjazenzmatrix als auch mit einer Adjazenzliste.



Lösungsvorschlag

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} * & - & 1 & - & 1 & - \\ 1 & * & - & - & - & - \\ - & - & * & 1 & - & - \\ 1 & - & - & * & - & 1 \\ - & 1 & - & - & * & - \\ - & - & 1 & - & - & * \end{pmatrix} \end{array}$$

A $\rightarrow$ C $\rightarrow$ E
B $\rightarrow$ A
C $\rightarrow$ D
D $\rightarrow$ A $\rightarrow$ F
E $\rightarrow$ B
F $\rightarrow$ C

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46114/2006/03/Thema-2/Aufgabe-6.tex`

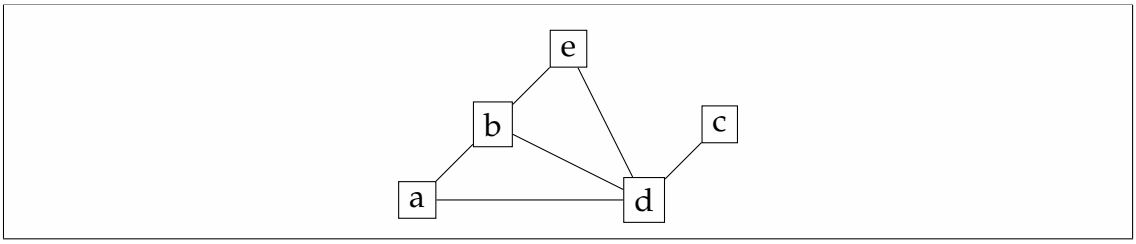
Examensaufgabe „Dijkstra“ (46114-2008-H.T1-A2)

Gegeben sei folgender Graph:

V: {a, b, c, d, e}
E: a → a, b
b → b, d, e
c → c, d
d → a, e

(a) Stellen Sie den Graphen grafisch dar!

Lösungsvorschlag



(b) Berechnen Sie mit dem Algorithmus von Dijkstra schrittweise die Länge der kürzesten Pfade ab dem Knoten a! Nehmen Sie dazu an, dass alle Kantengewichte 1 sind. Erstellen Sie eine Tabelle gemäß folgendem Muster:

ausgewählt |a| b c d e

Ergebnis:

Hinweis: Nur mit Angabe der jeweiligen Zwischenschritte gibt es Punkte. Es reicht also nicht, nur das Endergebnis hinzuschreiben.

Lösungsvorschlag

Nr.	ausgewählt	a	b	c	d	e
1	a	**0**	1	∞	1	∞
2	b	|	**1**	∞	1	2
3	d	|	|	2	**1**	2
4	c	|	|	**2**	|	2
5	e	|	|	|	|	**2**

(c) Welchen Aufwand hat der Algorithmus von Dijkstra bei Graphen mit |V| Knoten und |E| Kanten,

- wenn die Kantengewichte alle 1 sind? Mit welcher Datenstruktur und welchem Vorgehen lässt sich der Aufwand in diesem Fall reduzieren (mit kurzer Begründung)?

- wenn die Kantengewichte beliebig sind und als Datenstruktur eine Halde verwendet wird (mit kurzer Begründung)?

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46114/2008/09/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Prim nach Adjazenzmatrix, Tripelnotation“ (46115-2018-F.T1-A8)

Algorithmus von Prim

Berechnen Sie mithilfe des Algorithmus von Prim ausgehend vom Knoten s einen minimalen Spannbaum des ungerichteten Graphen G, der durch folgende Adjazenzmatrix gegeben ist:

$$\begin{array}{c} \\ s \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{ccccccccc} s & a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} * & - & 3 & - & - & 7 & - & - & - \\ - & * & - & 0 & 8 & - & 11 & - & - \\ 3 & - & * & - & 5 & - & 10 & - & - \\ - & 0 & - & * & - & - & 1 & - & - \\ - & 8 & 5 & - & * & 2 & 3 & - & 6 \\ 7 & - & - & - & 2 & * & - & - & 11 \\ - & 11 & 10 & 1 & 3 & - & * & 7 & - \\ - & - & - & - & - & - & 7 & * & 4 \\ - & - & - & - & 6 & 11 & - & 4 & * \end{pmatrix} \end{array}$$

(a) Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v, der vom Algorithmus als nächstes in den Ergebnisbaum aufgenommen wird (dieser sog. schwarze Knoten ist damit fertiggestellt) als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von v zu p an. Führen Sie in der zweiten Spalte alle anderen vom aktuellen Spannbaum direkt erreichbaren Knoten v (sog. graue Randknoten) ebenfalls als Tripel (v, p, δ) auf. Zeichnen Sie anschließend den entstandenen Spannbaum und geben Sie sein Gewicht an.

Lösungsvorschlag

Der Graph muss nicht gezeichnet werden. Der Algorithmus kann auch nur mit der Adjazenzmatrix durchgeführt werden. Möglicherweise geht das Lösen der Aufgabe schneller mit der Matrix von der Hand.

Kompletter Graph

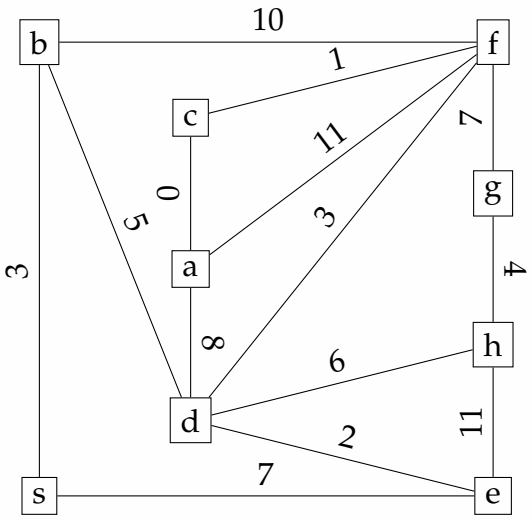
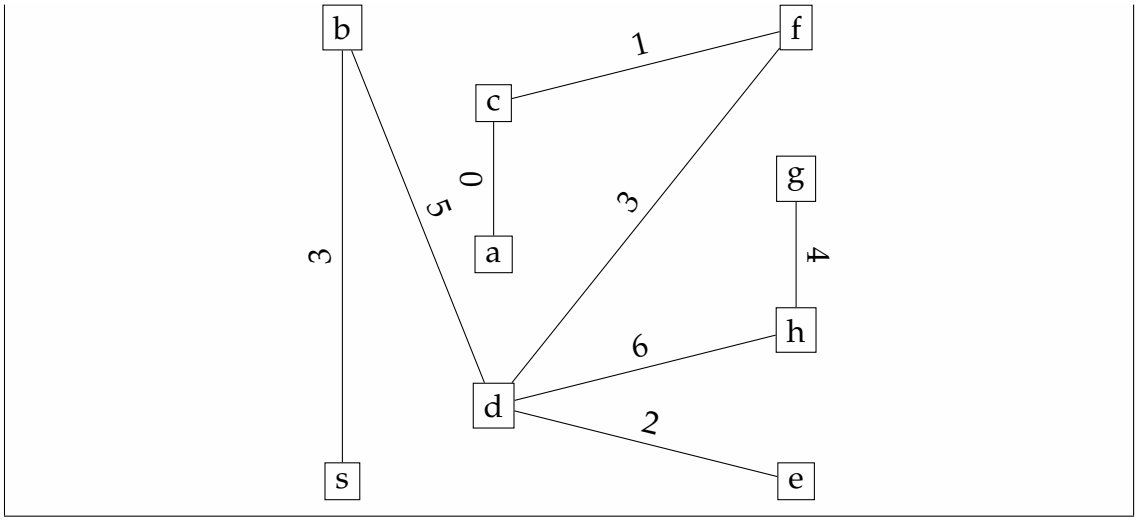


Tabelle schwarz-graue-Knoten

schwarze	graue
(s, null, -)	(b, s, 3); (e, s, 7);
(b, s, 3)	(d, b, 5); (e, s, 7); (f, b, 10);
(d, b, 5)	(a, d, 8); (e, d, 2); (f, d, 3); (h, d, 6);
(e, d, 2)	(a, d, 8); (f, d, 3); (h, d, 6);
(f, d, 3)	(a, d, 8); (c, f, 1); (g, f, 7); (h, d, 6);
(c, f, 1)	(a, c, 0); (g, f, 7); (h, d, 6);
(a, c, 0)	(g, f, 7); (h, d, 6);
(h, d, 6)	(g, h, 4);
(g, h, 4)	

Gewicht des minimalen Spannbaums: 24

Minimaler Spannbaum



(b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap nach Distanz verwaltet werden? Sei dabei n die Anzahl an Knoten und m die Anzahl an Kanten des Graphen. Eine Begründung ist nicht erforderlich.

Lösungsvorschlag

$\mathcal{O}(m + n \log n)$

(c) Beschreiben Sie kurz die Idee des alternativen Ansatzes zur Berechnung eines minimalen Spannbaumes von Kruskal.

Lösungsvorschlag

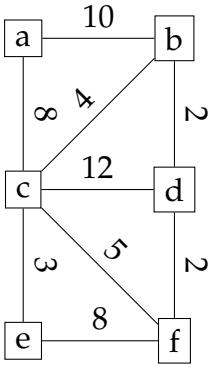
Kruskal wählt nicht die kürzeste an einen Teilgraphen anschließende Kante, sondern global die kürzeste verbliebene aller Kanten, die keinen Zyklus bildet, ohne dass diese mit dem Teilgraph verbunden sein muss.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2018/03/Thema-1/Aufgabe-8.tex`

Examensaufgabe „Graph a-f“ (46115-2018-F.T2-A4)

Minimaler Spannbaum
Algorithmus von Prim

Sei G der folgende Graph.



(a) Der Algorithmus von Prim ist ein Algorithmus zur Bestimmung des minimalen Spannbaums in einem Graphen. Geben Sie einen anderen Algorithmus zur Bestimmung des minimalen Spannbaums an.

Lösungsvorschlag

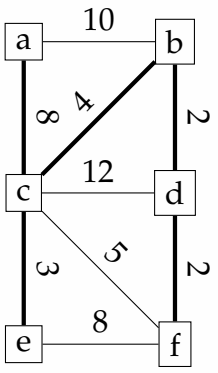
Zum Beispiel der Algorithmus von Kruskal

(b) Führen Sie den Algorithmus von Prim schrittweise auf G aus. Ausgangsknoten soll der Knoten a sein. Ihre Tabelle sollte wie folgt beginnen:

a	b	c	d	e	f	Warteschlange

Die Einträge der Tabelle geben an, wie weit der angegebene Knoten vom aktuellen Baum entfernt ist.

Lösungsvorschlag



a	b	c	d	e	f	Warteschlange
0	∞	∞	∞	∞	∞	a
0	10	8	∞	∞	∞	c, b
0	4	0	12	3	5	e, b, f, d
0	4	0	12	0	5	b, f, d
0	0	0	2	0	5	d, f
0	0	0	0	0	2	f
0	0	0	0	0	0	

(c) Erklären Sie, warum der Kürzeste-Wege-Baum (also das gezeichnete Ergebnis des Dijkstra-Algorithmus) und der minimale Spannbaum nicht notwendigerweise identisch sind.

Lösungsvorschlag

Die Wahl der nächsten Kante erfolgt nach völlig verschiedenen Kriterien:

- Beim Kürzeste-Wege-Baum orientiert sie sich an der Entfernung der einzelnen Knoten vom Startknoten.
- Beim Spannbaum orientiert sie sich an der Entfernung der einzelnen Knoten vom bereits erschlossenen Teil des Spannbaums.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2018/03/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Schwach zusammenhängend gerichteter Graph“ (46115-2021-F.T1-TA2-A3)

Wir betrachten eine Variante der Breitensuche (BFS), bei der die Knoten markiert werden, wenn sie das erste Mal besucht werden. Außerdem wird die Suche einmal bei jedem unmarkierten Knoten gestartet, bis alle Knoten markiert sind. Wir betrachten gerichtete Graphen. Ein gerichteter Graph G ist *schwach zusammenhängend*, wenn der ungerichtete Graph (der sich daraus ergibt, dass man die Kantenrichtungen von G ignoriert) zusammenhängend ist.

Exkurs: Schwach zusammenhängend gerichteter Graph
Beim gerichteten Graphen musst du auf die Kantenrichtung achten. Würde man die Richtungen der Kanten ignorieren wäre aber trotzdem jeder Knoten erreichbar. Einen solchen Graphen nennt man schwach zusammenhängend.[a]

Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist. [b]

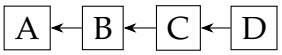
[a] `https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285`
[b] `https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)`

(a) Beschreiben Sie für ein allgemeines $n \in \mathbb{N}$ mit $n \geq 2$ den Aufbau eines schwach zusammenhängenden Graphen G_n, mit n Knoten, bei dem die Breitensuche $\Theta(n)$ mal gestartet werden muss, bis alle Knoten markiert sind.

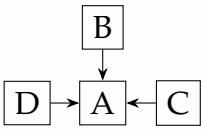
Lösungsvorschlag

?

Die Breitensuche benötigt einen Startknoten. Die unten aufgeführten Graphen finden immer nur einen Knoten nämlich den Startknoten.



Oder so:



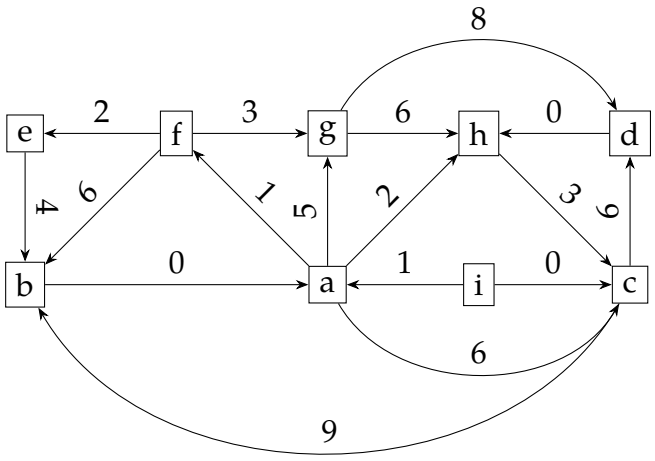
(b) Welche asymptotische Laufzeit in Abhängigkeit von der Anzahl der Knoten (n) und von der Anzahl der Kanten (m) hat die Breitensuche über alle Neustarts zusammen? Beachten Sie, dass die Markierungen nicht gelöscht werden. Geben Sie die Laufzeit in Θ-Notation an. Begründen Sie Ihre Antwort.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Graph a-i“ (46115-2021-F.T2-TA2-A4)

Algorithmus von Dijkstra

(a) Berechnen Sie im gegebenen gerichteten und gewichteten Graph $G = (V, E, w)$ mit Kantenlängen $w : E \to \mathbb{R}_0^+$ mittels des Dijkstra-Algorithmus die kürzesten (gerichteten) Pfade ausgehend vom Startknoten a.



Knoten, deren Entfernung von a bereits feststeht, seien als schwarz bezeichnet und Knoten, bei denen lediglich eine obere Schranke $\neq \infty$ für ihre Entfernung von a bekannt ist, seien als *grau* bezeichnet.

(i) Geben Sie als Lösung eine Tabelle an. Fügen Sie jedes mal, wenn der Algorithmus einen Knoten schwarz färbt, eine Zeile zu der Tabelle hinzu. Die Tabelle soll dabei zwei Spalten beinhalten: die linke Spalte zur Angabe des aktuell schwarz gewordenen Knotens und die rechte Spalte mit der bereits aktualisierten Menge grauer Knoten. Jeder Tabelleneintrag soll anstelle des nackten Knotennamens v ein Tripel $(v, v.d, v.\pi)$ sein. Dabei steht $v.d$ für die aktuell bekannte kürzeste Distanz zwischen a und v. $v.\pi$ ist der direkte Vorgänger von v auf dem zugehörigen kürzesten Weg von a.

Lösungsvorschlag

Nr.	besucht	a	b	c	d	e	f	g	h	i
0		0	∞	∞	∞	∞	∞	∞	∞	∞
1	a	**0**	∞	6	∞	∞	1	5	2	∞
2	f	|	7	6	∞	3	**1**	4	2	∞
3	h	|	7	5	∞	3	|	4	**2**	∞
4	e	|	7	5	∞	**3**	|	4	|	∞
5	g	|	7	5	12	|	|	**4**	|	∞
6	c	|	7	**5**	11	|	|	|	|	∞
7	b	|	**7**	|	11	|	|	|	|	∞
8	d	|	|	|	**11**	|	|	|	|	∞
9	i	|	|	|	|	|	|	|	|	∞

nach	Entfernung	Reihenfolge	Pfad
a → a	0	1	
a → b	7	7	a → f → b
a → c	5	6	a → h → c
a → d	11	8	a → h → c → d
a → e	3	4	a → f → e
a → f	1	2	a → f
a → g	4	5	a → f → g
a → h	2	3	a → h
a → i	2.147483647E9	9	a → i

(ii) Zeichnen Sie zudem den entstandenen Kürzeste-Pfade-Baum.

(b) Warum berechnet der Dijkstra-Algorithmus auf einem gerichteten Eingabegraphen mit potentiell auch negativen Kantengewichten $w : E \to \mathbb{R}$ nicht immer einen korrekten Kürzesten-Wege-Baum von einem gewählten Startknoten aus? Geben Sie ein Beispiel an, für das der Algorithmus die falsche Antwort liefert.

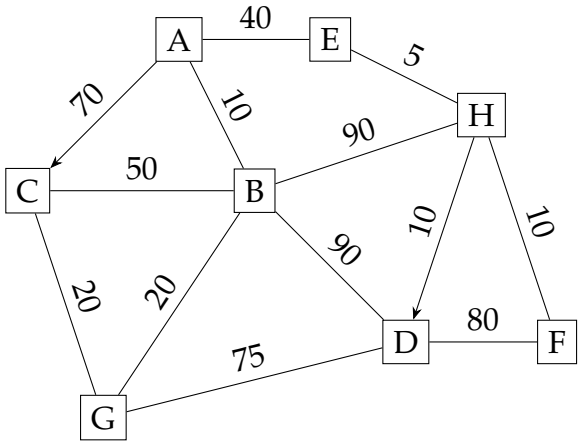
(c) Begründen Sie, warum das Problem nicht gelöst werden kann, indem der Betrag des niedrigsten (also des betragsmäßig größten negativen) Kantengewichts im Graphen zu allen Kanten addiert wird.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-2/Aufgabe-4.tex`

Examensaufgabe „Städte gemischt gerichtet / ungerichtet“ (66112-2004-F.T1-A5)

Algorithmus von Dijkstra
Adjazenzmatrix

Ein wichtiges Problem im Bereich der Graphalgorithmen ist die Berechnung kürzester Wege. Gegeben sei der folgende Graph, in dem Städte durch Kanten verbunden sind. Die Kantengewichte geben Fahrzeiten an. Außer den durch Pfeile als nur in eine Richtung befahrbar gekennzeichneten Straßen sind alle Straßen in beiden Richtungen befahrbar.



(a) Geben Sie zu dem obigen Graphen zunächst eine Darstellung als Adjazenzmatix an.

Lösungsvorschlag

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{pmatrix} * & 10 & 70 & - & 40 & - & - & - \\ 10 & * & 50 & 90 & - & - & 20 & 90 \\ - & 50 & * & - & - & - & 20 & - \\ - & 90 & - & * & - & 80 & 75 & - \\ 40 & - & - & - & * & - & - & 5 \\ - & - & - & 80 & - & * & - & 10 \\ - & 20 & 20 & 75 & - & - & * & - \\ - & 90 & - & 10 & 5 & 10 & - & * \end{pmatrix} \end{array}$$

(b) Berechnen Sie nun mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten A zu allen anderen Knoten.

Lösungsvorschlag

Nr.	besucht	A	B	C	D	E	F	G	H
0		0	∞	∞	∞	∞	∞	∞	∞
1	A	**0**	10	70	∞	40	∞	∞	∞
2	B	|	**10**	60	100	40	∞	30	100
3	G	|	|	50	100	40	∞	**30**	100
4	E	|	|	50	100	**40**	∞	|	45
5	H	|	|	50	55	|	55	|	**45**
6	C	|	|	**50**	55	|	55	|	|
7	D	|	|	|	**55**	|	55	|	|
8	F	|	|	|	|	|	**55**	|	|

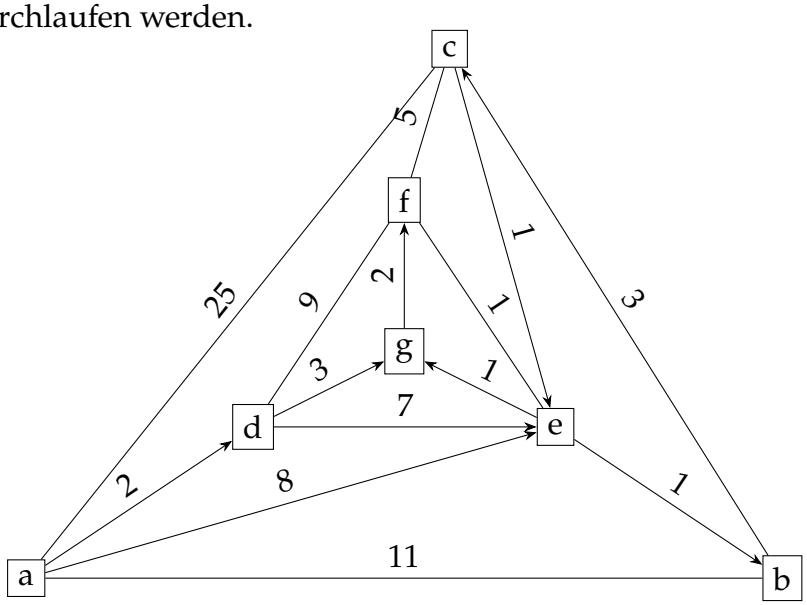
nach	Entfernung	Reihenfolge	Pfad
A → A	0	1	
A → B	10	2	A → B
A → C	50	6	A → B → G → C
A → D	55	7	A → E → H → D
A → E	40	4	A → E
A → F	55	8	A → E → H → F
A → G	30	3	A → B → G
A → H	45	5	A → E → H

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2004/03/Thema-1/Aufgabe-5.tex`

Examensaufgabe „Graph a-g“ (66115-2013-H.T2-A9)

Algorithmus von Dijkstra

Gegeben sei der unten stehende gerichtete Graph $G = (V, E)$ mit positiven Kantenlingen $l(e)$ für jede Kante $e \in E$. Kanten mit Doppelspitzen können in beide Richtungen durchlaufen werden.



(a) In welcher Reihenfolge werden die Knoten von G ab dem Knoten a durch den Dijkstra-Algorithmus bei der Berechnung der kürzesten Wege endgültig bearbeitet?

Lösungsvorschlag

Nr.	besucht	a	b	c	d	e	f	g
0		0	∞	∞	∞	∞	∞	∞
1	a	**0**	11	25	2	8	∞	∞
2	d	|	11	25	**2**	8	11	5
3	g	|	11	25	|	8	7	**5**
4	f	|	11	12	|	8	**7**	|
5	e	|	9	12	|	**8**	|	|
6	b	|	**9**	12	|	|	|	|
7	c	|	|	**12**	|	|	|	|

(b) Berechnen Sie die Länge des kürzesten Weges von a zu jedem Knoten.

Lösungsvorschlag

siehe oben

(c) Geben Sie einen kürzesten Weg von a nach c an.

Lösungsvorschlag

$a \rightarrow d \rightarrow g \rightarrow f \rightarrow c$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/09/Thema-2/Aufgabe-9.tex`

Examensaufgabe „Karlsruhe nach Kassel“ (66115-2016-F.T2-A6)

Algorithmus von Dijkstra

(a) Berechnen Sie für folgenden Graphen den kürzesten Weg von Karlsruhe nach Kassel und dokumentieren Sie den Berechnungsweg:

Verwendete Abkürzungen:

A Augsburg

EF Erfurt

F Frankfurt

KA Karlsruhe

KS Kassel

M München

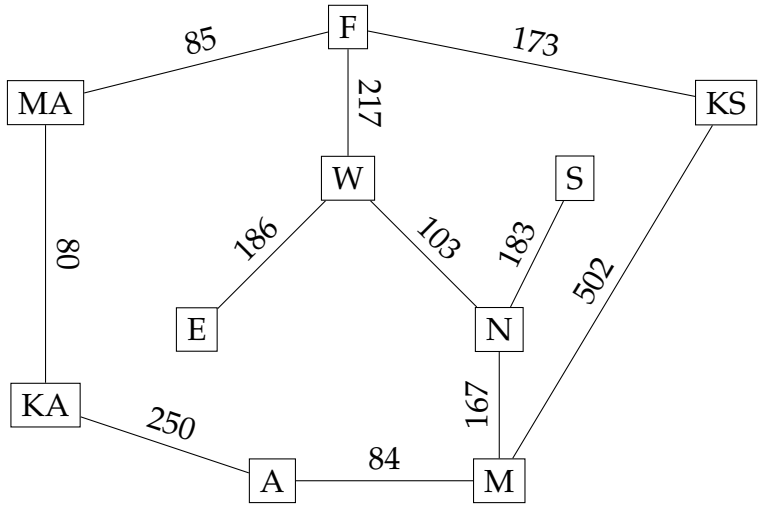
MA Mannheim

N Nürnberg

S Stuttgart

WÜ Würzburg

Zahl = Zahl in Kilometern



Lösungsvorschlag

Nr.	besucht	A	E	F	KA	KS	M	MA	N	S	W
0		∞	∞	∞	0	∞	∞	∞	∞	∞	∞
1	KA	**250**	∞	∞	**0**	∞	∞	80	∞	∞	∞
2	MA	|	∞	165	|	∞	∞	**80**	∞	∞	∞
3	F	|	∞	**165**	|	338	∞	|	∞	∞	382
4	A	|	∞	|	|	338	334	|	∞	∞	382
5	M	|	∞	|	|	338	**334**	|	501	∞	382
6	KS	|	∞	|	|	**338**	|	|	501	∞	382
7	W	|	568	|	|	|	|	|	485	∞	**382**
8	N	|	568	|	|	|	|	|	**485**	668	|
9	E	|	**568**	|	|	|	|	|	|	668	|
10	S	|	|	|	|	|	|	|	|	**668**	|

nach	Entfernung	Reihenfolge	Pfad
KA → A	250	0	KA → A
KA → E	568	9	KA → MA → F → W → E
KA → F	165	3	KA → MA → F
KA → KA	0	1	
KA → KS	338	6	KA → MA → F → KS
KA → M	334	5	KA → A → M
KA → MA	80	2	KA → MA
KA → N	485	8	KA → MA → F → W → N
KA → S	668	10	KA → MA → F → W → N → S
KA → W	382	7	KA → MA → F → W

(b) Könnte man den Dijkstra Algorithmus auch benutzen, um das Travelling-Salesman Problem zu lösen?

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-2/Aufgabe-6.tex`

Examensaufgabe „Bayerischee Autobahnen“ (66115-2017-F.T1-A1)

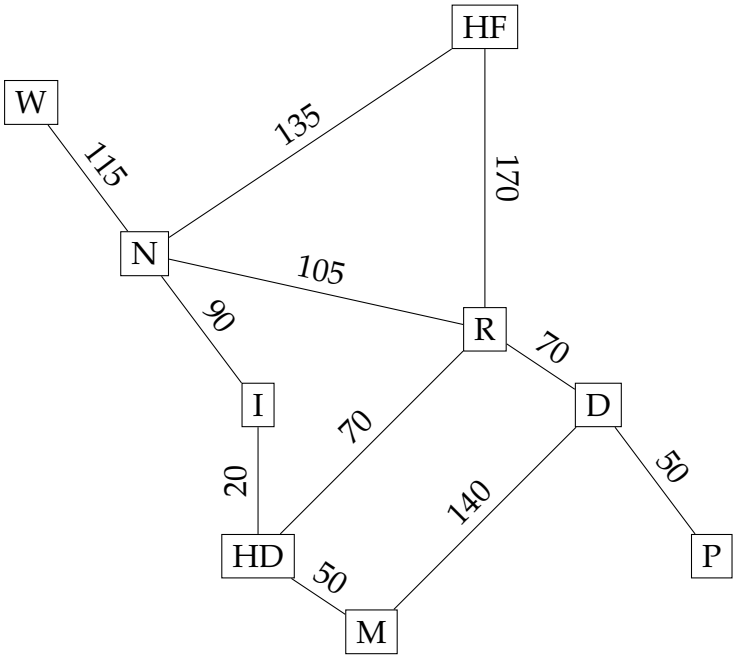
Die folgende Abbildung zeigt die wichtigsten bayerischen Autobahnen zusammen mit einigen anliegenden Orten und die Entfernungen zwischen diesen.

Entfernungstabelle

von	nach	km
Würzburg	Nürnberg	115
Nürnberg	Regensburg	105
Regensburg	AK Deggendorf	70
AK Deggendorf	Passau	50
Hof	Nürnberg	135
Nürnberg	Ingolstadt	90
Ingolstadt	AD Holledau	20
AD Holledau	München	50
München	AK Deggendorf	140
Hof	Regensburg	170
Regensburg	AD Holledau	70

Abkürzungen

D	Deggendorf
HF	Hof
HD	Holledau
I	Ingolstadt
M	München
N	Nürnberg
P	Passau
R	Regensburg
W	Würzburg



(a) Bestimmen Sie mit dem Algorithmus von *Dijkstra* den kürzesten Weg von Ingolstadt zu allen anderen Orten. Verwenden Sie zur Lösung eine Tabelle gemäß folgendem Muster und markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Ort. Setzen Sie für die noch zu bearbeitenden Orte eine Prioritätswarteschlange ein, ðbei gleicher Entfernung wird der ältere Knoten gewählt.

Lösungsvorschlag

Nr.	**besucht**	**D**	**HD**	**HF**	**I**	**M**	**N**	**P**	**R**	**W**
0		∞	∞	∞	0	∞	∞	∞	∞	∞
1	I	∞	20	∞	**0**	∞	90	∞	∞	∞
2	HD	∞	**20**	∞	|	70	90	∞	90	∞
3	M	210	|	∞	|	**70**	90	∞	90	∞
4	N	210	|	225	|	|	**90**	∞	90	205
5	R	160	|	225	|	|	|	∞	**90**	205
6	D	**160**	|	225	|	|	|	210	|	205
7	W	|	|	225	|	|	|	210	|	**205**
8	P	|	|	225	|	|	|	**210**	|	|
9	HF	|	|	**225**	|	|	|	|	|	|

(b) Die bayerische Landesregierung hat beschlossen, die eben betrachteten Orte mit einem breitbandigen Glasfaser-Backbone entlang der Autobahnen zu verbinden. Dabei soll aus Kostengründen so wenig Glasfaser wie möglich verlegt werden. Identifizieren Sie mit dem Algorithmus von Kruskal diejenigen Strecken, entlang

welcher Glasfaser verlegt werden muss. Geben Sie die Ortspaare (Autobahnsegmente) in der Reihenfolge an, in der Sie sie in Ihre Verkabelungsliste aufnehmen. Algorithmus von Kruskal

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(c) Um Touristen den Besuch aller Orte so zu ermöglichen, dass sie dabei jeden Autobahnabschnitt genau einmal befahren müssen, bedarf es zumindest eines sogenannten offenen Eulerzugs. Zwischen welchen zwei Orten würden Sie eine Autobahn bauen, damit das bayerische Autobahnnetz mindestens einen Euler-Pfad enthält?

> **Exkurs: offener Eulerzug**
>
> Ein offener Eulerzug ist gegeben, wenn Start- und Endknoten nicht gleich sein müssen, wenn also statt eines Zyklus lediglich eine Kantenfolge verlangt wird, welche jede Kante des Graphen genau einmal enthält. Ein bekanntes Beispiel ist das „Haus vom Nikolaus“.

Lösungsvorschlag

Zwischen Deggendorf und Würzburg

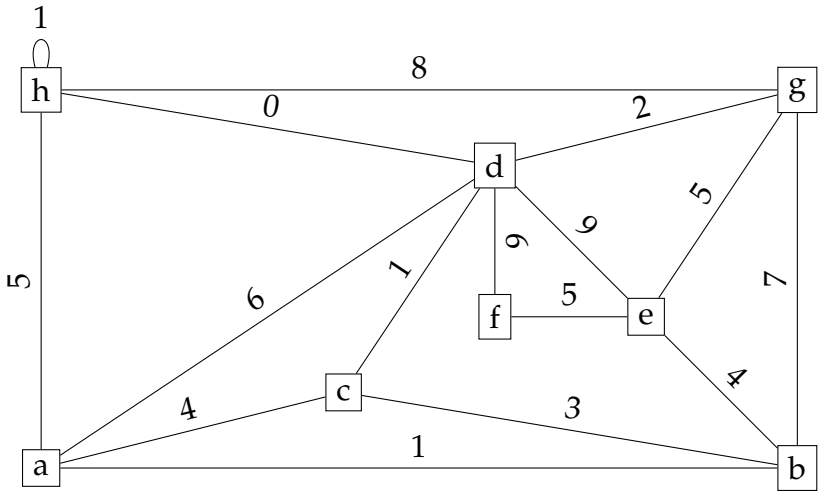
$$\mathrm{P} \rightarrow \mathrm{D} \rightarrow \mathrm{R} \rightarrow \mathrm{N} \rightarrow \mathbf{W} \rightarrow \mathbf{D} \rightarrow \mathrm{M} \rightarrow \mathrm{HD} \rightarrow \mathrm{R} \rightarrow \mathrm{HF} \rightarrow \mathrm{N} \rightarrow \mathrm{I} \rightarrow \mathrm{HD}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Graph a-h“ (66115-2018-F.T2-A10)

(a) Berechnen Sie mithilfe des Algorithmus von Prim ausgehend vom Knoten a einen minimalen Spannbaum des ungerichteten Graphen G, der durch folgende Adjazenzmatrix gegeben ist:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} * & 1 & 4 & 6 & - & - & - & 5 \\ 1 & * & 3 & - & 4 & - & 7 & - \\ 4 & 3 & * & 1 & - & - & - & - \\ 6 & - & 1 & * & 9 & 6 & 2 & 0 \\ - & 4 & - & 9 & * & 5 & 5 & - \\ - & - & - & 6 & 5 & * & - & - \\ - & 7 & - & 2 & 5 & - & * & 8 \\ 5 & - & - & 0 & - & - & 8 & 1 \end{pmatrix} \end{array}$$

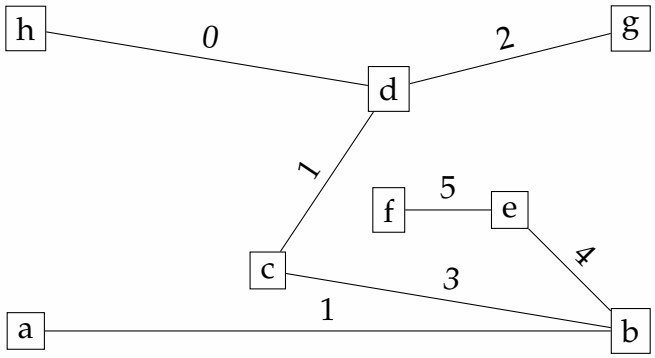


Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v, der vom Algorithmus als nächstes in den Ergebnisbaum aufgenommen wird (dieser sog. „schwarze“ Knoten ist damit fertiggestellt), als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von v zu p an. Führen Sie in der zweiten Spalte alle anderen vom aktuellen Spannbaum direkt erreichbaren Knoten v (sog. „graue Randknoten“) ebenfalls als Tripel (v, p, δ) auf.

Zeichnen Sie anschließend den entstandenen Spannbaum und geben sein Gewicht an.

Lösungsvorschlag

„schwarze“	„graue“ Randknoten
(a, NULL, ∞)	(b, a, 1) (c, a, 4) (h, a, 5) (d, a, 6)
(b, a, 1)	(c, b, 3) (e, b, 4) (h, a, 5) (d, a, 6) (g, b, 7)
(c, b, 3)	(d, c, 1) (e, b, 4) (h, a, 5) (g, b, 7)
(d, c, 1)	(h, d, 0) (g, d, 2) (e, b, 4) (f, d, 6)
(h, d, 0)	(g, d, 2) (e, b, 4) (f, d, 6)
(g, d, 2)	(e, b, 4) (f, d, 6)
(e, b, 4)	(f, e, 5)
(f, e, 5)	



Minimales Kantengewicht: 16

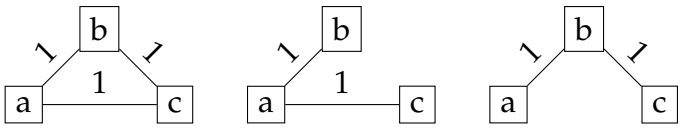
(b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap (= Halde) nach Distanz verwaltet werden? Sei dabei n die Anzahl an Knoten und m die Anzahl an Kanten des Graphen. Eine Begründung ist nicht erforderlich.

Lösungsvorschlag

$\mathcal{O}(n \cdot \log(n) + m)$

(c) Zeigen Sie durch ein kleines Beispiel, dass ein minimaler Spannbaum eines ungerichteten Graphen nicht immer eindeutig ist.

Lösungsvorschlag



(d) Skizzieren Sie eine Methode, mit der ein maximaler Spannbaum mit einem beliebigen Algorithmus für minimale Spannbäume berechnet werden kann. In welcher Laufzeitkomplexität kann ein maximaler Spannbaum berechnet werden?

Lösungsvorschlag

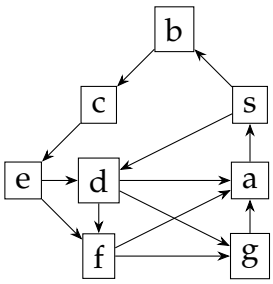
Alle Kantengewichte negieren. In $\mathcal{O}(n \cdot \log(n) + m)$ wie der Algorithmus von Prim.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-10.tex`

Examensaufgabe „Graph a-g, Startknoten s“ (66115-2018-F.T2-A11)

Graphen
Tiefensuche
Breitensuche

Gegeben sei der folgende gerichtete Graph G:



Traversieren Sie G ausgehend vom Knoten s mittels

(a) Tiefensuche (DFS),

Lösungsvorschlag

Rekursiv ohne Keller:

0	1	2	3	4	5	6	7
s	b	c	e	d	a	f	g

(b) Breitensuche (BFS)

Lösungsvorschlag

mit Warteschlange:

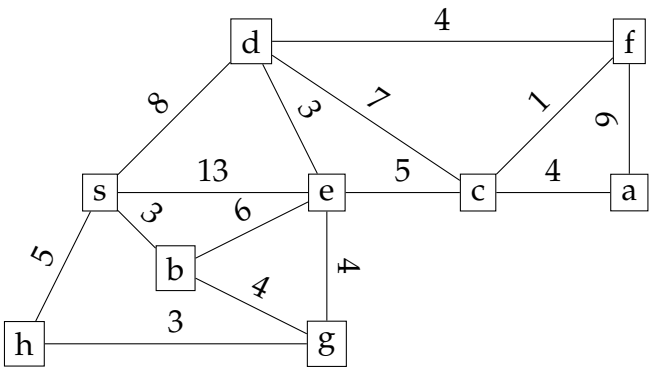
0	1	2	3	4	5	6	7
s	b	d	c	a	f	g	e

und geben Sie jeweils die erhaltene Nummerierung der Knoten an. Besuchen Sie die Nachbarn eines Knotens bei Wahlmöglichkeiten immer in alphabetisch aufsteigender Reihenfolge.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-11.tex`

Examensaufgabe „Negative Kantengewichte“ (66115-2018-F.T2-A9)

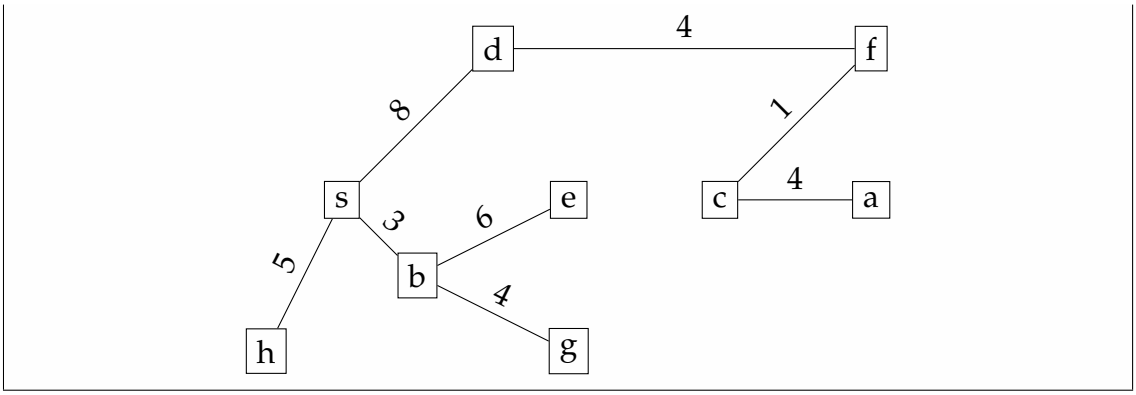
Gegeben sei folgender Graph G.



(a) Berechnen Sie mithilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten s zu allen anderen Knoten im Graphen G. Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte den jeweils als nächstes fertigzustellenden Knoten v (wird sog. „schwarz“) als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von s zu v über p an. Führen Sie in der zweiten Spalten alle anderen bisher erreichten Knoten v ebenfalls als Tripel (v, p, δ) auf, wobei diese sog. „grauen Randknoten“ in folgenden Durchgängen erneut betrachtet werden müssen. Zeichnen Sie anschließend den entstandenen Wegebaum, ðden Graphen G, in dem nur noch diejenigen Kanten vorkommen, die Teil der kürzesten Wege von s zu allen anderen Knoten sind.

Lösungsvorschlag

Nr	„schwarze“ Knoten	„graue“ Randknoten
1	**(s, -, 0)**	[(b, s, 3)] (d, s, 8) (e, s, 13) (h, s, 5)
2	**(b, s, 3)**	(d, s, 8) (e, **b**, **9**) (g, b, 7) [(h, s, 5)]
3	**(h, s, 5)**	(d, s, 8) (e, b, 9) [(g, b, 7)]
4	**(g, b, 7)**	[(d, s, 8)] (e, b, 9)
5	**(d, s, 8)**	(c, d, 15) [(e, b, 9)] (f, d, 12)
6	**(e, b, 9)**	(c, **e**, **14**) [(f, d, 12)]
7	**(f, d, 12)**	(a, f, 21) [(c, **f**, **13**)]
8	**(c, f, 13)**	[(a, **c**, **17**)]
9	**(a, c, 17)**	



Alternativer Lösungsweg

Lösungsvorschlag

Nr.	besucht	a	b	c	d	e	f	g	h	s
0		∞	∞	∞	∞	∞	∞	∞	∞	0
1	s	∞	3	∞	8	13	∞	∞	5	**0**
2	b	∞	**3**	∞	8	9	∞	7	5	|
3	h	∞	|	∞	8	9	∞	7	**5**	|
4	g	∞	|	∞	8	9	∞	**7**	|	|
5	d	∞	|	15	**8**	9	12	|	|	|
6	e	∞	|	14	|	**9**	12	|	|	|
7	f	21	|	13	|	|	**12**	|	|	|
8	c	17	|	**13**	|	|	|	|	|	|
9	a	**17**	|	|	|	|	|	|	|	|

nach	Entfernung	Reihenfolge	Pfad
s → a	17	9	s → d → f → c → a
s → b	3	2	s → b
s → c	13	8	s → d → f → c
s → d	8	5	s → d
s → e	9	6	s → b → e
s → f	12	7	s → d → f
s → g	7	4	s → b → g
s → h	5	3	s → h
s → s	0	1	

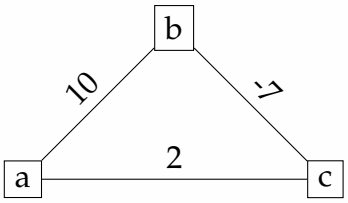
(b) Der Dijkstra-Algorithmus liefert bekanntlich auf Graphen mit negativen Kanten-

gewichten unter Umständen ein falsches Ergebnis.

(i) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein korrektes Ergebnis liefert.

Lösungsvorschlag

Startknoten a



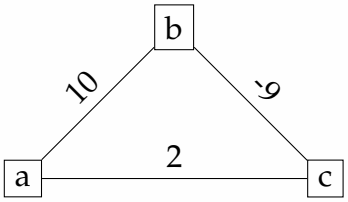
Nr.	besucht	a	b	c
0		0	∞	∞
1	a	**0**	10	2
2	c	|	10	**2**
3	b	|	**10**	|

Richtig: a - c: 2

(ii) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein falsches Ergebnis liefert.

Lösungsvorschlag

Startknoten a



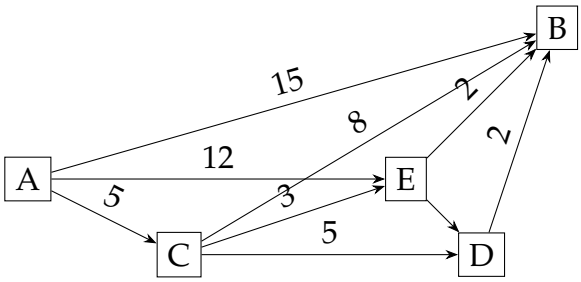
Nr.	besucht	a	b	c
0		0	∞	∞
1	a	**0**	10	2
2	c	|	10	**2**
3	b	|	**10**	|

falsch: a - c: müsste 1 (10 - 9) sein.

Ein Beweis oder eine Begründung ist jeweils nicht erforderlich.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-9.tex`

Examensaufgabe „Graph A-E“ (66115-2020-H.T1-TA2-A3)



(a) Ermitteln Sie mit dem Algorithmus von Dijkstra den kürzesten Weg vom Knoten A zu allen erreichbaren Knoten in G. Verwenden Sie zur Lösung eine Tabelle der folgenden Form. Markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Knoten und führen Sie die Prioritätswarteschlange der noch zu betrachtenden Knoten (aufsteigend sortiert).

Nr.	besucht	A	B	C	D	E
0		0	∞	∞	∞	∞

Lösungsvorschlag

Nr.	besucht	A	B	C	D	E
0		0	∞	∞	∞	∞
1	A	**0**	15	5	∞	12
2	C	|	13	**5**	10	8
3	E	|	10	|	9	**8**
4	D	|	10	|	**9**	|
5	B	|	**10**	|	|	|

(b) Geben Sie den kürzesten Pfad vom Knoten A zum Knoten B an.

Lösungsvorschlag

A → C → E → B: 10

nach	Entfernung	Reihenfolge	Pfad
A → A	0	0	
A → B	10	5	A → C → E → B
A → C	5	2	A → C
A → D	9	4	A → C → E → D
A → E	8	3	A → C → E

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Schwach zusammenhängend gerichteter Graph“ (66115-2021-F.T1-TA2-A3)

Wir betrachten eine Variante der Breitensuche (BFS), bei der die Knoten markiert werden, wenn sie das erste Mal besucht werden. Außerdem wird die Suche einmal bei jedem unmarkierten Knoten gestartet, bis alle Knoten markiert sind. Wir betrachten gerichtete Graphen. Ein gerichteter Graph G ist *schwach zusammenhängend*, wenn der ungerichtete Graph (der sich daraus ergibt, dass man die Kantenrichtungen von G ignoriert) zusammenhängend ist.

Exkurs: Schwach zusammenhängend gerichteter Graph
Beim gerichteten Graphen musst du auf die Kantenrichtung achten. Würde man die Richtungen der Kanten ignorieren wäre aber trotzdem jeder Knoten erreichbar. Einen solchen Graphen nennt man schwach zusammenhängend.[a]

Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist. [b]

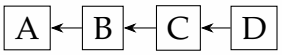
[a] `https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285`
[b] `https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)`

(a) Beschreiben Sie für ein allgemeines $n \in \mathbb{N}$ mit $n \geq 2$ den Aufbau eines schwach zusammenhängenden Graphen G_n, mit n Knoten, bei dem die Breitensuche $\Theta(n)$ mal gestartet werden muss, bis alle Knoten markiert sind.

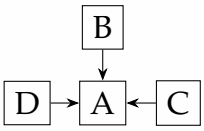
Lösungsvorschlag

?

Die Breitensuche benötigt einen Startknoten. Die unten aufgeführten Graphen finden immer nur einen Knoten nämlich den Startknoten.



Oder so:



(b) Welche asymptotische Laufzeit in Abhängigkeit von der Anzahl der Knoten (n) und von der Anzahl der Kanten (m) hat die Breitensuche über alle Neustarts zusammen? Beachten Sie, dass die Markierungen nicht gelöscht werden. Geben Sie die Laufzeit in Θ-Notation an. Begründen Sie Ihre Antwort.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Kürzeste-Wege-Bäume und minimale Spannbäume“ (66115-2021-F.T1-TA2-A4)

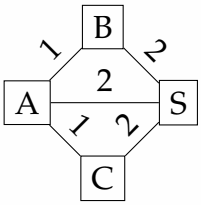
Graphen
Algorithmus von Dijkstra
Algorithmus von Prim

Die Algorithmen von Dijkstra und Jarník-Prim gehen ähnlich vor. Beide berechnen, ausgehend von einem Startknoten, einen Baum. Allerdings berechnet der Algorithmus von Dijkstra einen Kürzesten-Wege-Baum, während der Algorithmus von Jarník-Prim einen minimalen Spannbaum berechnet.

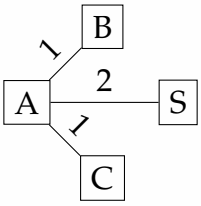
(a) Geben Sie einen ungerichteten gewichteten Graphen G mit höchstens fünf Knoten und einen Startknoten s von G an, so dass **Dijkstra**(G, s) und **Jarník-Prim**(G, s) ausgehend von s verschiedene Bäume in G liefern. Geben Sie beide Bäume an.

Lösungsvorschlag

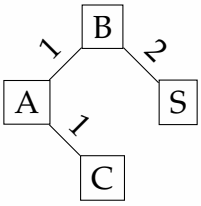
Originalgraph



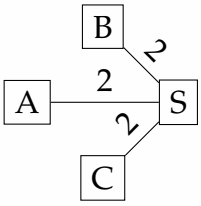
Minimaler Spannbaum 1



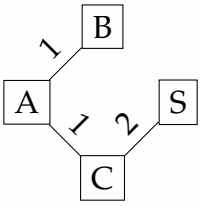
Minimaler Spannbaum 2



Dijkstra-Wegebaum von S aus:



Minimaler Spannbaum 3



(b) Geben Sie eine unendlich große Menge von Graphen an, auf denen der Algorithmus von Jarník-Prim asymptotisch schneller ist als der Algorithmus von Kruskal, der ebenfalls minimale Spannbäume berechnet.

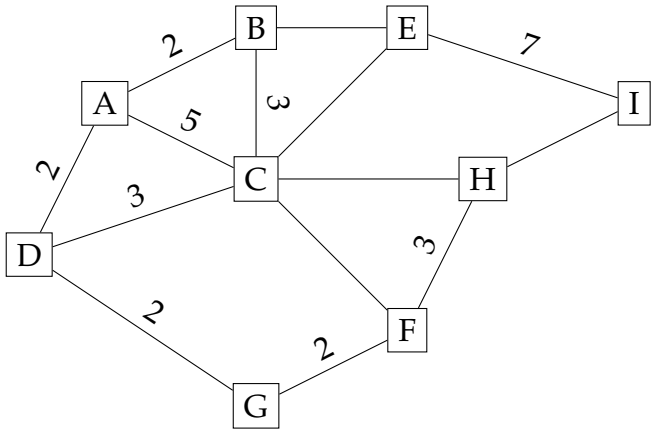
Hinweis: Für einen Graphen mit n Knoten und m Kanten benötigt Jarník-Prim $\mathcal{O}(m + n \log n)$ Zeit, Kruskal $\mathcal{O}(m \log m)$ Zeit.

(c) Sei Z die Menge der zusammenhängenden Graphen und $G \in Z$. Sei n die Anzahl der Knoten von G und m die Anzahl der Kanten von G. Entscheiden Sie mit Begründung, ob $\log m \in \Theta(\log n)$ gilt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-2/Aufgabe-4.tex

Übungsaufgabe „Graph A-I“ (Algorithmus von Dijkstra)

Führen Sie auf dem gegebenen Graphen die Suche nach der kürzesten Distanz aller Knoten zum Startknoten *A* mit dem Algorithmus von Dijkstra durch. Tragen Sie die Abarbeitungsreihenfolge, den unmittelbaren Vorgängerknoten, sowie die ermittelte kürzeste Distanz für jeden Knoten ein! Bei gleichen Distanzen arbeiten Sie die Knoten in lexikalischer Reihenfolge ab.



Lösungsvorschlag

Nr.	besucht	A	B	C	D	E	F	G	H	I
0		0	∞	∞	∞	∞	∞	∞	∞	∞
1	A	**0**	2	5	2	∞	∞	∞	∞	∞
2	B	|	**2**	5	2	3	∞	∞	∞	∞
3	D	|	|	5	**2**	3	∞	4	∞	∞
4	E	|	|	4	|	**3**	∞	4	∞	10
5	C	|	|	**4**	|	|	5	4	5	10
6	G	|	|	|	|	|	5	**4**	5	10
7	F	|	|	|	|	|	**5**	|	5	10
8	H	|	|	|	|	|	|	|	**5**	6
9	I	|	|	|	|	|	|	|	|	**6**

nach	Entfernung	Reihenfolge	Pfad
A → A	0	0	
A → B	2	2	A → B
A → C	4	5	A → B → E → C
A → D	2	3	A → D
A → E	3	4	A → B → E
A → F	5	7	A → B → E → C → F
A → G	4	6	A → D → G
A → H	5	8	A → B → E → C → H
A → I	6	9	A → B → E → C → H → I

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/90_Graphen/10_Dijkstra/Aufgabe_Graph-A-I.tex`

Übungsaufgabe „Städte A-F“ (Algorithmus von Dijkstra)

Nehmen Sie an, es gibt sieben Städte A, B, C, D, E, F und G. Sie wohnen in der Stadt A und möchten zu jeder der anderen Städte die preiswerteste Flugverbindung finden (einfach ohne Rückflug). Sie sind dazu bereit, beliebig oft umzusteigen. Folgende Direktflüge stehen Ihnen zur Verfügung:

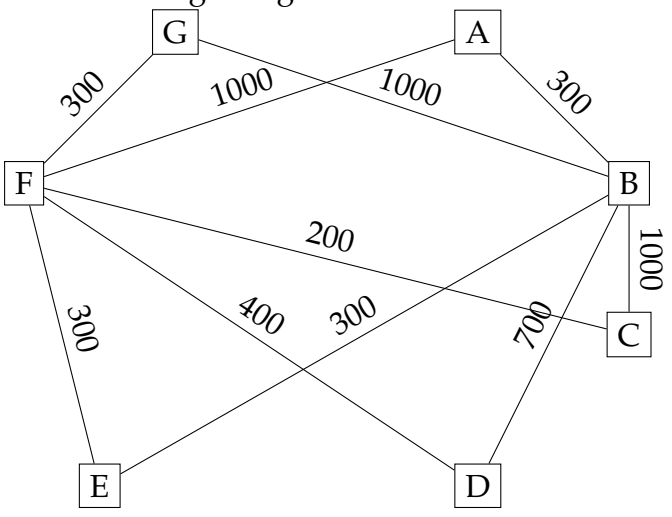
Städte	Preis
A ↔ B	300 €
A ↔ F	1000 €
B ↔ C	1000 €
B ↔ D	700 €
B ↔ E	300 €
B ↔ G	1000 €
C ↔ F	200 €
D ↔ F	400 €
E ↔ F	300 €
F ↔ G	300 €

Der Preis p in einer Zeile

Städte	Preis
x ↔ y	p

gilt dabei sowohl für einen einfachen Flug von x nach y als auch für einen einfachen Flug von y nach x. Bestimmen Sie mit dem Algorithmus von Dijkstra (führen Sie den Algorithmus händisch durch!) die Routen und die Preise für die preiswertesten Flugverbindungen von der Stadt A zu jeder der anderen Städte.

Als TikZ-Umgebung



Schritt	besuchte Knoten	A	B	C	D	E	F	G
Init		0	∞	∞	∞	∞	∞	∞
1	**A**	0	**300,A**	∞	∞	∞	1000,F	∞
2	A,**B**	0	|	1300,B	1000,B	**600,B**	1000,F	1300,B
3	A,B,**E**	0	|	1300,B	1000,B	|	**900,E**	1300,B
4	A,B,E,**F**	0	|	1100,F	**1000,B**	|	|	1200,F
5	A,B,E,F,**D**	0	|	**1100,F**	|	|	|	1200,F
6	A,B,E,F,D,**C**	0	|	|	|	|	|	**1200,F**
7	A,B,E,F,D,C,**G**	0	|	|	|	|	|	|

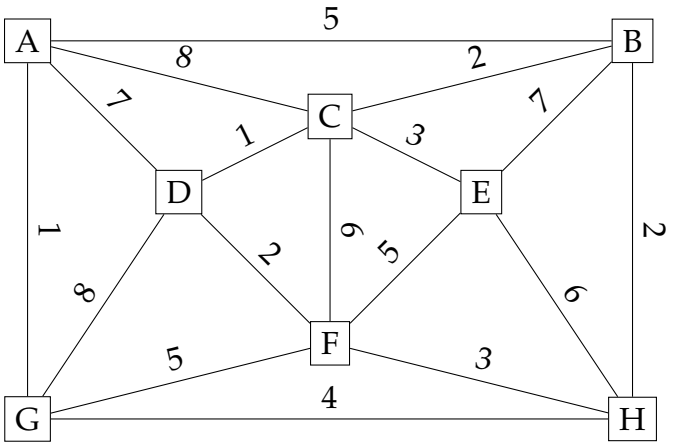
Städte	Preis
A → B	300
A → B → E → F → C	1100
A → B → D	1000
A → B → E	600
A → B → E → F	900
A → B → E → F → G	1200

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/90_Graphen/10_Dijkstra/Aufgabe_Staedte-A-F.tex`

Übungsaufgabe „Minimaler Spannbaum A-H“ (Minimaler Spannbaum, Algorithmus von Prim)

Minimaler Spannbaum

Ermitteln Sie einen minimalen Spannbaum des vorliegenden Graphen. Nutzen Sie den *Knoten A als Startknoten* in ihrem Algorithmus.



(a) Welches Gewicht hat der Spannbaum insgesamt?

Lösungsvorschlag

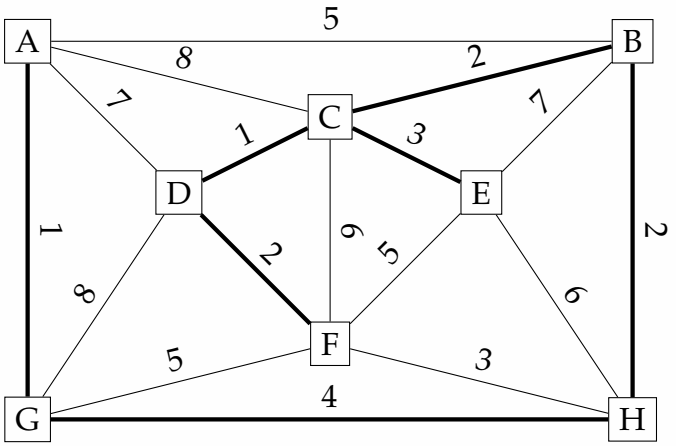
Das Kantengewicht des minimalen Spannbaums beträgt 15.

Wir setzen den Algorithmus von Prim ein. Der Algorithmus läuft folgendermaßen ab:

```
Besuche Knoten „A"
Füge Kante (A, B, 5) hinzu
Füge Kante (A, C, 8) hinzu
Füge Kante (A, D, 7) hinzu
Füge Kante (A, G, 1) hinzu

Besuche Knoten „G"
Füge Kante (G, F, 5) hinzu
Füge Kante (G, H, 4) hinzu

Besuche Knoten „H"
Aktualisiere Kante (H, B, 2)
Füge Kante (H, E, 6) hinzu
Aktualisiere Kante (H, F, 3)

Besuche Knoten „B"
Aktualisiere Kante (B, C, 2)

Besuche Knoten „C"
Aktualisiere Kante (C, D, 1)
Aktualisiere Kante (C, E, 3)
```

```
Besuche Knoten „D"
Aktualisiere Kante (D, F, 2)

Besuche Knoten „F"

Besuche Knoten „E"
```

schwarze	**graue**
(A, null, 0)	(B, A, 5); (C, A, 8); (D, A, 7); (G, A, 1);
(G, A, 1)	(B, A, 5); (C, A, 8); (D, A, 7); (F, G, 5); (H, G, 4);
(H, G, 4)	(B, H, 2); (C, A, 8); (D, A, 7); (E, H, 6); (F, H, 3);
(B, H, 2)	(C, B, 2); (D, A, 7); (E, H, 6); (F, H, 3);
(C, B, 2)	(D, C, 1); (E, C, 3); (F, H, 3);
(D, C, 1)	(E, C, 3); (F, D, 2);
(F, D, 2)	(E, C, 3);
(E, C, 3)	



(b) Welchen Algorithmus haben Sie zur Ermittlung eingesetzt?

Lösungsvorschlag

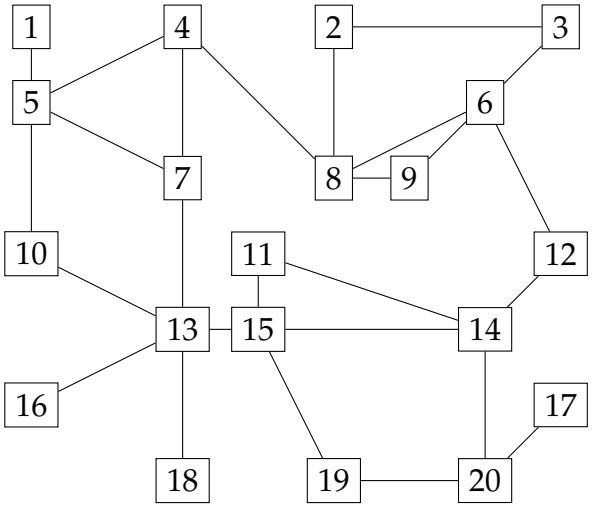
Algorithmus von Prim. Dieser Algorithmus benötigt einen Startknoten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/90_Graphen/20_Spannbaume/Aufgabe_Minimaler-Spannbaum-A-H.tex

Übungsaufgabe „Knoten-1-20“ (Breitensuche, Tiefensuche)

Breitensuche

(a) Geben Sie die Reihenfolge an, in der die Knoten besucht werden, wenn auf dem folgenden Graphen *Breitensuche* ausgehend von Knoten 1 ausgeführt wird. Wenn mehrere Knoten zur Wahl stehen, wählen Sie den Knoten mit dem kleinsten Schlüssel.



Lösungsvorschlag

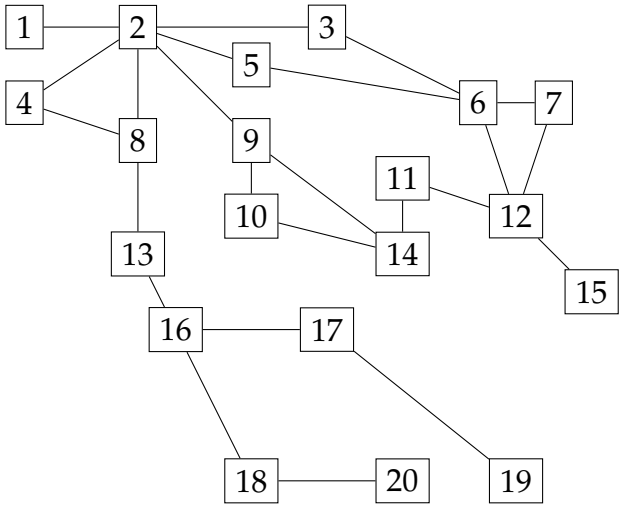
```
        add 1  [1]
del 1
        add 5  [5]
del 5
        add 4  [4]
        add 7  [4, 7]
        add 10 [4, 7, 10]
del 4
        add 8  [7, 10, 8]
del 7
        add 13 [10, 8, 13]
del 10
del 8
        add 2  [13, 2]
        add 6  [13, 2, 6]
        add 9  [13, 2, 6, 9]
del 13
        add 15 [2, 6, 9, 15]
        add 16 [2, 6, 9, 15, 16]
        add 18 [2, 6, 9, 15, 16, 18]
del 2
        add 3  [6, 9, 15, 16, 18, 3]
del 6
        add 12 [9, 15, 16, 18, 3, 12]
del 9
del 15
        add 11 [16, 18, 3, 12, 11]
        add 14 [16, 18, 3, 12, 11, 14]
        add 19 [16, 18, 3, 12, 11, 14, 19]
del 16
```

Tiefensuche

```
del 18
del 3
del 12
del 11
del 14
        add 20 [19, 20]
del 19
del 20
        add 17 [17]
del 17
```

Reihenfolge: 1, 5, 4, 7, 10, 8, 13, 2, 6, 9, 15, 16, 18, 3, 12, 11, 14, 19, 20, 17

(b) Geben Sie die Reihenfolge an, in der die Knoten besucht werden, wenn auf dem folgenden Graphen *Tiefensuche* ausgehend vom Knoten 1 ausgeführt wird. Wenn mehrere Knoten zur Wahl stehen, wählen Sie den Knoten mit dem kleinsten Schlüssel.



Lösungsvorschlag

Rekursive Tiefensuche:

```
add 1
add 2
add 3
add 6
add 5
        exit 5
add 7
add 12
add 11
add 14
add 9
add 10
        exit 10
        exit 9
        exit 14
        exit 11
```

```
add 15
        exit 15
        exit 12
        exit 7
        exit 6
        exit 3
add 4
add 8
add 13
add 16
add 17
add 19
        exit 19
        exit 17
add 18
add 20
        exit 20
        exit 18
        exit 16
        exit 13
        exit 8
        exit 4
        exit 2
        exit 1
```

Reihenfolge: 1, 2, 3, 6, 5, 7, 12, 11, 14, 9, 10, 15, 4, 8, 13, 16, 17, 19, 18, 20

Mit Stapel

```
        add 1  [1]
del 1
        add 2  [2]
del 2
        add 3  [3]
        add 4  [4, 3]
        add 5  [5, 4, 3]
        add 8  [8, 5, 4, 3]
        add 9  [9, 8, 5, 4, 3]
del 9
        add 10 [10, 8, 5, 4, 3]
        add 14 [14, 10, 8, 5, 4, 3]
del 14
        add 11 [11, 10, 8, 5, 4, 3]
del 11
        add 12 [12, 10, 8, 5, 4, 3]
del 12
        add 6  [6, 10, 8, 5, 4, 3]
        add 7  [7, 6, 10, 8, 5, 4, 3]
        add 15 [15, 7, 6, 10, 8, 5, 4, 3]
del 15
del 7
del 6
del 10
```

```
del 8
        add 13 [13, 5, 4, 3]
del 13
        add 16 [16, 5, 4, 3]
del 16
        add 17 [17, 5, 4, 3]
        add 18 [18, 17, 5, 4, 3]
del 18
        add 20 [20, 17, 5, 4, 3]
del 20
del 17
        add 19 [19, 5, 4, 3]
del 19
del 5
del 4
del 3
```

Reihenfolge: 1, 2, 3, 4, 5, 8, 9, 10, 14, 11, 12, 6, 7, 15, 13, 16, 17, 18, 20, 19

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/90_Graphen/30_Tiefen-Breitensuche/Aufgabe_Knoten-1-20.tex

Sonstige

Examensaufgabe „Algorithmenanalyse“ (66115-2020-H.T1-TA2-A1)

Betrachten Sie die folgende Prozedur `countup`, die aus zwei ganzzahligen Eingabewerten `n` und `m` einen ganzzahligen Ausgabewert berechnet:

```
procedure countup(n, m : integer): integer
var x, y : integer;
begin
  x := n;
  y := 0;
  while (y < m) do
    x := x - 1;
    y := y + 1;
  end while
  return x;
end
```

(a) Führen Sie `countup(3,2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen `n`, `m`, `x` und `y` zu Beginn der while-Schleife und den Rückgabewert der Prozedur an.

Lösungsvorschlag

n	m	x	y	ausgeführter Code, der Änderung bewirkte
3	2	3	0	
3	2	2	1	`x := x - 1; y := y + 1;`

Rückgabewert: 1

Java-Implementation der Prozedur

```
public class CountUp {

  public static int countup(int n, int m) {
    int x = n;
    int y = 0;
    while (y < m) {
      System.out.println(String.format("%s %s %s %s", n, m, x, y));
      x = x - 1;
      y = y + 1;
    }
    return x;
  }

  public static void main(String[] args) {
    System.out.println(countup(3, 2));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/counter/CountUp.java`

(b) Gibt es Eingabewerte von `n` und `m`, für die die Prozedur `countup` nicht terminiert? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. Mit jedem Schleifendurchlauf wird der Wert der Variablen `y` um eins hochgezählt. Die Werte, die `y` annimmt, sind streng monoton steigend. `y` nähert sich `m` an, bis `y` nicht mehr kleiner ist als `m` und die Prozedur terminiert. An diesem Sachverhält ändern auch sehr große Zahlen, die über die Variable `m` der Prozedur übergeben werden, nichts.

(c) Geben Sie die asymptotische worst-case Laufzeit der Prozedur `countup` in der Θ-Notation in Abhängigkeit von den Eingabewerten `n` und/oder `m` an. Begründen Sie Ihre Antwort.

Lösungsvorschlag

Die Laufzeit der Prozedur ist immer $\Theta(m)$. Die Laufzeit hängt nur von `m` ab. Es kann nicht zwischen best-, average and worst-case unterschieden werden.

(d) Betrachten Sie nun die folgende Prozedur `countdown`, die aus zwei ganzzahligen Eingabewerten `n` und `m` einen ganzzahligen Ausgabewert berechnet:

```
procedure countdown(n, m : integer) : integer
var x, y : integer;
begin
  x := n;
  y := 0;
  while (n > 0) do
    if (y < m) then
      x := x - 1;
      y := y + 1;
    else
      y := 0;
      n := n / 2; /* Ganzzahldivision */
    end if
  end while
  return x;
end
```

Führen Sie `countdown(3, 2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen `n`, `m`, `x` und `y` zu Beginn der `while`-Schleife und den Rückgabewert der Prozedur an.

Lösungsvorschlag

n	m	x	y	ausgeführter Code, der Änderung bewirkte
3	2	3	0	
3	2	2	1	`x := x - 1; y := y + 1;`
3	2	1	2	`x := x - 1; y := y + 1;`
1	2	1	0	`y := 0; n := n / 2;`
1	2	0	1	`x := x - 1; y := y + 1;`
1	2	-1	2	`x := x - 1; y := y + 1;`

Rückgabewert: -1

Java-Implementation der Prozedur

```java
public class CountDown {

  public static int countdown(int n, int m) {
    int x = n;
    int y = 0;
    while (n > 0) {
      System.out.println(String.format("%s %s %s %s", n, m, x, y));
      if (y < m) {
        x = x - 1;
        y = y + 1;
      } else {
        y = 0;
        n = n / 2; /* Ganzzahldivision */
      }
    }
    return x;
  }

  public static void main(String[] args) {
    System.out.println(countdown(3, 2));
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/counter/CountDown.java`

(e) Gibt es Eingabewerte von `n` und `m`, für die die Prozedur `countdown` nicht terminiert? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein.

`n <= 0` terminiert sofort

`m <= 0` Der Falsch-Block der Wenn-Dann-Bedingung erniedrigt `n n := n / 2;` bis `0` erreicht ist. Dann terminiert die Prozedur.

`m > 0` Der erste Wahr-Block der Wenn-Dann-Bedingung erhöht `y` streng monoton bis `y >= m`. 2. Falsch-Block der Wenn-Dann-Bedingung halbiert `n` bis `0`. 1. und 2. solange bis $n = 0$

(f) Geben Sie die asymptotische Laufzeit der Prozedur `countdown` in der Θ-Notation in Abhängigkeit von den Eingabewerten `n` und/oder `m` an unter der Annahme, dass `m >= 0` und `n > 0`. Begründen Sie Ihre Antwort.

Lösungsvorschlag

Anzahl der Wiederholungen der while-Schleife: $m + 1$:

- m oft: bis $y < m$
- $+1$ Halbierung von n und y auf 0 setzen

wegen dem `n/2` ist die Laufzeit logarithmisch, ähnlich wie der worst case bei der Binären Suche.

n	m	x	y	ausgeführter Code, der Änderung bewirkte
16	3	16	0	
16	3	15	1	
16	3	14	2	
16	3	13	3	
8	3	13	0	`y := 0; n := n / 2;`
8	3	12	1	
8	3	11	2	
8	3	10	3	
4	3	10	0	`y := 0; n := n / 2;`
4	3	9	1	
4	3	8	2	
4	3	7	3	
2	3	7	0	`y := 0; n := n / 2;`
2	3	6	1	
2	3	5	2	
2	3	4	3	
1	3	4	0	`y := 0; n := n / 2;`
1	3	3	1	
1	3	2	2	
1	3	1	3	

$\Theta((m+1)\log_2 n)$

Wegkürzen der Konstanten:

$\Rightarrow \Theta(m \log n)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Teil III

Softwaresysteme (SOSY)

Projektmanagement

Examensaufgabe „modernen Softwaretechnologie. 3 Begriffe in 3 Sätzen“ (46116-2013-F.T1-TA1-A2)

Erläutern Sie in jeweils ca. 3 Sätzen die folgenden Begriffe aus der modernen Softwaretechnologie: *Pair Programming*, *Refactoring*, *agile Softwareentwicklung*.

Lösungsvorschlag

Pair Programming Beim Pair Programming entwickeln zwei Personen die Software zusammen. Es gibt einen sogenannten Driver - die Person, die tippt - und einen sogenannten Navigator - die Person, die kritisch den Code überprüft. Das Pair Programming ist wichtiger in agilen Entwicklungsmethoden wie z. B dem Exterme Programming (XP).

Refactoring Unter Refactoring versteht man die Verbesserung der Code- und Systemstruktur mit dem Ziel einer besseren Wartbarkeit. Dabei sollen Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken. Im Refactoring werden keine neuen Funktionalitäten programmiert.

agile Softwareentwicklung Agile Entwicklungsprozesse gehen auf das Agile Manifest, das im Jahr 2001 veröffentlicht wurde, zurück. Dieses Agile Manifest bildet die Basis für agile Entwicklungsprozesse wie eXtreme Programming oder SCRUM. Das Manifest beinhaltet 12 Grundprinzipien für die moderne agile Software-Entwicklung. Hauptzielsetzung ist, rasch eine funktionsfähige Software-Lösung auszuliefern, um den Kundenwunsch bestmöglich zu erfüllen. Dabei spielt die Interaktion mit dem Kunden eine zentrale Rolle.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2013/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Multiple-Choice: Allgemeine SWT, Vorgehensmodelle und Requirements“ (46116-2014-H.T2-TA1-A1)

EXtreme Programming
V-Modell
SCRUM

Aufgabe 1: Allgemeine SWT, Vorgehensmodelle und Requirements Engineering

Kreuzen Sie für die folgenden Multiple-Choice-Fragen genau die richtigen Antworten deutlich an. Es kann mehr als eine Antwort richtig sein.

Jedes korrekt gesetzte oder korrekt nicht gesetzte Kreuz wird mit 1 Punkt gewertet. Jedes falsch gesetzte oder falsch nicht gesetzte Kreuz wird mit -1 Punkt gewertet. Eine Frage kann entwertet werden, dann wird sie nicht in der Korrektur berücksichtigt. Einzelne Antworten können nicht entwertet werden. Entwerten Sie eine Frage wie folgt

Die gesamte Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

(a) Welche Aussage ist wahr?

- ☐ Je früher ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Je später ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Der Zeitpunkt der Entdeckung hat keinen Einfluss auf die Kosten.

Lösungsvorschlag

2 ist richtig: Je später der Fehler entdeckt wird, desto mehr wurde er schon in das Projekt „eingearbeitet“, daher dauert das Beseitigen des Fehlers länger und das kostet mehr Geld.

(b) Mit welcher Methodik können Funktionen spezifiziert werden?

- ☐ Als Funktionsvereinbarung in einer Programmiersprache
- ☐ Mit den Vor- und Nachbedingungen von Kontrakten
- ☐ Als Zustandsautomaten

Lösungsvorschlag

2 und 3 ist richtig: Die Spezifikation soll unabhängig von einer Programmiersprache sein.

(c) Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen geeignet?

- ☐ Extreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Scrum

Lösungsvorschlag

Prototyping
Funktionale Anforderungen
Nicht-funktionale Anforderungen

1 und 3 ist richtig. Das V-Modell ist ein starres Vorgehensmodell, bei dem alle Anforderungen zu Beginn vorhanden sein müssen.

(d) Welche der folgenden Aussagen ist korrekt?

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

Lösungsvorschlag

2 und 3 ist richtig: Prototypen müssen auch getestet werden. Es kann nicht an Tests gespart werden. Durch das häufige Feedback des Kunden / der Stakeholder können die Anforderungen immer genauer und klarer erfasst werden.

(e) Welche der folgenden Aussagen ist korrekt?

- ☐ Bei der Architektur sollten funktionale und nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur soliten nur funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollten nur nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollte auf die mögliche Änderungen von Komponenten geachtet werden.

Lösungsvorschlag

1 und 4 ist richtig: Mögliche Änderungen werden durch klar definierte Schnittstellen und wenig Kopplung der Komponenten erleichtert. (Kopplung handelt von Abhängigkeiten zwischen Modulen. Kohäsion handelt von Abhängigkeiten zwischen Funktionen innerhalb eines Moduls.)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/09/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Vorgehensmodelle“ (46116-2014-H.T2-TA1-A2)

Prozessmodelle
Wasserfallmodell

Software wird oft in definierten Prozessen entwickelt. Diese nennt man Vorgehensmodelle.

Allgemein

(a) Was sind die Aufgaben eines Vorgehensmodells im Allgemeinen?

Lösungsvorschlag

- liefert Erfahrungen und bewährte Methoden
- beschreibt die am Projekt beteiligten Rollen
- legt Aufgaben und Aktivitäten fest
- definiert einheitliche Begriffe
- gibt Techniken, Werkzeuge, Richtlinien / Standards an

(b) Was sind die wesentlichen Bestandteile eines Vorgehensmodells und in welcher Beziehung stehen diese zueinander?

Lösungsvorschlag

Bestandteile: Anforderungsanalyse, Modellierung, Implementierung, Test, Auslieferung, Wartung

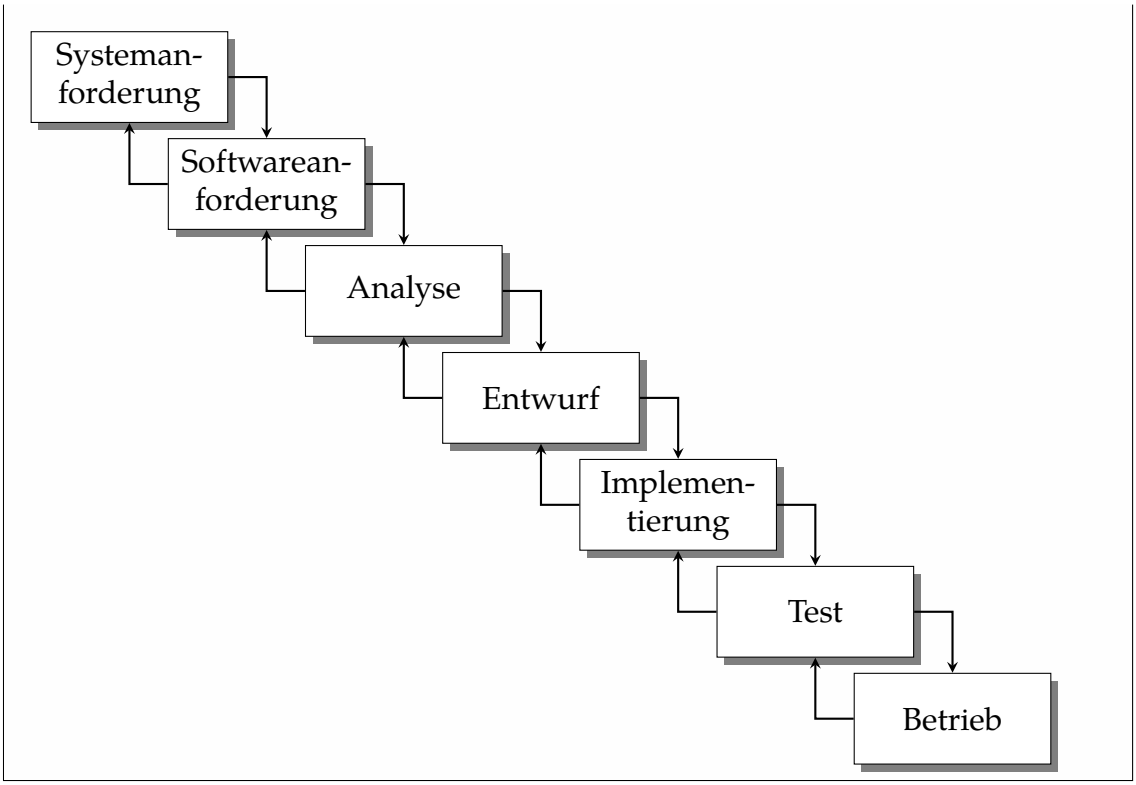
Die einzelnen Phasen bauen immer aufeinander aus. Je nach Vorgehensmodell können sich Phasen auch wiederholen (Prototyping, Scrum...).

Ein frühes Vorgehensmodell ist das von Dr. Winston Royce 1970 formalisierte Wasserfallmodell.

(a) Geben Sie eine schematische Darstellung des Wasserfallmodells an.

Lösungsvorschlag

V-Modell



(b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

Fehler werden ggf. erst am Ende des Entwicklungsprozesses erkannt, da erst dort das Testen stattfindet. Dadurch kann die Behebung eines Fehlers sehr aufwändig und somit teuer werden.

Der Kunde / Endanwender wird erst nach der Implementierung wieder eingebunden. Das bedeutet, dass er nach der Stellung der Anforderungen keinen Einblick mehr in den Prozess hat und somit auch nicht gegensteuern kann, falls ihm etwas nicht gefällt oder er etwas nicht bedacht hat.

(c) In welchen Situationen lässt sich das Wasserfallmodell gut einsetzen?

Lösungsvorschlag

Das Wasserfallmodell ist geeignet, wenn es sich um ein von Anfang an klar definiertes Projekt ohne große Komplexität handelt, bei dem alle Anforderungen, Aufwand und Kosten schon zu Beginn des Projekts feststehen bzw. abgeschätzt werden können.

Barry Boehm erweiterte das Wasserfallmodell 1979 zum so genannten V-Modell.

(a) Geben Sie eine schematische Darstellung des V-Modells an.

(b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

- Die Nachteile des Wasserfallmodells bestehen weiterhin!
- Nicht für kleine Projekte geeignet, da aufwändige Tests vorgesehen sind, die im Kleinen detailliert meist nicht stattfinden (können).

(c) Welchen Vorteil hat das V-Modell gegenüber dem Wasserfallmodell?

Lösungsvorschlag

Für jedes Dokument besteht ein entsprechender Test (Validierung / Verifikation). Dabei kann die Planung der Tests schon vor der eigentlichen Durchführung geschehen, so dass Aktivitäten im Projektteam parallelisiert werden können. So kann zum Beispiel der Tester die Testfälle für den Akzeptanztest (=Test des Systementwurfs) entwicklen, auch wenn noch keine Implementierung existiert.

In neuerer Zeit finden immer häufiger iterative und inkrementelle Vorgehensweisen Anwendung.

(a) Erklären Sie den Begriff iterative Softwareentwicklung.

Lösungsvorschlag

Iterativ heißt, dass der Entwikclungsprozess mehrfach wiederholt wird: statt den „Wasserfall“ einmal zu durchlaufen, werden „kleine Wasserfälle“ hintereinander gesetzt.

(b) Erklären Sie den Begriff inkrementelle Softwareentwicklung und grenzen Sie ihn von iterativer Softwareentwicklung ab.

Lösungsvorschlag

Bei der inkrementellen Entwicklung wird das System Schritt für Schritt fertig gestellt. D. h., dass ein Prototyp immer etwas mehr kann als der Prototyp davor. Dies wird durch die iterative Entwicklung unterstützt, da bei jeder Wiederholung des Entwicklungsprozesses ein neues Inkrement entsteht, d. h. ein neuer Prototyp, der mehr Funktionalitäten benutzt als der vorangegangene.

(c) Nennen Sie jeweils zwei Vor- und Nachteile eines iterativen und inkrementellen Vorgehens im Vergleich zum Wasserfallmodell.

Lösungsvorschlag

Vorteile:

- Risiken können früher erkannt werden.
- volatile Anforderungen können besser berücksichtigt werden.
- inkrementelle Auslieferung wird erleichtert.

Nachteile:

- komplexeres Projektmanagement
- schwerer messbar
- (Mehrarbeit)[a]

[a] Quelle: https://www.pst.ifi.lmu.de/Lehre/WS0607/pm/vorlesung/PM-02-Prozess.pdf

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/09/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Vermischte Softwaresysteme-Fragen“ (66116-2013-H.T1-TA2-A3)

Testen
Model Checking
Refactoring
EXtreme Programming
White-Box-Testing
Black-Box-Testing

(a) Nennen Sie jeweils einen Vorteil und einen Nachteil für Qualitätssicherung durch *„Testing“* bzw. durch *„Model Checking“*.

Lösungsvorschlag

Qualitätssicherung durch „Testing“

Vorteil schnell, Massentests

Nachteil keine 100% Garantie, dass alles getestet ist

Qualitätssicherung durch „Model Checking“

Vorteil mathematischer Beweis

Nachteil teilweise langwierig / nicht möglich

(b) Definieren Sie den Begriff „Refactoring“.

Lösungsvorschlag

Verbesserung der Code-/Systemstruktur mit dem Ziel einer besseren Wartbarkeit

- Dokumentation, Namen, Kommentare
- keine neuen Funktionalitäten
- bessere Struktur, einheitlich, einfacher

(c) Begründen Sie, warum bei der Entwicklung nach der Methode des „eXtreme Programming“ langfristig gesehen Refactorings zwingend notwendig werden.

Lösungsvorschlag

Im „eXtreme Programming“ wird das Projekt kontinuierlich aufgebaut, somit ist ein Refactoring, auch aufgrund des Pair-Programmings, auf lange Sicht gesehen notwendig.

(d) Wie wird in der Praxis während und nach erfolgtem Refactoring sichergestellt, dass keine neuen Defekte eingeführt werden bzw. wurden?

Lösungsvorschlag

Re-testing $\rightarrow$ erneute Verifikation mit Tests nach Refactoring

(e) Worin besteht der Unterschied zwischen „White-Box-Testing“ und „Black-Box-Testing“?

Lösungsvorschlag

Funktionale Anforderungen
Nicht-funktionale Anforderungen
Kontinuierliche Integration (Continuous Integration)
EXtreme Programming

White-Box-Testing: Struktur-Test → Wie funktioniert der Code?

Black-Box-Testing: Funktionstest → Das nach außen sichtbare Verhalten wird getestet.

(f) Nennen Sie vier Qualitätsmerkmale von Software.

Lösungsvorschlag

Änderbarkeit, Wartbarkeit, gute Dokumentation, Effizienz, Funktionalität (→ Korrektheit), Zuverlässigkeit, Portabilität

(g) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

(h) Was verbirgt sich hinter dem Begriff „Continuous Integration” ?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (→ neue Programmversion).

(i) Nennen Sie sechs Herausstellungsmerkmale des *„eXtreme Programming“* Ansatzes.

Lösungsvorschlag

- Werte: Mut, Respekt, Einfachheit, Feedback, Kommunikation
- geringe Bedeutung von formalisiertem Vorgehen, Lösen der Programmieraufgabe im Vordergrund
- fortlaufende Iterationen
- Teamarbeit und Kommunikation (auch mit Kunden)
- Ziel: Software schneller und mit höherer Qualität bereitstellen, höhere Kundenzufriedenheit
- Continuous Integration und Testing, Prototyping
- Risikoanalysen zur Risikominimierung

- YAGNI (You ain't gonna need it) → nur die Features, die gefordert sind, umsetzen; kein Vielleicht braucht man's... " Unit-Test

(j) Was versteht man unter einem Unit-Test? Begründen Sie, warum es unzureichend ist, wenn eine Test-Suite ausschließlich Unit-Tests enthält.

Lösungsvorschlag

Unter einem Unit-Test versteht man den Test eines einzelnen Software-Moduls oder auch nur einer Methode. Dies ist allein nicht ausreichend, da man so nichts über das Zusammenspiel der Module aussagen kann.

(k) Nennen Sie jeweils eine Methodik, mit welcher in der Praxis die Prozesse der *„Validierung"* und der *„Verifikation"* durchgeführt werden.

Lösungsvorschlag

Methodik der Verifikation: Testen, wp-Kalkül, Model Checking

Methodik der Validierung: Kundentest, Kundengespräch (Spezifiaktion durchsprechen)

(l) Grenzen Sie die Begriffe *„Fault"* und *„Failure"* voneinander ab.

Lösungsvorschlag

Fault: interner Fehlerzustand, der nach außen nicht nicht sichtbar werden muss, aber kann.

Failure: Systemfehler / Fehlerzustand, der nach außen sichtbar wird.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2013/09/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Softwaresysteme: Begriffe und Konzepte“ (66116-2016-H.T1-TA2-A1)

Pflichtenheft Softwaremaße

(a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

Lösungsvorschlag

Im Rahmen der Zielbestimmung werden die Ziele des Produktes in einer Art Prioritätenliste exakt eingegrenzt, die in drei weitere Kategorien gegliedert ist, die als **Muss-**, **Wunsch-** und **Abgrenzungskriterien** bezeichnet werden.

Musskriterien Zu den Musskriterien zählen die Produktfunktionen, die für ein Funktionieren des Produktes unabdingbar sind. Ohne ihre Umsetzung geht einfach nichts, deshalb müssen sie erfüllt werden.

Wunschkriterien Die nächste Kategorie, die Wunschkriterien, sind zwar entbehrlich, wurden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also genauso eine Pflicht, wie die der ersten Kategorie, das Produkt würde aber auch ohne ihre Implementierung seinen Betrieb korrekt ausführen.

Abgrenzungskriterien Die letzte Kategorie beschreiben die Abgrenzungskriterien. Sie sollen die Grenzen des Produktes klarmachen und sind von daher beinahe wichtiger als die beiden ersten Fälle denn sie hindern die Entwickler daran, über alle Ziele hinauszuschießen.

`https://st.inf.tu-dresden.de/SalesPoint/v3.2/tutorial/stepbystep2/pflh.html`

(b) Nennen Sie drei Einsatzziele von Softwaremaßen.

Lösungsvorschlag

Aus Baumann K. (1997) Softwaremaße. In: Unterstützung der objektorientierten Systemanalyse durch Softwaremaße. Beiträge zur Wirtschaftsinformatik, vol 23. Physica, Heidelberg. `https://doi.org/10.1007/978-3-662-13273-9_2`

Der Einsatz von Softwaremaßen kann vielfältige Vorteile mit sich bringen. Folgende Ziele können mit der Anwendung von Softwaremaßen verfolgt werden:

- Durch eine Bewertung kann überprüft werden, inwieweit einmal gestellte *Qualitätsanforderungen* erfüllt wurden.
- Eine Bewertung bereits erstellter oder noch zu erstellender Softwareprodukte kann Hilfestellung bei Entscheidungen, die für die *Projektplanung* oder das Entwicklungsmanagement zu treffen sind, leisten. So können Softwaremaße zur Einschätzung des notwendigen Aufwands, der zu erbringenden Kosten und des Personalbedarfs herangezogen werden.

- Ein möglichst frühzeitiges *Aufzeigen von Schwachstellen im Systemdesign*, die Beurteilung der Auswirkungen neuer Techniken oder Tools auf die Produktivität der Entwickler oder auf die Qualität des entwickelten Produkts sowie die Überprüfung der Einhaltung festgelegter Designstandards sind weitere mögliche Einsatzfelder für Softwaremaße.
- Darüber hinaus ist der Einsatz von Softwaremaßen *Teil umfassender Konzepte zum Softwarequalitätsmanagement*. Softwaremaße bzw. die zugehörige Meßergebnisse tragen zunächst dazu bei, die betrachteten Produkte oder Prozesse besser zu verstehen oder hinsichtlich interessierender Eigenschaften zu bewerten.

`https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9_2`

(c) Welche Arten von Softwaremaßen werden unterschieden?

Lösungsvorschlag

Die Unterscheidung nach dem Messgegenstand ergibt folgende Arten von Maßen:

Externe Produktmaße. Diese beschreiben Merkmale des Produkts, die von außen sichtbar sind, wenngleich eventuell nur indirekt. Dazu gehören insbesondere Maße für Qualitätsattribute wie die Wartbarkeit, die Zuverlässigkeit, die Benutzbarkeit, die Effizienz etc.

Interne Produktmaße. Basis für die Charakterisierung externer Produktattribute sind interne Attribute, die sich zumindest zum Teil besser messen lassen. Hierzu gehören z. B. die Größe, die Modularität und die Komplexität.

Prozessmaße. Diese charakterisieren Eigenschaften des Produktionsprozesses, z. B. die typische Produktivität, Kostenaufteilung auf bestimmte Arbeiten, Dauer von Arbeitsschritten etc.

Ressourcenmaße. Diese charakterisieren Eigenschaften der im Prozess verwendeten Ressourcen, z. B. die Auslastung von Rechnern, die typische Produktivität und Fehlerrate eines bestimmten Ingenieurs etc.

`http://page.mi.fu-berlin.de/prechelt/swt2/node5.html`

Was sind die 3 Arten der (einfachen) Maße zur Messung von Software?

https://quizlet.com/de/339799466/softwaretechnik-theorie-flash-cards/

- Größenorientiert/nach Größe
- Funktionsorientiert/nach Funktion
- Objektorientiert/nach Objekt

(d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

Lösungsvorschlag

Wie für jede Form der Software-Entwicklung stellt sich die Frage nach einem geeigneten methodischen Ansatz. Generell lässt sich der Prozess der Software-Entwicklung in die folgenden drei Abschnitte gliedern:

(i) von der Problembeschreibung bis zur Software-Spezifikation (auch Problemanalyse, Systemanalyse, Requirements Engineering bzw. frühe Phasen der Software-Entwicklung ge- nannt),

(ii) die Software-Entwicklung im engeren (technischen) Sinn,

(iii) die Integration der Software in den Anwendungszusammenhang.

Diese Einteilung gilt ebenso für die evolutionäre Software-Entwicklung, allerdings wird hier der Schwerpunkt anders gelegt. Steht beim klassischen Ansatz in der Regel die Software-Entwicklung im engeren Sinn im Mittelpunkt, so werden beim evolutionären Ansatz alle drei Abschnitte möglichst gleichgewichtig und im Zusammenhang betrachtet. Daraus resultiert eine natürliche Tendenz zu einer zyklischen Vorgehensweise.

Daneben gibt es weitere Querbezüge: Orientiert sich die Software-Entwicklung an einem festen Ziel, verfolgt dieses jedoch durch schrittweises Erweitern zunächst unvollkommener Teillösungen, so wird dieses Vorgehen inkrementell genannt. Unfertige Versuchsanordnungen oder Teilsysteme, die am Beginn einer solchen Entwicklung stehen, werden oft als Prototypen bezeichnet. Wählt man Prototyping als Vorgehensweise, so kann entweder das spätere Produkt inkrementell aus einem oder mehreren Prototypen weiterentwickelt werden oder man setzt nach einer (Wegwerf-) Prototypphase neu auf. Beides sind Formen evolutionärer Entwicklung.

`http://waste.informatik.hu-berlin.de/~dahme/edidahm.pdf`

(e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

Lösungsvorschlag

(i) Möglichkeit, auf ältere Entwicklungsversionen zurückzukehren, wenn sich die aktuelle als nicht lauffähig bzw. unsicher erwiesen hat (z. B. git revert)

(ii) Möglichkeit, dass mehrere Entwickler gleichzeitig an ein- und denselben Softwareprojekt arbeiten und Möglichkeit, dass ihre Entwicklungen durch das Versionsverwaltungssystem zusammengeführt werden können (z. B. git merge)

(iii) Möglichkeit, den Zeitpunkt oder den Urheber eines Softwarefehlers ausfindig zumachen (z. B. git blame)

(f) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Kontinuierliche Integration (Continuous Integration)

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

(g) Was verbirgt sich hinter dem Begriff *„Continuous Integration“*?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (*rightarrow* neue Programmversion).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Software Engineering
Agile Methoden
Spiralmodell
Nicht-funktionale Anforderungen
Entwurfsmuster
Schichtenarchitektur
Blackboard-Muster
Einbringen von Abhängigkeiten (Dependency Injection)

Examensaufgabe „Wahrheitsgehalt-Tabelle Software Engineering“ (66116, 2016-H.T2-TA2-A1)

Ordnen Sie die folgenden Aussagen entsprechend ihres Wahrheitsgehaltes in einer Tabelle der folgenden Form an:

Kategorie	WAHR	FALSCH
X	X1, X3	X2
Y	Y2	Y1
...	...	...

A Allgemein

- **A1** Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
- **A2** Software Engineering ist gleichbedeutend mit Programmieren.

B Vorgehensmodelle

- **B1** Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
- **B2** Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
- **B3** Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.

C Anforderungserhebung

- **C1** Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zu, Vorschein kommen könnten.
- **C2** Ein Szenario beinhaltet eine Menge von Anwendungsfällen.
- **C3** Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

D Architekturmuster

- **D1** Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
- **D2** Das Black Board Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
- **D3** „Dependency Injection“ bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.

E UML

Sequenzdiagramm
Zustandsdiagramm Wissen
Komponentendiagramm
Modell-Präsentation-Steuerung (Model-View-Controller)
Einzelstück (Singleton)
Kommando (Command)
Validation
Verifikation

E1 Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.

E2 Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.

E3 Komponentendiagramme beschreiben die Struktur eines Systems.

F Entwurfsmuster

F1 Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.

F2 Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.

F3 Im Kommando Enwurfsmuster (engl. „Command Pattern“) werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.

G Testen

G1 Validation dient der Überprüfung von Laufzeitfehlern.

G2 Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.

G3 Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Lösungsvorschlag

Kategorie	WAHR	FALSCH
A	A1	A2
B	B3	B1, B2
C	C3	C1, C2
D	D3	D1, D2
E	E1, E2, E3	
F	F2, F3	F1
G	G3	G1[a], G2[b]

[a] Validierung: Prüfung der Eignung beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck: „Wird das richtige Produkt entwickelt?“

[b] Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Lebenszyklus“ (66116-2020-H.T1-TA1-A5)

Projektplanung
SCRUM

(a) Nennen Sie fünf kritische Faktoren, die bei der Auswahl eines Vorgehensmodells helfen können und ordnen Sie plangetriebene und agile Prozesse entsprechend ein.

Lösungsvorschlag

(i) Vollständigkeit der Anforderungen
- bei vollständiger Kenntnis der Anforderungen: *plangetrieben*
- bei teilweiser Kenntnis der Anforderungen: *agil*

(ii) Möglichkeit der Rücksprache mit dem Kunden
- keine Möglichkeit: *plangetrieben*
- Kunde ist partiell immer wieder involviert: *agil*

(iii) Teamgröße
- kleine Teams (max. 10 Personen): *agil*
- größere Teams: *plangetrieben*

(iv) Bisherige Arbeitsweise des Teams
- bisher feste Vorgehensmodelle: *plangetrieben*
- flexible Arbeitsweisen: *agil*

(v) Verfügbare Zeit
- kurze Zeitvorgabe: *plangetrieben*
- möglichst schnell funktionierender Prototyp verlangt: *agil*
- beide Vorgehensmodelle sind allerdings zeitlich festgelegt

Mögliche weitere Faktoren: Projektkomplexität, Dokumentation

(b) Nennen und beschreiben Sie kurz die Rollen im Scrum.

Lösungsvorschlag

Product Owner Der Product Owner ist für die Eigenschaften und den wirtschaftlichen Erfolg des Produkts verantwortlich.

Entwickler Die Entwickler sind für die Lieferung der Produktfunktionalitäten in der vom Product Owner gewünschten Reihenfolge verantwortlich.

Scrum Master Der Scrum Master ist dafür verantwortlich, dass Scrum als Rahmenwerk gelingt. Dazu arbeitet er mit dem Entwicklungsteam zusammen, gehört aber selbst nicht dazu.

(c) Nennen und beschreiben Sie drei Scrum Artefakte. Nennen Sie die verantwortliche Rolle für jedes Artefakt.

Lösungsvorschlag

Product Backlog Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

Sprint Backlog Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

Product Increment Das Inkrement ist die Summe aller Product-Backlog-Einträge, die während des aktuellen und allen vorangegangenen Sprints fertiggestellt wurden.

(d) Beschreiben Sie kurz, was ein Sprint ist. Wie lange sollte ein Sprint maximal dauern?

Lösungsvorschlag

Ein Sprint ist ein Arbeitsabschnitt, in dem ein Inkrement einer Produktfunktionalität implementiert wird. Ein Sprint umfasst ein Zeitfenster von ein bis vier Wochen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-5.tex`

Examensaufgabe „Wissensfragen“ (66116-2020-H.T2-TA1-A1)

Wasserfallmodell
Prozessmodelle
Kontinuierliche Integration (Continuous Integration)
EXtreme Programming
Softwaremaße

(a) Nennen Sie fünf Phasen, die im Wasserfallmodell durchlaufen werden sowie deren jeweiliges Ziel bzw. Ergebnis(-dokument).

Lösungsvorschlag

Anforderung Lasten- und Pflichtenheft

Entwurf Softwarearchitektur in Form von Struktogrammen, UML-Diagrammen etc.

Implementierung Software

Überprüfung überarbeitete Software

Einsatz und Wartung erneut überarbeitete Software, verbesserte Wartung

(b) Nennen Sie drei Arten der Softwarewartung und geben Sie jeweils eine kurze Beschreibung an.

Lösungsvorschlag

korrektive Wartung Korrektur von Fehlern, die schon beim Kunden in Erscheinung getreten sind

präventive Wartung Korrektur von Fehlern, die beim Kunden noch nicht in Erscheinung getreten sind

adaptive Wartung Anpassung der Software an neue Anforderungen

perfektionierende Wartung Verbesserung von Performance und Wartbarkeit und Behebung von technical depts[a]

[a] `https://de.wikipedia.org/wiki/Softwarewartung`

(c) Eine grundlegende Komponente des *Extreme Programming* ist *„Continuous Integration“*. Erklären Sie diesen Begriff und warum man davon einen Vorteil erwartet.

Lösungsvorschlag

Die Software wird nach jeder Änderung (push) automatisch kompiliert und auch getestet. Man erwartet sich davon einen Vorteil, weil Fehler schneller erkannt werden und die aktuelle Version des Systems ständig verfügbar ist.

(d) Nennen Sie zwei Softwaremetriken und geben Sie jeweils einen Vor- und Nachteil an, der im Projektmanagement von Bedeutung ist.

Lösungsvorschlag

Anzahl der Code-Zeilen

Vorteil: Ist sehr einfach zu bestimmen.

Nachteil: Entwickler können die Zeilenzahl manipulieren (zusätzliche Leerzeilen, Zeilen aufteilen), ohne dass sich die Qualität des Codes verän-

dert.

Zyklomatische Komplexität

Vorteil: Trifft eine Aussage über die Qualität des Algorithmus.

Nachteil: Ist für Menschen nicht intuitiv zu erfassen. Zwei Codes, die dasselbe Problem lösen können die gleiche Zyklomatische Komplexität haben, obwohl der eine wesentlich schlechter zu verstehen ist (Spaghetticode!).

(e) Nennen und beschreiben Sie kurz drei wichtige Aktivitäten, welche innerhalb einer Sprint-Iteration von Scrum durchlaufen werden.

Lösungsvorschlag

Sprint Planning Im Sprint Planning werden zwei Fragen beantwortet:

- Was kann im kommenden Sprint entwickelt werden?
- Wie wird die Arbeit im kommenden Sprint erledigt?

Die Sprint-Planung wird daher häufig in zwei Teile geteilt. Sie dauert in Summe maximal 2 Stunden je Sprint-Woche, beispielsweise maximal acht Stunden für einen 4-Wochen-Sprint.

Daily Scrum Zu Beginn eines jeden Arbeitstages trifft sich das Entwicklerteam zu einem max. 15-minütigen Daily Scrum. Zweck des Daily Scrum ist der Informationsaustausch.

Sprint Review Das Sprint Review steht am Ende des Sprints. Hier überprüft das Scrum-Team das Inkrement, um das Product Backlog bei Bedarf anzupassen. Das Entwicklungsteam präsentiert seine Ergebnisse und es wird überprüft, ob das zu Beginn gesteckte Ziel erreicht wurde.

(f) Erläutern Sie den Unterschied zwischen dem Product-Backlog und dem Sprint-Backlog.

Lösungsvorschlag

Product Backlog Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

Sprint Backlog Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

(g) Erläutern Sie, warum eine agile Entwicklungsmethode nur für kleinere Teams (max. 10 Personen) gut geeignet ist, und zeigen Sie einen Lösungsansatz, wie auch größere Firmen agile Methoden sinnvoll einsetzen können.

Lösungsvorschlag

Bei agilen Entwicklungsmethoden finden daily scrums statt, die in der intendierten Kürze nur in kleinen Teams umsetzbar sind. Außerdem ist in größeren Teams oft eine Hierarchie vorhanden, die eine schnelle Entscheidungsfin-

dung verhindert. Man könnte die große Firma in viele kleine Teams einteilen, die jeweils an kleinen (Teil-)Projekten arbeiten. Eventuell können die product owner der einzelnen Teams nach agilen Methoden zusammenarbeiten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Entwicklungsprozesse“ (66116-2021-F.T2-TA1-A2)

Iterative Prozessmodelle
Inkrementelle Prozessmodelle
Prozessmodelle
Agiles Manifest

(a) Erklären Sie den Unterschied zwischen *iterativen* und *inkrementellen* Entwicklungsprozessen. Nennen Sie zudem je ein Prozessmodell als Beispiel.

Lösungsvorschlag

Iterativ: Ein Entwicklungszyklus wird immer wieder durchlaufen:

Planung → Implementierung → Testung → Evaluation.

Mit jeder Iteration wird das Produkt verfeinert. Das Wasserfallmodell in seiner normalen Form würde dies nicht erfüllen, da hier alle Phasen nur einmal durchlaufen werden.

Beispiel: Agile Programmierung, erweitertes Wasserfallmodell

Inkrementell: Das Projekt wird in einzelne Teile zerlegt. An jedem Teilprojekt kann bestenfalls separat gearbeitet werden. In den einzelnen Teilprojekten kann dann ebenfalls wieder iterativ gearbeitet werden, die beiden Methoden schließen sich gegenseitig also nicht aus.

Beispiel: Agile Programmierung, V-Modell XT, Aufteilung in Teilprojekte, die alle mit Wasserfallmodell bearbeitet werden

(b) Nennen und erklären Sie kurz die vier Leitsätze der agilen Softwareentwicklung laut *Agilem Manifest*.

Lösungsvorschlag

(i) **Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge**

Ein festgelegter Prozess, der nicht sinnvoll von den beteiligten Personen umgesetzt werden kann, ist nicht sinnvoll und sollte nicht durchgeführt werden. Es geht darum, die Stärken der Mitarbeiter einzusetzen und sich nicht sklavisch an Abläufen zu orientieren. Es geht darum, Freiräume zu schaffen, um das Potential des Einzelnen voll entfalten zu lassen.

(ii) **Funktionierende Software ist wichtiger als umfassende Dokumentation**

Der Kunde ist in erster Linie an einem funktionierenden Produkt interessiert. Es soll möglichst früh ein Prototyp entstehen, der dann im weiteren Prozess an die Bedürfnisse des Kunden angepasst wird. Eine umfassende Dokumentation kann am Ende immer noch ergänzt werden.

(iii) **Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlung**

Anforderungen und Spezifikationen können sich flexibel ändern, dadurch kann der Kunde direkt Rückmeldung geben, ðes ist nicht wichtig im Vertrag kleinste Details zu regeln, sondern auf die Bedürfnisse des Kunden

einzugehen. Daraus folgt direkt der letzte Punkt: SCRUM SCRUM

(iv) **Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans**
Softwareentwicklung ist ein dynamischer Prozess. Das starre Befolgen eines Plans widerspricht der Grundidee der agilen Programmierung.

(c) Beschreiben Sie die wesentlichen Aktivitäten in *Scrum* (agiles Entwicklungsmodell) inklusive deren zeitlichen Ablaufs. Gehen Sie dabei auch auf die *Artefakte* ein, die im Verlauf der Entwicklung erstellt werden.

Lösungsvorschlag

(i) **Initiale Projekterstellung:**
Festlegen eines Product Backlog durch den Product Owner, Erstellen von ersten User Stories. Diese entstehen gegebenenfalls durch den Kontakt mit den Stakeholdern, falls der Product Owner nicht selbst der Kunde ist.

(ii) **Sprint planning:**
Der nächste Sprint (zwischen 1 und 4 Wochen) wird geplant. Dabei wird ein Teil des Product Backlog als *Sprint Backlog* definiert und auf die einzelnen Entwickler verteilt. Es wird festgelegt, *was* implementiert werden soll (Product Owner anwesend) und *wie* das geschehen soll (Entwicklerteam). Während des Sprints findet jeden Tag ein *Daily Scrum* statt, ein fünfzehnminütiger Austausch zum aktuellen Stand. Am Ende des Sprints gibt es ein *Sprint Review* und das *Product Inkrement* wird evaluiert und das *Product Backlog* gegebenenfalls angepasst in Absprache mit *Product Owner* und *Stakeholdern*.

Artefakte:

- Product Backlog
- Sprint Backlog
- Product Increment

(d) Nennen und erklären Sie die *Rollen* in einem Scrum-Team.

Lösungsvorschlag

Product Owner Verantwortlich für das Projekt, für die Reihenfolge der Bearbeitung und Implementierung, ausgerichtet auf Maximierung und wirtschaftlichen Erfolg. Er führt und aktualisiert das *Product Backlog*.

Scrum Master Führungsperson, die das Umsetzen des Scrum an sich leitet und begleitet. Er mischt sich nicht mit konkreten Arbeitsanweisungen ein, sondern moderiert und versucht Hindernisse zu beseitigen, die einem Gelingen der Sprintziele im Weg sind.

Entwickler Sie entwicklen und implementieren die einzelnen Produktfunk-

tionalitäten, die vom Product Owner festgelegt wurden, in eigenverantwortlicher Zeit. Sie müssen die Qualitätsstandards beachten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Übungsaufgabe „Multiple-Choice Allgemeine Software-Technologie“ (EXtreme Programming, V-Modell, Wasserfallmodell, SCRUM, Prototyping, Unit-Test, Anforderungsanalyse)

EXtreme Programming
V-Modell
Wasserfallmodell
SCRUM
Prototyping
Unit-Test
Anforderungsanalyse

Kreuzen Sie bei der folgenden Multiple-Choice-Frage die richtige(n) Antwort(en) an. Auf falsch gesetzte Kreuze gibt es je einen Minuspunkt. Die Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

(a) Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen gedacht?

- ☐ EXtreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Wasserfallmodell
- ☐ Scrum

Lösungsvorschlag

- ☒ EXtreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Wasserfallmodell
- ☒ Scrum

(b) Welche der folgenden Aussagen ist korrekt?

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

Lösungsvorschlag

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☒ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☒ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

(c) Welche der folgenden Aussagen ist korrekt?

- ☐ Das Wasserfallmodell sollte nur für große Projekte eingesetzt werden, da der Einarbeitungsaufwand sehr groß ist.

- ☐ Eine gute Anforderungsspezifikation muss vor allem für Ingenieure verständlich sein, da die Anforderungsspezifikation die Grundlage der Systementwicklung bildet.
- ☐ Verifikation ist der Prozess der Beurteilung eines Systems mit dem Ziel festzustellen, ob die spezifizierten Anforderungen erfüllt sind.
- ☐ Durch Validierung kann überprüft werden, ob das Produkt den Erwartungen des Kunden entspricht.
- ☐ Mit Hilfe eines Black-Box-Tests kann man die Korrektheit eines Programmcodes beweisen.

Lösungsvorschlag

- ☐ Das Wasserfallmodell sollte nur für große Projekte eingesetzt werden, da der Einarbeitungsaufwand sehr groß ist.
- ☐ Eine gute Anforderungsspezifikation muss vor allem für Ingenieure verständlich sein, da die Anforderungsspezifikation die Grundlage der Systementwicklung bildet.
- ☐ Verifikation ist der Prozess der Beurteilung eines Systems mit dem Ziel festzustellen, ob die spezifizierten Anforderungen erfüllt sind.
- ☒ Durch Validierung kann überprüft werden, ob das Produkt den Erwartungen des Kunden entspricht.
- ☐ Mit Hilfe eines Black-Box-Tests kann man die Korrektheit eines Programmcodes beweisen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/01_Projektmanagement/Aufgabe_Allgemeine-Software-Technik.tex`

Übungsaufgabe „Teacher-Data“ (Nicht-funktionale Anforderungen, Funktionale Anforderungen, Wasserfallmodell, Spiralmodell, V-Modell, Evolutionäre Softwaremodelle, Inkrementelle Prozessmodelle, SCRUM)

Das Entwicklerteam von Teacher-Data hat einen neuen Auftrag bekommen. Sie sollen für die Automaten-Videothek DVDRental die Software entwickeln. Nach den ersten Gesprächen konnten folgende Informationen zusammengetragen werden:

- DVD-Verleih-Automat:

    - Der Kunde soll einfach, schnell und günstig am Automaten 24 Stunden, sieben Tage die Woche DVDs ausleihen können. Der Kunde hat des Weiteren die Möglichkeit, Filme vorab über das Internet zu reservieren. In der DVD-Verleihstation gibt es drei verschiedene Automaten (siehe Zeichnung).
    - Um DVDs ausleihen zu können, müssen sich Neukunden während der Service- und Anmeldezeiten der DVD-Verleihstation bei einem Mitarbeiter anmelden. Dabei werden die benötigten Daten (*Name*, *Vorname*, *Geburtsdatum*, *Anschrift*, *Personalausweisnummer*, *Guthaben*, *Kennwort*) des Kunden aufgenommen und im System gespeichert. Zur persönlichen Identifikation des Kunden wird anschließend der Fingerabdruck am Automaten eingelesen. Der Kunde bekommt eine Magnetkarte ausgehändigt, um sich an den Automaten anmelden zu können.
    - Beim DVD-Ausleih muss sich der Kunde mit der Magnetkarte und seinem Fingerabdruck am Automaten für die Filmauswahl identifizieren. Am Eingabe-Terminal hat der Kunde die Möglichkeit Filme nach *Genre*, *Schauspieler*, *Titel*, *TOP-10*, *Neuerscheinungen* und *Stichworten* zu suchen. Suchergebnisse werden mit folgenden Informationen auf dem Display präsentiert: *Titel des Films*, *Regisseur*, *Hauptdarsteller*, *Kurzbeschreibung*, *Erscheinungsjahr*, *Altersfreigabe* und *Verfügbarkeit*. Nach der Wahl eines Films, erhält der Kunde seine Magnetkarte zurück und kann den Film am Automaten für die Ausund Rückgabe nach erneuter Identifikation durch die Karte und den Fingerabdruck abholen. Die DVD wird über den Ausgabeschacht ausgegeben.
    - Bei der Rückgabe einer DVD muss sich der Kunde am Automaten für die Aus- und Rückgabe von Filmen mit der Magnetkarte und seinem Fingerabdruck am Automaten identifizieren. Nach erfolgreicher Identifikation führt der Kunde die DVD in den Rückgabeschacht ein. Der Automat liest daraufhin den Barcode auf der DVD und sortiert diese eigenständig wieder ein und bucht die Leihgebühr vom Guthaben auf der Karte ab.
    - Das Guthaben kann am Automaten zum Aufladen des Guthabens aufgeladen werden. Nach Eingabe der Kundenkarte können Münzen (5, 10, 20, 50 Cent, 1 und 2€) und Scheine (5, 10, 20, 50€) bar eingezahlt werden. Bei einem Betrag von 20€ gibt es einen Bonus von 3€, bei 30€ einen von 5€ und bei einem von 50€ von 10€. Die Karte wird bei beendeter Geldeingabe nach Drücken des roten Knopfes wieder ausgeworfen.

    - Reservierung von Filmen über das Internet: Nach erfolgreicher Anmeldung mit Kundennummer und Geheimzahl auf der betreibereigenen Homepage kann der Kunde nach *Genre*, *Schauspieler*, *Titel*, *TOP-10*, *Neuerscheinungen* und *Stichworten* Filme suchen. Suchergebnisse werden mit folgenden Informationen präsentiert: *Titel des Films*, *Regisseur*, *Hauptdarsteller*, *Kurzbeschreibung*, *Erscheinungsjahr*, *Altersfreigabe* und *Verfügbarkeit*. Der Kunde kann nach der Suche einzelne Filme reservieren, die er innerhalb von zwei Stunden abholen muss. Sollte dies nicht in dem Zeitraum geschehen, so werden die Filme nach zwei Stunden wieder zum Ausleihen freigegeben.

- Weitere Rahmenbedingungen:
    - Wird die Kundenkarte im Automaten vergessen, so wird sie zur Sicherheit automatisch eingezogen und kann während der Servicezeiten wieder abgeholt werden.
    - Die maximale Leihdauer beträgt 10 Tage.
    - Hat der Kunde versehentlich den falschen Film ausgeliehen, so kann er diese bis 10 Minuten nach Ausgabe der DVD ohne Berechnung einer Leihgebühr wieder am Automaten zurückgeben.
    - Die Leihgebühr für jeden angefangenen Tag (24h) beträgt 2€.
    - Zu jedem Film gibt es mehrere Exemplare.
    - Die Ausgabe und Einsortierung von DVDs übernimmt ein DVD-Archiv-Roboter, der über eine Schnittstelle angesteuert werden muss.

(a) Identifizieren Sie die Aktoren des oben beschriebenen Systems.

Lösungsvorschlag

Kunde, Service-Mitarbeiter, Wartung, DVD-Archiv-Roboter

(b) Identifizieren Sie die nicht-funktionalen und funktionalen Anforderungen (als Use Cases).

Lösungsvorschlag

Nicht-funktional: Bedienoberfläche Web / Automat gleich, einfache Handhabbarkeit, schnell, Dauerbetrieb, Ausfallquote, Leihdauer

Funktional: Geldkarte aufladen, Film suchen, Film auswählen, Film entnehmen, Film zurückgeben, Kunden aufnehmen, Film reservieren

(c) Geben Sie eine formale Beschreibung für drei Use Cases an. Orientieren können Sie sich an folgendem Beispiel:

Lösungsvorschlag

Geldkarte aufladen

Use Case:	Geldkarte aufladen
Ziel:	Auf der Geldkarte befindet sich ein gewünschter Betrag an Geld.
Kategorie:	primär
Vorbedingung:	Geldkarte befindet sich im Automaten.
Nachbedingung Erfolg:	Auf der Geldkarte befindet sich der gewünschte Betrag.
Nachbedingung Fehlschlag:	Betrag auf der Karte ist der gleiche wie davor → Fehlermeldung
Akteure:	Kunde
Auslösendes Ereignis:	Geldkarte wird in den Automaten zum Aufladen des Guthabens geschoben.
Beschreibung:	(i) Kunde schiebt Karte in den Automaten. (ii) Kunde wirft Münzen oder gibt einen Geldschein ein. (iii) Kunde drückt Knopf zum Auswerfen der Karte. (iv) Karte wird ausgegeben.
Erweiterungen:	Schein oder Münze wird nicht akzeptiert. → Fehlermeldung
Alternativen:	Mindest-Guthaben auf der Karte

Film suchen

Use Case:	Film suchen
Ziel:	Anzeigen des gesuchten Films
Kategorie:	primär
Vorbedingung:	Kunde hat sich mit Magnetkarte und Fingerabdruck identifiziert.
Nachbedingung Erfolg:	Eine Liste von Filmen wird angezeigt.
Nachbedingung Fehlschlag:	Kein Film gefunden → Fehlermeldung
Akteure:	Kunde
Auslösendes Ereignis:	Kunde will einen Film ausleihen.
Beschreibung:	Kunde gibt Genre, Schauspieler, Titel oder Stichwort ein oder lässt sich Neuerscheinungen und TOP10 auflisten.
Erweiterungen:	–
Alternativen:	Vorgaben von weiteren Suchkriterien

Kunden aufnehmen

Use Case:	Kunden aufnehmen
Ziel:	Kunde ist in der Kundendatei.
Kategorie:	primär
Vorbedingung:	Kunde ist noch nicht in der Kundendatei vorhanden.
Nachbedingung Erfolg:	Kunde ist als Mitglied in der Kundendatei aufgenommen.
Nachbedingung Fehlschlag:	Kunde kann nicht in die Datei aufgenommen werden.
Akteure:	Kunde, Service-Mitarbeiter
Auslösendes Ereignis:	Kunde stellt zu den Öffnungszeiten bei einem Service-Mitarbeiter einen Mitgliedsantrag.
Beschreibung:	(i) Kunde meldet sich zu den Öffnungszeiten bei einem Service-Mitarbeiter an. (ii) Service-Mitarbeiter erfasst nötige Daten (Name, Vorname, Geburtsdatum, Anschrift, Personalausweisnummer, Kennwort). (iii) Service-Mitarbeiter kassiert einen Startbetrag und erfasst ihn bei den Kundendaten. (iv) Der Fingerabdruck des Kunden wird erfasst und gespeichert. (v) Kunde bekommt eine Magnetkarte mit seinen Daten.
Erweiterungen:	-
Alternativen:	Kunde ist bereits im System und lässt nur die Kundendaten ändern.

(d) Verfeinern Sie nun die Use Cases, indem Sie ähnliche Teile identifizieren und daraus weitere Use Cases bilden.

Lösungsvorschlag

Identifizierung des Kunden an den Automaten, Filmsuche

(e) Nennen Sie mindestens drei offene Fragen, die in einem weiteren Gespräch mit dem Kunden noch geklärt werden müssten.[1]

[1] Quelle: Universität Stuttgart, Institut für Automatisierungs- und Softwaretechnik Prof. Dr.-Ing. Dr. h. c. P. Göhner

Lösungsvorschlag Wasserfallmodell

- Was passiert nach der maximlaen Leihdauer?
- Was passiert, wenn der Finger nicht zur Karte passt?
- Was passiert mit defekten DVDs?
- Was passiert bei Rückgabe der falschen DVD?

Wasserfallmodell

Die Firma Teacher-Data soll für einen Auftraggeber ein Informationssystem entwickeln. Das Entwicklerteam entscheidet sich, nach dem *Wasserfallmodell* vorzugehen.

(a) Geben Sie an, welche *Phasen* dabei durchlaufen werden und erläutern Sie kurz deren *Zweck / Intention / Aufgabe*.

Lösungsvorschlag

(i) Problem- und Systemanalyse

Grobe Formulierung der Ziele (z. B. Einsatzplattform Windows, einheitliche Software zur Unterstützung des Verkaufs, ...)

- Erfassung des Problembereichs (z. B. Interaktion zwischen Händler und Hersteller soll komplett über das Netz funktionieren.)
- Machbarkeitsstudie (z. B. Händler sind bereit, sich an das zu entwickelnde System anzupassen.)
- Analyse des Ist-Zustandes (z. B. Händler A bietet Interface X wohingegen Händler B Interface Z bietet. Es wird bereits teilweise bei Händler C über das Netz bestellt. ...)
- Anforderungsspezifikation (Lastenheft) (z. B. Einheitliche Oberfläche für alle Benutzer beim Hersteller, möglichst wenige Veränderungen für die Mitarbeiter (= schnelle Eingewöhnungsphase) ...)
- Systemspezifikation (Pflichtenheft) (z. B. soll der Ablauf eines Kaufvorgangs folgendermaßen aussehen: Händlerauswahl, Bestellung, Abrechnung ... Die Benutzeroberfläche soll folgende Informationen anzeigen: Lieferstatus, ...)

(ii) Systementwurf

- Systemarchitektur (z. B. Klassendiagramm, Festlegung von Komponenten ...)
- Schnittstellen der Komponenten festlegen (z. B. Datenbank, Middleware, ...)
- Modulkonzept (z. B. Modulstruktur und Art der Module (funktional, prozedural, objektorientiert)
- Definition der Einzelschnittstellen (z. B. Interfaces und abstrakte Klassen in Java)

(iii) Implementierung

- Strukturierung der Komponenten in Module (z. B. Klassenrümpfe)
- Spezifikation einzelner Module (z. B. exakte Beschreibung eines Moduls)
- Codierung, Generierung, Wiederverwendung einzelner Module (z. B. Methoden in Klassen, ...)

(iv) Integration und Test

- Integration von Modulen (z. B. Zusammensetzen von Datenbank und Applikation zum Server)
- Integration von Komponenten (z. B. Einsetzen von Client und Server ins Gesamtsystem.)
- Testen und Verifikation (z. B. Testen, ob die Implementierung zusammen mit der Umgebung funktioniert. Zusichern, dass gewisse Eigenschaften gültig sind.)

(v) Wartung und Weiterentwicklung

- Wartung (z. B. Korrektur eines Fehlers im Programm, Portierung auf anderes Betriebssystem, ...)
- Erweitern der Funktionalität (z. B. Einbeziehen einer neuen Option im Geschäftsprozess)
- Anpassung der Funktionalität (z. B. Anpassung an Änderung im Geschäftsprozess)

(b) Während des Entwicklungsprozesses werden nach und nach im Folgenden aufgelistete Teilprodukte entstehen. Ordnen Sie diese den jeweiligen Phasen zu.

- Schätzung der Entwicklungskosten
- Dokumentation der Änderungen nach Produktabnahme
- Pflichtenheft
- Beschreibung der Datenstruktur in der Datenbank
- Integration von Modulen/Komponenten
- Betriebsbereites Informationssystem
- Beschreibung der Schnittstelle einzelner Softwarekomponenten
- Quellcode für die GUI
- Durchführbarkeitsstudie
- Systemarchitektur

Lösungsvorschlag

- Schätzung der Entwicklungskosten → *Anforderungsdefinition*
- Dokumentation der Änderungen nach Produktabnahme → *Betrieb und Wartung*
- Pflichtenheft → *Anforderungsdefinition*

Wasserfallmodell
Spiralmodell
V-Modell
Evolutionäre Softwaremodelle
Inkrementelle Prozessmodelle
SCRUM

- Beschreibung der Datenstruktur in der Datenbank → *System- und Softwareentwurf*
- Integration von Modulen/Komponenten → *Integration und Systemtest*
- Betriebsbereites Informationssystem → *Betrieb und Wartung*
- Beschreibung der Schnittstelle einzelner Softwarekomponenten → *System- und Softwareentwurf*
- Quellcode für die GUI → *Implementierung und Komponententest*
- Durchführbarkeitsstudie → *Anforderungsdefinition*
- Systemarchitektur → *System- und Softwareentwurf*

Vorgehensmodelle

Dem Team von Teacher-Data kommen Zweifel, ob die Entscheidung für das Wasserfallmodell für die Umsetzung des Informationssystems richtig war, oder ob ein anderes Vorgehen eventuell besser wäre. Beurteilen und vergleichen Sie die Vorgehensmodelle

- Wasserfallmodell
- Spiralmodell
- V-Modell
- Evolutionäres Modell oder Inkrementelles Modell
- agile Entwicklung, wie z. B. Scrum

nach den folgenden Gesichtspunkten:

(a) Größe des Entwicklerteams

(b) Komplexität des Projekts

(c) Bekanntheit der Anforderungen

(d) Änderung der Anforderungen

(e) Zeitspielraum (Time-to-Market; was muss bis wann fertig sein?)

(f) Dokumentation

(g) IT-Kenntnisse des Kunden

(h) Durchschnittliche Anzahl an Iterationen

Lösungsvorschlag

	Wasserfallmodell	Spiralmodell	V-Modell	evolutionär / inkrementell	agil
Größe des Entwicklerteams	diskutiert die Projektgröße nicht	mittlere Teams	mittlere und große Teams	diskutiert die Projektgröße nicht	ca. 3-9 Entwickler (ohne Product Owner und Scrum Master)
Komplexität des Projekts	einfache Projekte mit stabilen Anforderungen	einfache Projekte mit stabilen Anforderungen; Anforderungen können jedoch in den Iterationen angepasst werden.	komplexe Projekte; bei einfachen Projekten relativ viel bürokratischer Overhead.	große, lange und komplexe Projekte	hohe Komplexität möglich
Bekanntheit der Anforderungen	Alle Anforderungen müssen zu Beginn des Projekts bekannt sein.	Es muss eine initiale Menge von Anforderung bekannt sein. Der Kunde kann die Anforderungen aber aufgrund von Prototypen erweitern.	evolutionär: Alle Anforderungen müssen zu Beginn des Projekts bekannt sein. inkrementell: Anforderungen müssen nicht von Anfang an bekannt sein.	Anforderungen müssen von Anfang an bekannt sein.	Es muss eine initiale Menge von Anforderung bekannt sein. Der Kunde kann die Anforderungen aber laufend erweitern / ggf. ändern.
Änderungen der Anforderungen	Schwer möglich. Durch den strikten Top-Down-Ansatz von der Anforderung hin zum Code zieht eine Änderung einen tiefen Eingriff in den Prozess mit sich.	Gut möglich. In jeder Runde der Spirale können die Anforderungen neu definiert werden.	Schwer möglich → strikter Top-Down-Ansatz	evolutionär: häufige Änderungen möglich. inkrementell: Änderungen teilweise möglich.	rasche Anpassung an neue Anforderungen möglich (deshalb agil!)
Zeitspielraum	Wenig Spielraum. Üblicherweise muss der ganze Prozess durchlaufen werden. Es ist höchstens möglich, den Prozess zu beschleunigen, indem man Anforderungen streicht. Es wird also alles zum Schluss fertig.	Relativ flexibel. Sobald ein Prototyp existiert, der akzeptabel ist, kann dieser auf den Markt gebracht werden.	Wenig Spielraum (vgl. Wasserfallmodell)	Einsatzfähige Produkte in kurzen Zeitabständen, daher relativ flexibel.	Unterteilung in Sprints, daher schnell lauffähige Prototyoen vorhanden. Bei der Planung des nächsten Sprints kann auf neue zeitliche Gegebenheiten relativ flexibel eingegangen werden.
Dokumentation	Viel Dokumentation – Für jede Phase wird eine komplette Dokumentation erstellt.	Sehr viel Dokumentation – In jedem Zyklus werden alle Phasen (inkl. Dokumentation) durchlaufen, aber nur im Umfang des Prototypen. Es werden alle Artefakte und Tätigkeiten dokumentiert.	Viel Dokumentation – Gerade in den ersten Phasen wird sehr viel über die Software schriftlich festgehalten. Aber auch für die anderen Phasen wird eine komplette Dokumentation erstellt.		Anforderungsdokumentation ist sehr wichtig. Ansonsten ist funktionierende Software höher zu bewerten als eine umfangreiche Dokumentation.
IT-Kenntnisse des Kunden	Wenig Kenntnisse nötig. Meistens schreibt der Kunde das Lastenheft und der Entwickler versucht dieses dann in einem Pflichtenheft umzusetzen. Wenn das Lastenheft vom Entwickler als machbar befunden wird, bekommt der Kunde das fertige Produkt.	Wenig Kenntnisse nötig. – Kunde sieht immer nur die fertigen Prototypen und äußert dann seine Wünsche.	Wenig Kenntnisse nötig (vgl. Wasserfallmodell)	Wenig Kenntnisse nötig	Wenig Kenntnisse nötig, aber von Vorteil, da enge Zusammenarbeit mit dem Kunden
Durchschnittliche Anzahl an Iterationen	vgl. V-Modell	ca. 3-5 Iterationen	Nur lange Zyklen. – Mit dem V-Modell sind nur sehr lange Zyklen handhabbar, da immer wieder der ganze Entwicklungsprozess durchlaufen werden muss.		variable Anzahl an Sprints, je nach Projektgröße

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/01_Projektmanagement/Aufgabe_Teacher-Data.tex`

Übungsaufgabe „Grundwissen“ (White-Box-Testing, Black-Box-Testing, Funktionalorienteres Testen, V-Modell)

White-Box-Testing
Black-Box-Testing
Funktionalorienteres Testen
V-Modell

(a) Nennen Sie die wesentlichen Unterschiede zwischen *White-Box-Testen* und *Black-Box-Testen* und geben Sie jeweils zwei Beispiele an.

Lösungsvorschlag

White-Box-Tests sind *strukturorientiert*, z. B. Kontrollflussorientiertes Testen oder Datenflussorientiertes Testen.

Black-Box-Test sind *spezifikations- und funktionsorientiert* (nur Ein- und Ausgabe relevant), z. B. Äquivalenzklassenbildung oder Grenzwertanalyse.

(b) Geben Sie drei nicht-funktionalorientierte Testarten an.

Lösungsvorschlag

(i) Performanztest

(ii) Lasttest

(iii) Stresstest

(c) Nennen Sie die vier verschiedenen Teststufen aus dem V-Modell und erläutern Sie deren Ziele.

Lösungsvorschlag

Komponenten-Test: Fehlerzustände in Modulen finden.

Integrations-Test: Fehlerzustände in Schnittstellen und Interaktionen finden.

System-Test: Abgleich mit Spezifikation.

Abnahme-Test: Vertrauen in System und nicht-funktionale Eigenschaften gewinnen.

(d) Nennen Sie fünf Aktivitäten des Testprozesses.

Lösungsvorschlag

(i) Testplanung und Steuerung,

(ii) Testanalyse und Testentwurf,

(iii) Testrealisierung und Testdurchführung,

(iv) Bewertung von Endekriterien und Bericht,

(v) Abschluss der Testaktivitäten

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/20_Black_White-Box-Test/Aufgabe_Grundwissen.tex`

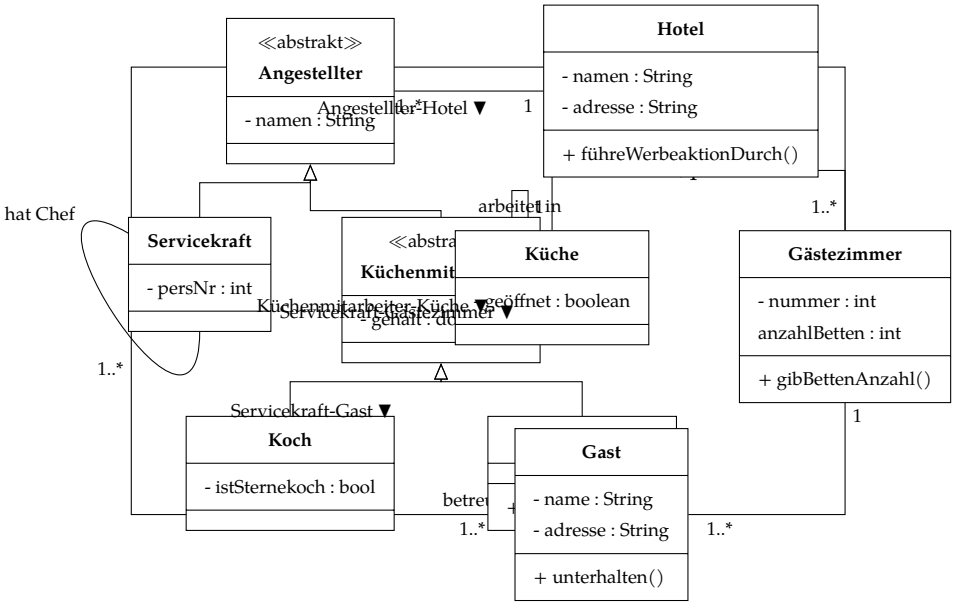
Modellierung

Examensaufgabe „Hotel-Verwaltung“ (46116-2012-F.T2-TA2-A1)

Hotel-Verwaltung

Erstellen Sie zu der folgenden Beschreibung eines Systems zur Organisation eines Hotels ein Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten sowie Methoden enthält. Setzen Sie dabei das Konzept der Vererbung sinnvoll ein.

- Ein **Hotel** besteht aus genau einer Küche und mehreren Gästezimmern.
- Es hat einen Namen, eine Adresse und kann Werbeaktionen durchführen.
- Alle Mitarbeiter, sowohl Servicekräfte als auch Küchenmitarbeiter, sind Angestellte des Hotels.
- Die Küche kann geöffnet oder geschlossen sein.
- Gästezimmer haben eine Nummer und eine bestimmte Anzahl an Betten, die ausgegeben werden kann.
- In diesen Gästezimmern wohnen Gäste, die von einer oder mehreren Servicekräften umsorgt werden.
- Servicekräfte sind nur für die ihnen zugeordneten Gästezimmer verantwortlich.
- Jede Servicekraft hat einen Namen und eine Personalnummer sowie einen Vorgesetzten, der auch eine Servicekraft ist.
- In der Küche arbeiten Küchenmitarbeiter, die einen Namen haben und ein Gehalt bekommen.
- Küchenmitarbeiter sind entweder Köche oder Aushilfen. Köche können zudem Sterneköche sein, also mit einem oder mehreren Sternen dekoriert sein.
- Aushilfen bauen die Büffets für die Mahlzeiten auf.
- Gäste können sich unterhalten. Jeder Gast hat einen Namen und eine Adresse und ist seinem Zimmer zugeordnet.



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2012/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „CreditCard, Order“ (46116-2013-F.T2-TA1-A1)

Klassendiagramm

Gegeben sei folgendes Klassendiagramm:

(a) Implementieren Sie die Klassen „Order“, „Payment“, „Check“, „OrderDetail“ und „Item“ in einer geeigneten objektorientierten Sprache Ihrer Wahl. Beachten Sie dabei insbesondere Sichtbarkeiten, Klassen- vs. Instanzzugehörigkeiten und Schnittstellen bzw. abstrakte Klassen.

Lösungsvorschlag

```java
public class Check extends Payment {
  String bankName;
  long bankId;
  public boolean authorized() {
    return true;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Check.java`

```java
@SuppressWarnings("unused")
public class Item {
  private double weight;
  public String description;

  public double getPrice() {
    return 42;
  }

  public double getWeight() {
    return 23;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Item.java`

```java
import java.util.Date;

public class Order {
  public static double VAT = 0.19;

  protected Date date;

  protected boolean shipped;

  public double calcPrice() {
    return 0.1d;
  }

  public double calcWeight() {
    return 0.2d;
  }

  public double calcTax(double tax) {
```

```
    return 0.3d;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Order.java`

```
@SuppressWarnings("unused")
public class OrderDetail {
  private long quantity;

  Item item;

  public double calcPrice() {
    return 0.1d;
  }

  public double calcWeight() {
    return 0.2d;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/OrderDetail.java`

```
public abstract class Payment {
  double amount;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Payment.java`

(b) Erstellen Sie ein Sequenzdiagramm (mit konkreten Methodenaufrufen und Nummerierung der Nachrichten) für folgendes Szenario:

(i) „Erika Mustermann“ aus „Rathausstraße 1, 10178 Berlin“ wird als neue Kundin angelegt.

(ii) Frau Mustermann bestellt heute 1 kg Gurken und 2 kg Kartoffeln.

(iii) Sie bezahlt mit ihrer Visa-Karte, die im August 2014 abläuft und die Nummer „1234 567891 23456“ hat — die Karte erweist sich bei der Prüfung als gültig.

(iv) Am Schluss möchte sie noch wissen, wie viel ihre Bestellung kostet — dabei wird der Anteil der Mehrwertsteuer extra ausgewiesen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2013/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Bestellsystem“ (46116-2014-H.T2-TA1-A3)

UML-Diagramme
Anwendungsfalldiagramm

Gegeben sei folgender Sachverhalt:

Für eine Verwaltungssoftware einer Behörde soll ein Bestellsystem entwickelt werden. Dabei sollen die Nutzer ihre Raummaße eingeben können. Anschließend können die Nutzer über ein Web-Interface das Büro gestalten und Möbel (wie zum Beispiel Wandschränke) und andere Einrichtungsgegenstände in einem virtuellen Büro platzieren. Aus dem Web-Interface kann die Einrichtung dann direkt bestellt werden. Dazu müssen die Nutzer ihre Büro-Nummer und den Namen und die Adresse der Behörde eingeben und die Bestellung bestätigen.

Weiterhin können Nutzer auch Büromaterialien über das Web-Interface bestellen. Dazu ist anstatt der Eingabe der Raummaße nur das Eingeben von Büro-Nummer und des Namens und der Adresse der Behörde erforderlich.

Zusätzlich zum Standard-Nutzer können sich auch Administratoren im System anmelden und Möbel zur Kollektion hinzufügen und aus der Kollektion entfernen. Die Möbel können eindeutig durch ihre Inventurnummer identifiziert werden.

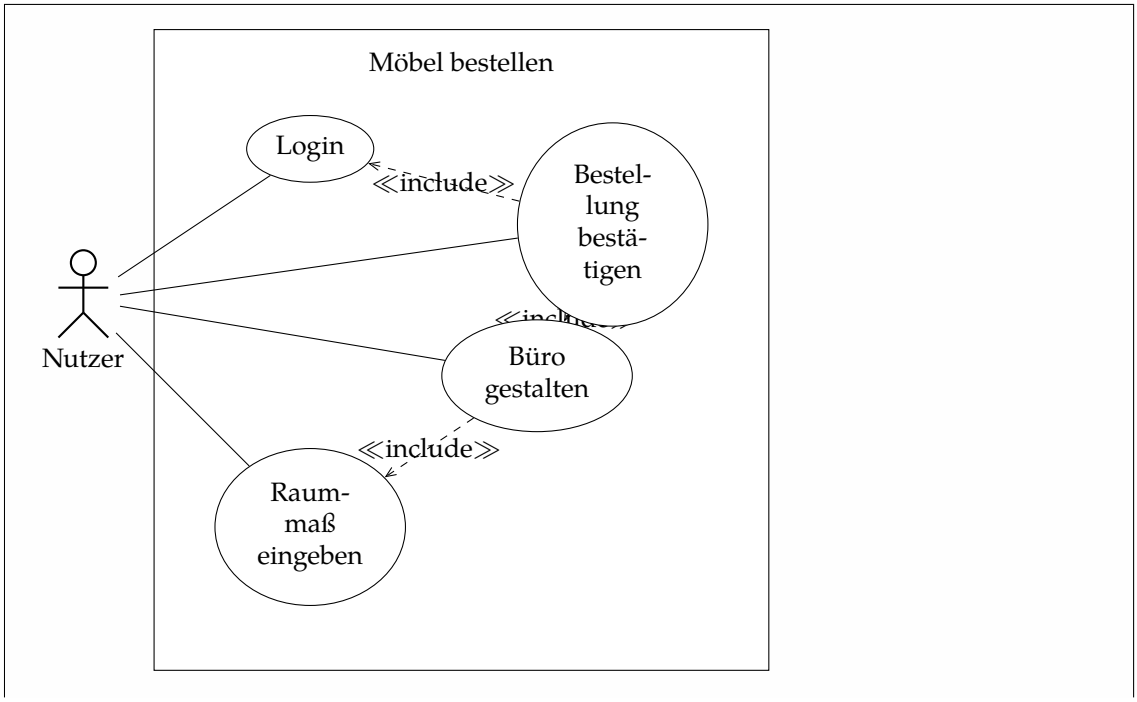
Um jegliche Veränderungen im System protokollieren zu können müssen Nutzer und Administratoren zur Bestätigung eingeloggt sein.

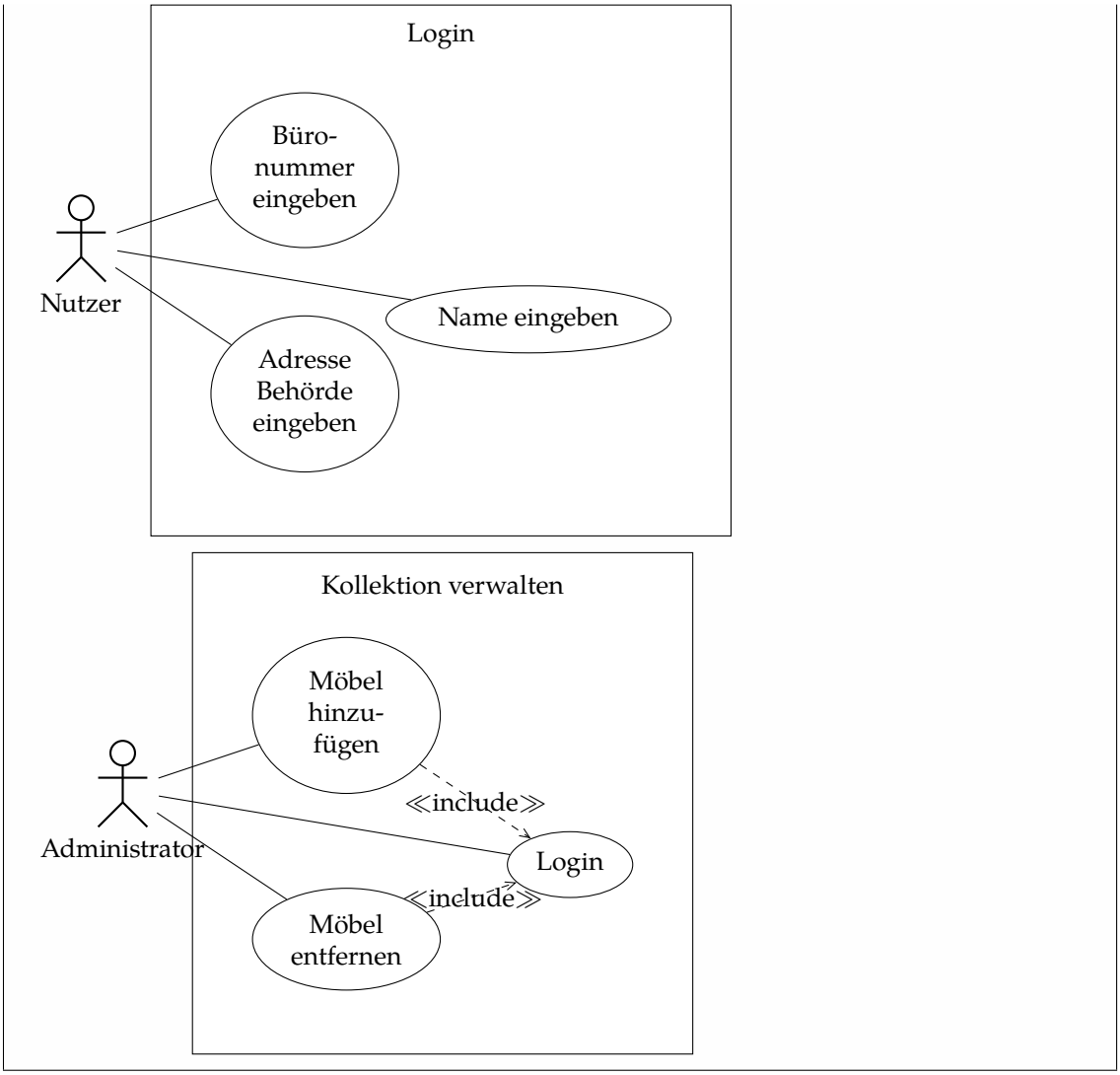
(a) Erfassen Sie die drei Systemfunktionen *Möbel bestellen*, *Login* und *Kollektion verwalten* in einem UML-konformen Use Case Diagramm.

Login

Kollektion verwalten

Lösungsvorschlag





(b) Erstellen Sie ein UML-Klassendiagramm, welches die Beziehungen und sinnvolle Attribute der Klassen „Nutzer, Büro, Möbelstück und Wandschrank“ darstellt.

(c) Instanziieren Sie das Klassendiagramm in einem Objektdiagramm mit den zwei Nutzern mit Namen Ernie und Bernd, einem Büro mit der Nummer CAPITOL2 und zwei Schränken mit den Inventurnummern S1.88 und S1.77.

(d) Geben Sie für ein Büromöbelstück ein Zustandsdiagramm an. Überlegen Sie dazu, welche möglichen Zustände ein Möbelstück während des Bestellvorgangs haben kann und finden Sie geeignete Zustandsübergänge.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/09/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

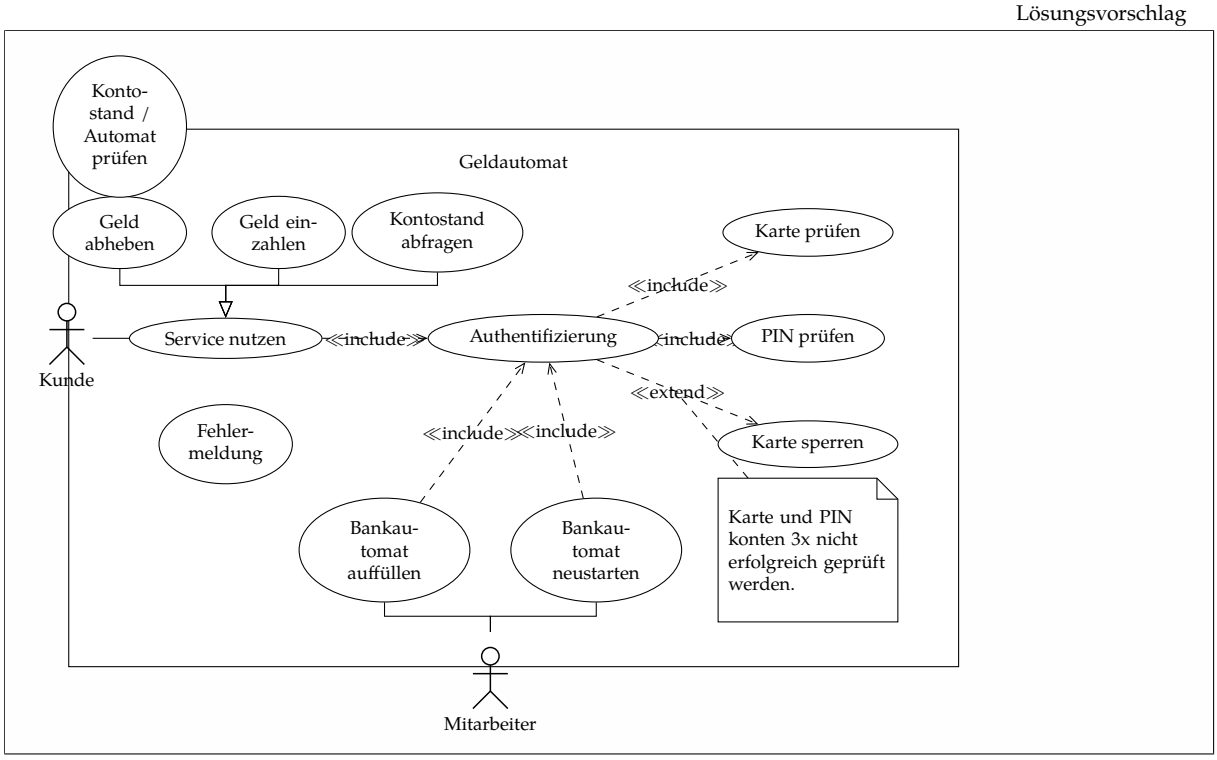
Examensaufgabe „Geldautomat“ (46116-2017-F.T1-TA1-A2)

Anwendungsfalldiagramm

Im Folgenden sehen Sie ein fehlerhaftes Use-Case-Diagramm für das System „Geldautomat”. Geben Sie ein korrektes Use-Case-Diagramm (Beziehungen und Beschriftungen) entsprechend der folgenden Beschreibung an. Sollte es mehrere Möglichkeiten geben, begründen Sie Ihre Entscheidung kurz.

Kunden können am Geldautomat verschiedene Services nutzen, es kann Geld abgehoben und eingezahlt sowie der Kontostand abgefragt werden. Für jeden dieser Services ist eine Authentifizierung notwendig. Diese besteht aus der Prüfung der Bankkarte und des eingegebenen PINs. Manche Bankautomaten können Karten bei zu vielen Fehlversuchen (3) bei der Anmeldung sperren, andere geben Fehlermeldungen aus, falls die Karte gesperrt wurde oder nicht mehr genügend Geld auf dem Konto oder im Bankautomaten vorhanden ist. Mitarbeiter bzw. Mitarbeiterinnen der Bank können sich ebenfalls über PIN und Karte authentifizieren, um dann den Bankautomaten neu zu starten oder mit Geld aufzufüllen. Bevor Geld abgehoben werden kann, ist sicherzustellen, dass auf dem Konto und im Bankautomaten genügend Geld vorhanden ist.

Lösungsvorschlag



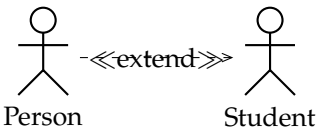
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Korrektheit von UML-Diagrammen“ (46116-2017-F.T1-TA1-A3)

UML-Diagramme
Klassendiagramm
Anwendungsfalldiagramm

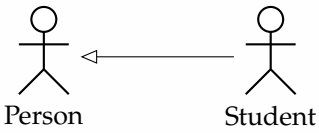
Betrachten Sie die folgenden UML-Diagramme. Sind diese korrekt? Falls nein, begründen Sie warum nicht. Geben Sie in diesem Fall außerdem eine korrigierte Version an. Falls ja, erklären Sie die inhaltliche Bedeutung des Diagramms.

(a)

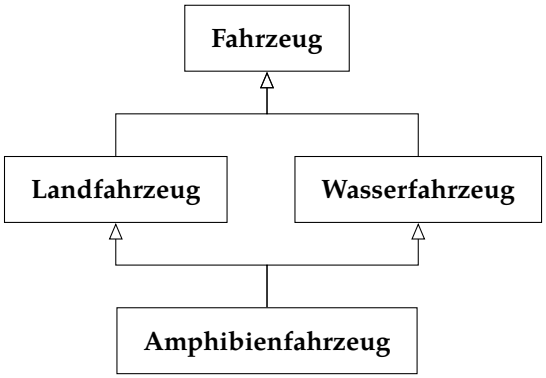


Lösungsvorschlag

Falsch, den verwendeten „extends“-Pfeil gibt es nur zwischen Anwendungsfällen.



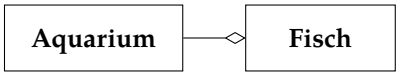
(b)



Lösungsvorschlag

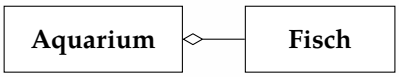
Die dargestellte Modellierung ist *korrekt*. Sowohl Land- als auch Wasserfahrzeuge sind Fahrzeuge und erben somit von dieser Klasse. Da ein Amphibienfahrzeug eine „Mischung“ aus beidem ist, erbt diese Klasse auch von beiden Klassen. Diese Mehrfachvererbug kann allerdings nicht in jeder Programmiersprache (z. B. nicht in Java) umgesetzt werden.

(c)

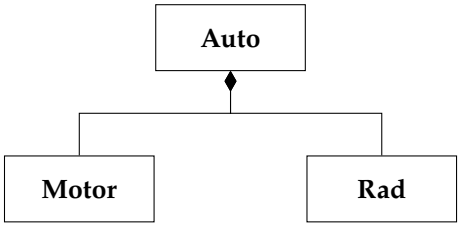


Lösungsvorschlag

Bei der dargestellten Aggregation befindet sich die Raute an der *falschen* Seite der Beziehung. Das Diagramm würde bedeuten, dass ein Fisch mehrere Aquarien enthält. Die Umkehrung ist aber korrekt:

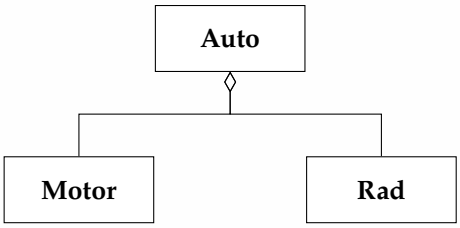


(d)



Lösungsvorschlag

Hier wurde für die Modellierung eine Komposition gewählt. Dies bedeutet, dass die Existenz der Teile vom Ganzen abhängt. Einen Raum kann es z. B. ohne ein Gebäude nicht geben. In diesem Fall ist die Darstellung *falsch*, da Motor und Rad auch ohne Auto existieren können. Die Modellierung muss also mittels Aggregation erfolgen:



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Entwurfsmuster bei Bankkonten, Hockeyspiel, Dateisystem“ (46116-2017-H.T2-TA1-A3)

Entwurfsmuster
Klassendiagramm
Abstrakte Fabrik (Abstract Factory)
Beobachter (Observer)
Kompositum (Composite)

Verwenden Sie geeignete **Entwurfsmuster**, um die folgenden Sachverhalte mit Hilfe von **UML-Klassendiagrammen** zu beschreiben. Nennen Sie das zu verwendende Entwurfsmuster namentlich, wenden Sie es zur Lösung derjeweiligen *Fragestellung* an und erstellen Sie damit das problemspezifische UML-Klassendiagramm. Beschränken Sie sich dabei auf die statische Sicht, ðdefinieren Sie keinerlei Verhalten mit Ausnahme der Definition geeigneter Operationen.

(a) Es gibt unterschiedliche Arten von Bankkonten: Girokonto, Bausparkonto vmd Kreditkarte. Bei allen Konten ist der Name des Inhabers hinterlegt. Girokonten haben eine IBAN. Kreditkarten sind immer mit einem Girokonto verknüpft. Bei Bausparkonten werden ein Sparzins sowie ein Darlehenszins festgelegt. Es gibt eine *zentrale* Klasse, die die *Erzeugung* unterschiedlicher Typen von Bankkonten steuert.

(b) Beim Ticker für ein Hockeyspiel können sich verschiedene Geräte registrieren und wieder abmelden, um auf *Veränderungen* des Spielstands *zu reagieren*. Hierzu werden im Ticker die Tore der Heim- vmd Gastmannschaft sowie die aktuelle Spielminute vermerkt. Als konkrete Geräte sind eine Smartphone-App sowie eine Stadionuhr bekannt.

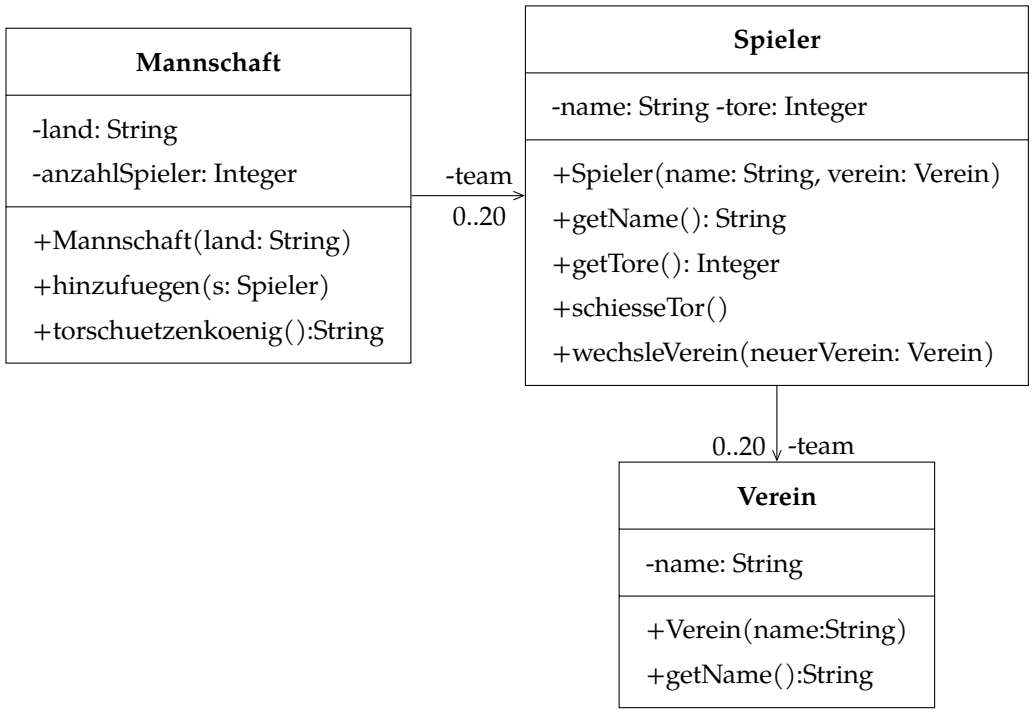
(c) Dateisysteme sind *baumartig* strukturiert. Verzeichnisse können wiederum selbst Verzeichnisse und/oder Dateien beinhalten. Sowohl Dateien als auch Verzeichnisse haben einen Namen.Das jeweilige Eltemverzeichnis ist eindeutig. Bei Dateien wird die Art(Binär, Text oder andere) sowie die Größe in Byte, bei Verzeichnissen die Anzahl enthaltener Dateien hinterlegt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/09/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Fußballweltmeisterschaft“ (46116-2018-F.T1-TA1-A3)

Klassendiagramm
Implementierung in Java

Für die nächste Fußballweltmeisterschaft möchte ein Wettbüro ein Programm zur Verwaltung von Spielern, Vereinen und (National-)Mannschaften entwickeln. Dazu wurde bereits das folgende UML-Klassendiagramm entworfen.



Es kann angenommen werden, dass die Klasse `Verein` bereits implementiert ist. In den folgenden Implementierungsaufgaben können Sie eine objektorientierte Programmiersprache Ihrer Wahl verwenden. Die verwendete Sprache ist anzugeben. Zu beachten sind jeweils die im Klassendiagramm angegebenen Sichtbarkeiten von Attributen, Rollennamen, Konstruktorn und Operationen.

(a) Es ist eine Implementierung der Klasse `Spieler` anzugeben. Der Konstruktor soll die Instanzvariablen mit den gegebenen Parametern initialisieren, wobei die. Anzahl der Tore nach der Objekterzeugung gleich 0 sein soll. Ansonsten kann die Funktionalität der einzelnen Operationen aus deren Namen geschlossen werden.

Lösungsvorschlag

```
public class Spieler {
  private String name;
  private int tore;
  @SuppressWarnings("unused")
  private Verein verein;

  public Spieler(String name, Verein verein) {
    this.name = name;
    this.verein = verein;
    tore = 0;
  }
```

```
  public String getName() {
    return name;
  }

  public int getTore() {
    return tore;
  }

  public void schiesseTor() {
    tore++;
  }

  public void wechsleVerein(Verein neuerVerein) {
    verein = neuerVerein;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2018/fruehjahr/verein/Spieler.java`

Feld (Array)

(b) Es ist eine Implementierung der Klasse `Mannschaft` anzugeben. Der Konstruktor soll das Land initialisieren, die Anzahl der Spieler auf 0 setzen und das Team mit einem noch „leeren “Array der Länge 20 initialisieren. Das Team soll mit der Methode `hinzufuegen` um einen Spieler erweitert werden. Die Methode `torschuetzenkoenig` soll den Namen eines Spielers aus dem Team zurückgeben, der die meisten Tore für die Mannschaft geschossen hat. Ist das für mehrere Spieler der Fall, dann kann der Name eines beliebigen solchen Spielers zurückgegeben werden. Ist noch kein Spieler im Team, dann soll der String `"Kein Spieler vorhanden"` zurückgegeben werden.

Lösungsvorschlag

```
public class Mannschaft {
  @SuppressWarnings("unused")
  private String land;
  private int anzahlSpieler;
  private Spieler[] team;

  public Mannschaft(String land) {
    this.land = land;
    anzahlSpieler = 0;
    team = new Spieler[20];
  }

  public void hinzufuegen(Spieler s) {
    team[anzahlSpieler] = s;
    anzahlSpieler++;
  }

  public String torschuetzenkoenig() {
    if (anzahlSpieler == 0) {
      return "Kein Spieler vorhanden";
    }
    Spieler koenig = team[0];
    for (int i = 0; i < anzahlSpieler; i++) {
```

main-Methode

```
      Spieler spieler = team[i];
      if (spieler.getTore() > koenig.getTore()) {
        koenig = spieler;
      }
    }
    return koenig.getName();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2018/fruehjahr/verein/Mannschaft.java`

(c) Schreiben Sie den Rumpf einer `main`-Methode, so dass nach Ausführung der Methode eine deutsche Mannschaft existiert mit zwei Spielern Namens `"Hugo Meier"` und `"Frank Huber"`. Beide Spieler sollen zum selben Verein `"FC Staatsexamen"` gehören. `"Hugo Meier"` soll nach Aufnahme in die deutsche Mannschaft genau ein Tor geschossen haben, während `"Frank Huber"` noch kein Tor erzielt hat. (Wir abstrahieren hier von der Realität, in der ein 2-er Team noch gar nicht spielbereit ist.)

Lösungsvorschlag

```
public class Verein {
  private String name;

  public Verein(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }

  public static void main(String[] args) {
    Verein verein1 = new Verein("FC Staatsexamen");
    Spieler spieler1 = new Spieler("Hugo Meier", verein1);
    Spieler spieler2 = new Spieler("Frank Huber", verein1);
    Mannschaft deutscheMannschaft = new Mannschaft("Deutschland");
    deutscheMannschaft.hinzufuegen(spieler1);
    deutscheMannschaft.hinzufuegen(spieler2);
    spieler1.schiesseTor();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2018/fruehjahr/verein/Verein.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Verhaltens-Modellierung mit Zustandsdiagrammen. Digitaluhr“ (46116-2018-H.T1-TA1-A3)

Zustandsdiagramm zeichnen

Eine Digitaluhr kann alternativ entweder die Zeit (Stunden und Minuten) oder das Datum (Tag, Monat und Jahr) anzeigen. Zu Beginn zeigt die Uhr die Zeit an. Sie besitzt drei Druckknöpfe **A**, **B** und **C**. Mit Knopf **A** kann zwischen Zeit- und Datumsanzeige hin und her gewechselt werden.

Wird die Zeit angezeigt, dann kann mit Knopf **B** der Reihe nach erst in einen Stundenmodus, dann in einen Minutenmodus und schließlich zurück zur Zeitanzeige gewechselt werden. Im Stundenmodus blinkt die Stundenanzeige. Mit Drücken des Knopfes **C** können dann die Stunden schrittweise inkrementiert werden. Im Minutenmodus blinkt die Minutenanzeige und es können mit Hilfe des Knopfes **C** die Minuten schrittweise inkrementiert werden.

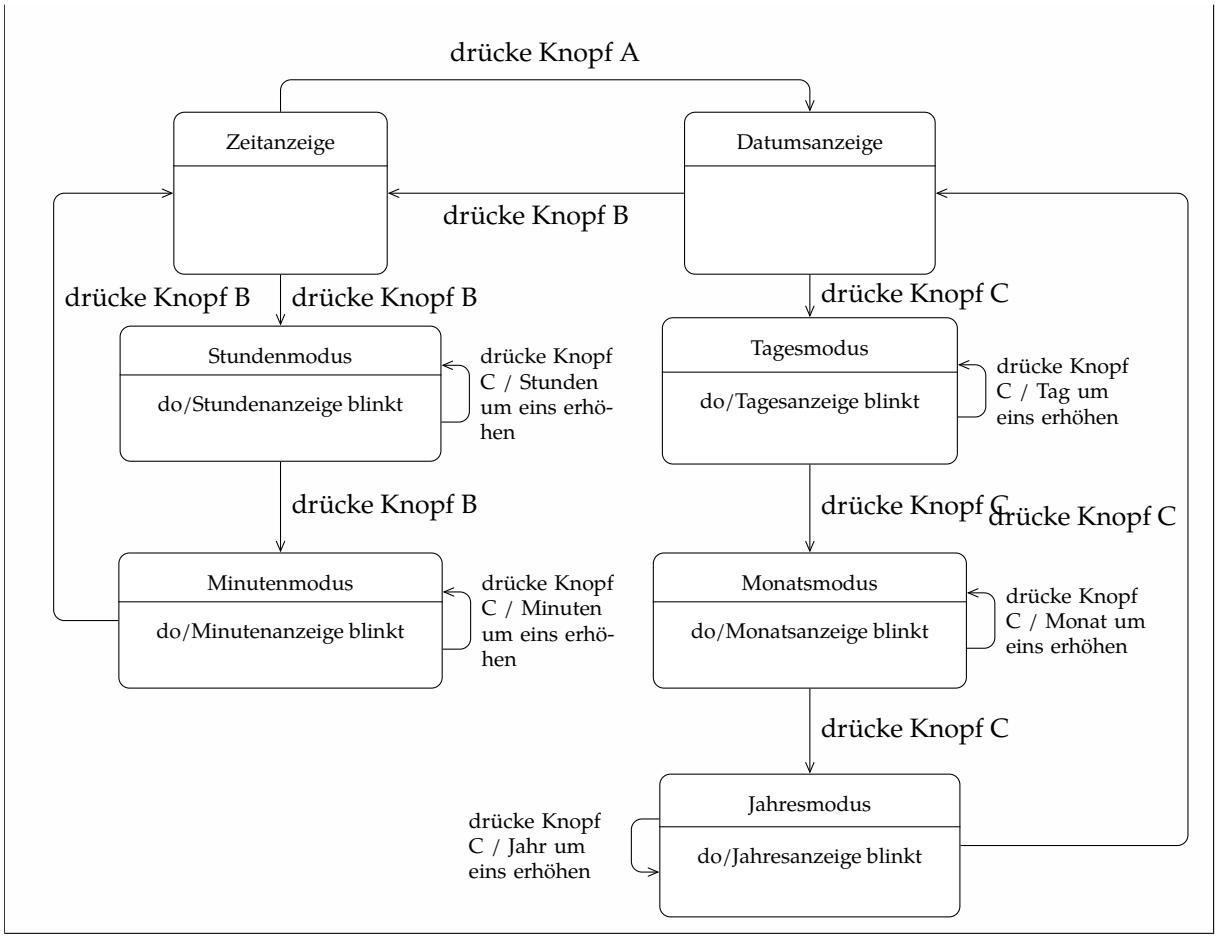
Die Datumsfunktionen sind analog. Wird das Datum angezeigt, dann kann mit Knopf **B** der Reihe nach in einen Tagesmodus, Monatsmodus, Jahresmodus und schließlich zurück zur Datumsanzeige gewechselt werden. Im Tagesmodus blinkt die Tagesanzeige. Mit Drücken des Knopfes **C** können dann die Tage schrittweise inkrementiert werden. Analog blinken mit Eintritt in den entsprechenden Einstellmodus der Monat oder das Jahr, die dann mit Knopf **C** schrittweise inkrementiert werden können.

Wenn sich die Uhr in einem Einstellmodus befindet, hat das Betätigen des Knopfes **A** keine Wirkung. Ebenso wirkungslos ist Knopf **C**, wenn gerade Zeit oder Datum angezeigt wird.

Beschreiben Sie das Verhalten der Digitaluhr durch ein UML-Zustandsdiagramm. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. Deren Bedeutung soll durch die Verwendung von sprechenden Namen klar sein. Für die Inkrementierung von Stunden, Minuten, Tagen etc. brauchen keine konkreten Berechnungen angegeben werden. Der kontinuierliche Zeitfortschritt des Uhrwerks ist nicht zu modellieren.

Zustände sind, wie in der UML üblich, durch abgerundete Rechtecke darzustellen. Sie können unterteilt werden in eine obere und eine untere Hälfte, wobei der Name des Zustands in den oberen Teil und eine in dem Zustand auszuführende Aktivität in den unteren Teil einzutragen ist.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2018/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Banksystem“ (66112-2002-H.T1-A4)

Vererbung
Abstrakte Klasse
Implementierung in Java
Klassendiagramm

In einer Anforderungsanalyse für ein Banksystem wird der folgende Sachverhalt beschrieben:

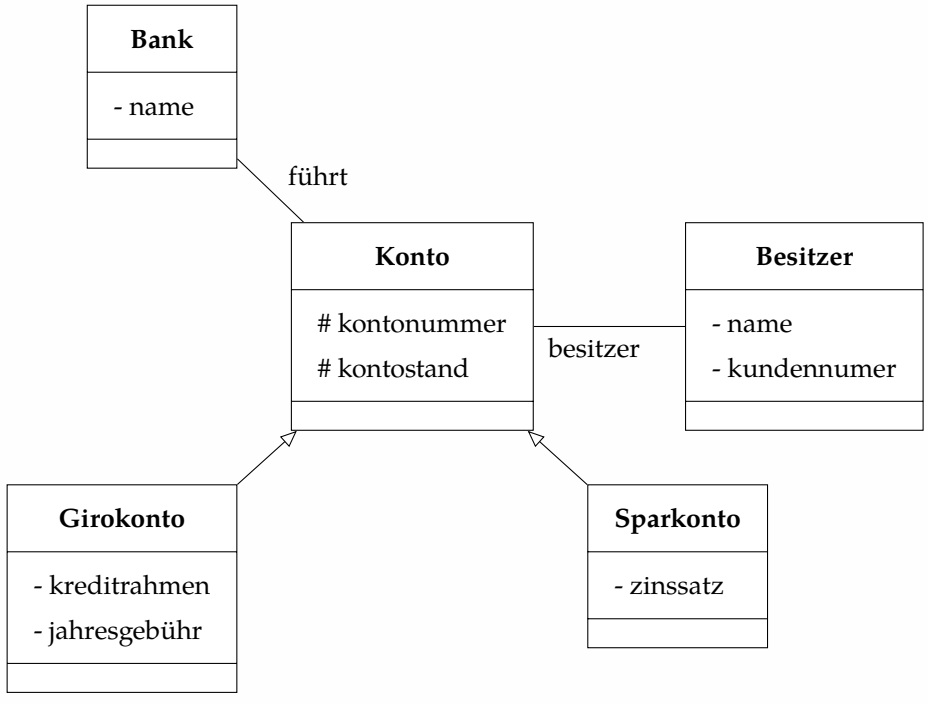
Eine Bank hat einen Namen und sie führt Konten. Jedes Konto hat eine Kontonummer, einen Kontostand und einen Besitzer. Der Besitzer hat einen Namen und eine Kundennummer. Ein Konto ist entweder ein Sparkonto oder ein Girokonto. Ein Sparkonto hat einen Zinssatz, ein Girokonto hat einen Kreditrahmen und eine Jahresgebühr.

(a) Deklarieren Sie geeignete Klassen in Java [oder in C++], die die oben beschriebenen Anforderungen widerspiegeln! Nutzen Sie dabei das Vererbungskonzept aus, wo es sinnvoll ist! Gibt es Klassen, die als abstrakt zu verstehen sind?

[Hinweis: Geben Sie sowohl ein Klassendiagramm als auch den Quellcode für die beteiligten Klassen inkl. Attributen, ohne Methoden an! Achtung: in Teilaufgabe b) werden anschließend die benötigten Konstruktoren verlangt; Um die verschiedenen Konten in einer Bank zu verwalten, eignet sich das Array NICHT als Datenstruktur. Informieren Sie sich über die Datenstruktur „ArrayList“ und verwenden Sie diese.]

Lösungsvorschlag

Die Klasse Konto ist abstrakt, da ein Konto immer entweder ein Spar- *oder* ein Girokonto ist. Ein Objekt der Klasse Konto ist deshalb nicht sinnvoll.



(b) Geben Sie für alle nicht abstrakten Klassen benutzerdefinierte Konstruktoren an mit Parametern zur Initialisierung der folgenden Werte: der Name einer Bank, die Kontonummer, der Kontostand, der Besitzer und der Zinssatz (bzw. Kreditrahmen und Jahresgebühr) eines Sparkontos (bzw. Girokontos), der Name und die Kundennummer eines Kontobesitzers.

Ergänzen Sie die Klassen um Methoden für die folgenden Aufgaben! Nutzen Sie wann immer möglich das Vererbungskonzept aus und verwenden Sie ggf. abstrakte (bzw. virtuelle) Methoden!

[Achtung: Ergänzen Sie ggf. alle benötigten Getter und Setter in den beteiligten Klassen!]

Lösungsvorschlag

Konstruktoren ergänzen:

Bemerkung: Auch eine abstrakte Klasse kann einen Konstruktor besitzen, dieser kann nur nicht ausgeführt werden. In den abgeleiteten Klassen kann dieser Super-Konstruktor aber verwendet werden.

(c) Auf ein Konto soll ein Betrag eingezahlt und ein Betrag abgehoben werden können. Soll von einem Sparkonto ein Betrag abgehoben werden, dann darf der Kontostand nicht negativ werden. Bei einer Abhebung von einem Girokonto darf der Kreditrahmen nicht überzogen werden.

Lösungsvorschlag

Bemerkung: Die Methode `einzahlen` ist in der Klasse `Konto` implementiert, da sie sich für Spar- und Girokonten nicht unterscheidet im Gegensatz zur Methode `abheben`, die in beiden Klassen unterschiedlich implementiert ist. [Sie wird als abstrakte Methode in der Klasse Konto angegeben, um ihre Implementierung (Überschreiben) zu gewährleisten.] Kreditrahmen wird als negativer Wert gespeichert. Die Methoden zum Abheben liefern zusätzlich zur Änderung des Kontostandes eine Rückmeldung bezüglich Erfolg oder Misserfolg der Abbuchung.

(d) Ein Konto kann eine Jahresabrechnung durchführen. Bei der Jahresabrechnung eines Sparkontos wird der Zinsertrag gutgeschrieben, bei der Jahresabrechnung eines Girokontos wird die Jahresgebühr abgezogen (auch wenn dadurch der Kreditrahmen überzogen wird).

Lösungsvorschlag

Anmerkung: Im Folgenden nur noch Angabe der gesuchten Methoden, alle vorherigen Implementierungen (Attribute, Konstruktoren, Methoden, s. o.) wurden nicht nochmals aufgeführt.

(e) Eine Bank kann einen Jahresabschluss durchführen. Dieser bewirkt, dass für jedes Konto der Bank eine Jahresabrechnung durchgeführt wird.

(f) Eine Bank kann ein Sparkonto eröffnen. Die Methode soll die folgenden fünf Parameter haben: den Namen und die Kundennummer des Kontobesitzers, die Kontonummer, den (anfänglichen) Kontostand und den Zinssatz des Sparkontos. Alle Parameter sind als String-Objekte oder als Werte eines Grunddatentyps zu übergeben! Das Sparkonto muss nach seiner Eröffnung in den Kontenbestand der Bank aufgenommen sein.

[Hinweis: Falls der Kunde schon mit einem Konto in der Bank geführt ist, können die Werte für das `Besitzer`-Objekt übernommen werden. Schreiben Sie daher eine Hilfsmethode `Besitzer schonVorhanden(String name, int kunr)`, die prüft, ob der Kunde mit `name` und `kunr` schon in der Liste vorhanden ist und diesen bzw. andernfalls einen neu erzeugten Besitzer zurückgibt.]

[Sinnvolle/notwendige Methoden der Klasse ArrayList für diese Aufgabe:
`public int size()` : Returns the number of elements in this list.
`public boolean isEmpty()` : Returns true if this list contains no elements.
`public E get(int index)` : Returns the element at the specified position in this list.
`public boolean add(E e)` : Appends the specified element to the end of this list.
(siehe auch Java-Api: https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html)]

Lösungsvorschlag

```java
import java.util.ArrayList;

public class Bank {
  @SuppressWarnings("unused")
  private String name;
  private ArrayList<Konto> konten;

  public Bank(String name) {
    this.name = name;
    konten = new ArrayList<Konto>();
  }

  public void rechneAb() {
    for (int i = 0; i <= konten.size(); i++) {
      konten.get(i).rechneAb();
    }
  }

  public void legeAn(String name, int kundenNummer, int kontoNummer, float
  ↪ kontoStand, float zinssatz) {
    Besitzer besitzer = schonVorhanden(name, kundenNummer);
    konten.add(new Sparkonto(kontoNummer, kontoStand, besitzer, zinssatz));
  }

  public Besitzer schonVorhanden(String name, int kundenNummer) {
    if (!konten.isEmpty()) {
      for (int i = 0; i < konten.size(); i++) {
        if (konten.get(i).besitzer.gibKundenNummer() == kundenNummer) {
          return konten.get(i).gibBesitzer();
        }
      }
    }
    return new Besitzer(name, kundenNummer);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Bank.java`

```java
public abstract class Konto {
```

```
  protected int kontoNummer;
  protected float kontoStand;
  protected Besitzer besitzer;

  public Konto(int kontoNummer, float kontoStand, Besitzer besitzer) {
    this.kontoNummer = kontoNummer;
    this.kontoStand = kontoStand;
    this.besitzer = besitzer;
  }

  public void zahleEin(float betrag) {
    kontoStand += betrag;
  }

  public Besitzer gibBesitzer() {
    return besitzer;
  }

  public abstract boolean hebeAb(float betrag);

  public abstract void rechneAb();

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Konto.java`

```
public class Sparkonto extends Konto {
  private float zinssatz;

  public Sparkonto(int kontoNummer, float kontoStand, Besitzer besitzer,
↪   float zinssatz) {
    super(kontoNummer, kontoStand, besitzer);
    this.zinssatz = zinssatz;
  }

  public boolean hebeAb(float betrag) {
    if (kontoStand - betrag >= 0) {
      kontoStand -= betrag;
      return true;
    } else {
      return false;
    }
  }

  public void rechneAb() {
    kontoStand += kontoStand * (1 + zinssatz);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Sparkonto.java`

```
public class Girokonto extends Konto {
  private float kreditRahmen;
```

```
  private float jahresGebühr;

  public Girokonto(int kontoNummer, float kontoStand, Besitzer besitzer,
↪   float kreditRahmen, float jahresGebühr) {
    super(kontoNummer, kontoStand, besitzer);
    this.kreditRahmen = kreditRahmen;
    this.jahresGebühr = jahresGebühr;
  }

  public boolean hebeAb(float betrag) {
    if (kontoStand - betrag >= kreditRahmen) {
      kontoStand -= betrag;
      return true;
    } else {
      return false;
    }
  }

  public void rechneAb() {
    kontoStand -= jahresGebühr;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Girokonto.java`

```
@SuppressWarnings("unused")
public class Besitzer {
  private String name;
  private int kundenNummer;
  private Konto hatKonto;

  public Besitzer(String name, int kundenNummer) {
    this.name = name;
    this.kundenNummer = kundenNummer;
  }

  public int gibKundenNummer() {
    return kundenNummer;
  }
}
```

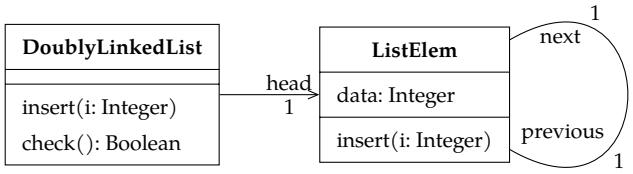
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Besitzer.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2002/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Klasse „DoublyLinkedList““ (66112-2005-F.T1-A1)

Klassendiagramm

Betrachten Sie folgendes Klassendiagramm, das doppelt-verkettete Listen spezifiziert. Die Assoziation `head` zeigt auf das erste Element der Liste. Die Assoziationen `previous` und `next` zeigen auf das vorherige bzw. folgende Element.



Implementieren Sie die doppelt-verketteten Listen in einer geeigneten objektorientierten Sprache (z. B. Java oder C++), das heißt:

(a) Implementieren Sie die Klasse `ListElem`. Die Methode `insert` ordnet eine ganze Zahl `i` in eine aufsteigend geordnete doppelt-verkettete Liste `l` an die korrekte Stelle ein. Sei z. B. das Objekt `l` eine Repräsentation der Liste `[0, 2, 2, 6, 8]` dann liefert `l.insert(3)` eine Repräsentation der Liste `[0, 2, 2, 3, 6, 8]`.

Lösungsvorschlag

```
public class ListElem {
  private int data;
  private ListElem previous;
  private ListElem next;

  public ListElem(int i) {
    data = i;
  }

  public ListElem() {
  }

  public void insert(int i) {
    ListElem newElement = new ListElem(i);
    if (i <= data) {
      if (previous != null) {
        newElement.next = this;
        newElement.previous = previous;
        previous.next = newElement;
        previous = newElement;
      } else {
        newElement.next = this;
        previous = newElement;
      }
    } else {
      if (next != null) {
        next.insert(i);
      } else {
        newElement.previous = this;
```

```
        next = newElement;
      }
    }
  }

  public ListElem getPrevious() {
    return previous;
  }

  public ListElem getNext() {
    return next;
  }

  public int getData() {
    return data;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java`

(b) Implementieren Sie die Klasse `DoublyLinkedList`, wobei die Methode `insert` eine Zahl `i` in eine aufsteigend geordnete Liste einordnet. Die Methode `check` überprüft, ob eine Liste korrekt verkettet ist, ðob für jedes ListElem-Objekt `o`, das über den `head` der Liste erreichbar ist, der Vorgänger des Nachfolgers von `o` gleich `o` ist.

Lösungsvorschlag

```
public class DoublyLinkedList {
  private ListElem head;

  public DoublyLinkedList() {
  }

  public void insert(int i) {
    if (head != null) {
      // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist als
↪   head.
      head.insert(i);
      // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen am
↪   Anfang
      // nicht eingefügt werden.
      if (i <= head.getData()) {
        head = head.getPrevious();
      }
    } else {
      head = new ListElem(i);
    }
  }

  public boolean check() {
    ListElem current = head;
    while (current.getNext() != null) {
      if (current.getNext().getPrevious() != current) {
```

```
        return false;
      } else {
        current = current.getNext();
      }
    }
    return true;
  }

  public ListElem getHead() {
    return head;
  }

  public static void main(String[] args) {
    DoublyLinkedList list = new DoublyLinkedList();
    // int[] numbers = new int[] { 1 };
    // int[] numbers = new int[] { 1, 1, 1, 1, };
    // int[] numbers = new int[] { 1, 1, 1, 2, };
    // int[] numbers = new int[] { 2, 1, 1, 1, };
    // int[] numbers = new int[] { 2, 1 };
    int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
    for (int number : numbers) {
      list.insert(number);
    }
    list.insert(3);

    ListElem current = list.getHead();
    while (current.getNext() != null) {
      System.out.println(current.getData());
      current = current.getNext();
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2005/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Wahlsystem“ (66116-2014-H.T2-TA2-A2)

Modell-Präsentation-Steuerung (Model-View-Controller) Implementierung in Java

Sie sollen das Design für ein einfaches Wahlsystem entwerfen. Das System soll dabei die Verteilung der Stimmen auf die einzelnen Parteien ermöglichen. Zusätzlich soll es verschiedene Darstellungen dieser Daten erlauben: Eine Tabelle, in der die Daten gelesen und auch eingegeben werden können, und ein Diagramm als alternative Darstellung der Informationen. Das System soll mit dem *Model-View-Controller* Muster modelliert werden.

(a) Beschreiben Sie das *Model-View-Controller* Muster:

(i) Beschreiben Sie das Problem, welches das Muster adressiert.

Lösungsvorschlag

> Das MVC-Muster wird verwendet, um spätere Änderungen bzw. Erweiterungen zu vereinfachen. Dies unterstützt somit auch die Wiederverwendbarkeit.

(ii) Beschreiben Sie die Aufgaben der Komponenten, die im Muster verwendet werden.

Lösungsvorschlag

> Im Modell werden die Daten verwaltet, die View ist für die Darstellung der daten sowie die Benutzerinteraktion zuständig und der Controller übernimmt die Steue- rung zwischen View und Modell.
>
> Das MVC-Muster ist aus den drei Entwurfsmustern Beobachter, Kompositum und Strategie zusammengesetzt.

(b) Modellieren Sie das System unter Anwendung des Musters:

(i) Entwerfen Sie ein UML Klassendiagramm.

(ii) Implementieren Sie eine setup-Methode, die das Objektmodell erstellt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2014/09/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

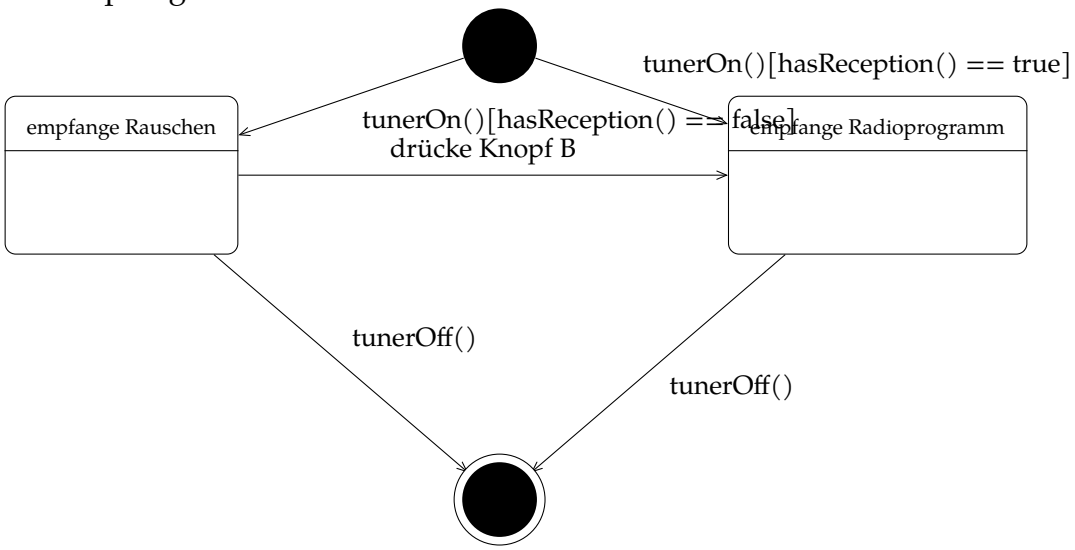
Examensaufgabe „Radiotuner“ (66116-2015-F.T1-TA2-A2)

Zustandsdiagramm zeichnen

Erstellen Sie ein UML-Zustandsdiagramm für einen Radiotuner. Mit dem Radiotuner können Sie ein Radioprogramm auf der Frequenz f empfangen. Beim Einschalten wird f auf 87,5 MHz gesetzt und der Tuner empfängt. Sie können nun die Frequenz in 0,5 MHz Schritten erhöhen oder senken. Bitte beachten Sie, dass das Frequenzband für den Radioempfang von 87,5 MHz bis 108 MHz reicht. Sobald f 87,5 MHz unter- bzw. 108 MHz überschreitet, soll f auf 108 MHz bzw. 87,5 MHz gesetzt werden. Weiterhin kann ein Suchmodus gestartet werden, der automatisch die Frequenz erhöht, bis ein Sender empfangen wird. Wird während des Suchmodus die Frequenz verändert oder erneut Suchen ausgeführt, dann wird die Suche beendet (und, je nach Knopf, ggf. noch die Frequenz um 0,5 MHz verändert).
Die Klasse RadioTuner besitzt folgende Methoden:

- tunerOn()
- tunerOff()
- increaseFrequency()
- decreaseFrequency()
- seek()
- hasReception()

Hinweis: Die Hilfsmethode hasReception() liefert true zurück, genau dann wenn ein Sender empfangen wird.



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Kunden und Angestellte einer Firma“ (66116-2015-F.T1-TA2-A3)

Implementierung in Java
Vererbung
Interface
Abstrakte Klasse

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine *Person* hat einen *Namen* und ein *Geschlecht* (männlich oder weiblich).
- Ein *Angestellter* ist eine *Person*, zu der zusätzlich das monatliche *Gehalt* gespeichert wird.
- Ein *Kunde* ist eine *Person*, zu der zusätzlich eine *Kundennummer* hinterlegt wird.

(a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:

- Schreiben Sie ein Interface `Person` sowie zwei davon erbende Interfaces `Angestellter` und `Kunde`. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.

Lösungsvorschlag

```java
public interface Person {
  String getName();

  char getGeschlecht();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Person.java`

```java
public interface Angestellter extends Person {
  int getGehalt();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Angestellter.java`

```java
public interface Kunde extends Person {
  int getKundennummer();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Kunde.java`

- Schreiben Sie eine abstrakte Klasse `PersonImpl`, die das Interface `Person` implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Lösungsvorschlag

```java
public abstract class PersonImpl implements Person {
  protected String name;
  protected char geschlecht;
```

```
  public PersonImpl(String name, char geschlecht) {
    this.name = name;
    this.geschlecht = geschlecht;
  }

  public String getName() {
    return name;
  }

  public char getGeschlecht() {
    return geschlecht;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/PersonImpl.java`

- Schreiben Sie zwei Klassen `AngestellterImpl` und `KundeImpl`, die von `PersonImpl` erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder initialisieren und dabei auf den Konstruktor der Basisklasse `PersonImpl` Bezug nehmen.

Lösungsvorschlag

```
public class AngestellterImpl extends PersonImpl implements Angestellter
↪  {
  protected int gehalt;

  public AngestellterImpl(String name, char geschlecht, int gehalt) {
    super(name, geschlecht);
    this.gehalt = gehalt;
  }

  public int getGehalt() {
    return gehalt;
  }

}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/AngestellterImpl.java`

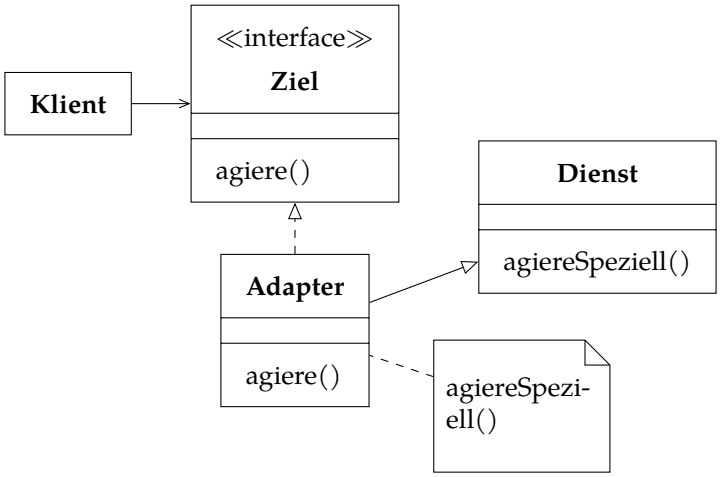
```
public class KundeImpl extends PersonImpl implements Kunde {
  protected int kundennummer;

  public KundeImpl(String name, char geschlecht, int kundennummer) {
    super(name, geschlecht);
    this.kundennummer = kundennummer;
  }

  public int getKundennummer() {
    return kundennummer;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/KundeImpl.java`

Adapter

(b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

Exkurs: Entwurfsmuster „Adapter“



Ziel (Target) Das Ziel definiert die Schnittstelle, die der Klient nutzen kann.

Klient (Client) Der Klient nutzt Dienste über inkompatible Schnittstellen und greift dabei auf adaptierte Schnittstellen zurück.

Dienst (Adaptee) Der Dienst bietet wiederzuverwendende Dienstleistungen mit fest definierter Schnittstelle an.

Adapter Der Adapter adaptiert die Schnittstelle des Dienstes auf die Schnittstelle zum Klienten.

```java
/**
 * GoF: Adaptee
 */
public class Dienst {

  public void agiereSpeziell() {
      System.out.println("Agiere speziell!");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Dienst.java`

```java
public interface Ziel {
  public void agiere();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Ziel.java`

```java
public class Adapter extends Dienst implements Ziel {

  @Override
  public void agiere() {
      System.out.print("agiere: ");
      agiereSpeziell();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Adapter.java`

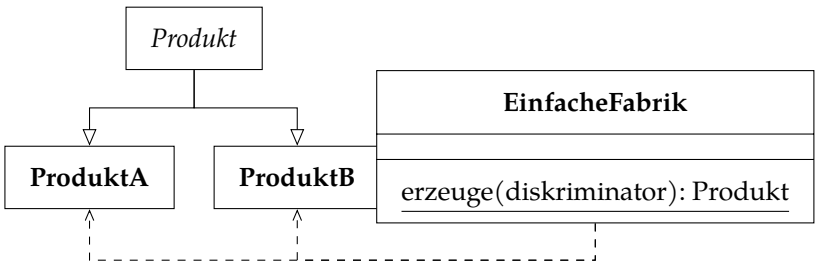
```java
public class Klient {

  public static void main(String[] args) {
    new Adapter().agiere();
    // agiere: Agiere speziell!
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java`

(c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z. B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface `Person` notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

Exkurs: Entwurfsmuster „Einfache Fabrik“



EinfacheFabrik Eine Klasse mit einer Erzeugunsmethode, die über eine größere Bedingung verschiedene Objekt instanziert.

Produkt Eine abstrakte Klasse, die von den konkreten Produkten geerbt wird.

KonkretesProdukt Ein konkretes Produkt, das von der einfachen Fabrik erzeugt wird.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „OOP/OOD - Reverse Engineering“ (66116-2015-F.T2-TA2-A2)

Klassendiagramm

Leider ist das Klassendiagramm der folgenden Klassen verloren gegangen. Führen Sie ein Reverse Engineering durch und erstellen Sie aus dem Quellcode ein vollständiges UML-Klassendiagramm inklusive aller Klassen, Schnittstellen, Attribute, Methoden, Konstruktoren, Sichtbarkeiten, Assoziationen, Rollennamen, Multiplizitäten, Navigationspfeilen und evtl. Stereotypen. Der Quellcode innerhalb von Methoden und Konstruktoren soll nicht übertragen werden, wohl aber die Methodensignaturen. Assoziationsnamen und deren Leserichtung lassen sich aus dem Quellcode nur schwer erahnen und sollen deshalb ebenfalls weggelassen werden.

```
public abstract class Display implements PixelPainter {
  protected HardwareMatrix hardwareMatrix;
  protected int lastPaintedX;
  protected int lastPaintedY;

  public Display(HardwareMatrix hardwareMatrix) {
    this.hardwareMatrix = hardwareMatrix;

  }

  public int getWidth() {
    return hardwareMatrix.getWidth() / getWidthFactor();
  }

  public int getHeight() {
    return hardwareMatrix.getHeight() / getHeightFactor();
  }

  public void clear() {
    // some longer code
  }

  protected abstract int getWidthFactor();

  protected abstract int getHeightFactor();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/Display.java`

```
import java.awt.Color;

public interface PixelPainter {
  void set(int x, int y, Color color);

  int getHeight();

  int getWidth();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/PixelPainter.java`

```
public interface HardwareMatrix {
  void set(int x, int y, int v);

  int getWidth();

  int getHeight();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/HardwareMatrix.java`

```
import java.awt.Color;

@SuppressWarnings({ "unused" })
public class DisplayUnion extends RGBDisplay {
  public static final int MAX_DISPLAY_COUNT = 50;

  private int currentDisplayCount;

  private Display[] displays;

  public DisplayUnion(Display[] displays) {
    super(null);
  }

  public int getDisplayCount() {
    return 0;
  }

  protected int getWidthFactor() {
    return 1;
  }

  protected int getHeightFactor() {
    return 1;
  }

  public void set(int x, int y, Color color) {
  }
}
```

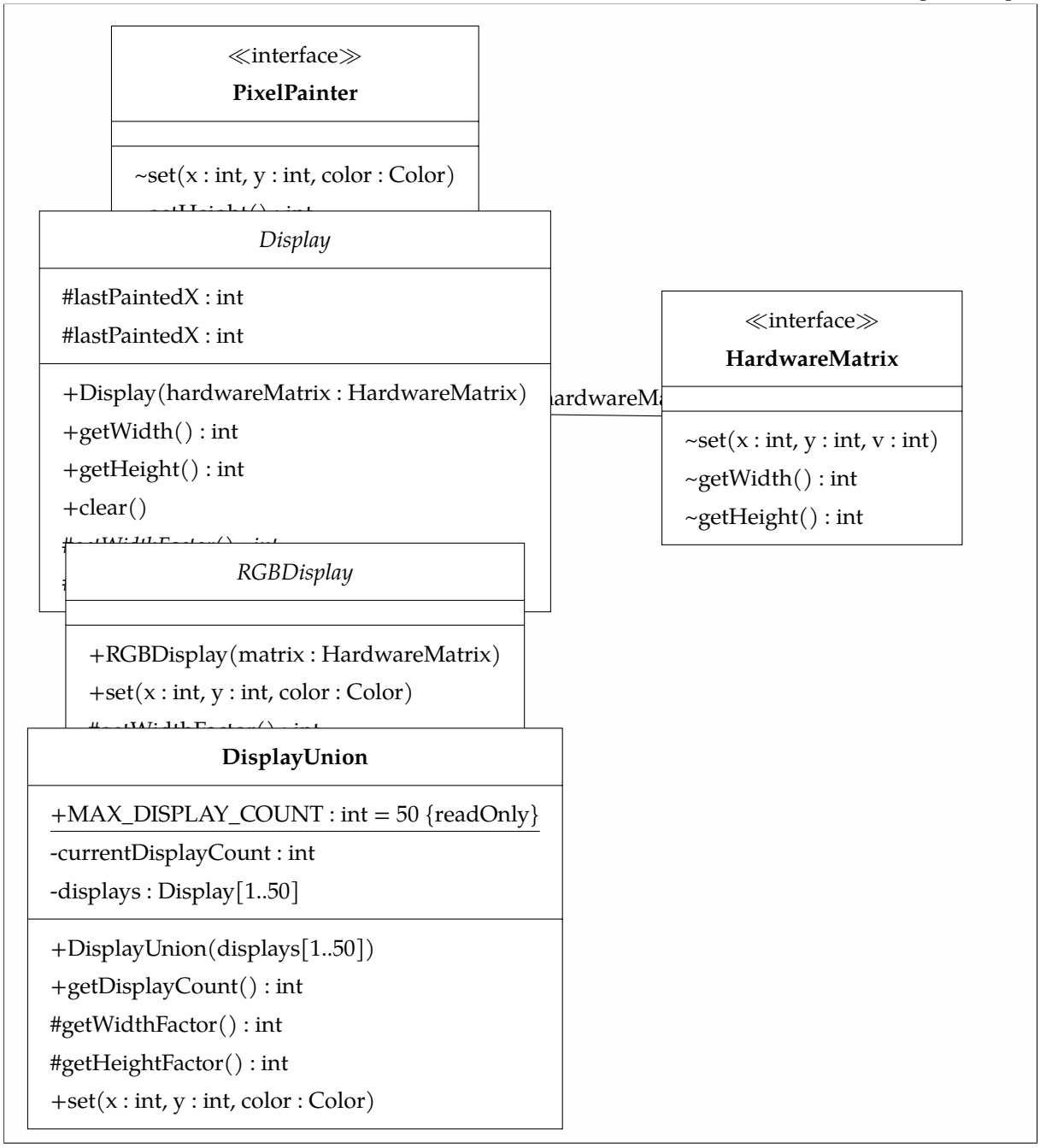
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/DisplayUnion.java`

```
import java.awt.Color;

public class RGBDisplay extends Display {
  public RGBDisplay(HardwareMatrix matrix) {
    super(matrix);
  }

  public void set(int x, int y, Color color) {
  }

  protected int getWidthFactor() {
    return 3;
  }
```

```
  protected int getHeightFactor() {
    return 1;
  }
}
```

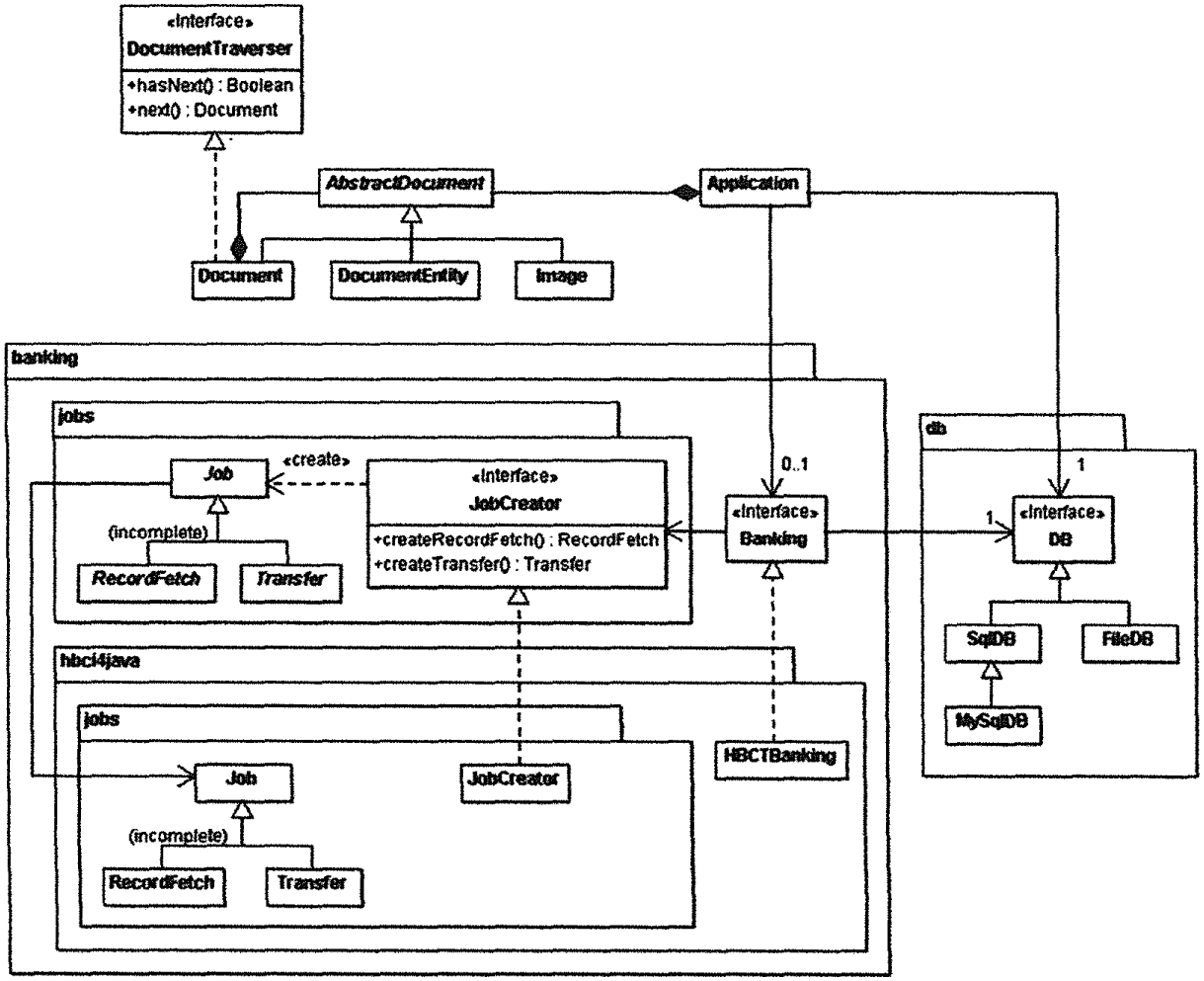
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/RGBDisplay.java`

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Entwurfsmuster in UML-Diagramm erkennen“ (66116, 2016-F.T1-TA2-A2)



(a) Kennzeichnen Sie im folgenden Klassendiagramm die Entwurfsmuster *„Abstrakte Fabrik“*, *„Iterator“*, *„Adapter “* und *„Kompositum“*. Geben Sie die jeweils beteiligten Klassen und deren Zuständigkeit im entsprechenden Muster an.

Lösungsvorschlag

Iterator

DocumentTraverser (interface) Schnittstelle zur Traversierung und zum Zugriff auf Dokumente

Document implementiert die Schnittstelle

Kompositum

AbstractDocument abstrakte Basisklasse, die gemeinsames Verhalten der beteiligten Klassen definiert

Document enthält wiederum weitere Documente bzw. DocumentEntities und Images

DocumentEntity, Image primitive Unterklassen, besitzen keine Kind-

objekte

Adapter (Objektadapter)

Banking (interface) vom Client (hier Application) verwendete Schnittstelle

HBCTBanking passt Schnittstelle der unpassenden Klasse an Zielschnittstelle (Banking) an

DB (interface) anzupassende Schnittstelle

abstrakte Fabrik

JobCreator (interface) abstrakte Fabrik

Job (abstrakt) mit Unterklassen RecordFetch und Transfer abstraktes Produkt

Job (konkret) mit Unterklassen konkretes Produkt

(b) (i) Beschreiben Sie die Funktionsweise der folgenden Entwurfsmuster und geben Sie ein passendes UML-Diagramm an.

- Dekorierer
- Klassenadapter
- Objektadapter

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(ii) Erklären Sie mit maximal zwei Sätzen den Unterschied zwischen Klassenadapter und Objektadapter.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(c) Implementieren Sie einen Stapel in der Programmiersprache Java. Nutzen Sie dazu ein Array mit fester Größe. Auf eine Überlaufprüfung darf verzichtet werden. Implementieren Sie in der Klasse das Iterator Entwurfsmuster, um auf die Inhalte zuzugreifen, sowie eine Funktion zum Hinzufügen von Elementen. Als Typ für den Stapel kann zur Vereinfachung ein Integertyp verwendet werden.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkor-

rekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „PKI-System Lehrer Schüler“ (66116-2016-H.T1-TA2-A2)

UML-Diagramme
Anwendungsfalldiagramm
Klassendiagramm

(a) Gegeben sei folgende natürlichsprachliche Spezifikation eines PKI-Systems:

Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „Öffentlichen“ (ÖS) und einem zugehörigen „privaten“ Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z. B. per E-Mail oder USB-Stick) an den Schüler übermitteln.

Der Schüler kann nun seine Nachricht (z. B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit seinem PS beliebige Informationen (z. B. die Note) „digital signieren“. Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.

Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Aktoren in einem Use-Case-Diagramm. Betrachten Sie den Keyserver zunächst ebenfalls als Aktor, welcher gegenüber PKI als „externer Vermittler“ auftritt.

(b) Erstellen Sie ein geeignetes Klassendiagramm für das obige PKI. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:

(i) Die Schlüssel werden mit einer E-Mail-Adresse „benannt“, damit der Schüler den richtigen Schlüssel abrufen kann.

(ii) Es gibt genau einen Keyserver; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.

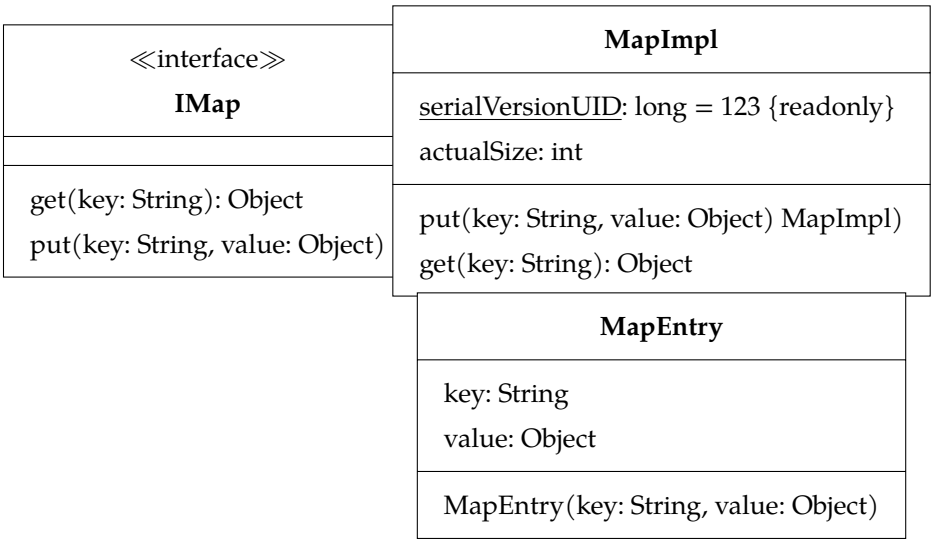
(iii) Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.

(iv) Eine Nachricht kann (muss aber nicht) eine Signatur oder einen ÖS als „Anhang“ zusätzlich zum eigentlichen Inhalt (zur Vereinfachung: String) mit sich führen.

(v) Für das Signieren bzw. Entschlüsseln ist der PS (das zugehörige Objekt selbst) zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.

(vi) Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.

(c) Übertragen Sie folgendes UML-Klassendiagramm in Programm-Code einer gängigen und geeigneten objektorientierten Sprache Ihrer Wahl. Die Methodenrümpfe dürfen Sie dabei leer lassen (auch wenn das Programm dann nicht übersetzbar bzw. ausführbar wäre).



Lösungsvorschlag

Interface „IMap“

```
interface IMap {
  Object get(String key);

  void put(String key, Object value);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/IMap.java`

Klasse „MapImpl“

```
import java.util.ArrayList;
import java.util.List;

class MapImpl implements IMap {
  final static long serialVersionUID = 123;

  private List<MapEntry> entries;

  MapImpl() {
    entries = new ArrayList<MapEntry>();
```

```
  }

  public Object get(String key) {
    return new Object();
  }

  public void put(String key, Object value) {
    // Nicht verlangt in der Aufgabenstellung.
    entries.add(new MapEntry(key, value));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapImpl.java`

Klasse „MapEntry“

```
class MapEntry {
  String key;
  Object value;

  MapEntry(String key, Object value) {
    // Nicht verlangt in der Aufgabenstellung.
    this.key = key;
    this.value = value;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapEntry.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2016/09/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „UML-Diagramme entspreched Java-Code zeichnen“ (66116-2018-F.T2-TA2-A1)

Vererbung
Interface
Abstrakte Klasse
Klassendiagramm

Gegeben sei das folgende Java-Programm:

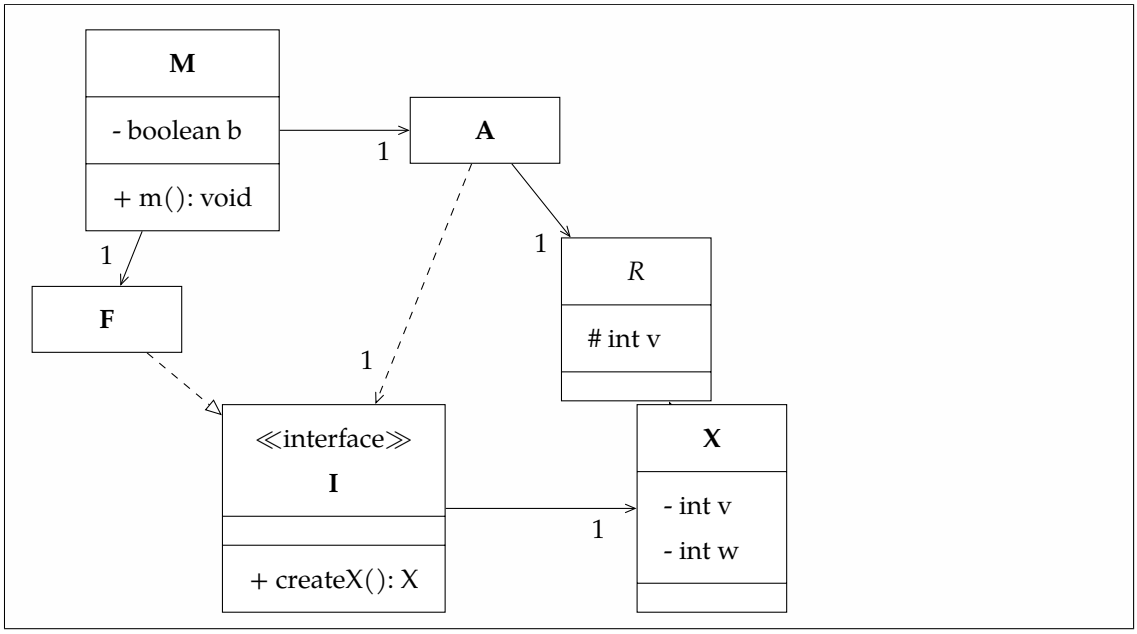
```
class M {
  private boolean b;
  private F f;
  private A a;

  public void m() {
    f = new F();
    a = new A(f);
    b = true;
  }
}

class A {
  private R r;
  public A(I i) {
    r = i.createX();
  }
}

interface I {
  public X createX();
}

class F implements I {
  public X createX() {
    return new X(0, 0);
  }
}

abstract class R {
  protected int v;
}

class X extends R {
  private int v, w;
  public X(int v, int w) {
    this.v = v;
    this.w = w;
  }
}
```

(a) Das Subtypprinzip der objektorientierten Programmierung wird in obigem Programmcode zweimal ausgenutzt. Erläutern Sie wo und wie dies geschieht.

(b) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des obigen Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen (insbesondere Interfaces, abstrakte Klassen, Zugriffsrech-

te, benutzerdefinierte Konstruktoren und Methoden) sollen in dem Klassendiagramm abgebildet werden.

Lösungsvorschlag



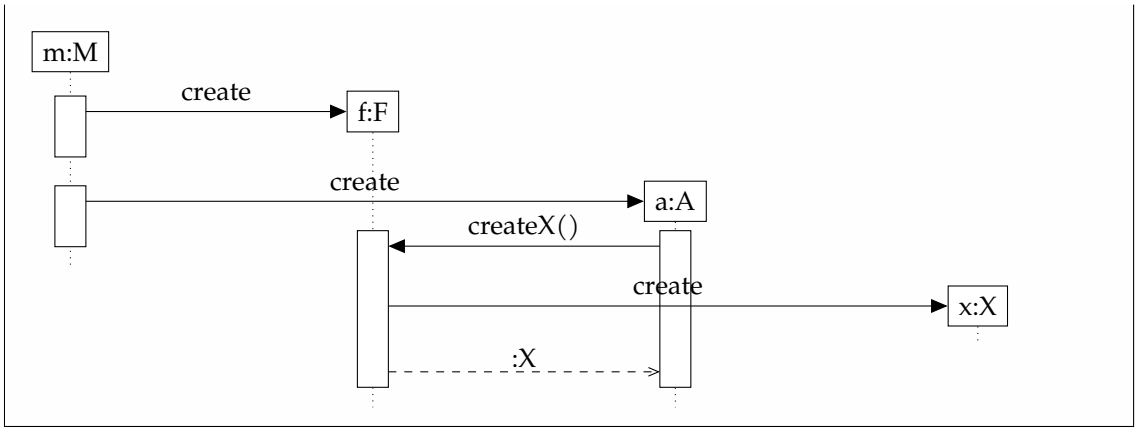
(c) Es wird angenommen, dass ein Objekt der Klasse `M` existiert, für das die Methode `m()` aufgerufen wird. Geben Sie ein Instanzendiagramm (Objektdiagramm) an, das alle nach der Ausführung der Methode `m` existierenden Objekte und deren Verbindungen (Links) zeigt.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(d) Wie in Teil c) werde angenommen, dass ein Objekt der Klasse M existiert, für das die Methode `m()` aufgerufen wird. Diese Situation wird in Abb. 1 dargestellt. Zeichnen Sie ein Sequenzdiagramm, das Abb. 1 so ergänzt, dass alle auf den Aufruf der Methode `m()` folgenden Objekterzeugungen und Interaktionen gemäß der im Programmcode angegebenen Konstruktor- und Methodenrümpfe dargestellt werden. Aktivierungsphasen von Objekten sind durch längliche Rechtecke deutlich zu machen.

Lösungsvorschlag



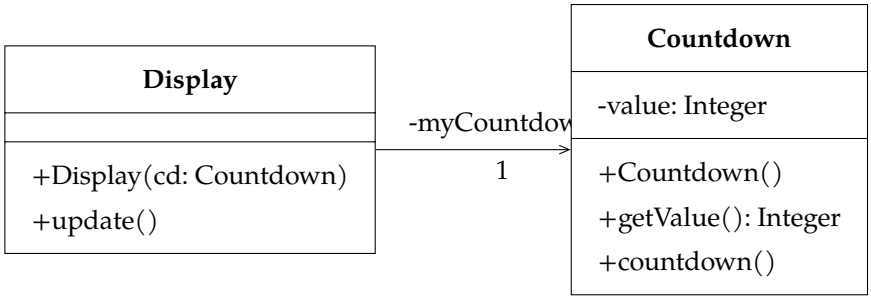
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Beobachter (Observer)

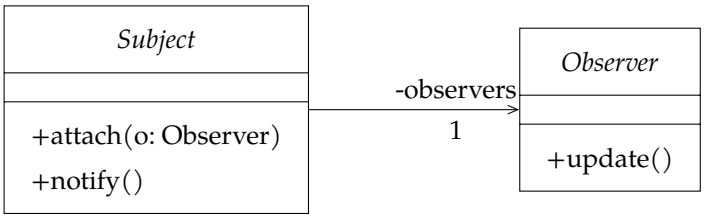
Examensaufgabe „Countdown und Observer“ (66116-2018-F.T2-TA2-A2)

Es soll eine (kleine) Anwendung entwickelt werden, in der ein Zähler in 1-er Schritten von 5000 bis 0 herunterzählt. Der Zähler soll als Objekt der Klasse `Countdown` realisiert werden, die in UML-Notation dargestellt ist. Das Attribut value soll den aktuellen Zählerstand speichern, der mit dem Konstruktor zu initialisieren ist. Die Methode `getValue` soll den aktuellen Zählerstand liefern und die Methode `countdown` soll den Zähler von 5000 bis 0 herunterzählen.

Der jeweilige Zählerstand soll von einem Objekt der in untenstehender Abbildung angegebenen Klasse `Display` am Bildschirm ausgegeben werden. Bei der Konstruktion eines `Display`-Objekts soll es mit einem `Countdown`-Objekt verbunden werden, indem dessen Referenz unter `myCountdown` abgespeichert wird. Die Methode `update` soll den aktuellen Zählerstand vom `Countdown`-Objekt holen und mit `System.out.println` am Bildschirm ausgeben. Dies soll zu Beginn des Zählprozesses und nach jeder Änderung des Zählerstands erfolgen.



Damit das `Display`-Objekt über Zählerstände des `Countdown`-Objekts informiert wird, soll das Observer-Pattern angewendet werden. Untenstehende Abbildung zeigt die im Observer-Pattern vorkommenden abstrakten Klassen. (Kursivschreibweise bedeutet abstrakte Klasse bzw. abstrakte Methode.)



(a) Welche Wirkung haben die Methoden `attach` und `notify` gemäß der Idee des Observer-Patterns?

Lösungsvorschlag

Das beobachtete Objekt bietet mit der Methode `attach` einen Mechanismus, um Beobachter anzumelden und diese über Änderungen zu informieren.

Mit der Methode `notify` werden alle Beobachter benachrichtigt, wenn sich das beobachtete Objekt ändert.

(b) Welche der beiden Klassen `Display` und `Countdown` aus obenstehender Abbildung spielt die Rolle eines `Subject` und welche die Rolle eines `Observer`?

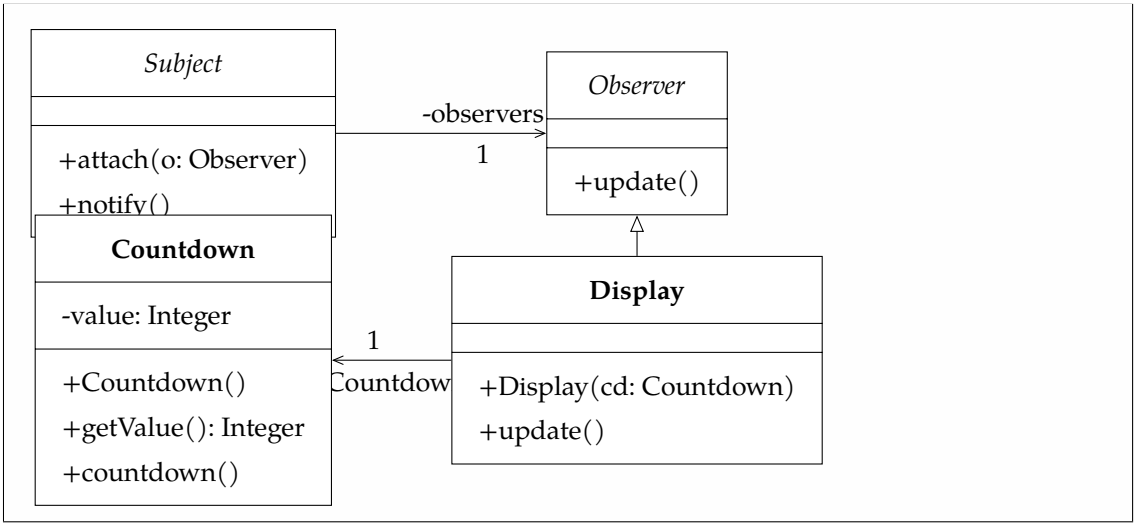
Klassendiagramm Implementierung in Java

Lösungsvorschlag

> Die Klasse `Countdown` spielt die Rolle des `Subject`s, also des Gegenstands, der beobachtet wird.
>
> Die Klasse `Display` spielt die Rolle eines `Observer`, also die Rolle eines Beobachters.

(c) Erstellen Sie ein Klassendiagramm, das die beiden obenstehenden gegebenen Diagramme in geeigneter Weise, ðentsprechend der Idee des Observer-Patterns, zusammenfügt. Es reicht die Klassen und deren Beziehungen anzugeben. Eine nochmalige Nennung der Attribute und Methoden ist nicht notwendig.

Lösungsvorschlag



(d) Unsere Anwendung soll nun in einer objektorientierten Programmiersprache Ihrer Wahl (z. B. Java oder C++) implementiert werden. Dabei soll von folgenden Annahmen ausgegangen werden:

- Das Programm wird mit einer main-Methode gestartet, die folgenden Rumpf hat:

```
public static void main(String[] args){
  Countdown cd = new Countdown();
  new Display(cd);
  cd.countdown();
}
```

- Die beiden Klassen Subject und Observer sind bereits gemäß der Idee des Observer-Patterns implementiert.

Geben Sie auf dieser Grundlage eine Implementierung der beiden Klassen `Display` und `Countdown` an, so dass das gewünschte Verhalten, ðAnzeige der Zählerstände und Herunterzählen des Zählers, realisiert wird. Die Methoden der Klassen `Subject` und `Observer` sind dabei auf geeignete Weise zu verwenden bzw. zu implementieren. Geben Sie die verwendete Programmiersprache an.

Lösungsvorschlag

```
public class Client {
  public static void main(String[] args){
    Countdown cd = new Countdown();
    new Display(cd);
    cd.countdown();
    cd.countdown();
    cd.countdown();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Client.java`

```
import java.util.ArrayList;
import java.util.List;

public abstract class Subject {
  private final List<Observer> observers = new ArrayList<Observer>();

  public void attach(Observer o) {
    observers.add(o);
  }

  public void notifyObservers() {
    for (Observer o : observers) {
      o.update();
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Subject.java`

```
public abstract class Observer {
  public abstract void update();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Observer.java`

```
public class Countdown extends Subject {

  private int value;

  public Countdown() {
    value = 5000;
  }

  public int getValue() {
    return value;
  }

  public void countdown() {
    if (value > 0) {
      notifyObservers();
```

```
      value--;
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Countdown.java`

```
public class Display extends Observer {
  Countdown myCountdown;
  public Display(Countdown cd) {
    myCountdown = cd;
    myCountdown.attach(this);
  }

  public void update() {
    System.out.println(myCountdown.getValue());
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Display.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/03/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Beatles“ (66116-2018-H.T1-TA1-A2)

Zustand (State)

Gegeben sei das Java-Programm:

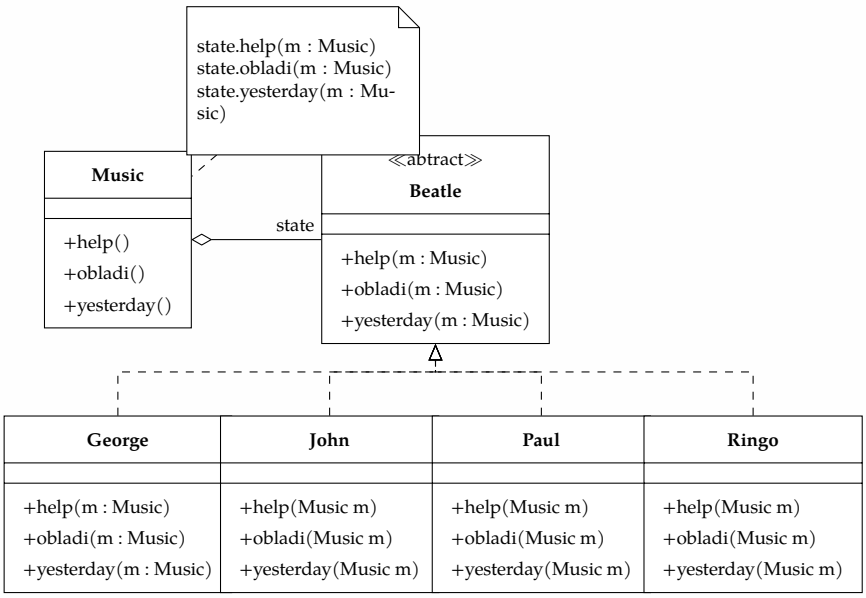
```
public class Music {
  private Beatle state;

  public Music() {
    state = new Paul();
  }

  public void help() {
    this.state.help(this);
  }

  public void obladi() {
    this.state.obladi(this);
  }

  public void yesterday() {
    this.state.yesterday(this);
  }

  public void setBeatle(Beatle b) {
    state = b;
  }
}

abstract class Beatle {
  public abstract void help(Music m);

  public abstract void obladi(Music m);

  public abstract void yesterday(Music m);
}

class George extends Beatle {
  public void help(Music m) {
    System.out.println("help");
    m.setBeatle(new John());
  }

  public void obladi(Music m) {
  }

  public void yesterday(Music m) {
    System.out.println("yesterday");
    m.setBeatle(new Paul());
  }
}

class John extends Beatle {
  public void help(Music m) {
    System.out.println("help");
    m.setBeatle(new Paul());
```

```
  }

  public void obladi(Music m) {
    System.out.println("obladi");
    m.setBeatle(new Ringo());
  }

  public void yesterday(Music m) {
  }
}

class Paul extends Beatle {
  public void help(Music m) {
    System.out.println("help");
    m.setBeatle(new George());
  }

  public void obladi(Music m) {
  }

  public void yesterday(Music m) {
    System.out.println("yesterday");
  }
}

class Ringo extends Beatle {
  public void help(Music m) {
    System.out.println("help");
    m.setBeatle(new John());
  }

  public void obladi(Music m) {
  }

  public void yesterday(Music m) {
    System.out.println("yesterday");
    m.setBeatle(new Paul());
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/herbst/beatles/Music.java`

(a) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und passender Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen sollen in dem Klassendiagramm dargestellt werden.

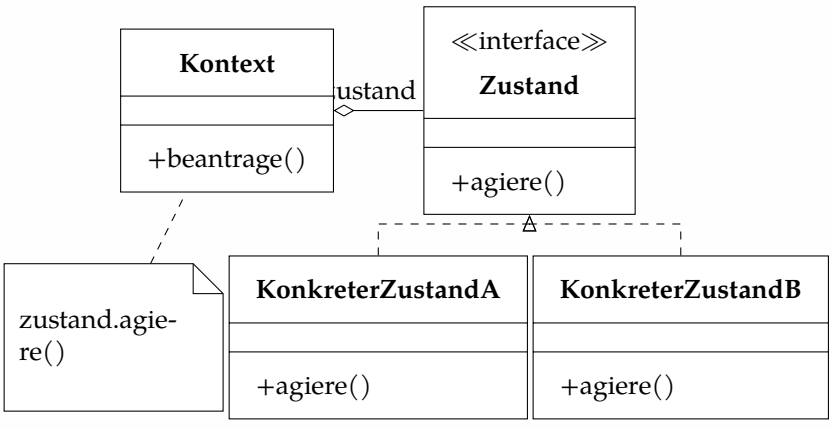
Lösungsvorschlag



Das Programm implementiert ein Zustandsdiagramm, das das Verhalten von Objekten der Klasse `Music` beschreibt. Für die Implementierung wurde das Design-Pattern `STATE` angewendet.

(c) Geben Sie die statische Struktur des STATE-Patterns an und erläutern Sie, welche Rollen aus dem Entwurfsmuster den Klassen des gegebenen Programms dabei zufallen und welche Operationen aus dem Entwurfsmuster durch (ggf. mehrere) Methoden in unserem Beispielprogramm implementiert werden. Es ist von den z. B. im Design-Pattern-Katalog von Gamma et al. verwendeten Namen auszugehen, das heißt von Klassen mit Namen `Context`, `State`, `ConcreteStateA`, `ConcreteStateB` und von Operationen mit Namen `request` und `handle`.

Lösungsvorschlag



Kontext (Context) definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

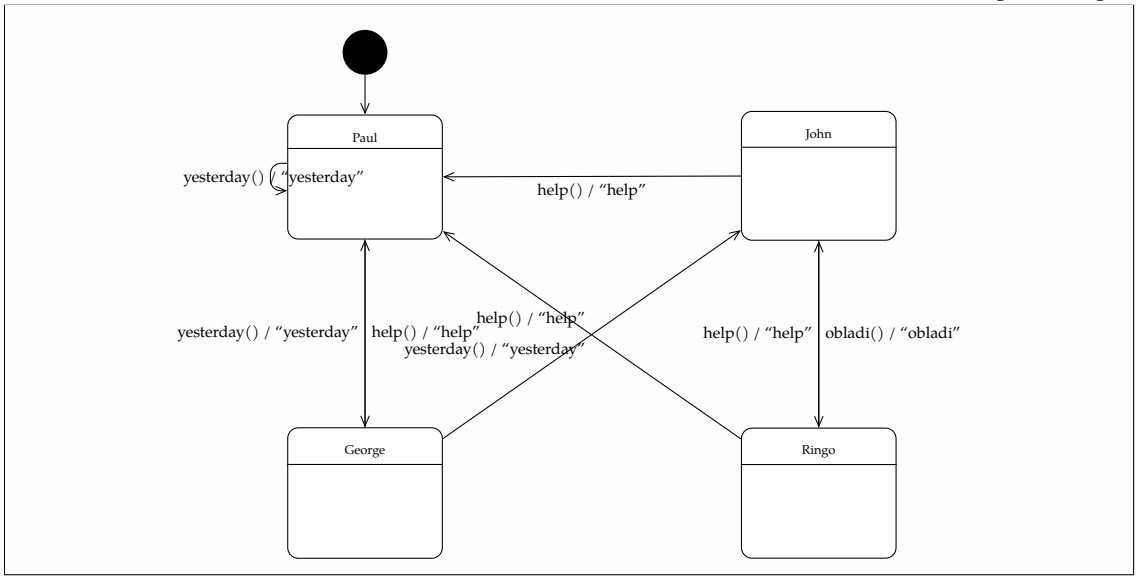
State (Zustand) definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

KontreterZustand (**ConcreteState**) implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Zustandsdiagramm zeichnen

(d) Zeichnen Sie das UML-Zustandsdiagramm (mit Anfangszustand), das von dem Programm implementiert wird. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. In dem Diagramm kann zur Vereinfachung statt `System.out.println ("x")` einfach `"x"` geschrieben werden.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/09/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Kompositum (Composite)

Examensaufgabe „Grafik: Kreis, Quadrat, Dreieck“ (66116-2019-F.T1-TA2-A1)

Gegeben sei folgender Sachverhalt: Eine Grafik ist entweder ein Kreis, ein Quadrat oder ein Dreieck. Eine Grafik kann zudem auch eine Kombination aus diesen Elementen sein. Des Weiteren können Sie aus mehreren Grafiken auch neue Grafiken zusammenbauen. Sie denken sich: Ich möchte eine Menge von Grafiken genauso wie eine einzelne Grafik behandeln können.

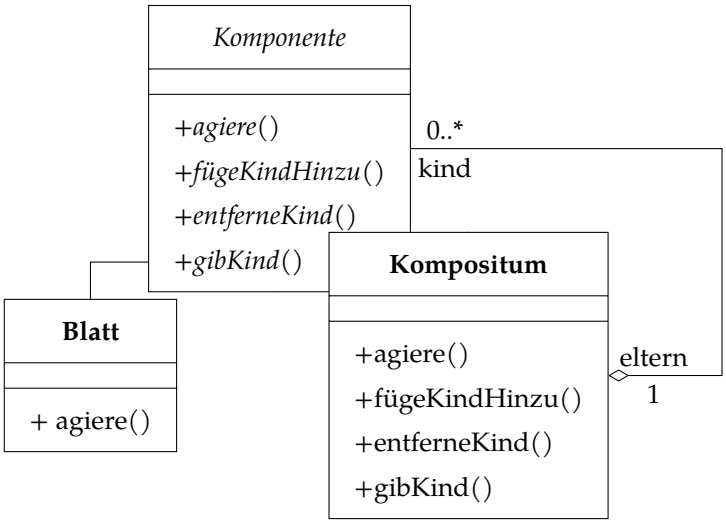
(a) Welches Entwurfsmuster sollten Sie zur Modellierung verwenden?

Lösungsvorschlag

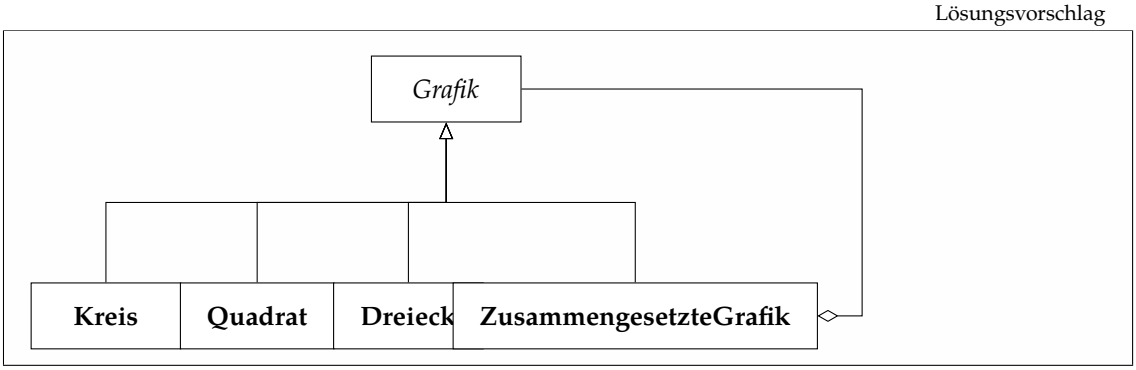
Kompositum

(b) Zeichnen Sie das entsprechende Klassendiagramm. Es reicht, nur die Klassennamen mit ihren Assoziationen und Vererbungsbeziehungen anzugeben; ðohne Methoden und Attribute.

Exkurs: Kompositum



Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-1/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „Roboter in einer Montagehalle“ (66116-2019-F.T1-TA2-A2)

Klassendiagramm

Hintergrundinformationen für die Aufgaben 2-5: Modellierung und Implementierung eines Programms

Ein autonomer Roboter in einer Montagehalle bekommt einen Auftrag, eine Menge von Objekten aus dem Lager (Material, z. B. Schrauben oder Werkzeug, z. B. Bohrer) zu holen und anschließend an seinen Ausgangspunkt zu bringen. Der Roboter hat einen Namen, der ihn eindeutig identifiziert, kennt seine aktuelle Position (x- und y-Koordinate) und hat als weitere Eigenschaft den Auftrag, der aus einer Liste von Auftragspositionen besteht, die er holen soll. Der Roboter besitzt folgende Fähigkeiten (Methoden):

(a) Er kann sich zu einer angegebenen Position bewegen.

(b) Er hat eine Methode, um einen Auftrag abzuarbeiten, d.h. alle Auftragspositionen zu holen.

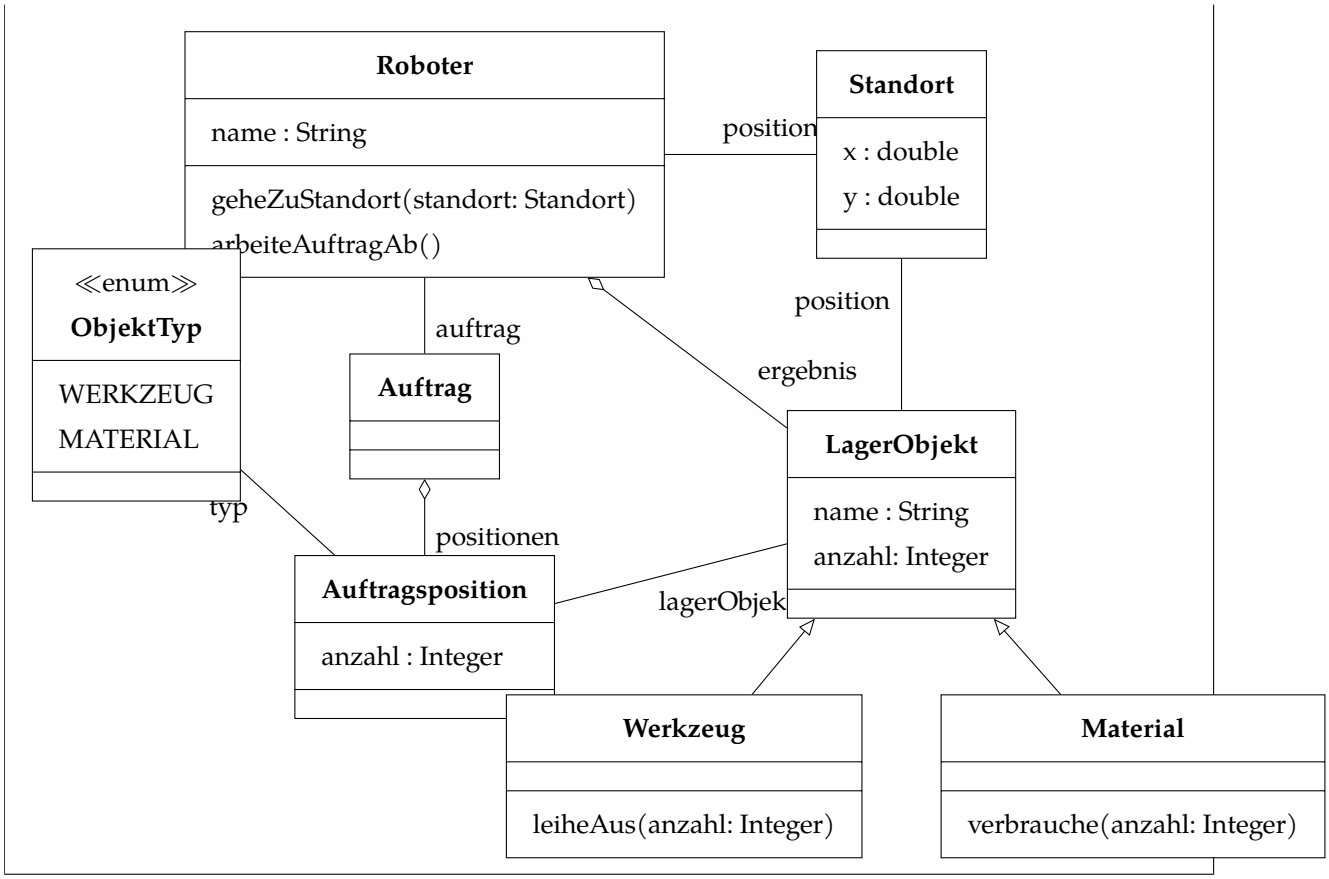
Alle Objekte im Lager haben jeweils einen Namen, einen Standort mit Angabe einer Position (s. oben) und speichern außerdem die jeweils noch vorhandene Stückzahl. Es gibt zwei Typen von Objekten: Werkzeuge mit einer Methode, mit der ein einzelnes Werkzeug ausgeliehen werden kann, und Materialien mit einer Methode, mit der sie verbraucht werden, wobei eine gewünschte Stückzahl angegeben wird. Diese vorgegebenen Methoden aktualisieren die Stückzahlen der Werkzeuge (Reduktion um 1) bzw. Materialien (Reduktion um verbrauchte Stückzahl), wobei hier vereinfachend angenommen wird, dass die Stückzahlen der Werkzeuge und Materialien immer ausreichend groß sind, um die geforderten Mengen bedienen zu können (d.h. Sie können diese beiden Methoden nutzen, ohne deren Implementierung angeben zu müssen).

Ein Auftrag (z. B. *„Hole Bohrer Typ B1, 100 × Schrauben M6, 10 × Schrauben M10 und 2 × Blech B72“*) besteht aus einer Menge von Auftragspositionen. Eine Auftragsposition besteht aus dem Typ des zu holenden Objekts (Werkzeug oder Material, soll als Enumeration modelliert werden), einem Verweis auf das zu holende Objekt und der zu holenden Stückzahl (Quantität; bei Werkzeugen wird diese ignoriert, da sie immer 1 ist). Der Roboter soll über die Auftragsposition außerdem die Position bestimmen, zu der er fahren muss, um das Objekt zu holen.

Der Roboter arbeitet die Auftragspositionen in der Reihenfolge ab, indem er sich von seinem aktuellen Standpunkt immer zur am nächsten liegenden Auftragsposition bewegt, um dort das nächste Objekt zu holen. Wir gehen der Einfachheit halber davon aus, dass die Montagehalle gut aufgeräumt ist und der Roboter sich quasi entlang der Luftlinie bewegen kann, d.h. die Entfernung zwischen zwei Positionen entspricht der euklidischen Distanz (Wurzel aus der Summe der Quadrate der Differenzen der x- und y-Koordinaten der Positionen). Der Roboter soll zur Kontrolle als weitere Eigenschaft „Ergebnis“ die Liste der eingesammelten Objekte zu einem Auftrag in der Reihenfolge speichern, in der er sie geholt hat, z.B. (M6 M10 B72 B1).

Geben Sie ein UML-Klassendiagramm zu der Aufgabenstellung an. Hinweis: Bei den Aufgaben 4 und 5 wird Konsistenz des Aktivitätsdiagramms bzw. des Codes mit dem Klassendiagramm verlangt.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-1/Teilaufgabe-2/Aufgabe-2.tex`

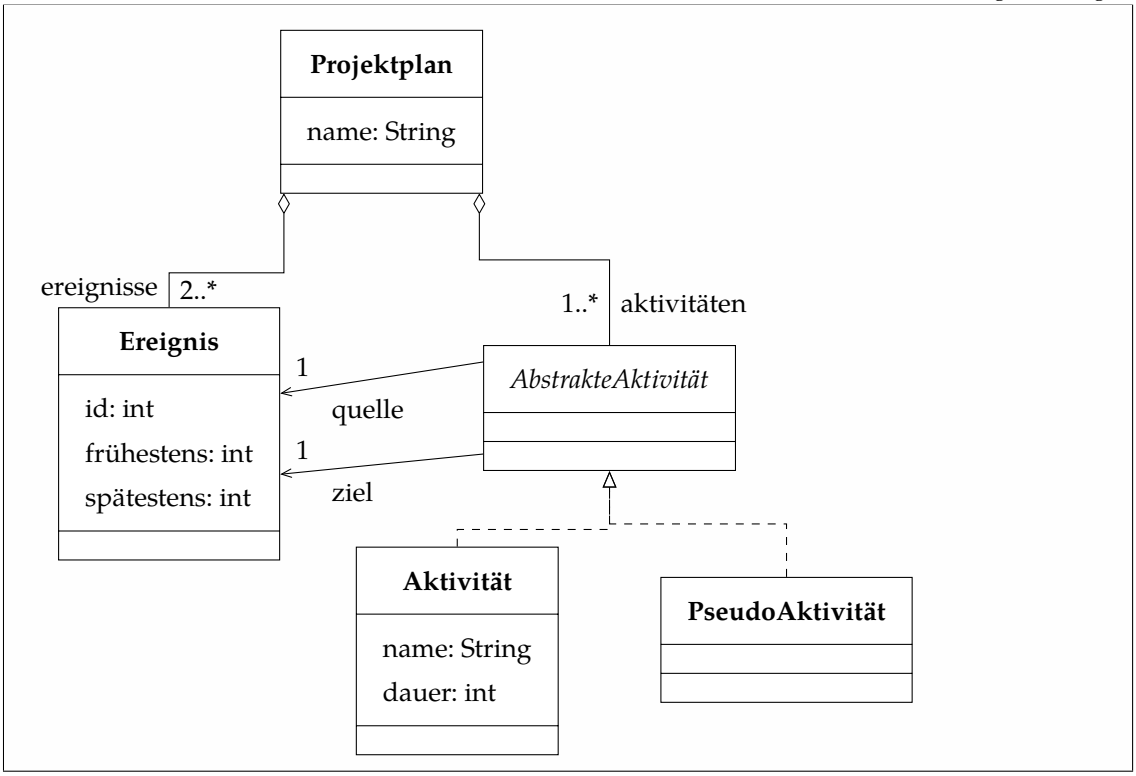
Examensaufgabe „Critical Path Method“ (66116-2019-H.T1-TA1-A1)

Klassendiagramm
Objektdiagramm

Ein CPM-Netzwerk („Critical Path Method“) ist ein benannter Projektplan, der aus Ereignissen und Aktivitäten besteht. Ein Ereignis wird durch eine ganze Zahl > 0 identifiziert. Jede Aktivität führt von einem Quellereignis zu einem Zielereignis. Eine reale Aktivität hat einen Namen und eine Dauer (eine ganze Zahl > 0). Eine Pseudoaktivität ist anonym. Ereignisse und Pseudoaktivitäten verbrauchen keine Zeit. Zu jedem Ereignis gibt es einen frühesten und einen spätesten Zeitpunkt (eine ganze Zahl > 0), deren Berechnung nicht Gegenstand der Aufgabe ist.

(a) Erstellen Sie ein UML-Klassendiagramm zur Modellierung von CPM-Netzwerken. Geben Sie für Attribute jeweils den Namen und den Typ an. Geben Sie für Assoziationen den Namen und für jedes Ende den Rollennamen und die Multiplizität an. Nutzen Sie ggf. abstrakte Klassen, Vererbung, Komposition oder Aggregation. Verzichten Sie auf Operationen und Sichtbarkeiten.

Lösungsvorschlag



(b) Erstellen Sie für das Klassendiagramm aus a) und das Beispiel aus der Aufgabenstellung ein Objektdiagramm. Geben Sie Rollennamen nur an, wenn es notwendig ist, um die Enden eines Links (Instanz einer Assoziation) zu unterscheiden.

Lösungsvorschlag

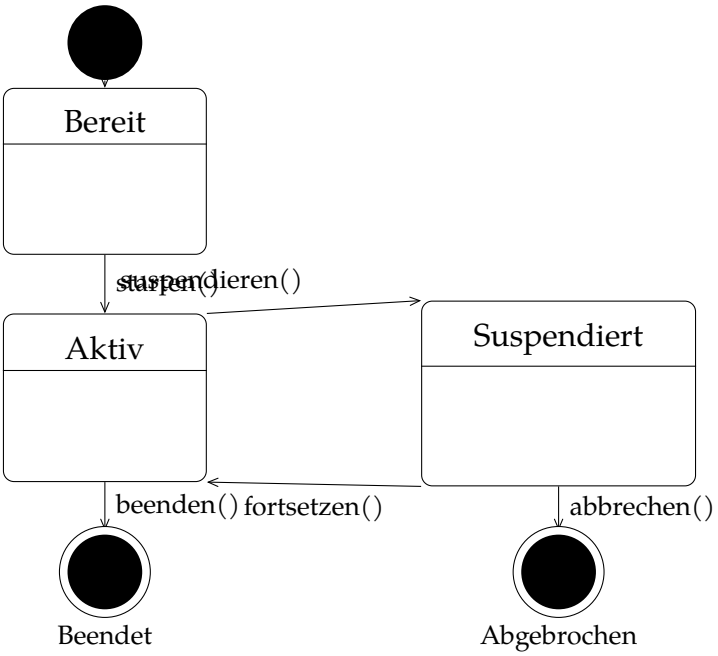
Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Zustand-Entwurfsmuster bei Verwaltung von Prozessen“ (66116-2019-H.T1-TA1-A4)

Zustandsdiagramm Wissen
Zustand (State)
Implementierung in Java

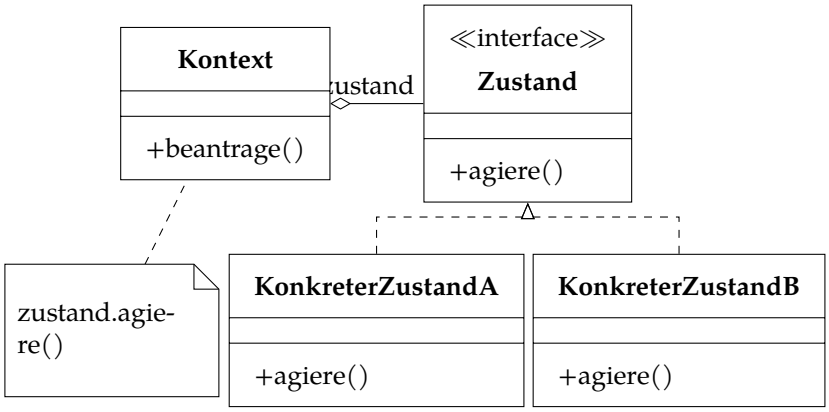
Zu den Aufgaben eines Betriebssystems zählt die Verwaltung von Prozessen. Jeder Prozess durchläuft verschiedene Zustände; Transitionen werden durch Operationsaufrufe ausgelöst. Folgendes Zustandsdiagramm beschreibt die Verwaltung von Prozessen:



Implementieren Sie dieses Zustandsdiagramm in einer Programmiersprache Ihrer Wahl mit Hilfe des Zustandsmusters; geben Sie die gewählte Sprache an. Die Methoden für die Transitionen sollen dabei die Funktionalität der Prozessverwaltung simulieren, indem der Methodenaufruf auf der Standardausgabe protokolliert wird. Falls Transitionen im aktuellen Zustand undefiniert sind, soll eine Fehlermeldung ausgegeben werden.

Exkurs: Zustand-(State)-Entwurfsmuster

UML-Klassendiagramm



Teilnehmer

Kontext (Context) definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

State (Zustand) definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

KontreterZustand (ConcreteState) implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Lösungsvorschlag

Implementierung in der Programmiersprache „Java“:

Methoden	Zustände	Klassennamen
	Bereit	ZustandBereit
starten(), fortsetzen()	Aktiv	ZustandAktiv
suspendieren()	Suspendiert	ZustandSuspendiert
beenden()	Beendet	ZustandBeendet
abbrechen()	Abgebrochen	ZustandAbgebrochen

```
/**
 * Entspricht der „Kontext"-Klasse in der Terminologie der „Gang of
 * Four".
 */
public class Prozess {

  private ProzessZustand aktuellerZustand;

  public Prozess() {
    aktuellerZustand = new ZustandBereit(this);
  }

  public void setzeZustand(ProzessZustand zustand) {
    aktuellerZustand = zustand;
  }
```

```
  public void starten() {
    aktuellerZustand.starten();
  }

  public void suspendieren() {
    aktuellerZustand.suspendieren();
  }

  public void fortsetzen() {
    aktuellerZustand.fortsetzen();
  }

  public void beenden() {
    aktuellerZustand.beenden();
  }

  public void abbrechen() {
    aktuellerZustand.abbrechen();
  }

  public static void main(String[] args) {
    Prozess prozess = new Prozess();
    prozess.starten();
    prozess.suspendieren();
    prozess.fortsetzen();
    prozess.beenden();
    prozess.starten();

    // Ausgabe:
    // Der Prozess ist im Zustand „bereit"
    // Der Prozess wird gestartet.
    // Der Prozess ist im Zustand „aktiv"
    // Der Prozess wird suspendiert.
    // Der Prozess ist im Zustand „suspendiert"
    // Der Prozess wird fortgesetzt.
    // Der Prozess ist im Zustand „aktiv"
    // Der Prozess wird beendet.
    // Der Prozess ist im Zustand „beendet"
    // Im Zustand „beendet" kann die Transition „starten" nicht ausführt werden!
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/Prozess.java`

```
/**
 * Entspricht der „Zustand"-Klasse in der Terminologie der "Gang of
 * Four".
 */
abstract class ProzessZustand {

  Prozess prozess;

  String zustand;
```

```
  public ProzessZustand(String zustand, Prozess prozess) {
    this.zustand = zustand;
    this.prozess = prozess;
    System.out.println(String.format("Der Prozess ist im Zustand „%s"", zustand));
  }

  private void gibFehlermeldungAus(String transition) {
    System.err.println(
↪   String.format("Im Zustand „%s" kann die Transition „%s" nicht ausführt werden!",
          zustand, transition));
  }

  public void starten() {
    gibFehlermeldungAus("starten");
  }

  public void suspendieren() {
    gibFehlermeldungAus("suspendieren");
  }

  public void fortsetzen() {
    gibFehlermeldungAus("fortsetzen");
  }

  public void beenden() {
    gibFehlermeldungAus("beenden");
  }

  public void abbrechen() {
    gibFehlermeldungAus("abbrechen");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ProzessZustand.java`

```
/**
 * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der "Gang of
 * Four".
 */
public class ZustandAbgebrochen extends ProzessZustand {

  public ZustandAbgebrochen(Prozess prozess) {
    super("abgebrochen", prozess);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandAbgebrochen.java`

```
/**
 * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der „Gang of
 * Four".
 */
```

```
public class ZustandAktiv extends ProzessZustand {

  public ZustandAktiv(Prozess prozess) {
    super("aktiv", prozess);
  }

  public void suspendieren() {
    System.out.println("Der Prozess wird suspendiert.");
    prozess.setzeZustand(new ZustandSuspendiert(prozess));
  }

  public void beenden() {
    System.out.println("Der Prozess wird beendet.");
    prozess.setzeZustand(new ZustandBeendet(prozess));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandAktiv.java`

```
/**
 * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der „Gang of
 * Four".
 */
public class ZustandBeendet extends ProzessZustand {

  public ZustandBeendet(Prozess prozess) {
    super("beendet", prozess);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandBeendet.java`

```
/**
 * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der „Gang of
 * Four".
 */
public class ZustandBereit extends ProzessZustand {

  public ZustandBereit(Prozess prozess) {
    super("bereit", prozess);
  }

  public void starten() {
    System.out.println("Der Prozess wird gestartet.");
    prozess.setzeZustand(new ZustandAktiv(prozess));
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandBereit.java`

```
/**
 * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der „Gang of
 * Four".
 */
```

```java
public class ZustandSuspendiert extends ProzessZustand {

  public ZustandSuspendiert(Prozess prozess) {
    super("suspendiert", prozess);
  }

  public void fortsetzen() {
    System.out.println("Der Prozess wird fortgesetzt.");
    prozess.setzeZustand(new ZustandAktiv(prozess));
  }

  public void abbrechen() {
    System.out.println("Der Prozess wird abgebrochen.");
    prozess.setzeZustand(new ZustandAbgebrochen(prozess));
  }
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandSuspendiert.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-1/Aufgabe-4.tex`

Examensaufgabe „Bankautomat“ (66116-2020-F.T1-TA1-A1)

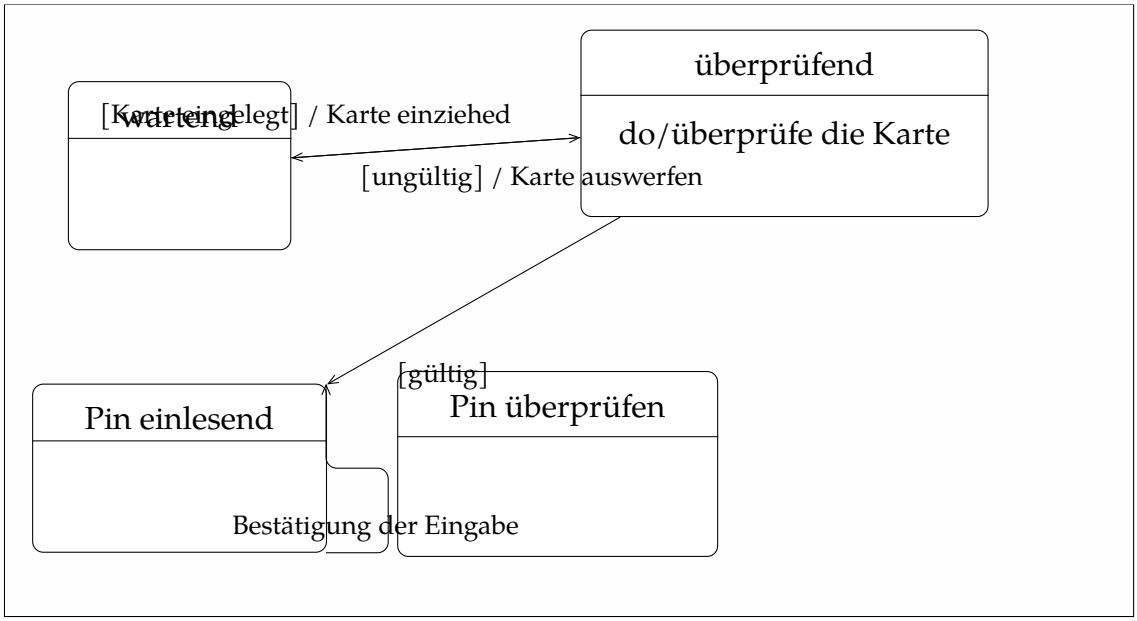
Zustandsdiagramm zeichnen

(a) Basisfunktion

Erstellen Sie ein Zustandsdiagramm für einen Bankautomat, welcher den im Folgenden beschriebenen Authentifizierungsvorgang von Bankkunden realisiert. Modellieren Sie dazu soweit nötig sowohl Zustände und Transitionsbedingungen als auch die Aktionen der Zustände.

Der Bankautomat startet im Grundzustand und wartet auf das Einlegen einer Bankkarte. Wird eine Karte eingelegt, wird diese eingezogen und der Automat startet die Überprüfung der Bankkarte. Ist die Karte ungültig, wird die Karte ausgeworfen und der Automat wechselt in den Grundzustand. Ist die Karte gültig, kann die vierstellige PIN eingelesen werden. Nach der Bestätigung der Eingabe wird diese überprüft. Ist die PIN gültig, so stoppt der Automat und zeigt eine erfolgreiche Authentifizierung an. Ist die PIN ungültig, zeigt der Automat einen Fehler an und erlaubt eine erneute Eingabe der PIN.

Lösungsvorschlag



(b) Erweiterung

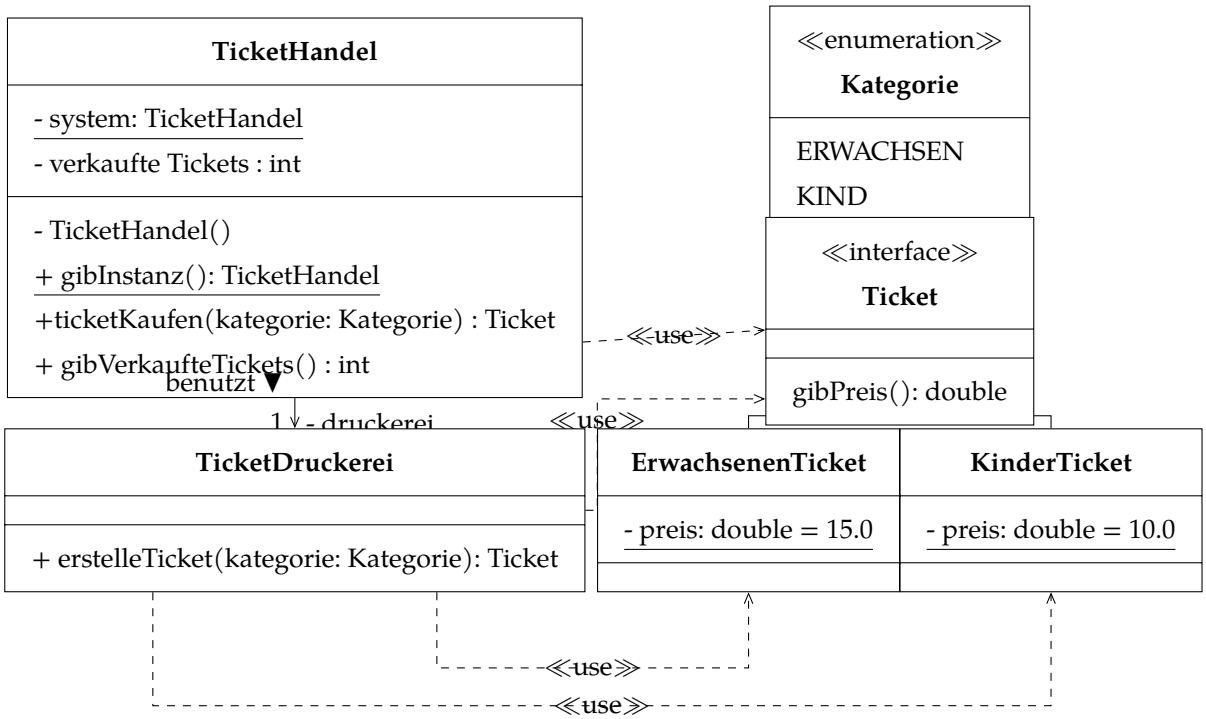
Der Bankautomaten aus Aufgabe a) soll nun so verändert werden, dass ein Bankkunde nach dem ersten fehlerhaften Eingeben der PIN die PIN erneut eingeben muss. Bei erneuter Falscheingabe, wird eine dritte Eingabe möglich. Bei der dritten Falscheingabe der PIN wird die Karte vom Automaten eingezogen und der Automat geht wieder in den Ausgangszustand über.

Hinweis: Für diese Aufgabe dürfen Sie Ihr Zustandsdiagramm aus a) weiter verwenden, wenn Sie eindeutig, z. B. durch den Einsatz von Farben kennzeichnen, was nur zur Aufgabe a) gehört und was Abänderungen des Zustandsdiagramms aus a) sind. Sie können, falls Sie einen neuen Automaten zeichnen, Zustände und Übergänge, die inhaltsgleich zur Lösung des Aufgabenteils a) sind mit einem “W“ markieren, statt sie zu beschriften. In diesem Fall wird der Text aus der Lösung zu Aufgabenteil a) an dieser Stelle wiederholt gedacht.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/03/Thema-1/Teilaufgabe-1/Aufgabe-1.tex

Examensaufgabe „Ticket-Handel“ (66116-2020-H.T1-TA1-A3)

Entwurfsmuster
Einzelstück (Singleton)
Fabrikmethode (Factory Method)

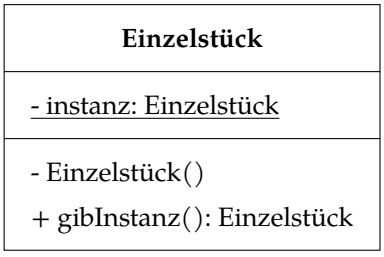


Ihnen sei ein UML-Klassendiagramm zu folgendem Szenario gegeben. Ein Benutzer (nicht im Diagramm enthalten) kann über einen `TicketHandel` Tickets erwerben. Dabei muss der Benutzer eine der zwei Ticketkategorien angeben. Das Handelsystem benutzt eine `TicketDruckerei`, um ein passendes `Ticket` für den Benutzer zu erzeugen.

(a) Im angegebenen Klassendiagramm wurden zwei unterschiedliche Entwurfsmuster verwendet. Um welche Muster handelt es sich? Geben Sie jeweils den Namen des Musters sowie die Elemente des Klassendiagramms an, mit denen diese Muster im Zusammenhang stehen. ACHTUNG: Es handelt sich dabei *nicht* um das *Interface*- oder das *Vererbungs*muster.

Lösungsvorschlag

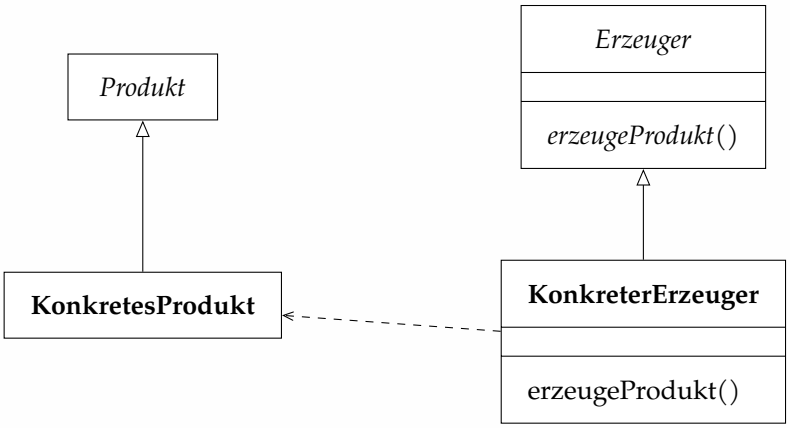
Einzelstück (Singleton)



Einzelstück (Singleton) stellt eine statische Methode bereit, mit deren Hilfe die Klienten nur auf eine einzige Instanz der Klasse zugreifen können.

Klasse	Akteur
`TicketHandel`	Einzelstück

Fabrikmethode (Factory Method)



Produkt Das Produkt ist der Basistyp (Klasse oder Schnittstelle) für das zu erzeugende Produkt.

KonkretesProdukt KonkretesProdukt implementiert die Produkt-Schnittstelle.

Erzeuger Der Erzeuger deklariert die Fabrikmethode, um ein solches Produkt zu erzeugen und kann eine Default-Implementierung beinhalten.

KonkreterErzeuger KonkreterErzeuger überschreibt die Fabrikmethode, um die ihm entsprechenden konkreten Produkte zu erzeugen (z. B. indem er den Konstruktor einer konkreten Produkt-Klasse aufruft).

Klasse	Akteur
`ErwachsenenTicket`	KonkretesProdukt
`KinderTicket`	KonkretesProdukt
`TicketDruckerei`	KonkreterErzeuger
`Ticket`	Produkt
-	Erzeuger

(b) Nennen Sie zwei generelle Vorteile von Entwurfsmustern.

Lösungsvorschlag

- Wiederverwendung einer bewährten Lösung für eine bestimmte Problemstellungen
- Verbesserung der Kommunikation unter EntwicklerInnen

(c) Geben Sie eine Implementierung der Klasse `TicketHandel` an. Bei der Methode `ticketKaufen()` wird die Anzahl der verkauften Tickets um 1 erhöht und ein

entsprechendes Ticket erstellt und zurückgegeben. Beachten Sie den Hinweis auf der nächsten Seite.

Lösungsvorschlag

```java
public class TicketHandel {
  private static TicketHandel system;
  private int verkaufteTickets;
  private TicketDruckerei druckerei;

  private TicketHandel() {
    druckerei = new TicketDruckerei();
    verkaufteTickets = 0;
  }

  public static TicketHandel gibInstanz() {
    if (system == null) {
      system = new TicketHandel();
    }
    return system;
  }

  public Ticket ticketKaufen(Kategorie kategorie) {
    verkaufteTickets++;
    return druckerei.erstelleTicket(kategorie);
  }

  public int gibVerkaufteTickets() {
    return verkaufteTickets;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketHandel.java`

(d) Geben Sie eine Implementierung der Klasse `TicketDruckerei` an.

Lösungsvorschlag

```java
public class TicketDruckerei {
  public Ticket erstelleTicket(Kategorie kategorie) {
    if (kategorie == Kategorie.ERWACHSEN) {
      return new ErwachsenenTicket();
    } else {
      return new KinderTicket();
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketDruckerei.java`

(e) Geben Sie eine Implementierung der Klasse `KinderTicket` an.

Lösungsvorschlag

```
public class KinderTicket implements Ticket {
  private static double preis = 10.0;
```

```
  public double gibPreis() {
    return preis;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/KinderTicket.java`

Hinweis: Die Implementierungen *müssen* sowohl dem Klassendiagramm, als auch den Konzepten der verwendeten Muster entsprechen. Verwenden Sie eine objektorientierte Programmiersprache, vorzugsweie *Java*. Sie müssen sich an der nachfolgenden Testmethode und ihrer Ausgabe orientieren. Die Testmethode muss mit Ihrer Implementierung ausführbar sein und sich semantisch korrekt verhalten.

Quelltext der Testmethode:

```
public static void main(String[] args) {
  TicketHandel.gibInstanz().ticketKaufen(Kategorie.ERWACHSEN);
  TicketHandel.gibInstanz().ticketKaufen(Kategorie.KIND);
  System.out.println("Anzahl verkaufter Tickets: " +
↪ TicketHandel.gibInstanz().gibVerkaufteTickets());
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/Test.java`

Konsolenausgabe:

```
Anzahl verkaufter Tickets: 2
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Objektorientierte Analyse“ (66116-2020-H.T2-TA1-A4)

Objektorientierung

Betrachten Sie das folgende Szenario:

Entwickeln Sie für einen Kunden eine einheitliche Online-Plattform, in welcher mehrere Restaurants angebunden sind. Die Nutzer des Systems sollen Essen wie Pizzen, Burger oder Pasta bestellen und dabei aus einer Liste von verschiedenen Gerichten auswählen können. Die Plattform soll zusätzliche Optionen (z. B. Lieferung durch einen Lieferdienst oder direkte Abholung, inkl. Salat oder einer Flasche Wein) ermöglichen. Die Bestellung soll dann von der Plattform an den jeweiligen Gaststättenbetreiber gesendet werden. Die Besteller sollen zudem eine Bestätigung als Nachricht erhalten. Die jeweiligen Gerichte und Optionen haben unterschiedliche Preise, die dem Internetnutzer angezeigt werden müssen, bevor er diese auswählt. Der Endpreis muss vor der endgültigen Zahlung des Auftrags angezeigt werden. Kunden können (optional) einen Benutzeraccount anlegen und erhalten bei häufigen Bestellungen einen Rabatt.

(a) Beschreiben Sie kurz ein Verfahren, wie Sie aus der Szenariobeschreibung mögliche Kandidaten für Klassen erhalten können.

Lösungsvorschlag

Verfahren nach Abbott; [a]

Objektorientierte Analyse und Design (OOAD)

[a] `http://info.johpie.de/stufe_q1/info_01_verfahren_abbott.pdf`

(b) Beschreiben Sie kurz ein Verfahren, um Vererbungshierarchien zu identifizieren.

Lösungsvorschlag

Eine Begriffshierachie mit den Substantiven des Textes bilden.

(c) Geben Sie fünf geeignete Klassen für das obige Szenario an. Nennen Sie dabei keine Klassen, welche durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag

- Benutzer (Kunde, Gaststättenbetreiber)
- Restaurant
- Gericht (Pizza, Burger, Pasta)
- ZusatzOption (Lieferung, Salat, Wein)
- Bestellung

(d) Nennen Sie drei Klassen für das obige Szenario, die direkt durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag Sequenzdiagramm

- Nachricht
- Preis
- Rabatt

(e) Erstellen Sie ein Sequenzdiagramm für das gegebene System mit folgendem Anwendungsfall: Ein Nutzer bestellt zwei Pizzen und eine Flasche Wein. Beginnen Sie mit der „Auswahl des Gerichts“ bis hin zur „Bestätigung der Bestellung“ sowie „Lieferung an die Haustür“. Das Diagramm soll mindestens je einen Akteur für Benutzer (Browser), Applikation (Webserver) und Restaurant (Koch und Lieferdienst) vorsehen. Die Bezahlung selbst muss nicht modelliert werden.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-4.tex`

Examensaufgabe „Terme über die Rechenarten“ (66116-2020-H.T2-TA1-A5)

Entwurfsmuster

Wir betrachten Terme über die Rechenarten $op \in \{+, -, \cdot, \div\}$, die rekursiv definiert sind:

- Jedes Literal ist ein Term, z. B. „4“.
- Jedes Symbol ist ein Term, z. B. „x“.
- Ist t ein Term, so ist „(t)“ ein (geklammerter) Term.
- Sind t_1, t_2 Terme, so ist „$t_1\ op\ t_2$“ ebenso ein Term.

Beispiele für gültige Terme sind also „$4 + 8$“, „$4 \cdot x$“ oder „$4 + (8 \cdot x)$“.

(a) Welches Design-Pattern eignet sich hier am besten zur Modellierung dieses Sachverhalts?

Lösungsvorschlag

Kompositum

(b) Nennen Sie drei wesentliche Vorteile von Design-Pattern im Allgemeinen.

(c) Modellieren Sie eine Klassenstruktur in UML, die diese rekursive Struktur von *Termen* abbildet. Sehen Sie mindestens einzelne Klassen für die *Addition* und *Multiplikation* vor, sowie weitere Klassen für *geklammerte Terme* und *Literale*, welche ganze Zahlen repräsentieren. Gehen Sie bei der Modellierung der Klassenstruktur davon aus, dass eine objektorientierte Programmiersprache wie Java zu benutzen ist.

(d) Erstellen Sie ein Objektdiagramm, welches den Term $t := 4 + (3 \cdot 2) + (12 \cdot y/(8 \cdot x))$ entsprechend Ihres Klassendiagramms repräsentiert.

(e) Überprüfen Sie, ob das Objektdiagramm für den in Teilaufgabe d) gegebenen Term eindeutig definiert ist. Begründen Sie Ihre Entscheidung.

(f) Die gegebene Klassenstruktur soll mindestens folgende Operationen unterstützen:

- das Auswerten von Termen,
- das Ausgeben in einer leserlichen Form,
- das Auflisten aller verwendeten Symbole.

Welches Design-Pattern ist hierfür am besten geeignet?

(g) Erweitern Sie Ihre Klassenstruktur um die entsprechenden Methoden, Klassen und Assoziationen, um die in Teilaufgabe f) genannten zusätzlichen Operationen gemäß dem von Ihnen genannten Design Pattern zu unterstützen.

Lösungsvorschlag

```java
public class Addition extends Rechenart {
  public Addition(Term a, Term b) {
    super(a, b, "+");
  }

  public double auswerten() {
    return a.auswerten() + b.auswerten();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Addition.java`

```java
public class Divison extends Rechenart {
  public Divison(Term a, Term b ) {
    super(a, b, "/");
  }

  public double auswerten() {
    return a.auswerten() / b.auswerten();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Divison.java`

```java
public class GeklammerterTerm extends Term {
  Term term;

  public GeklammerterTerm(Term term) {
    this.term = term;
  }

  public double auswerten() {
    return term.auswerten();
  }

  public void ausgeben() {
    System.out.print("(");
    term.ausgeben();
    System.out.print(")");
  }
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/GeklammerterTerm.java`

```java
public class Literal extends Term {
  int wert;

  public Literal(int wert) {
    this.wert = wert;
  }

  public double auswerten() {
```

```
    return wert;
  }

  public void ausgeben() {
    System.out.print(wert);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Literal.java`

```
public class Multiplikation extends Rechenart {
  public Multiplikation(Term a, Term b) {
    super(a, b, "*");
  }

  public double auswerten() {
    return a.auswerten() * b.auswerten();
  }
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Multiplikation.java`

```
abstract class Rechenart extends Term {
  Term a;
  Term b;
  String operatorZeichen;

  public Rechenart (Term a, Term b, String operatorZeichen) {
    this.a = a;
    this.b = b;
    this.operatorZeichen = operatorZeichen;
  }

  public void ausgeben () {
    a.ausgeben();
    System.out.print(" " + operatorZeichen + " ");
    b.ausgeben();
  }

  abstract public double auswerten();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Rechenart.java`

```
public class Subtraktion extends Rechenart {
  public Subtraktion(Term a, Term b) {
    super(a, b, "-");
  }

  public double auswerten() {
    return a.auswerten() - b.auswerten();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Subtraktion.java`

```java
public class Symbol extends Term {
  String name;
  public Symbol(String name) {
    this.name = name;
  }

  public double auswerten() {
    return Klient.symbole.get(name);
  }

  public void ausgeben() {
    System.out.print(name);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Symbol.java`

```java
abstract class Term {
  abstract public double auswerten();

  abstract public void ausgeben();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Term.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-5.tex`

Examensaufgabe „Elementtypen UML-Klassendiagramm“ (66116-2021-F.T1-TA1-A3)

UML-Diagramme

Wählen Sie bis zu fünf unterschiedliche Elementtypen aus folgendem Diagramm aus und benennen Sie diese Elemente und ihre syntaktische (nicht semantische) Bedeutung.

Lösungsvorschlag

Klasse ConcreteCreator

Interface Produkt

Abstrakte Klasse Creator

Kommentar return new ConcreteProdukt()

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Wissen Erbauer“ (66116-2021-F.T1-TA1-A6)

(a) Erläutern Sie den Zweck (Intent) des Erzeugungsmusters Erbauer in max. drei Sätzen, ohne dabei auf die technische Umsetzung einzugehen.

Lösungsvorschlag

> Die Erzeugung komplexer Objekte wird vereinfacht, indem der Konstruktionsprozess in eine spezielle Klasse verlagert wird. Er wird so von der Repräsentation getrennt und kann sehr unterschiedliche Repräsentationen zurückliefern.

(b) Erklären Sie, wie das Erzeugungsmuster Erbauer umgesetzt werden kann (Implementierung). Die Angabe von Code ist hierbei NICHT notwendig!

Lösungsvorschlag

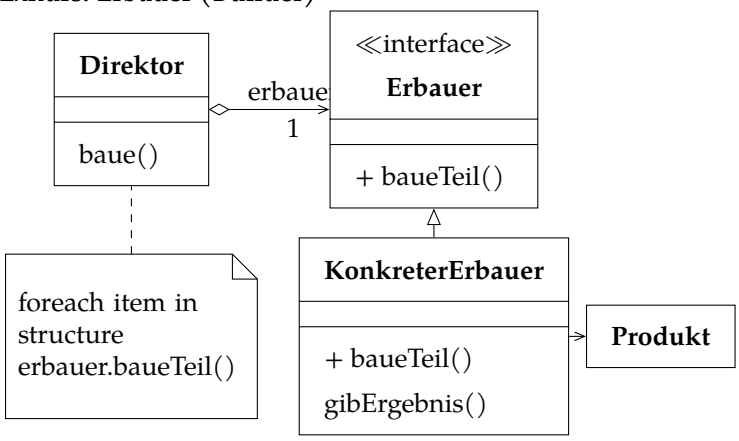
> Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(c) Nennen Sie jeweils einen Vor- und einen Nachteil des Erzeugungsmusters Erbauer im Vergleich zu einer Implementierung ohne dieses Muster.

Lösungsvorschlag

> **Vorteil** Die Implementierungen der Konstruktion und der Repräsentationen werden isoliert. Die Erbauer verstecken ihre interne Repräsentation vor dem Direktor.
>
> **Nachteil** Es besteht eine enge Kopplung zwischen Produkt, konkretem Erbauer und den am Konstruktionsprozess beteiligten Klassen.

Exkurs: Erbauer (Builder)



Erbauer Der Erbauer spezifiziert eine abstrakte Schnittstelle zur Erzeugung der Teile eines komplexen Objektes.

KonkreterErbauer Der konkrete Erbauer erzeugt die Teile des komplexen Objekts durch Imple-

mentierung der Schnittstelle. Außerdem definiert und verwaltet er die von ihm erzeugte Repräsentation des Produkts. Er bietet auch eine Schnittstelle zum Auslesen des Produkts.

Direktor Der Direktor konstruiert ein komplexes Objekt unter Verwendung der Schnittstelle des Erbauers. Der Direktor arbeitet eng mit dem Erbauer zusammen: Er weiß, welche Baureihenfolge der Erbauer verträgt oder benötigt. Der Direktor entkoppelt somit den Konstruktionsablauf vom Klienten.

Produkt Das Produkt repräsentiert das zu konstruierende komplexe Objekt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-6.tex`

Examensaufgabe „MyParser“ (66116-2021-F.T1-TA1-A7)

Einbringen von Abhängigkeiten (Dependency Injection)

Lesen Sie die folgenden alternativen Codestücke.

(a)
```
public class MyParser {
  private InputStream input;

  public MyParser(String filePath) {

  }
}
```

(b)
```
public class MyParser {
  private InputStream input;

  public MyParser(InputStream stream) {
  }
}
```

Beide Codestücke zeigen die Initialisierung einer Klasse namens `MyParser`. Das zweite Codestück nutzt jedoch hierfür eine Technik namens Abhängigkeits-Injektion (Dependency Injection).

(a) Erklären Sie den Unterschied zwischen beiden Initialisierungen. Hinweis: Sie können diese Aufgabe auch lösen, falls Sie die Technik nicht kennen.

Lösungsvorschlag

> Die Abhängigkeit von einer Instanz der Klasse InputStream wird erst bei der Initialisierung des Objekt übergeben.

(b) Benennen Sie einen Vorteil dieser Technik.

Lösungsvorschlag

> Die Kopplung zwischen einer Klasse und ihrer Abhängigkeit wird verringert.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-7.tex`

Übungsaufgabe „Fußballmeisterschaft“ (Klassendiagramm)

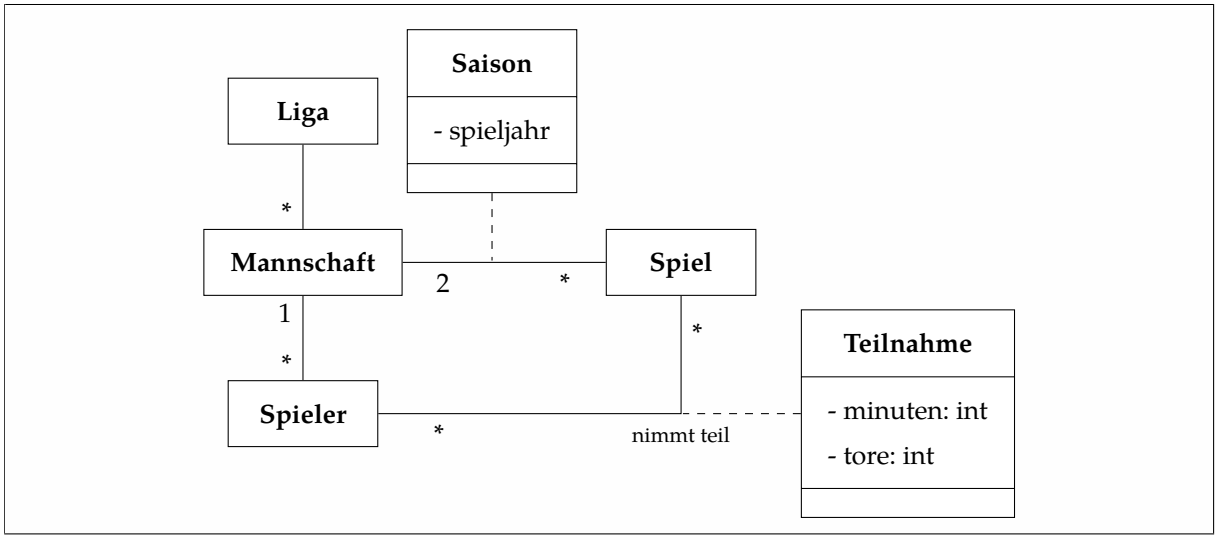
Klassendiagramm

Modellieren Sie die folgende Situation als Klassendiagramm:

Fußballmannschaften einer Liga bestreiten während einer Meisterschaftsrunde Spiele gegen andere Mannschaften. Dabei werden in jeder Mannschaft Spieler für einen bestimmten Zeitraum (in Minuten) eingesetzt, die dabei eventuell Tore schießen.

Die Modellierung soll es ermöglichen, festzustellen, welcher Spieler in welchem Spiel wie lange auf dem Feld war und wie viele Tore geschossen hat. Ebenso soll es möglich sein, für jede Mannschaft festzustellen, gegen welche Mannschaft welche Ergebnisse erzielt wurden.

Lösungsvorschlag



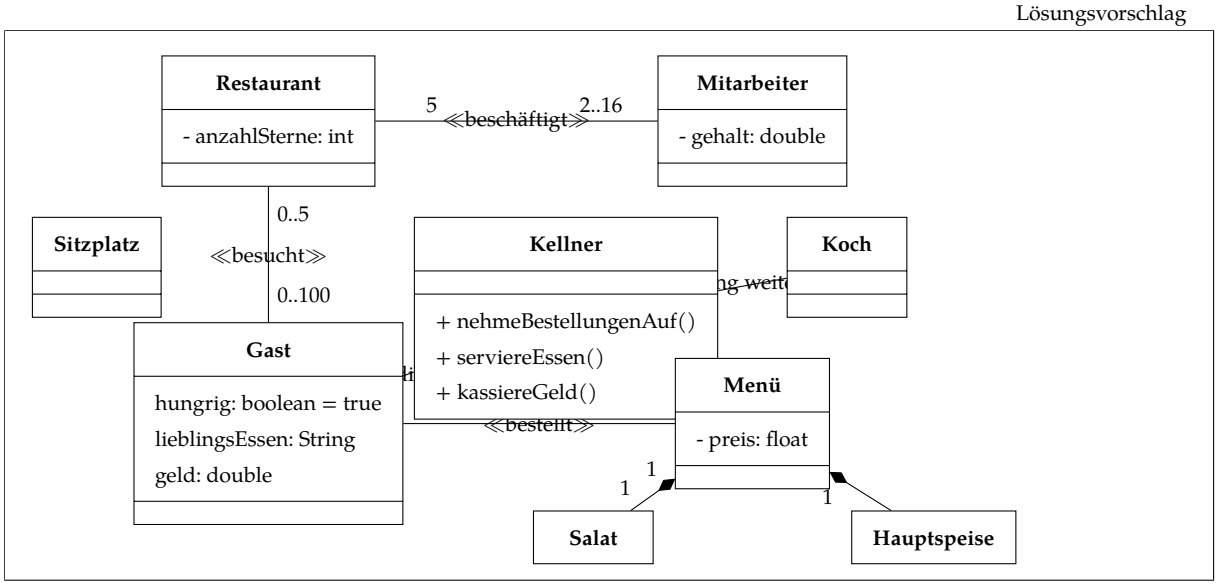
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Diagramme/10_Struktur/10_Klassendiagramm/Aufgabe_Fußballmeisterschaft.tex`

Übungsaufgabe „Gasthausen“ (Klassendiagramm)

Klassendiagramm

In diesem kleinen Städtchen gibt es fünf Restaurants, aber egal, ob es sich um ein 2*-Restaurant, oder um ein 5*-Restaurant handelt, alle haben Gemeinsamkeiten. In jedem Restaurant sind mehrere **Mitarbeiter** angestellt. Im kleinsten Restaurant nur zwei, aber im größten verdienen 16 Mitarbeiter ihr Geld. Die Mitarbeiter bekommen in jedem Restaurant ein anderes *Gehalt*, aber alle machen ihre Arbeit. Die **Kellner** *nehmen bestellungen entgegen*, *servieren das Essen* und *kassieren das Geld* von den Gästen. Jeder Kellner gibt die Bestellungen der Gäste an die Köche weiter. Die **Köche** kennen verschiedene **Rezepte**, nach denen sie die **Menüs** dann zubereiten. Jedes Menü hat einen anderen *Preis*, aber es besteht nach Tradition immer aus einem **Salat** und einer **Hauptspeise**. An manchen Tagen ist kein **Gast** da. Dann werden Sie in ganz familiärem Rahmen bewirtet. Aber auch an guten Tagen gibt es, bei 100 Plätzen im größten Restaurant, sicher für jeden hungrigen Gast einen **Platz**. Wer in Gasthausen Essen geht sollte hungrig sein! Natürlich hat jeder Gast unterschiedlich viel Geld zur Verfügung, aber bei entsprechender Wahl des Restaurants reicht es sicher für sein Lieblingsessen.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Diagramme/10_Struktur/10_Klassendiagramm/Aufgabe_Gasthausen.tex`

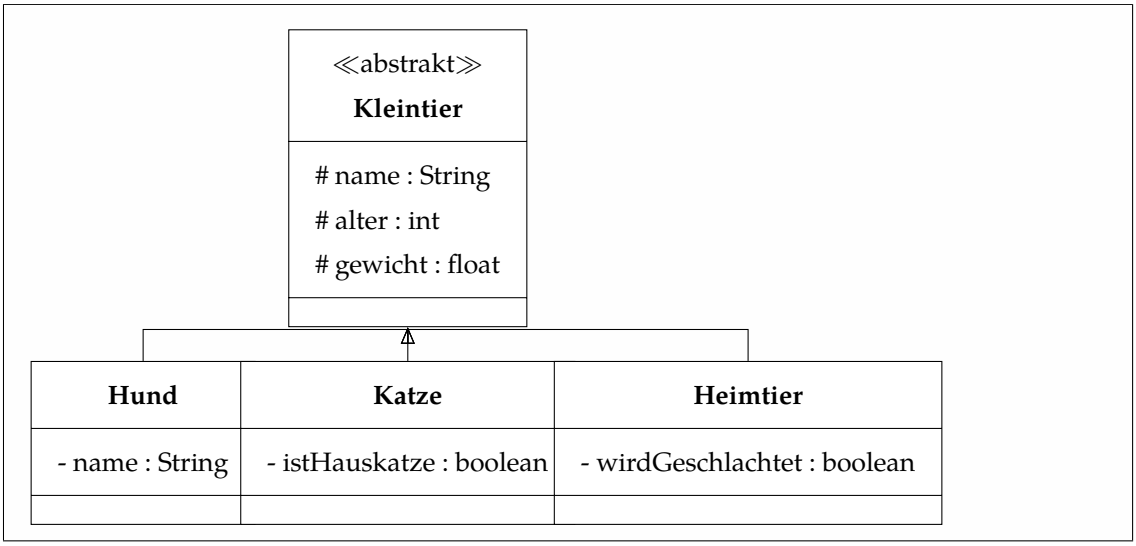
Übungsaufgabe „Kleintierpraxis“ (Klassendiagramm, Vererbung)

Klassendiagramm
Vererbung

In der Kleintierpraxis Dr. Huf werden *Hunde*, *Katzen* und *Heimtiere*[1] behandelt. In der Praxissoftware werden von jedem Tier *Name*, *Alter* (in Jahren) und *Gewicht* (in kg mit mindestens 2 Dezimalen) erfasst. Bei einem *Hund* wird zusätzlich aufgenommen, ob er *reinrassig* ist, bei einer *Katze*, ob sie ausschließlich *in der Wohnung gehalten* wird, und beim *Heimtier*, ob es zur *Lebensmittellieferung* dient.

(a) Erstellen Sie ein entsprechendes Klassendiagramm (zunächst ohne Methoden), das den oben beschriebenen Sachverhalt geeignet erfasst.

Lösungsvorschlag



(b) Implementieren Sie die Klassen gemäß des Modells aus Teilaufgabe a in der Programmierumgebung BlueJ. Beachten Sie dabei die abstrakte Oberklasse!

Lösungsvorschlag

Klasse *Kleintier*

```
public abstract class Kleintier {
  protected String name;
  protected int alter;
  protected float gewicht;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java`

```
  public Kleintier(String name, int alter, float gewicht) {
    this.name = name;
    this.alter = alter;
    this.gewicht = gewicht;
```

[1] Unter Heimtiere versteht man hier Kleintiere (d. h. keine Rinder, Pferde etc.), die zu Hause gehalten werden und keine Hunde oder Katzen sind (beispielsweise Meerschweinchen oder Kaninchen). Heimtiere können auch Nutztiere sein, die zur Schlachtung gehalten werden (z. B. Hühner). In diesem Fall muss ein Tierarzt darauf achten, nur Medikamente anzuwenden, die für Lebensmittel liefernde Tiere zugelassen sind.

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java`

Klasse *Hund*

```
public class Hund extends Kleintier {
  private boolean reinrassig;

  public Hund(String name, int alter, float gewicht, boolean reinrassig) {
    super(name, alter, gewicht);
    this.reinrassig = reinrassig;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Klasse *Katze*

```
public class Katze extends Kleintier {
  @SuppressWarnings("unused")
  private boolean istHauskatze;

  public Katze(String name, int alter, float gewicht, boolean istHauskatze) {
    super(name, alter, gewicht);
    this.istHauskatze = istHauskatze;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Katze.java`

Klasse *Heimtier*

```
public class Heimtier extends Kleintier {
  @SuppressWarnings("unused")
  private boolean wirdGeschlachtet;

  public Heimtier(String name, int alter, float gewicht, boolean
↪   wirdGeschlachtet) {
    super(name, alter, gewicht);
    this.wirdGeschlachtet = wirdGeschlachtet;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Heimtier.java`

(c) Die Praxissoftware verfügt über eine Methode `datenAusgeben()`, die den Namen und das Alter eines Tieres auf dem Bildschirm ausgibt. Fügen Sie im Klassendiagramm die Methode an geeigneter Stelle ein und implementieren Sie sie.

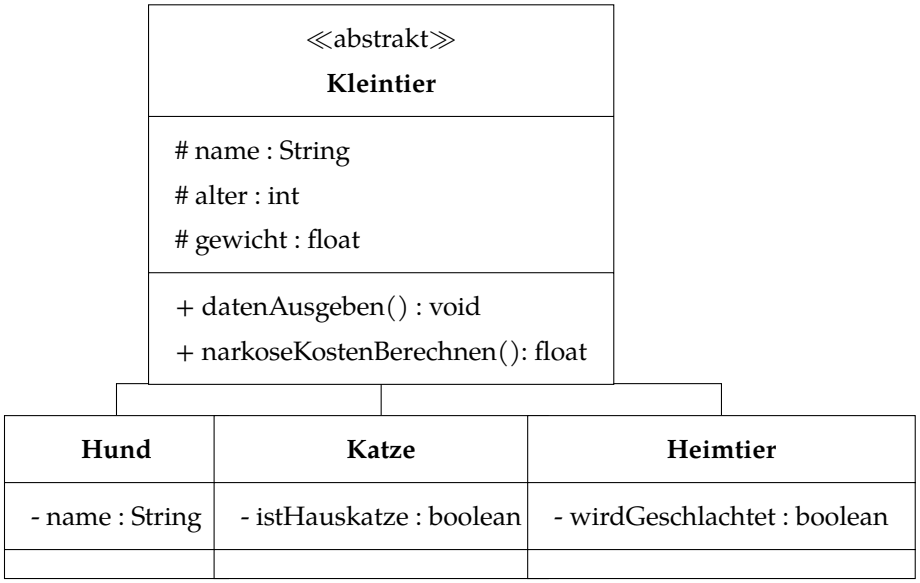
Lösungsvorschlag

Einfügen in der abstrakten Klasse *Heimtier*. Fertiges Klassendiagramm siehe 1 (d)

```
    public void datenAusgeben()
    {
```

```
        System.out.println("Name: " + name + ", Alter: " + alter);
    }
```

(d) Die Kosten für eine Narkose setzen sich zusammen aus einer Grundgebühr, die von der Tierart abhängt, und aus den Kosten für das verwendete Narkotikum. Diese wiederum berechnen sich aus dem Preis des Narkotikums pro kg Körpergewicht multipliziert mit dem Gewicht des Tieres. Ergänzen Sie das Klassendiagramm entsprechend um die Methode `narkosekostenBerechnen()`, die die Kosten für eine Narkose auf dem Bildschirm ausgibt, und implementieren Sie sie.



Lösungsvorschlag

Einfügen in der abstrakten Klasse *Heimtier*.

```
public class Hund extends Kleintier {
  private boolean reinrassig;

  public Hund(String name, int alter, float gewicht, boolean reinrassig) {
    super(name, alter, gewicht);
    this.reinrassig = reinrassig;
    narkoseGrundGebuehr = 1.50f;
  }

  public void datenAusgeben() {
    System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

    if (reinrassig) {
      System.out.println("Der Hund ist reinrassig.");
    } else {
      System.out.println("Der Hund ist nicht reinrassig.");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Einfügen in den Konstruktor der Klasse *Hund*:

```
narkoseGrundGebuehr = 1.50f;
```

Einfügen in den Konstruktor der Klasse *Katze*:

```
narkoseGrundGebuehr = 1f;
```

Einfügen in den Konstruktor der Klasse *Heimtier*:

```
narkoseGrundGebuehr = 2f;
```

(e) Falls es sich bei dem zu behandelnden Tier um einen Hund handelt, soll die Methode `datenAusgeben()` (siehe Teilaufgabe c) dies zusammen mit dem Namen und dem Alter des Tieres auf dem Bildschirm ausgeben. Außerdem soll in diesem Fall ausgegeben werden, ob der Hund reinrassig ist oder nicht.

Lösungsvorschlag

Einfügen in der Klasse *Hund*:

```
public void datenAusgeben() {
  System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

  if (reinrassig) {
    System.out.println("Der Hund ist reinrassig.");
  } else {
    System.out.println("Der Hund ist nicht reinrassig.");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Diagramme/10_Struktur/10_Klassendiagramm/Aufgabe_Kleintierpraxis.tex`

Übungsaufgabe „Universitätsverwaltung“ (Klassendiagramm)

Klassendiagramm

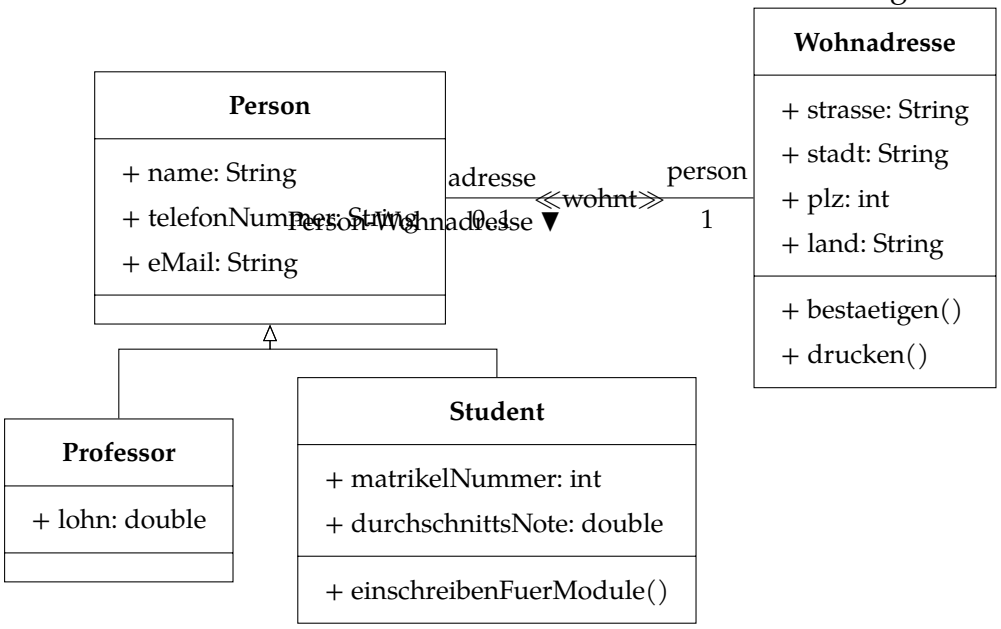
Gegeben ist der folgende Sachverhalt.

Jede **Person** hat einen *Namen*, eine *Telefonnummer* und *E-Mail*.

Jede **Wohnadresse** wird von nur einer Person bewohnt. Es kann aber sein, dass einige Wohnadressen nichtbewohnt sind. Den Wohnadressen sind je eine *Strasse*, eine *Stadt*, eine *PLZ* und ein *Land* zugeteilt. Alle Wohnadressen können bestätigt werden und als Beschriftung (für Postversand) gedruckt werden.

Es gibt zwei Sorten von **Personen**: **Student**, welcher sich für ein *Modul einschreiben* kann und **Professor**, welcher einen *Lohn* hat. Der Student besitzt eine *Matrikelnummer* und eine *Durchschnittsnote*.

Modellieren Sie diesen Sachverhalt mit einem UML Klassendiagramm.



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Diagramme/10_Struktur/10_Klassendiagramm/Aufgabe_Universitätsverwaltung.tex`

Übungsaufgabe „Alle UML-Diagramme“ (UML-Diagramme, Klassendiagramm, Objektdiagramm, Zustandsdiagramm Wissen, Sequenzdiagramm, Aktivitätsdiagramm, Anwendungsfalldiagramm, Kommunikationsdiagramm)

UML-Diagramme

Ordnen Sie die gegebenen Diagrammarten der Fragestellung zu, die das jeweilige Diagramm am besten beantwortet.

(a) Wie sind die Daten und das Verhalten meines Systems im Detail strukturiert?

Lösungsvorschlag

Klassendiagramm

(b) Wie sieht ein Schnappschuss meines Systems zur Ausführungszeit aus?

Lösungsvorschlag

Objektdiagramm

(c) Wie verhält sich das System in einem bestimmten Zustand bei gewissen Ereignissen?

Lösungsvorschlag

Zustandsdiagramm

(d) Wie läuft die Kommunikation in meinem System ab?

Lösungsvorschlag

Sequenzdiagramm

(e) Wie realisiert mein System ein bestimmtes Verhalten?

Lösungsvorschlag

Aktivitätsdiagramm

(f) Was soll mein geplantes System leisten?

Lösungsvorschlag

Anwendungsfalldiagramm

(g) Welche Teile einer komplexen Struktur arbeiten wie zusammen, um eine bestimmte Funktion zu erfüllen?

Lösungsvorschlag

Kommunikationsdiagramm

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Diagramme/Aufgabe_Alle-UML-Diagramme.tex`

Übungsaufgabe „Bankkonten“ (Vererbung, Generalisierung, Spezialisierung, Klassendiagramm, Implementierung in Java)

Vererbung
Generalisierung
Spezialisierung
Klassendiagramm

Eine kleine Bank bietet drei Arten von Konten an: Girokonten, Sparkonten und Geschäftskonten. Alle drei Kontoarten haben die Methoden `Einzahlen`, `Abheben` und `KontostandGeben` sowie die Attribute `kontostand` und `kontonummer`.

- Sparkonten haben einen Zinssatz und eine Methode `Verzinsen`, die den Jahreszins zum Guthaben addiert. Maximalbetrag beim Abheben ist der aktuelle Kontostand.
- Girokonten können um bis 2000 € überzogen werden (Dispokredit).
- Geschäftskonten haben einen variablen Dispokredit, der über die Methode `DispoKreditSetzen` festgelegt wird; der Startwert für den Dispokredit wird mit dem Konstruktor beim Einrichten des Kontos als Parameter mitgegeben.

(a) Überlege dir, welche Konten Generalisierungen bzw. Spezialisierungen anderer Konten sind. Warum ist es sinnvoll, eine Klasse Konto als oberste Klasse Generalisierungshierarchie einzuführen?

Lösungsvorschlag

Da alle Konten die Methoden `einzahlen()`, `abheben()` und `kontostandGeben()` sowie die Attribute `kontostand` und `kontonummer` besitzen, bietet sich deren Verwaltung in einer einzigen Klasse an. Da jede Kontoart aber auch individuelle Eigenschaften bzw. Methoden hat, muss es für jede auch eine eigene Klasse geben. Daher bietet es sich an, die Gemeinsamkeiten in eine Oberklasse `Konto` auszulagern (Generalisierung).

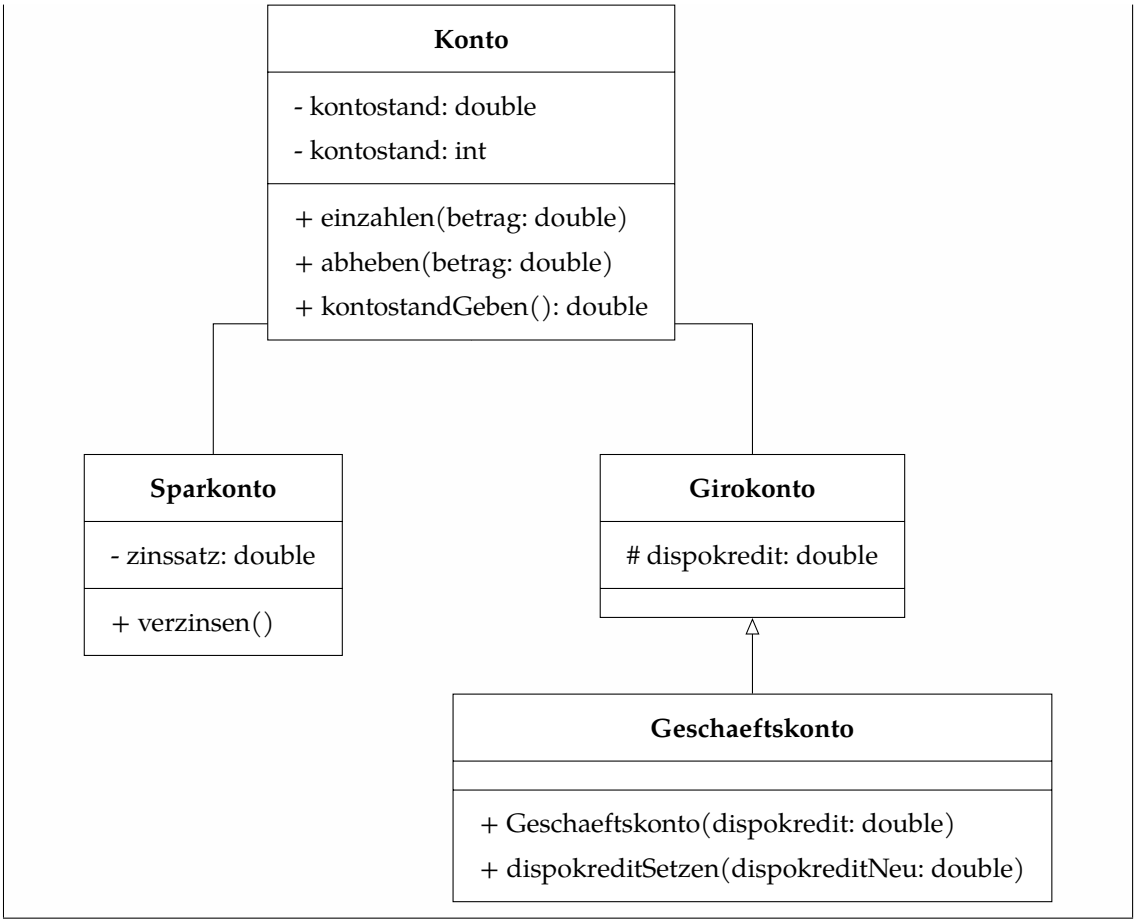
(b) Entwirf ein Klassendiagramm für die Klassen `Konto`, `Girokonto`, `Sparkonto`, `Geschaeftskonto`.

Lösungsvorschlag

Da das Geschäftskonto genauso wie das Girokonto einen Dispokredit besitzt, dieser nur anders festgelegt wird, wurde bei der Modellierung die Klasse `Geschaeftskonto` als Unterklasse der Klasse `Girokonto` umgesetzt.

Die Oberklasse `Konto` wurde abstrakt modelliert, da von ihr direkt keine Objekte erzeugt werden können.

Die Attribute `kontostand` und `kontonummer` in der Klasse `Konto` haben den Sichtbarkeitsmodifikator `private`, da die Unterklassen nie direkt auf die Attribute zugreifen, sondern die zur Verfügung stehenden Methoden dafür verwenden.



(c) Implementiere die Klassen in einem eigenen Projekt und teste die vorhandenen Methoden.

Lösungsvorschlag

```
public abstract class Konto {
  private double kontostand;
  @SuppressWarnings("unused")
  private int kontonummer;

  public Konto(int kontonummer) {
    this.kontonummer = kontonummer;
    kontostand = 0;
  }

  public void einzahlen(double betrag) {
    kontostand = kontostand + betrag;
  }

  public void abheben(double betrag) {
    // Ob abgehoben werden darf, entscheidet die Methode der jeweiligen
↪   Unterklasse.
    kontostand = kontostand - betrag;
  }

  public double kontostandGeben() {
```

```
    return kontostand;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Konto.java`

```
public class Sparkonto extends Konto {
  private double zinssatz;

  public Sparkonto(int kontonummer, double zinssatz) {
    super(kontonummer);
    // Aufruf des Konstruktors der Oberklasse
    this.zinssatz = zinssatz;
  }

  /**
   * Überschreiben der Methode abheben() aus der Oberklasse. Ist genügend
↪  Geld auf
   * dem Sparkonto...? dann darf man den gewünschten Betrag abheben, sonst
↪  nicht.
   */
  public void abheben(double betrag) {
    if (kontostandGeben() >= betrag) {
      super.abheben(betrag);
    }
  }

  /**
   * Der aktuelle Kontostand wird mit dem Zinssatz verrechnet und der sich
↪  daraus
   * ergebende Betrag (= Zinsen) dem Konto gutgeschrieben.
   */
  public void verzinsen() {
    einzahlen(kontostandGeben() * zinssatz);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Sparkonto.java`

```
public class Girokonto extends Konto {
  /**
   * Die Unterklasse muss auch auf das Attribut zugreifen können.
   */
  protected double dispokredit;

  public Girokonto(int kontonummer) {
    super(kontonummer);
    dispokredit = 2000;
  }

  /**
   * Die Methode abheben() kann direkt von der Oberklasse GIROKONTO und die
```

```
   * Methoden einzahlen() und kontostandGeben() von der Oberklasse KONTO
↪  genutzt
   * werden.
   *
   * Überschreiben der Methode abheben() der Klasse KONTO
   * Ist genügend Geld auf dem Konto bzw. reicht der Dispokredit aus...
   *  ...dann darf man den gewünschten Betrag abheben, sonst nicht.
   */
  public void abheben(double betrag) {//
    if (kontostandGeben() + dispokredit >= betrag) {
      super.abheben(betrag);
    }
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Girokonto.java`

```
public class Geschaeftskonto extends Girokonto {

  /**
   * Für das Geschäftskonto wird der Dispo individuell festgelegt.
   *
   * @param kontonummer Die Kontonummer
   * @param dispo Der maximale Rahme des Dispokredits.
   */
  public Geschaeftskonto(int kontonummer, double dispo) {
    super(kontonummer);
    dispokredit = dispo;
  }

  public void dispokreditSetzen(double dispokreditNeu) {
    dispokredit = dispokreditNeu;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Geschaeftskonto.java`

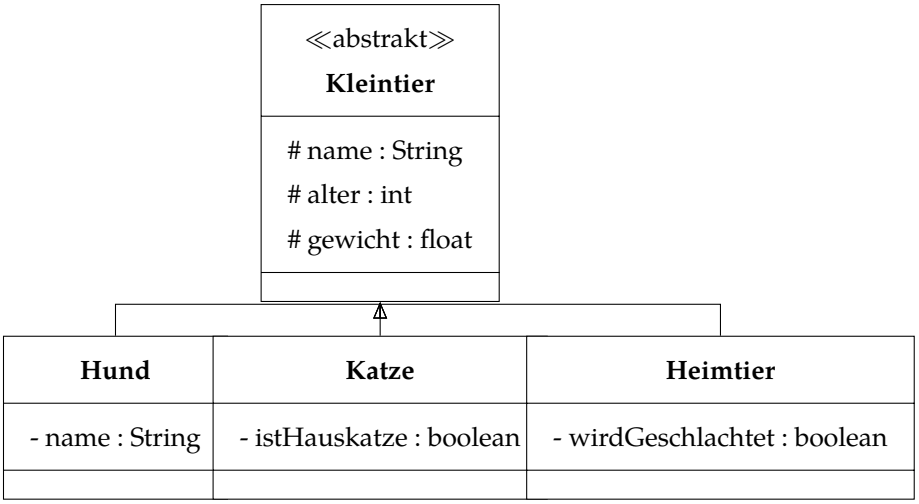
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Vererbung/Aufgabe_Bankkonten.tex`

Übungsaufgabe „Kleintierpraxis“ (Vererbung, Klassendiagramm, Implementierung in Java)

Vererbung
Klassendiagramm
Implementierung in Java

In der Kleintierpraxis Dr. Huf werden *Hunde*, *Katzen* und *Heimtiere*[2] behandelt. In der Praxissoftware werden von jedem Tier *Name*, *Alter* (in Jahren) und *Gewicht* (in kg mit mindestens 2 Dezimalen) erfasst. Bei einem *Hund* wird zusätzlich aufgenommen, ob er *reinrassig* ist, bei einer *Katze*, ob sie ausschließlich *in der Wohnung gehalten* wird, und beim *Heimtier*, ob es zur *Lebensmittellieferung* dient.

(a) Erstellen Sie ein entsprechendes Klassendiagramm (zunächst ohne Methoden), das den oben beschriebenen Sachverhalt geeignet erfasst.



(b) Implementieren Sie die Klassen gemäß des Modells aus Teilaufgabe a in der Programmierumgebung BlueJ. Beachten Sie dabei die abstrakte Oberklasse!

Klasse *Kleintier*

```
public abstract class Kleintier {
  protected String name;
  protected int alter;
  protected float gewicht;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java`

```
  public Kleintier(String name, int alter, float gewicht) {
    this.name = name;
    this.alter = alter;
    this.gewicht = gewicht;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java`

[2] Unter Heimtiere versteht man hier Kleintiere (d. h. keine Rinder, Pferde etc.), die zu Hause gehalten werden und keine Hunde oder Katzen sind (beispielsweise Meerschweinchen oder Kaninchen). Heimtiere können auch Nutztiere sein, die zur Schlachtung gehalten werden (z. B. Hühner). In diesem Fall muss ein Tierarzt darauf achten, nur Medikamente anzuwenden, die für Lebensmittel liefernde Tiere zugelassen sind.

Klasse *Hund*

```
public class Hund extends Kleintier {
  private boolean reinrassig;

  public Hund(String name, int alter, float gewicht, boolean reinrassig) {
    super(name, alter, gewicht);
    this.reinrassig = reinrassig;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Klasse *Katze*

```
public class Katze extends Kleintier {
  @SuppressWarnings("unused")
  private boolean istHauskatze;

  public Katze(String name, int alter, float gewicht, boolean istHauskatze) {
    super(name, alter, gewicht);
    this.istHauskatze = istHauskatze;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Katze.java`

Klasse *Heimtier*

```
public class Heimtier extends Kleintier {
  @SuppressWarnings("unused")
  private boolean wirdGeschlachtet;

  public Heimtier(String name, int alter, float gewicht, boolean
↪   wirdGeschlachtet) {
    super(name, alter, gewicht);
    this.wirdGeschlachtet = wirdGeschlachtet;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Heimtier.java`

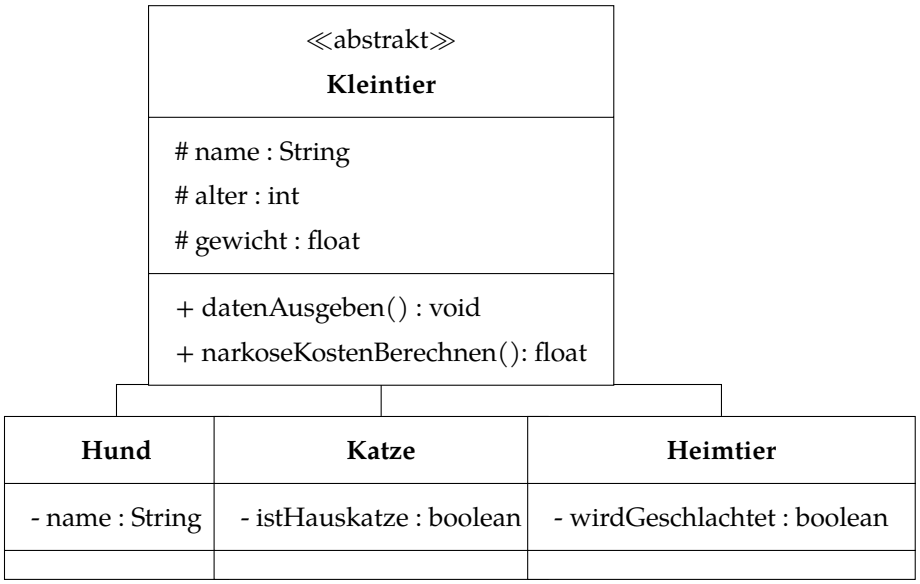
(c) Die Praxissoftware verfügt über eine Methode `datenAusgeben()`, die den Namen und das Alter eines Tieres auf dem Bildschirm ausgibt. Fügen Sie im Klassendiagramm die Methode an geeigneter Stelle ein und implementieren Sie sie.

Lösungsvorschlag

Einfügen in der abstrakten Klasse *Heimtier*. Fertiges Klassendiagramm siehe 1 (d)

```
    public void datenAusgeben()
    {
        System.out.println("Name: " + name + ", Alter: " + alter);
    }
```

(d) Die Kosten für eine Narkose setzen sich zusammen aus einer Grundgebühr, die von der Tierart abhängt, und aus den Kosten für das verwendete Narkotikum. Diese wiederum berechnen sich aus dem Preis des Narkotikums pro kg Körpergewicht multipliziert mit dem Gewicht des Tieres. Ergänzen Sie das Klassendiagramm entsprechend um die Methode `narkosekostenBerechnen()`, die die Kosten für eine Narkose auf dem Bildschirm ausgibt, und implementieren Sie sie.



Lösungsvorschlag

Einfügen in der abstrakten Klasse *Heimtier*.

```
public class Hund extends Kleintier {
  private boolean reinrassig;

  public Hund(String name, int alter, float gewicht, boolean reinrassig) {
    super(name, alter, gewicht);
    this.reinrassig = reinrassig;
    narkoseGrundGebuehr = 1.50f;
  }

  public void datenAusgeben() {
    System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

    if (reinrassig) {
      System.out.println("Der Hund ist reinrassig.");
    } else {
      System.out.println("Der Hund ist nicht reinrassig.");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Einfügen in den Konstruktor der Klasse *Hund*:

```
narkoseGrundGebuehr = 1.50f;
```

Einfügen in den Konstruktor der Klasse *Katze*:

```
narkoseGrundGebuehr = 1f;
```

Einfügen in den Konstruktor der Klasse *Heimtier*:

```
narkoseGrundGebuehr = 2f;
```

(e) Falls es sich bei dem zu behandelnden Tier um einen Hund handelt, soll die Methode `datenAusgeben()` (siehe Teilaufgabe c) dies zusammen mit dem Namen und dem Alter des Tieres auf dem Bildschirm ausgeben. Außerdem soll in diesem Fall ausgegeben werden, ob der Hund reinrassig ist oder nicht.

Lösungsvorschlag

Einfügen in der Klasse *Hund*:

```
public void datenAusgeben() {
  System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

  if (reinrassig) {
    System.out.println("Der Hund ist reinrassig.");
  } else {
    System.out.println("Der Hund ist nicht reinrassig.");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/20_OOMUP/Vererbung/Aufgabe_Kleintierpraxis.tex`

Übungsaufgabe „Grundwissen“ (Entwurfsmuster)

Entwurfsmuster

(a) Erklären Sie, was man bei der Entwicklung von Softwaresystemen unter einem Entwurfsmuster versteht und gehen Sie dabei auch auf die Vorteile ein.

Lösungsvorschlag

Entwurfsmuster sind bewährte Lösungsschablonen für wiederkehrende Entwurfsprobleme sowohl in der Architektur als auch in der Softwarearchitektur und -entwicklung. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist.

Der primäre Nutzen eines Entwurfsmusters liegt in der Beschreibung einer Lösung für eine bestimmte Klasse von Entwurfsproblemen. Weiterer Nutzen ergibt sich aus der Tatsache, dass jedes Muster einen Namen hat. Dies vereinfacht die Diskussion unter Entwicklern, da man abstrakt über eine Struktur sprechen kann. So sind etwa Software-Entwurfsmuster zunächst einmal unabhängig von der konkreten Programmiersprache. Wenn der Einsatz von Entwurfsmustern dokumentiert wird, ergibt sich ein weiterer Nutzen dadurch, dass durch die Beschreibung des Musters ein Bezug zur dort vorhandenen Diskussion des Problemkontextes und der Vor- und Nachteile der Lösung hergestellt wird.

(b) Nennen Sie die drei ursprünglichen Typen von Entwurfsmuster, erklären Sie diese kurz und geben Sie zu jedem Typ drei Beispiele an.

Lösungsvorschlag

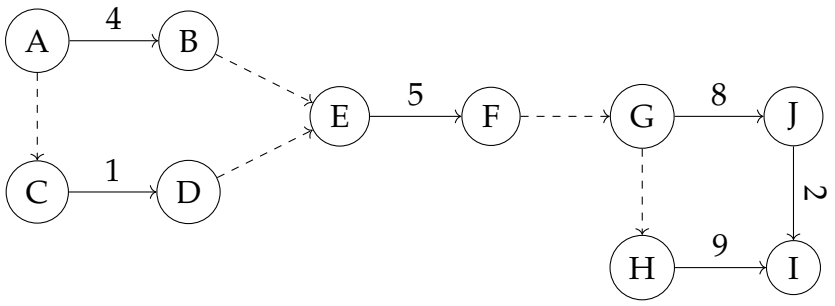
Typ	**Erlärung**	**Beispiele**
Erzeugungsmuster	Dienen der Erzeugung von Objekten; diese wird dadurch gekapselt und ausgelagert, um den Kontext der Objekterzeugung unabhängig von der konkreten Implementierung zu halten	abstrakte Fabrik, Singleton, Prototyp
Strukturmuster	Erleichtern den Entwurf von Software durch vorgefertigte Schablonen für Beziehungen zwischen Klassen.	Adapter, Kompositum, Stellvertreter
Verhaltensmuster	Modellieren komplexes Verhalten der Software und erhöhen damit die Flexibilität der Software hinsichtlich ihres Verhaltens.	Beobachter, Interpreter, Iterator

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/Entwurfsmuster/Aufgabe_Grundwissen.tex`

Projektplanung

Examensaufgabe „Gantt und CPM“ (46116-2015-F.T1-TA1-A3)

Betrachten Sie folgendes CPM-Netzwerk:



(a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

Lösungsvorschlag

i	Nebenrechnung	FZ_i
A		0
B		4
C		0
D		1
E	$\max(4,1)$	4
F		9
G		9
H		9
J		17
I	$\max(9_{(\to H)}+9, 17_{(\to J)}+2)$	19

(b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten.

Lösungsvorschlag

Gantt-Diagramm

i	Nebenrechnung	SZ_i
A	$\min(3,0)$	0
B		4
C		3
D		4
E		4
F		9
G	$\min(10,9)$	9
H		10
J		17
I		19

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten.

Lösungsvorschlag

i	A	B	C	D	E	F	G	H	J	I
FZ_i	0	4	0	1	4	9	9	9	17	19
SZ_i	0	4	3	4	4	9	9	10	17	19
GP	0	0	3	3	0	0	0	1	0	0

(d) Bestimmen Sie den kritischen Pfad.

Lösungsvorschlag

$A \to B \to E \to F \to G \to J \to I$

(e) Was ist ein Gantt-Diagramm? Worin unterscheidet es sich vom CPM-Netzwerk?

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Gantt und PERT“ (46116-2015-H.T1-TA1-A3)

Gantt-Diagramm

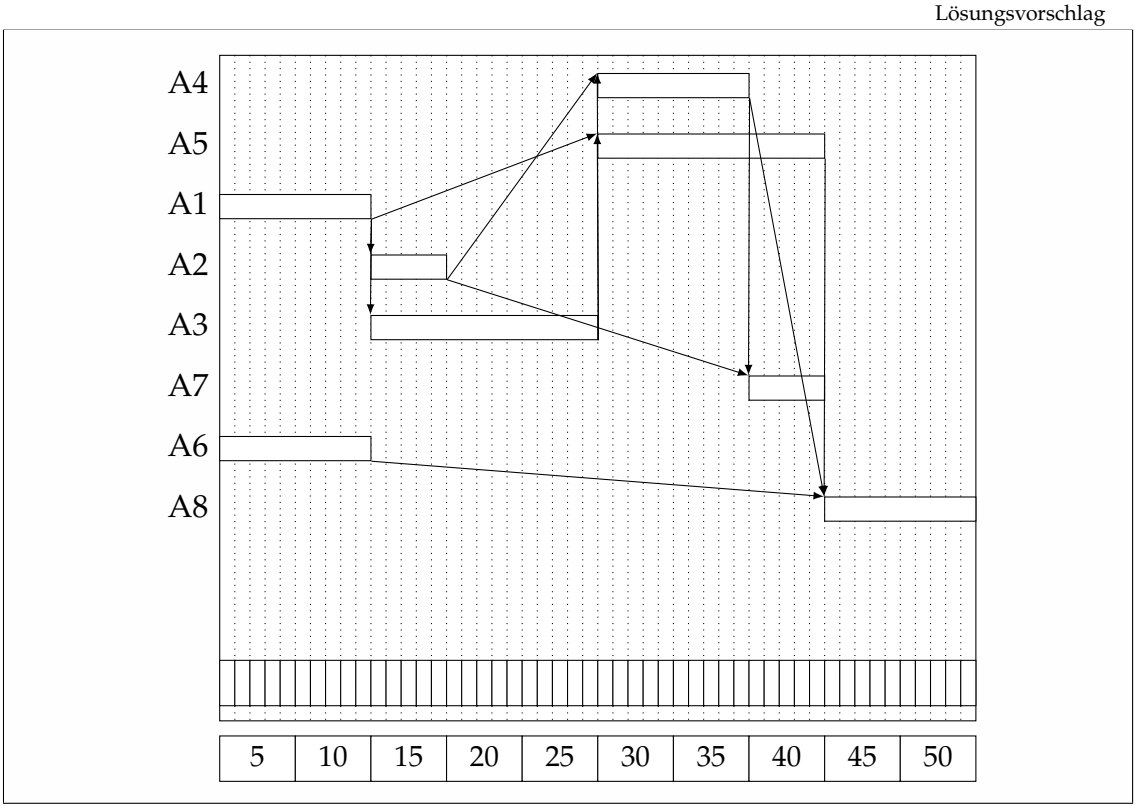
Betrachten Sie die folgende Tabelle zum Projektmanagement:

Name	Dauer (Tage)	Abhängig von
A1	10	
A2	5	A1
A3	15	A1
A4	10	A2, A3
A5	15	A1, A3
A6	10	
A7	5	A2, A4
A8	10	A4, A5, A6

Tabelle 1: Übersicht Arbeitspakete

(a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

Lösungsvorschlag



(b) Wie lange dauert das Projekt mindestens?

(c) Geben Sie den oder die kritischen Pfad(e) an.

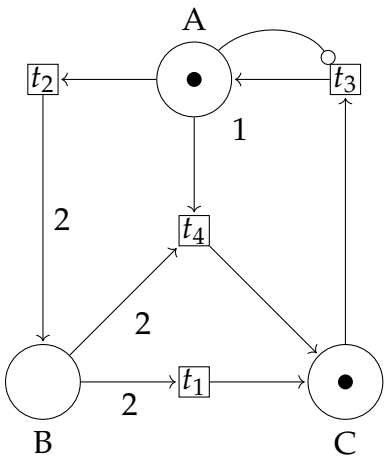
(d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Petri-Netz“ (46116-2016-F.T2-TA1-A2)

Petri-Netz
Erreichbarkeitsgraph

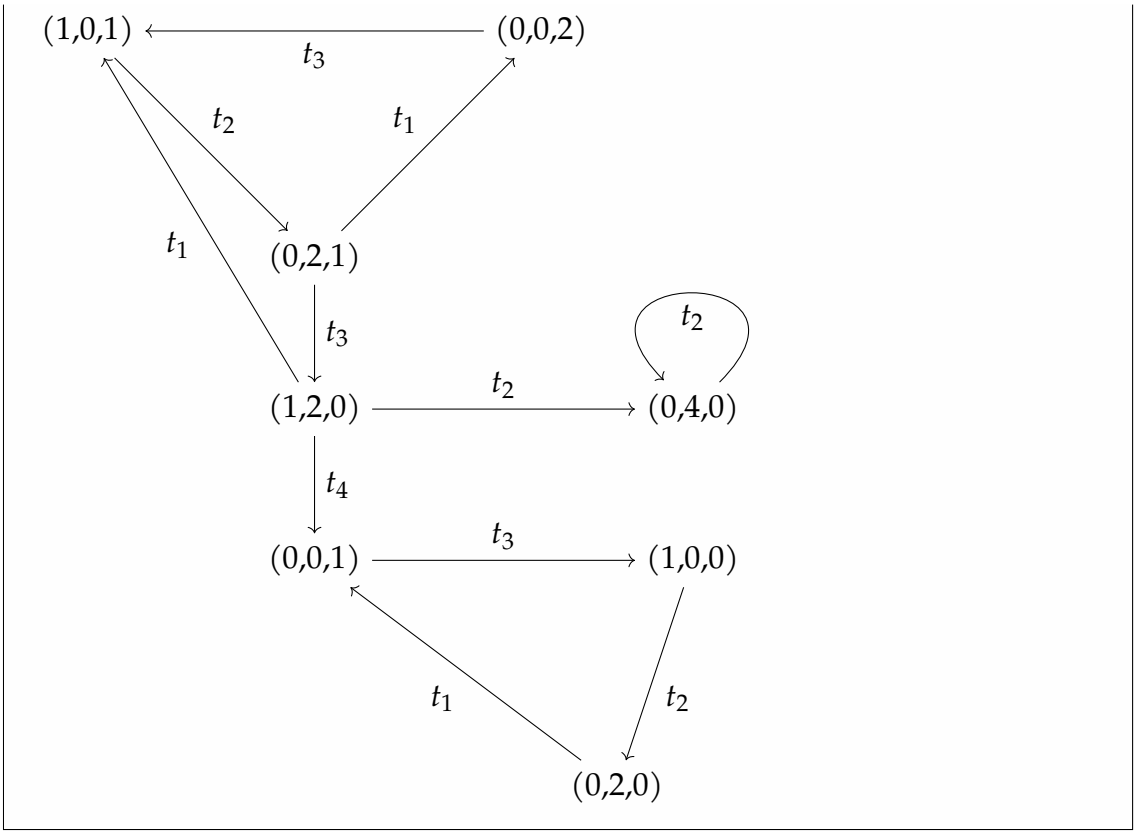
Gegeben sei das folgende Petri-Netz:



(a) Erstellen Sie den zum Petri-Netz gehörenden Erreichbarkeitsgraphen. Die Belegungen sind jeweils in der Form [A, B, C] anzugeben. Beschriften Sie auch jede Kante mit der zugehörigen Transition. Beachten Sie die auf 1 beschränkte Kapazität von Stelle A oder alternativ die Inhibitor-Kante von A zu t_3 (beides ist hier semantisch äquivalent).

Lösungsvorschlag

$$
\begin{array}{lcll}
(0,0,1) & \rightarrow & t_3 & \rightarrow (1,0,0) \\
(0,0,2) & \rightarrow & t_3 & \rightarrow (1,0,1) \\
(0,2,0) & \rightarrow & t_1 & \rightarrow (0,0,1) \\
(0,2,1) & \rightarrow & t_1 & \rightarrow (0,0,2) \\
(0,2,1) & \rightarrow & t_3 & \rightarrow (1,2,0) \\
(1,0,0) & \rightarrow & t_2 & \rightarrow (0,2,0) \\
(1,0,1) & \rightarrow & t_2 & \rightarrow (0,2,1) \\
(1,2,0) & \rightarrow & t_1 & \rightarrow (1,0,1) \\
(1,2,0) & \rightarrow & t_2 & \rightarrow (0,4,0) \\
(1,2,0) & \rightarrow & t_4 & \rightarrow (0,0,1) \\
(0,4,0) & \rightarrow & t_2 & \rightarrow (0,4,0)
\end{array}
$$



(b) Wie kann man mit Hilfe des Erreichbarkeitsgraphen feststellen, ob ein Petri-Netz lebendig ist?

(c) Aufgrund von Transition t_4 ist das gegebene Petri-Netz nicht stark lebendig. Wie müssten die Pfeilgewichte der Transition t_4 verändert werden, damit das Petri-Netz mit der gegebenen Startmarkierung beschränkt bleibt und lebendig wird?

Lösungsvorschlag

t_4 nach C mit Gewicht 2 versehen

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2016/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Gantt und PERT“ (46116-2017-F.T1-TA1-A5)

Gantt-Diagramm

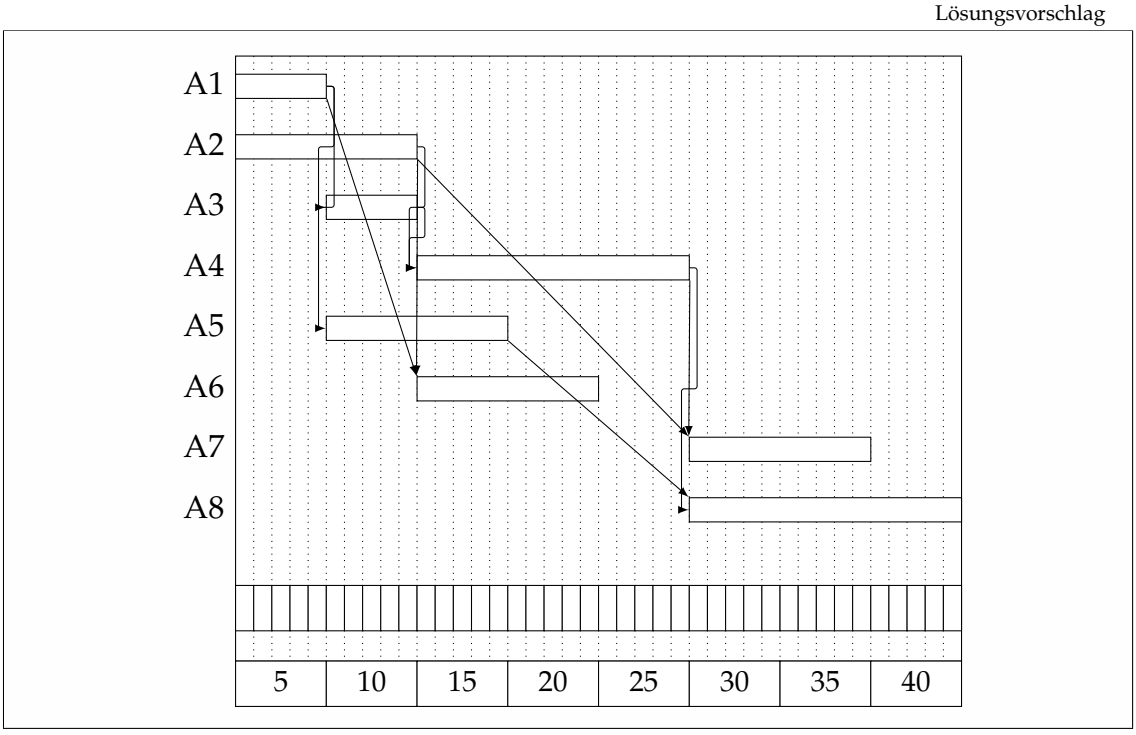
Betrachten Sie die folgende Tabelle zum Projektmanagement:

Name	Dauer (Tage)	Abhängig von
A1	5	
A2	10	
A3	5	A1
AA	15	A2, A3
AS	10	A1
A6	10	A1, A2
A7	10	A2, A4
A8	15	A4, A5

Tabelle 1: Übersicht Arbeitspakete

(a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

Lösungsvorschlag



(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

40 Tage

(c) Geben Sie den oder die kritischen Pfad(e) an.

Lösungsvorschlag

A2 A4 A8

A1 A3 A4 A8

(d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

Lösungsvorschlag

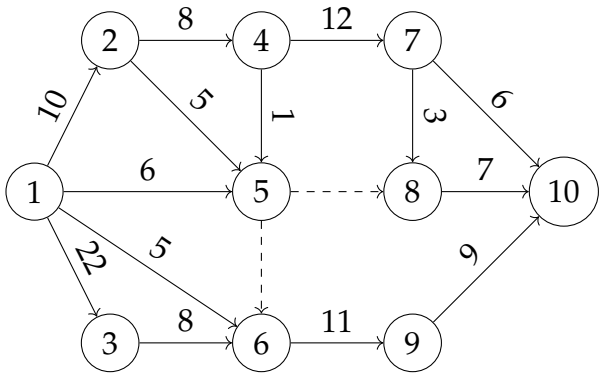
Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2017/03/Thema-1/Teilaufgabe-1/Aufgabe-5.tex`

Examensaufgabe „Gantt und CPM“ (66116-2012-H.T2-TA2-A2)

CPM-Netzplantechnik

Die unten stehende Abbildung stellt ein CPM-Netzwerk dar. Die Ereignisse sind fortlaufend nummeriert (Nummer im Inneren der Kreise) und tragen keine Namen. Gestrichelte Linien stellen Pseudo-Aktivitäten mit einer Dauer von 0 dar.



(a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet!

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** i: Ereignis i; FZ_i: Frühester Zeitpunkt, zu dem Ereignis i eintreten kann. —

i	Nebenrechnung	FZ_i
1		0
2		10
3		22
4		18
5	$\max(15_2, 6_1, 19_4)$	19
6	$\max(5_1, 30_6, 19_5)$	30
7		30
8	$\max(33_7, 19_5)$	33
9		41
10	$\max(36_7, 40_8, 50_9)$	50

(b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten!

Lösungsvorschlag

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** i: Ereignis i; SZ_i: Spätester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————

i	Nebenrechnung	SZ_i
10	siehe FZ_{10}	50
9		41
8		43
7	$\min(44_{10}, 40_8)$	40
6		30
5	$\min(30_6, 43_8)$	30
4	$\min(29_5, 28_7)$	28
3		22
2	$\min(20_4, 25_5)$	20
1	$\min(10_2, 24_5, 0_3, 25_6)$	0

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten!

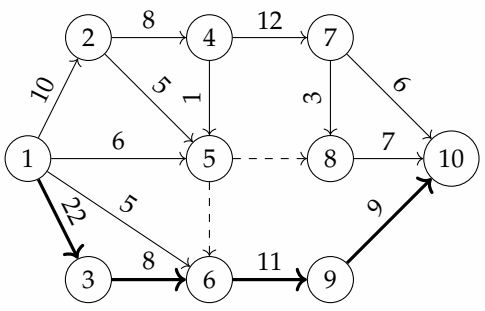
Lösungsvorschlag

i	1	2	3	4	5	6	7	8	9	10
FZ_i	0	10	22	18	19	30	30	33	41	50
SZ_i	0	20	22	28	30	30	40	43	41	50
GP	0	10	0	10	11	0	10	10	0	0

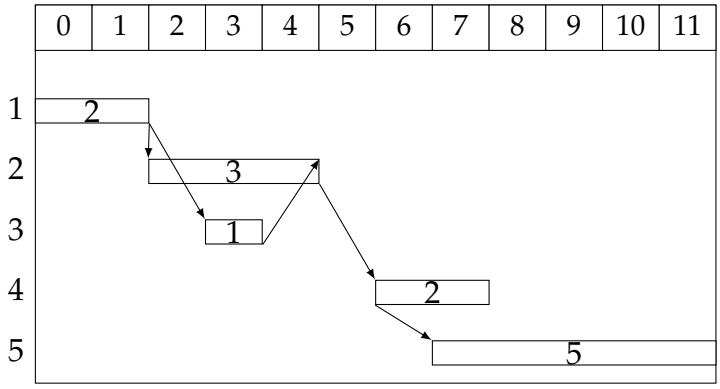
(d) Bestimmen Sie den kritischen Pfad!

Lösungsvorschlag

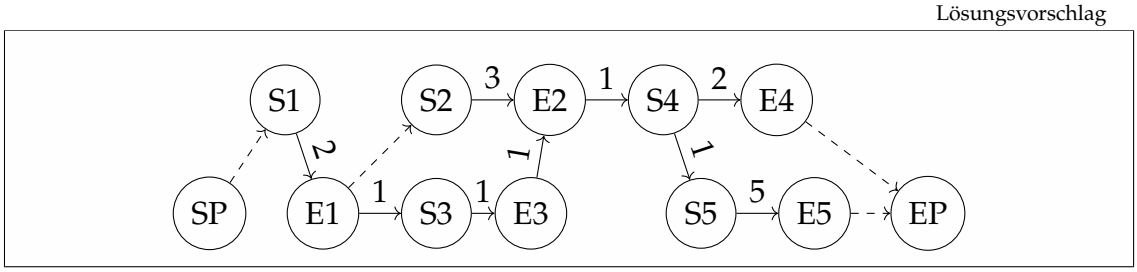
$1 \to 3 \to 6 \to 9 \to 10$



(e) Konvertieren Sie das Gantt-Diagramm aus Abbildung 3 in ein CPM-Netzwerk! Gantt-Diagramm



Lösungsvorschlag



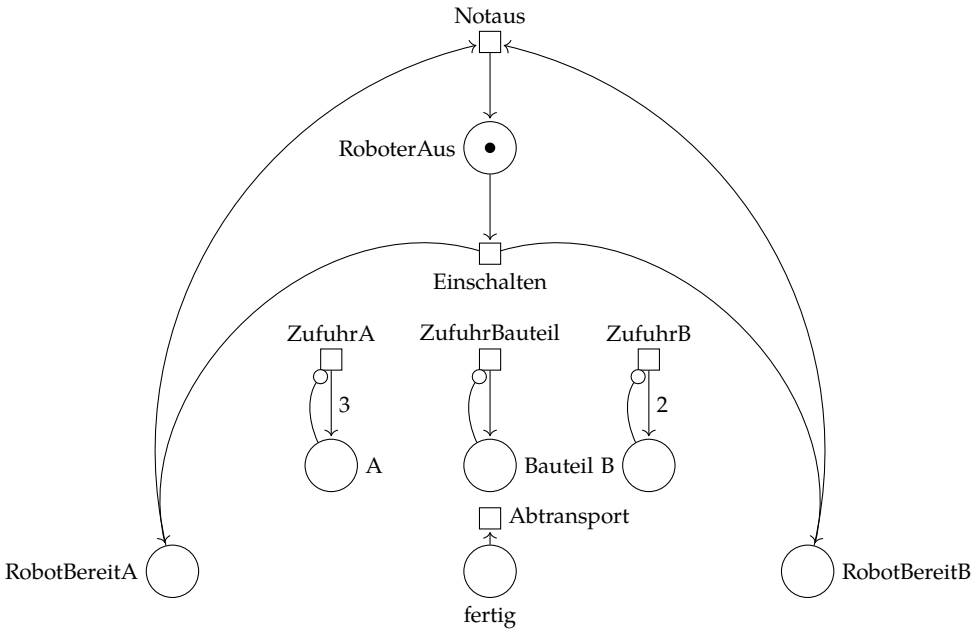
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2012/09/Thema-2/Teilaufgabe-2/Aufgabe-2.tex`

Examensaufgabe „Automatisierungsanlage mit zwei Robotern“ (66116-2015-F.T2-TA2-A3)

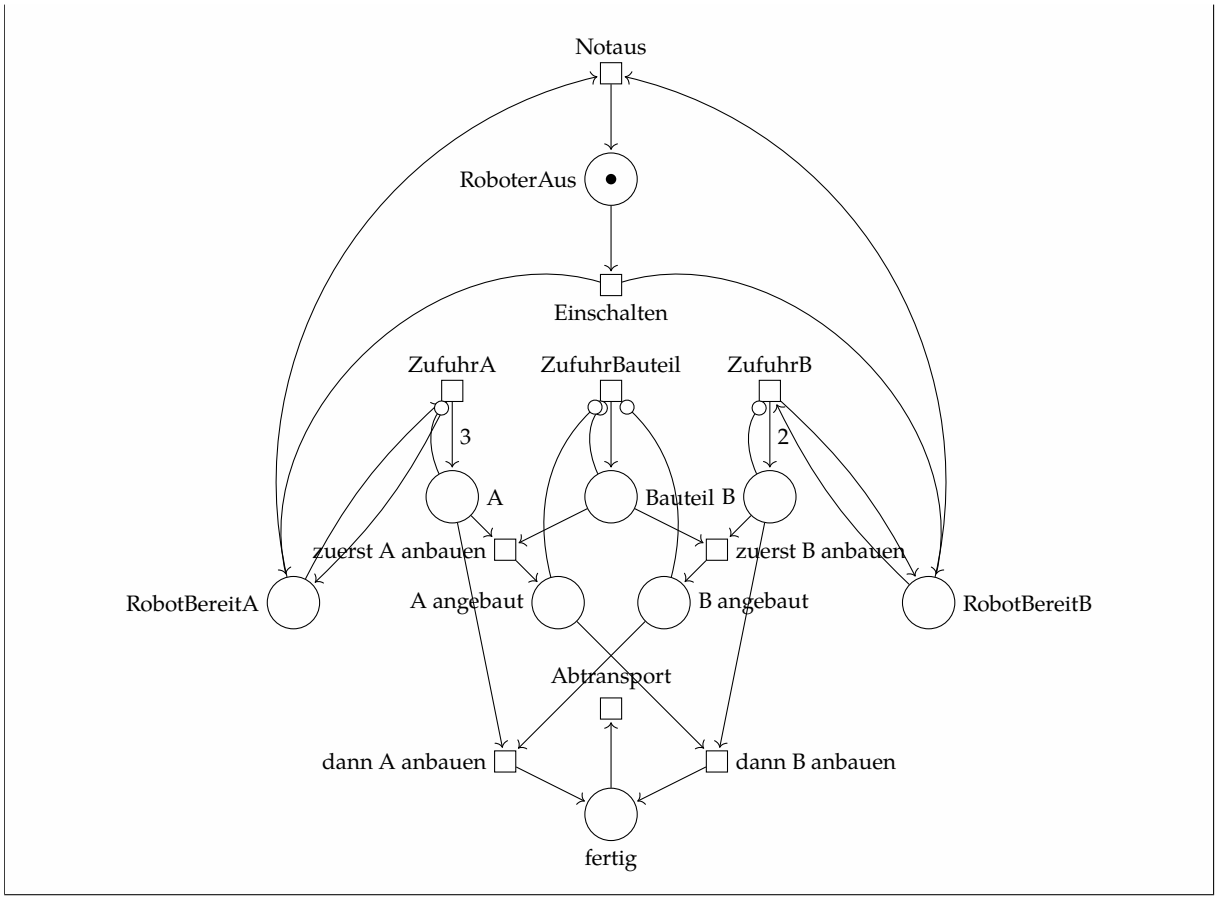
Petri-Netz

Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten werden:

In der Anlage arbeiten zwei Roboter A und B, die über einen Schalter „Einschalten“ aktiviert und *jederzeit* über einen „Notaus “-Schalter deaktiviert werden können müssen. Aufgabe der Roboter ist es, jeweils abwechselnd an Bauteilen zu arbeiten, die einzeln über ein Förderband „ZufuhrBauteil“ ins System eingefahren und nach der Fertigstellung mittels „Abtransport“ aus dem System abgeführt werden. Roboter A bringt genau 3 Anbauten vom Typ „A“ an jedes Bauteil an und Roboter B macht das entsprechend mit genau 2 Anbauten vom Typ „B“. Die Anbauten werden jeweils passend über „ZufuhrA“ auf „A“ bzw. mittels „ZufuhrB“ auf „B“ bereitgestellt. Die beiden Roboter dürfen *niemals* gleichzeitig an einem Bauteil arbeiten — die Reihenfolge, in der sie darauf zugreifen, ist jedoch beliebig und darf von Bauteil zu Bauteil frei varüeren. Sobald einer der Roboter mit der Arbeit an einem neuen Bauteil begonnen hat, darf kein weiteres Bauteil angenommen werden, ehe der jeweils andere Roboter nicht ebenfalls mit seiner Arbeit am gleichen Bauteil fertig geworden ist (und das fertige Bauteil „auf den Platz ‚fertig‘ legt“). Aus Platzgründen darf höchstens ein Bauteil auf dem Ausgangsplatz „fertig“ abgestellt werden, ehe es abtransportiert wird.



Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/03/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Gantt und zwei Softwareentwickler“ (66116-2018-H.T2-TA1-A1)

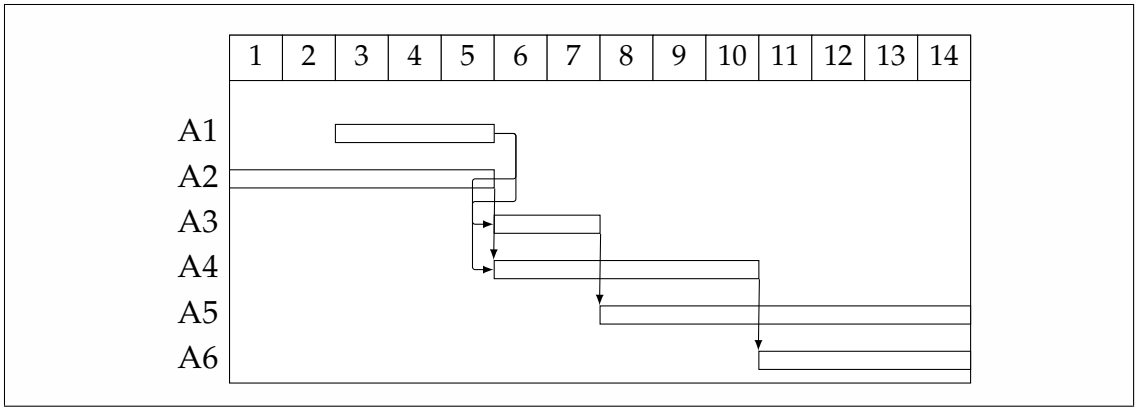
Gantt-Diagramm
CPM-Netzplantechnik

Ein Team von zwei Softwareentwicklern soll ein Projekt umsetzen, das in sechs Arbeitspakete unterteilt ist. Die Dauer der Arbeitspakete und ihre Abhängigkeiten können Sie aus folgender Tabelle entnehmen:

Name	Dauer in Wochen	Abhängig von
A1	2	-
A2	5	-
A3	2	A1
A4	5	A1, A2
A5	7	A3
A6	4	A4

(a) Zeichnen Sie ein Gantt-Diagramm, das eine kürzestmögliche Projektabwicklung beinhaltet.

Lösungsvorschlag



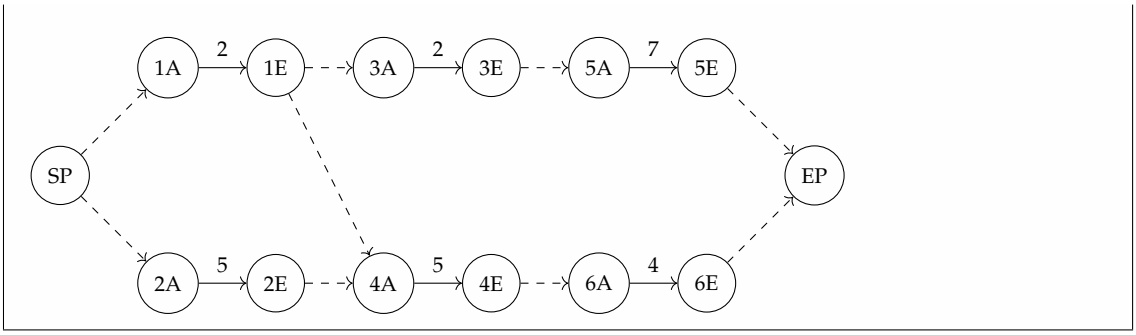
(b) Bestimmen Sie die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Lösungsvorschlag

Auf dem kritischen Pfad befinden die Arbeitspakete A2, A4 und A6. Die Länge des kritischen Pfades ist 14.

(c) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzplan um.

Lösungsvorschlag



(d) Berechnen Sie für jedes Ereignis den *frühesten Termin* und den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

Lösungsvorschlag

i	SP	1A	1E	2A	2E	3A	3E	4A	4E	5A	5E	6A	6E	EP
FZ_i	0	0	2	0	5	2	4	5	10	4	11	10	14	14
SZ_i	0	3	5	0	5	5	7	5	10	7	14	10	14	14
GP	0	3	3	0	0	3	3	0	0	3	3	0	0	0

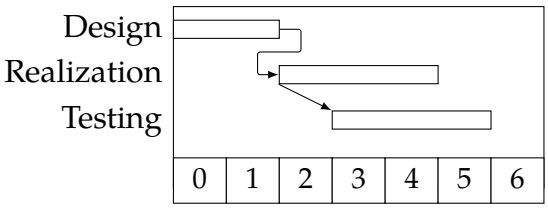
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2018/09/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Projektplanung“ (66116-2020-H.T1-TA1-A2)

Projektplanung
CPM-Netzplantechnik
Gantt-Diagramm

Die Planung eines Softwareprojekts kann z. B. in Form von Gantt-Diagrammen oder CPM-Netzwerken (kritischer Pfad Methode) festgehalten werden.

Folgendes Gantt-Diagramm zeigt einen Teil der Projektplanung in einem klassischen Softwareentwicklungsprozess:



(a) Im Diagramm werden 3 Phasen aus dem klassischen Softwareentwicklungsprozess genannt. Welche Phase sollte dem Design (Entwurf) immer vorangehen?

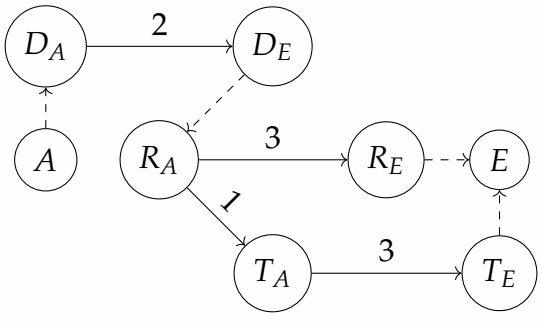
Lösungsvorschlag

Die Anforderungsdefinition

(b) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzwerk um. Fügen Sie dazu einen zusätzlichen Start- und Endknoten hinzu. Das Ende des Projekts ist durch das Ende aller Aktivitäten bedingt.

Lösungsvorschlag

D_A Design Anfang
R_A Realization Anfang
T_A Testing Anfang
D_E Design Ende
R_E Realization Ende
T_E Testing Ende



(c) Welche im obigen Gantt-Diagramm nicht enthaltenen Beziehungsarten zwischen Aktivitäten können in einem Gantt-Diagramm noch auftreten? Nennen Sie auch deren Bedeutung.

Lösungsvorschlag

Diese Beziehungsarten sind im obigen Gantt-Diagramm vorhanden:

Normalfolge EA: *end-to-start relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Anfang seines Nachfolgers.

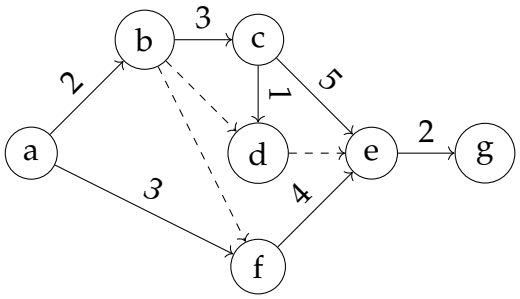
Anfangsfolge AA: *start-to-start relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Anfang seines Nachfolgers.

Diese Beziehungsarten sind im obigen Gantt-Diagramm *nicht* vorhanden:

Endefolge EE: *finish-to-finish relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Ende seines Nachfolgers.

Sprungfolge AE: *start-to-finish relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Ende seines Nachfolgers

Gegeben sei nun das folgende CPM-Netzwerk:



(d) Geben Sie für jedes Ereignis die früheste Zeit an.

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** i: Ereignis i; FZ_i: Frühester Zeitpunkt, zu dem Ereignis i eintreten kann. —

i	Nebenrechnung	FZ_i
a		0
b		2
c		5
d	$\max(2_b, 6_c)$	6
e	$\max(6_d, 10_e, 7_f)$	10
f	$\max(3_f, 2_b)$	3
g		12

(e) Geben Sie für jedes Ereignis die späteste Zeit an.

Lösungsvorschlag

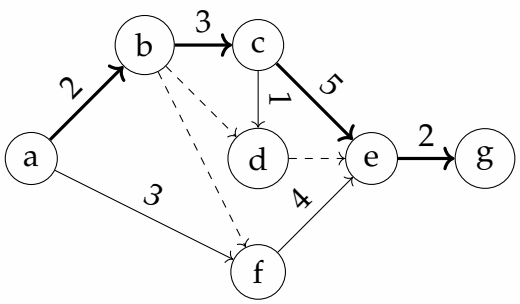
— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** i: Ereignis i; SZ_i: Spätester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————

i	Nebenrechnung	SZ_i
g		12
f		6
e		10
d		10
c	$\min(9_d, 5_e)$	5
b	$\min(2_c, 10_d, 6_f)$	2
a		0

(f) Geben Sie einen kritischen Pfad durch das Netz an! Wie wirkt sich eine Verzögerung von 5 Zeiteinheiten auf dem kritischen Pfad auf das Projektende aus?

Lösungsvorschlag

i	a	b	c	d	e	f	g
FZ_i	0	2	5	6	10	3	12
SZ_i	0	2	5	10	10	6	12
GP	0	0	0	3	0	3	0



Kritischer Pfad: a $\rightarrow$ b $\rightarrow$ c $\rightarrow$ e $\rightarrow$ g

Das Projekt verlängert sich um 5 Zeiteinheiten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Projektmanagement“ (66116-2020-H.T2-TA1-A2)

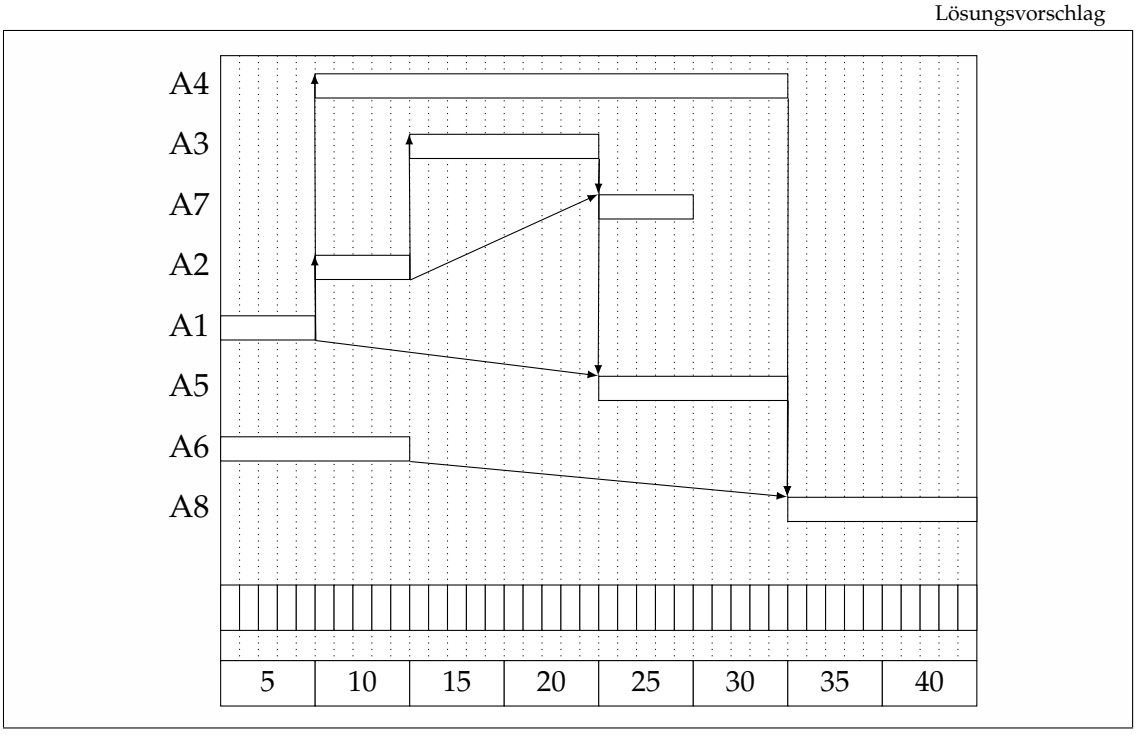
Gantt-Diagramm

Betrachten Sie die folgende Tabelle zum Projektmanagement:

Arbeitspaket	Dauer (Tage)	abhängig von
A1	5	
A2	5	A1
A3	10	A2
A4	25	A1
A5	10	A1, A3
A6	10	
A7	5	A2, A3
A8	10	A4, A5, A6

(a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt. Das Diagramm muss nicht maßstabsgetreu sein, jedoch jede Information aus der gegebenen Tabelle enthalten.

Lösungsvorschlag



(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

Es dauert mindestens 40 Tage.

(c) Geben Sie alle kritischen Pfade an.

Lösungsvorschlag

$A1 \rightarrow A4 \rightarrow A8$

$A1 \rightarrow A2 \rightarrow A3 \rightarrow A5 \rightarrow A8$

(d) Bewerten Sie folgende Aussage eines Projektmanagers: „Falls unser Projekt in Verzug gerät, bringen uns neue Programmierer auch nicht weiter.“

Lösungsvorschlag

Ist der Verzug in einem Arbeitspaket, in dem genügend Pufferzeit vorhanden ist (hier A6), so hilft ein neuer Programmierer nicht unbedingt weiter. Geht allerdings die Pufferzeit zu Ende oder ist erst gar nicht vorhanden (z. B. im kritischen Pfad), so kann ein/e neue/r ProgrammerIn helfen, falls deren/dessen Einarbeitsungszeit gering ist. Muss sich der/die Neue erste komplett einarbeiten, so wird er/sie wohl auch keine große Hilfe sein.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Projektmanagement“ (66116-2021-F.T2-TA1-A1)

CPM-Netzplantechnik

Gegeben seien folgende Tätigkeiten mit ihren Abhängigkeiten und Dauern:

Task	Dauer (in h)	Abhängigkeiten
T1	3	/
T2	6	/
T3	2	T1
T4	2	T2
T5	5	T1
T6	3	T4, T5
T7	6	T3
T8	7	T4
T9	4	T6, T8
T10	1	T7, T9

(a) Zeichnen Sie ein CPM-Diagramm basierend auf der gegebenen Aufgabenliste. Benutzen Sie explizite Start- und Endknoten.

Lösungsvorschlag

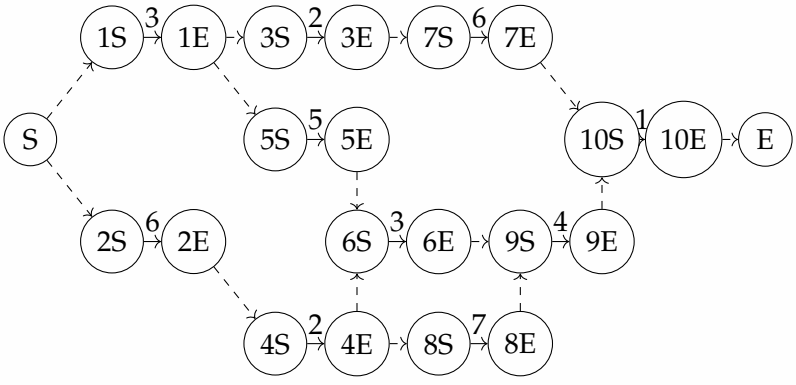
Abkürzungen

S Start

1S Start von T1

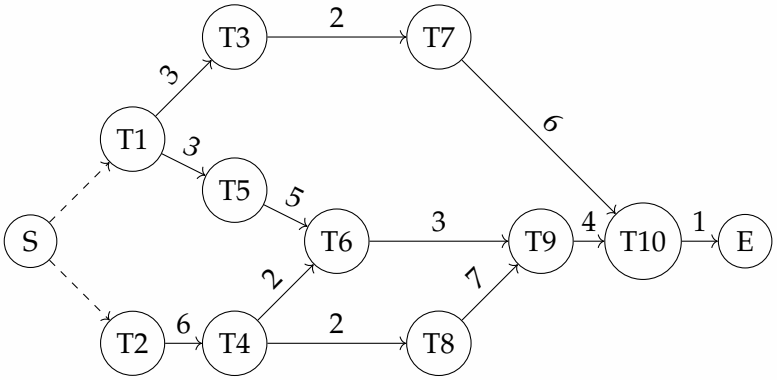
1E Ende von T1

E Ende

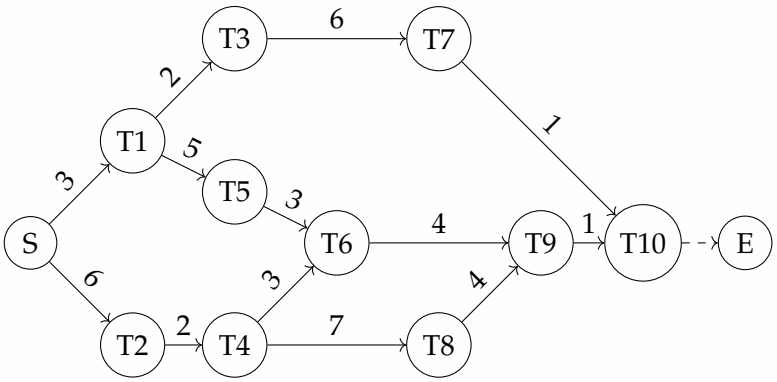


Teilen wir einen Task in zwei Knoten auf, so wird das Diagramm sehr unübersichtlich. Wir verwenden pro Task nur einen Knoten. Es gibt zwei Möglichkeiten:

Knoten sind Anfang der Tasks



Knoten sind Ende der Tasks



(b) Als *Slack* bezeichnet man die Zeit, um die eine Aufgabe bezüglich ihres frühesten Startzeitpunktes verzögert werden kann, ohne dass es Probleme bei der fristgerechten Fertigstellung des Projektes gibt. Berechnen Sie den Slack für alle Aktivitäten und ergänzen Sie ihn in Ihrem Diagramm.

Lösungsvorschlag

Knoten sind Anfang der Tasks

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** i: Ereignis i; FZ_i: Frühester Zeitpunkt, zu dem Ereignis i eintreten kann. —

i	Nebenrechnung	FZ_i
T1		0
T2		0
T3		3
T4		6
T5		3
T6	$\max(8_{T4}, 8_{T5})$	8
T7		5
T8		8
T9	$\max(11_{T6}, 15_{T4})$	15
T10	$\max(19_{T9}, 11_{T7})$	19
E		20

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** i: Ereignis i; SZ_i: Spätester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————

i	Nebenrechnung	SZ_i
E		20
T10		19
T9		15
T8		8
T7		13
T6		12
T5		7
T4	$\min(12_{T6}, 6_{T8})$	6
T3		11
T2		0
T1	$\min(8_{T3}, 4_{T5})$	4

i	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	E
FZ_i	0	0	3	6	3	8	5	8	15	19	20
SZ_i	4	0	11	6	7	12	13	8	15	19	20
GP	4	0	8	0	4	4	8	0	0	0	0

Knoten sind Ende der Tasks

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** i: Ereignis i; FZ_i: Frühester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————————————

i	Nebenrechnung	FZ_i
T1		3
T2		6
T3		5
T4		8
T5		8
T6	$\max(11_{T4}, 11_{T5})$	11
T7		11
T8		15
T9	$\max(15_{T6}, 19_{T8})$	19
T10	$\max(20_{T9}, 12_{T7})$	20
E		20

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** i: Ereignis i; SZ_i: Spätester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————————

i	Nebenrechnung	SZ_i
E		20
T10		20
T9		19
T8		15
T7		19
T6		15
T5		12
T4	$\min(12_{T6}, 8_{T8})$	8
T3		13
T2		6
T1	$\min(11_{T3}, 7_{T5})$	7

i	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	E
FZ_i	3	6	5	8	8	11	11	15	19	20	20
SZ_i	7	6	13	8	12	15	19	15	19	20	20
GP	4	0	8	0	4	4	8	0	0	0	0

(c) Zeichnen Sie den kritischen Pfad in Ihr Diagramm ein oder geben Sie die Tasks des kritischen Pfades in der folgenden Form an: **Start** ! . . . ! **Ende**. Sollte es mehrere kritische Pfade geben, geben Sie auch diese an. Wie lange ist die Dauer des kritischen Pfades bzw. der kritischen Pfade?

Lösungsvorschlag

Kritischer Pfad: **Start** ! T2 ! T4 ! T8 ! T9 ! T10 ! **Ende**

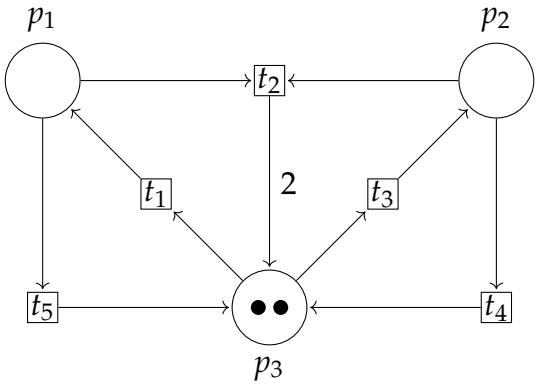
Dauer: 20 h

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Übungsaufgabe „Alles“ (Petri-Netz, Erreichbarkeitsgraph)

Petri-Netz
Erreichbarkeitsgraph

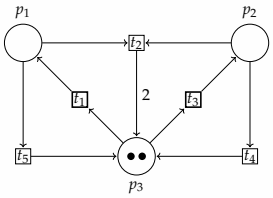
Gegeben Sei das folgende Petri-Netz:



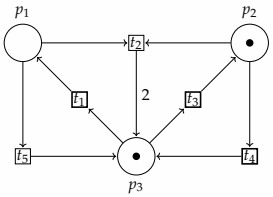
(a) Zeichnen Sie den Erreichbarkeitsgraphen des Petri-Netzes.

Lösungsvorschlag

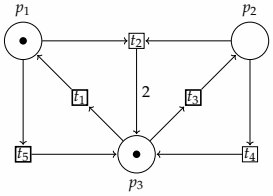
(0,0,2):



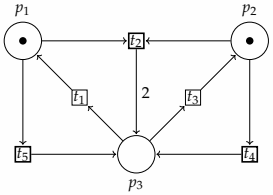
(0,1,1):

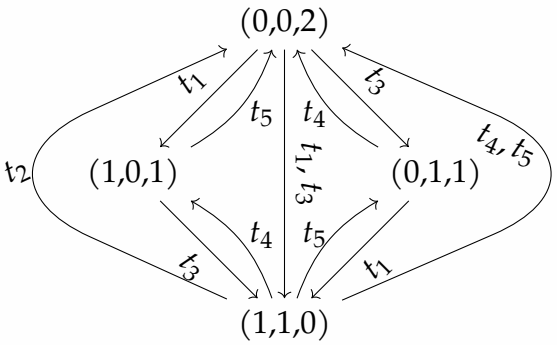


(1,0,1):



(1,1,0):





(b) Begründen Sie anhand des Erreichbarkeitsgraphen, ob das Petri-Netz lebendig, umkehrbar und/oder verklemmungsfrei ist.

Lösungsvorschlag

lebenig Ja. Es gibt im Erreichbarkeitsgraphen keine Senke, also keinen Zustand, aus dem man nicht mehr heraus kommt.

umkehrbar Im Erreichbarkeitsgraphen kommt man von $(0,0,2)$ auf verschiedenen Wegen wieder zurück zu $(0,0,2)$. Man kann sich unendlich oft im Graph bewegen. Die Anfangsmarkierung kann wieder erreicht werden ($t_1 \to t_3 \to t_2$ oder $t_1 \to t_3 \to t_5 \to t_4$).

verklemmungsfrei Ja. Es gibt im Erreichbarkeitsgraphen keine Senke, also keinen Zustand, aus dem man nicht mehr herauskommt.

(c) Geben Sie die Darstellungsmatrix A sowie den Belegungsvektor v an und berechnen Sie damit die Belegung nach Schaltung von $t_1 \to t_3 \to t_2$.

Lösungsvorschlag

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & -1 & 0 \\ -1 & 2 & -1 & 1 & 1 \end{pmatrix} \end{array}, \quad v = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

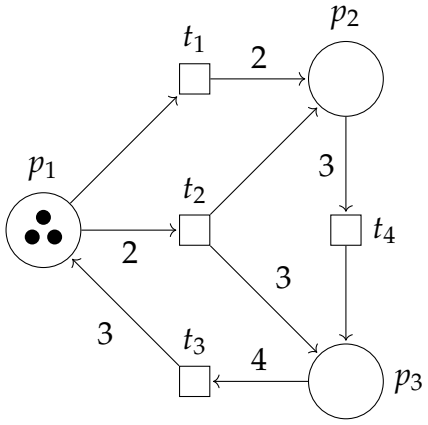
$$v_{\text{neu}} = v + A \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} = v$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/10_Petri-Netze/Aufgabe_Alles.tex`

Übungsaufgabe „Erreichbarkeitsgraph“ (Petri-Netz, Erreichbarkeitsgraph)

Petri-Netz
Erreichbarkeitsgraph

Gegeben ist das folgende Petri-Netz:



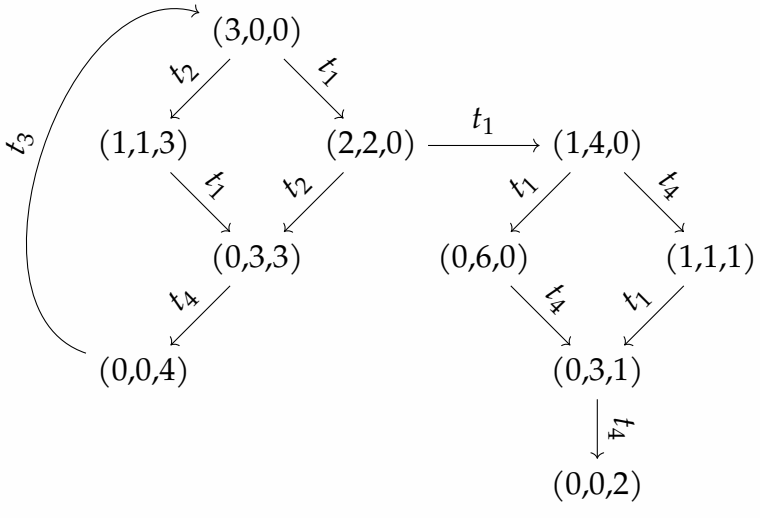
(a) Geben Sie die dazugehörige Darstellungsmatrix sowie den Belegungsvektor an.

Lösungsvorschlag

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & -2 & 3 & 0 \\ 2 & 1 & 0 & -3 \\ 0 & 3 & -4 & 1 \end{pmatrix} \end{array}, \quad v = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$$

(b) Skizzieren Sie den Erreichbarkeitsgraphen des Petri-Netzes.

Lösungsvorschlag



(c) Begründen Sie anhand des Erreichbarkeitsgraphen, ob das Petri-Netz verklemmungsfrei ist oder nicht.

Lösungsvorschlag

Durch Schalten von $t_1 \rightarrow t_1 \rightarrow t_1 \rightarrow t_4 \rightarrow t_4$ wird beispielsweise eine Verklemmung erreicht. Das Petri-Netz ist also nicht verklemmungsfrei. Am Erreichbarkeitsgraphen erkennt man das anhand der Senke im Knoten $[0,0,2]$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/10_Petri-Netze/Aufgabe_Erreichbarkeitsgraph.tex

Übungsaufgabe „Modellierung“ (Petri-Netz)

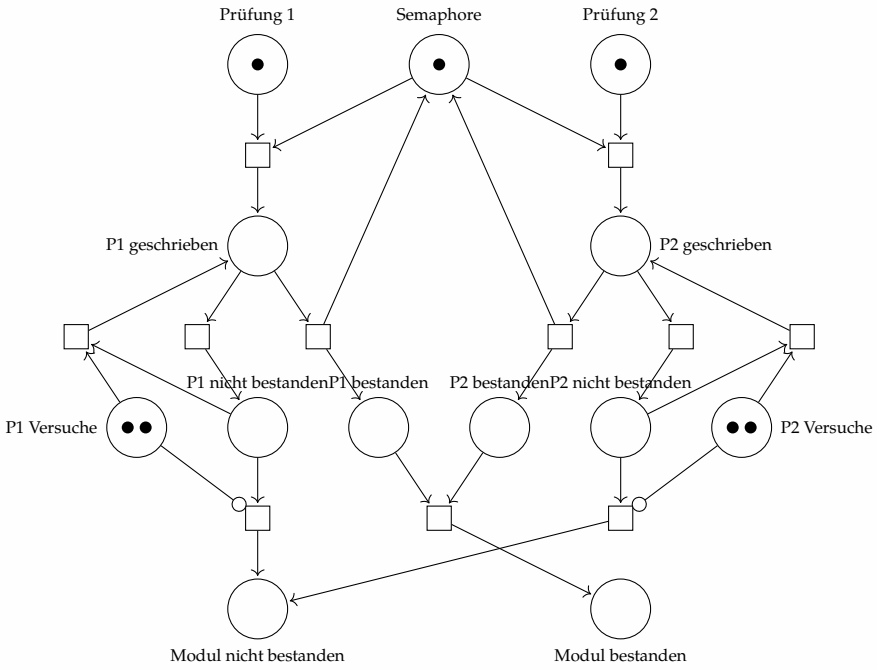
Petri-Netz

Modellieren Sie folgendes Szenario als Petri-Netz:

Ein Modul gilt als bestanden, wenn beide Prüfungen P_1 und P_2 bestanden sind. Beide Prüfungen dürfen bei Nicht-Bestehen jeweils maximal zwei mal wiederholt werden. Die Prüfungen dürfen nicht gleichzeitig geschrieben werden. Erst wenn eine von beiden bestanden wurde, darf die nächste begonnen werden. Wurde eine der beiden Prüfungen insgesamt drei mal nicht bestanden, so gilt das gesamte Modul als nicht bestanden.

Lösungsvorschlag

Man beachte die sog. Inhibitorkanten Linien mit Punkt am Ende). Z. B. kann t 7 nur schalten, wenn die Stelle „Wdh. Versuche P1“ keine Markierung enthält.



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/10_Petri-Netze/Aufgabe_Modellierung.tex`

Übungsaufgabe „Rechnen“ (Petri-Netz)

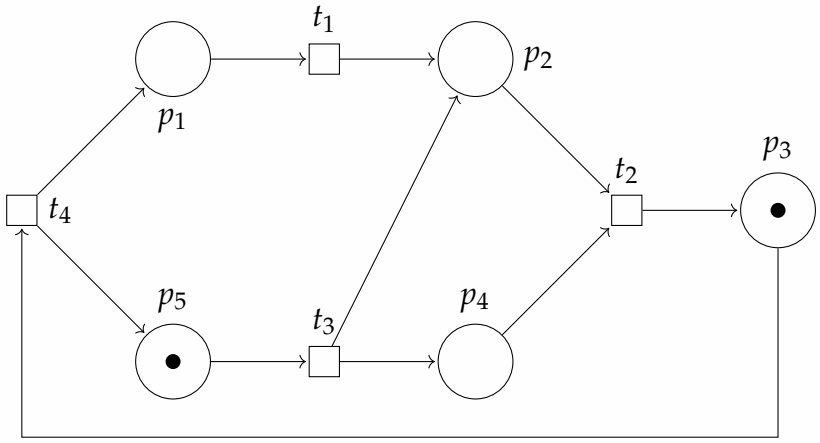
Petri-Netz

Gegeben sei die Darstellungsmatrix A und der Belegungsvektor v eines Petri-Netzes:

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \end{array}, \quad v = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

(a) Skizzieren Sie das zugehörige Petri-Netz.

Lösungsvorschlag



(b) Berechnen Sie mithilfe der Darstellungsmatrix A und zum Belegungsvektor v, die Belegung nach Schaltung von $t_3 \rightarrow t_2 \rightarrow t_4$.

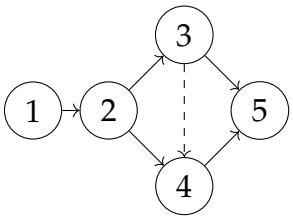
Lösungsvorschlag

$$v_{\text{neu}} = v + A \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/10_Petri-Netze/Aufgabe_Rechnen.tex`

Übungsaufgabe „CPM und Gantt“ (CPM-Netzplantechnik)

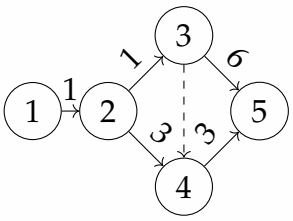
(a) Gegeben ist folgender (unvollständiger) CPM-Netzplan, sowie die frühesten und spätesten Termine und die Pufferzeiten aller Ereignisse:



Ereignis	1	2	3	4	5
frühester Termin	0	1	2	4	8
spätester Termin	0	1	2	5	8
Puffer	0	0	0	1	0

Vervollständigen Sie den CPM-Netzplan, indem Sie mit Hilfe obiger Tabelle die Zeiten der Vorgänge berechnen.

Lösungsvorschlag



Frühester Termin/Zeitpunkt

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:** i: Ereignis i; FZ_i: Frühester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————————————

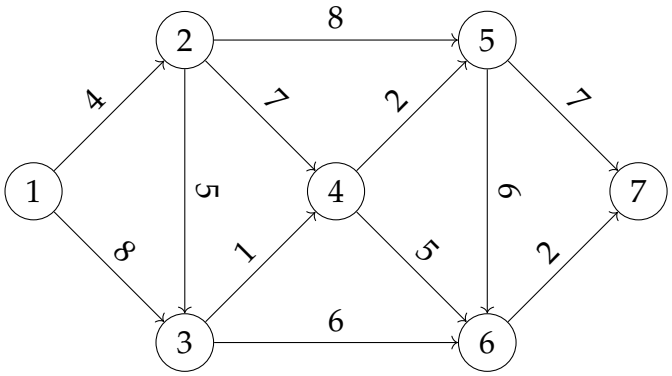
i	Nebenrechnung	FZ_i
1		0
2		1
3		2
4	$\max(4_2, 2_3)$	4
5	$\max(8_3, 7_4)$	8

Spätester Termin/Zeitpunkt

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:** i: Ereignis i; SZ_i: Spätester Zeitpunkt, zu dem Ereignis i eintreten kann. ———————————————

i	Nebenrechnung	SZ_i
5	siehe FZ_5	8
4		5
3	$\min(2_5, 5_4)$	2
2	$\min(1_3, 2_4)$	1
1		0

(b) Bestimmen Sie zum nachfolgenden CPM-Netzplan für jedes Ereignis den *frühesten Termin*, den *spätesten Termin* sowie die *Gesamtpufferzeit*. Geben Sie außerdem den *kritischen Pfad* an.



Lösungsvorschlag

i	1	2	3	4	5	6	7
FZ_i	0	4	9	11	13	19	21
SZ_i	0	4	10	11	13	19	21
GP	0	0	1	0	0	0	0

Frühester Termin/Zeitpunkt

i	Nebenrechnung	FZ_i
1		0
2		4
3	$\max(8, 4_{(\to 2)} + 5) = \max(8, 9)$	9
4	$\max(9_{(\to 3)} + 1, 4_{(\to 2)} + 7) = \max(10, 11)$	11
5	$\max(4_{(\to 2)} + 8, 11_{(\to 4)} + 2) = \max(12, 13)$	13
6	$\max(13_{(\to 5)} + 6, 11_{(\to 4)} + 5, 9_{(\to 3)} + 6) = \max(19, 16, 15)$	19
7	$\max(13_{(\to 5)} + 7, 19_{(\to 6)} + 2) = \max(20, 21)$	21

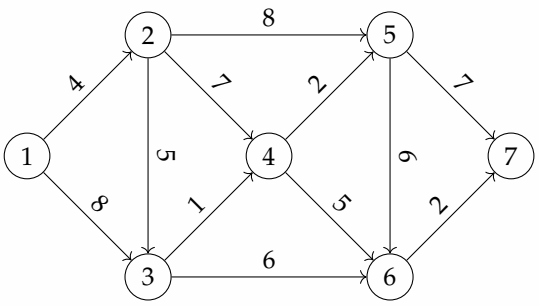
Spätester Termin/Zeitpunkt

i	Nebenrechnung	SZ_i
1	$\min(4_{(\to 2)} - 4, 10_{(\to 3)} - 8) = \min(0, 2)$	0
2	$\min(13_{(\to 5)} - 8, 11_{(\to 4)} - 7, 10_{(\to 3)} - 5) = \min(5, 4, 5)$	4
3	$\min(11_{(\to 4)} - 1, 19_{(\to 6)} - 6) = \min(10, 13)$	10
4	$\min(13_{(\to 5)} - 2, 19_{(\to 6)} - 5) = \min(11, 14)$	11
5	$\min(21_{(\to 7)} - 7, 19_{(\to 6)} - 6) = \min(14, 13)$	13
6	$21_{(\to 7)} - 2$	19
7	siehe FZ_7	21

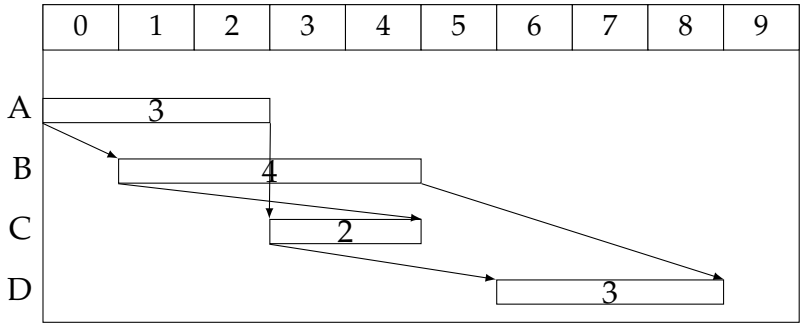
Kritischer Pfad

$1 \to 2 \to 4 \to 5 \to 6 \to 7$

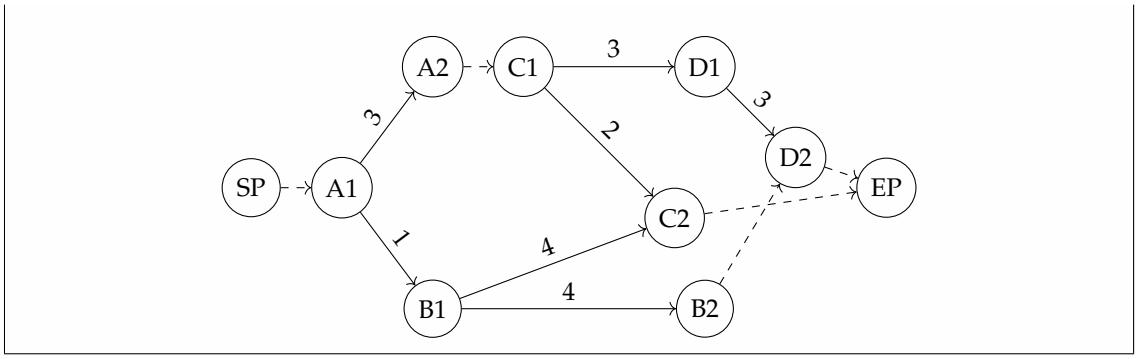
$4_{(1\to 2)} + 7_{(2\to 4)} + 2_{(4\to 5)} + 6_{(5\to 6)} + 2_{(6\to 7)} = 21$



(c) Konvertieren Sie das nachfolgende Gantt-Diagramm in ein CPM-Netzwerk. Als Hilfestellung ist die Anordnung der Ereignisse bereits vorgegeben.

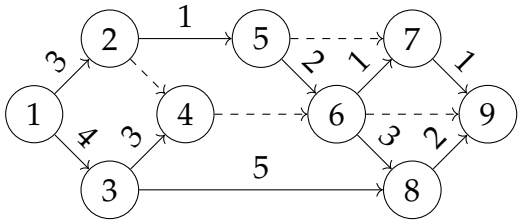


Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/20_CPM-Netzplantechnik/Aufgabe_CPM-Gantt.tex`

Übungsaufgabe „CPM-Netzwerk“ (CPM-Netzplantechnik)



(a) Welche Scheinvorgänge könnten aus dem Netzwerk entfernt werden, ohne dass Informationen verloren gehen?

- ☒ $2 \to 4$
- ☐ $4 \to 6$
- ☒ $5 \to 7$
- ☒ $6 \to 9$

(b) Berechnen Sie für jedes Ereignis den frühesten Termin, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

Lösungsvorschlag

i	Nebenrechnung	FZ_i
1		0
2	$0 + 3_{(1\to2)} = 3$	3
3	$0 + 4_{(1\to3)} = 4$	4
4	$3_{(1\to2)} + 0_{(2\to4)} = 3$ $4_{(1\to3)} + 3_{(3\to4)} = 7$ $\max(3,7)$	7
5	$3_{(1\to2)} + 1_{(2\to5)} = 4$	4
6	$\max(7+0, 4+2)$	7
7	$\max(4+0, 7+1)$	8
8	$\max(4+5, 7+3)$	10
9	$\max(8+1, 7+0, 10+2)$	12

(c) Berechnen Sie für jedes Ereignis auch die spätesten Zeiten, indem Sie für das letzte Ereignis den frühesten Termin als spätesten Termin ansetzen.

Lösungsvorschlag

i	Nebenrechnung	SZ_i
1	$\min(4-3, 4-4)$	0
2	$\min(5-1, 7-0)$	4
3	$\min(10-5, 7-3)$	4
4	$7-0$	7
5	$\min(11-0, 7-2)$	5
6	$\min(12-0, 11-1, 10-3)$	7
7	$12-1$	11
8	$12-2$	10
9	siehe FZ_9	12

(d) Geben Sie nun die Pufferzeiten der Ereignisse an.

Lösungsvorschlag

Ereignis	1	2	3	4	5	6	7	8	9
frühester Termin	0	3	4	7	4	7	8	10	12
spätester Termin	0	4	4	7	5	7	11	10	12
Puffer	0	1	0	0	1	0	3	0	0

(e) Wie verläuft der kritische Pfad durch das Netzwerk?

Lösungsvorschlag

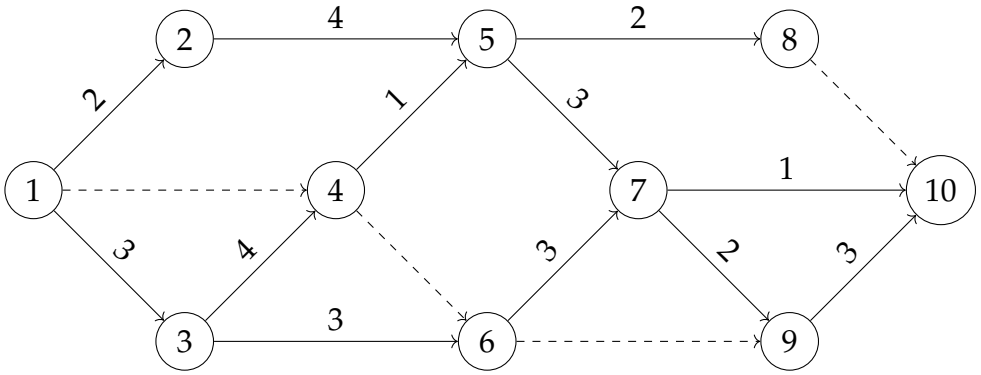
1 3 4 6 8 9

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/20_CPM-Netzplantechnik/Aufgabe_CPM-Netzwerk.tex`

Übungsaufgabe „CPM mit Scheinvorgang“ (CPM-Netzplantechnik)

CPM-Netzplantechnik

Gegeben ist das nachfolgende CPM-Netz. Gestrichelte Linien zwischen Ereignissen stellen Scheinvorgänge mit einer Dauer von 0 dar.



(a) Begründen Sie, welche Scheinvorgänge aus dem Netzplan ohne Informationsverlust gestrichen werden könnten.

Lösungsvorschlag

Die Scheinvorgänge zwischen den Ereignissen 1 und 4 bzw. zwischen 6 und 9 können jeweils gestrichen werden, da Ereignis 4 schon auf 1 wartet (über 3) und 9 wartet auf 6 (über 7).

(b) Berechnen Sie für jedes Ereignis den *frühesten Termin*, den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

Lösungsvorschlag

i	Nebenrechnung	FZ_i
1		0
2		2
3		3
4		7
5	$\max(3_{(\to 3)} + 3, 7_{(\to 4)} + 1)$	8
6	$\max(3_{(\to 3)} + 3, 7_{(\to 4)} + 0)$	7
7	$\max(8_{(\to 5)} + 3, 7_{(\to 6)} + 3)$	11
8	$8_{(\to 5)} + 2$	10
9	$\max(7_{(\to 6)} + 0, 11_{(\to 7)} + 2)$	13
10	$\max(10_{(\to 7)} + 1, 8_{(\to 8)} + 0, 13_{(\to 9)} + 3)$	16

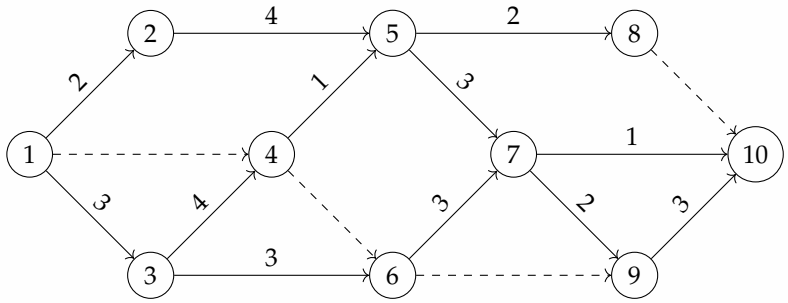
i	Nebenrechnung	SZ_i
1		0
2	$\min(8_{(\to 5)} - 4$	4
3	$\min(8_{(\to 6)} - 3, 7_{(\to 4)} - 4)$	3
4	$\min(8_{(\to 5)} - 1, 8_{(\to 6)} - 0)$	7
5	$\min(16_{(\to 8)} - 2, 11_{(\to 7)} - 3)$	8
6	$\min(11_{(\to 7)} - 3, 13_{(\to 9)} - 0)$	8
7	$\min(16_{(\to 10)} - 1, 13_{(\to 9)} - 2)$	11
8	$16_{(\to 10)} - 0$	16
9	$16_{(\to 10)} - 3$	13
10	siehe FZ_10	16

i	1	2	3	4	5	6	7	8	9	10
FZ_i	0	2	3	7	8	7	11	10	13	16
SZ_i	0	4	3	7	8	8	11	16	13	16
GP	0	2	0	0	0	1	0	6	0	0

(c) Bestimmen Sie den kritischen Pfad.

Lösungsvorschlag

$1 \to 3 \to 4 \to 5 \to 7 \to 9 \to 10$



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/20_CPM-Netzplantechnik/Aufgabe_CPM-Scheinvorgang.tex

Übungsaufgabe „Anordnungsbeziehungen“ (Gantt-Diagramm)

Gantt-Diagramm

In Gantt-Diagrammen unterscheidet man vier Anordnungsbeziehungen: Normalfolge (EA), Anfangsfolge (AA), Endfolge (EE) und Sprungfolge (AE). Ordnen Sie folgende Beispielen den Anordnungsbeziehungen (EA, AA, EE, AE) zu.

(a) Tests durchführen und Dokumentation erstellen

Lösungsvorschlag

EE (Das Testen muss abgeschlossen sein bevor die Erstellung der Dokumentation beendet werden kann.)

(b) Systementwurf und Implementierung

Lösungsvorschlag

AA (Der Beginn des Systementwurfs ist Voraussetzung für den Beginn der Implementierung.)

(c) neue Anwendung in Betrieb nehmen und alte Anwendung abschalten

Lösungsvorschlag

AE (Die alte Anwendung kann erst abgeschaltet werden, wenn die neue Anwendung in Betrieb genommen wurde.)

(d) Implementierung und Dokumentation

Lösungsvorschlag

EE (Die Implementierung muss abgeschlossen sein bevor die Dokumentation beendet werden kann.)

(e) Abitur schreiben und Studieren

Lösungsvorschlag

EA (Das Abitur muss abgeschlossen sein bevor mit dem Studium begonnen werden kann.)

(f) Führerschein machen und selbstständiges Autofahren

Lösungsvorschlag

EA (Der Führerschein muss bestanden sein bevor mit dem selbstständigen Autofahren begonnen werden kann.)

(g) Studieren und Buch aus Uni-Bibliothek ausleihen

Lösungsvorschlag

AA (Der Beginn des Studiums ist Voraussetzung für das Ausleihen eines Buches aus der Uni-Bibliothek.)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/03_Projektplanung/30_Gantt/Aufgabe_Anordnungsbeziehungen.tex

Softwarearchitektur

Examensaufgabe „Softwarearchitektur und Agilität“ (66116-2019-H.T2-TA1-A3)

Die Komponentenarchitektur eines Softwaresystems beschreibt die unterschiedlichen Softwarebausteine, deren Schnittstellen und die Abhängigkeiten von Softwarekomponenten wie beispielsweise Klassen. Wir unterscheiden zwischen der Soll- und der Ist-Architektur eines Systems.

(a) Nennen und definieren Sie drei Qualitätsattribute von Software, die durch die Architektur beeinflusst werden und charakterisieren Sie jeweils eine „schlechte“ Architektur, die diese Qualitätsattribute negativ beeinflusst.

Lösungsvorschlag

Skalierbarkeit Definition: Ob die Software auch für große Zugriffslasten konzipiert wurde. Charakterisierung einer schlechten Architektur: Eine Anwendung läuft nur auf einem Server.

Modifizierbarkeit Definition: Ob die Software leicht verändert, erweitert werden kann. Charakterisierung einer schlechten Architektur: Monolithische Architektur, dass kein Laden von Modulen zulässt.

Verfügbarkeit Definition: Ob die Software ausfallsicher ist, im Laufenden Betrieb gewartet werden kann. Charakterisierung einer schlechten Architektur: Ein-Server-Architektur, die mehrmal neugestartet werden muss, um ein Update einzuspielen.

(b) Erläutern Sie, was Information Hiding ist. Wie hängt Information Hiding mit Softwarearchitektur zusammen”? Wie wird Information Hiding auf Klassenebene in Java implementiert? Gibt es Situationen, in denen Information Hiding auch negative Effekte haben kann?

Lösungsvorschlag

Das Prinzip der Trennung von Zuständigkeiten (engl. separation of concerns) sorgt dafür, dass jede Komponente einer Architektur nur für eine einzige Aufgabe zuständig ist. Das Innenleben von Komponenten wird durch Schnittstellen verkapselt, was auf das Prinzip des Verbergens von Informationen (engl. information hiding) zurückgeht.

Java: Durch die Sichbarkeits-Schlüsselwörter: private, protected

Zusätzlicher Overhead durch Schnittestellen API zwischen den Modulen.

(c) Erklären Sie, was Refactoring ist.

Lösungsvorschlag

> Verbesserungen des Code durch bessere Lesbarkeit, Wartbarkeit, Performanz, Sicherheit. Keine neuen Funktionen werden programmiert.

(d) Skizzieren Sie die Kernideen von Scrum inkl. der wesentlichen Prozessschritte, Artefakte und Rollen. Beschreiben Sie dann die Rolle von Ist- und Soll-Architektur in agilen Entwicklungskontexten wie Scrum.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „AJAX“ (66116-2021-F.T1-TA1-A10)

Client-Server-Modell

(a) Was bedeutet die Abkürzung AJAX?

Lösungsvorschlag

Asynchronous JavaScript and XML

(b) Erklären Sie in max. drei Sätzen die grundlegende Funktion von AJAX.

Lösungsvorschlag

Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden. [a]

[a] `https://de.wikipedia.org/wiki/Ajax_(Programmierung)`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-10.tex`

Examensaufgabe „HTTP“ (66116-2021-F.T1-TA1-A11)

Client-Server-Modell

(a) Was ist das Hypertext Transfer Protocol (HTTP) und wozu dient es?

Lösungsvorschlag

Das Hypertext Transfer Protocol ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es wird hauptsächlich eingesetzt, um Webseiten (Hypertext-Dokumente) aus dem World Wide Web (WWW) in einen Webbrowser zu laden. Es ist jedoch nicht prinzipiell darauf beschränkt und auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet. [a]

[a] `https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol`

(b) Betrachten Sie die folgende Zeile Text. Um welche Art von Text handelt es sich?

`https://developer.mozilla.org/en-US/search?q=client+servertoverview`

Lösungsvorschlag

Es handelt sich um eine HTTP-URL (Uniform Resource Locator). Die URL lokalisiert eine Ressource, beispielsweise eine Webseite, über die zu verwendende Zugriffsmethode (zum Beispiel das verwendete Netzwerkprotokoll wie HTTP oder FTP) und den Ort (engl. location) der Ressource in Computernetzwerken. [a]

[a] `https://de.wikipedia.org/wiki/Uniform_Resource_Locator`

(c) Was sind die vier wesentlichen Bestandteile des Texts aus der vorigen Teilaufgabe?

Lösungsvorschlag

Schema `https://`

Host `developer.mozilla.org`

Pfad `/en-US/search`

Query `?q=client+servertoverview`

a

[a] `https://de.wikipedia.org/wiki/Uniform_Resource_Locator`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-11.tex`

Examensaufgabe „Richtig-Falsch“ (66116-2021-F.T1-TA1-A12)

Peer-to-Peer-Architektur

Es gibt Softwaresysteme, welche auf peer-to-peer (P2P) Kommunikation basieren und eine entsprechende Architektur aufweisen.

(a) Bewerten Sie die folgenden Aussagen als entweder richtig oder falsch.

(i) Mithilfe des Befehls “lookup” können Peers sich gegenseitig identifizieren.

Lösungsvorschlag

richtig

(ii) In einem P2P-System, wie auch bei Client-Server, sind alle Netzwerkteilnehmer gleichberechtigt.

Lösungsvorschlag

falsch. Im Client-Servermodell sind nicht alle Netzwerkteilnehmer gleichberechtig. Der Server hat mehr Privilegien wie der Client.

(iii) Alle P2P-Systeme funktionieren grundsätzlich ohne einen zentralen Verwaltungs-Peer.

Lösungsvorschlag

falsch. Es gibt zentralisierte P2P-Systeme (Beispiel: Napster), welche einen zentralen Server zur Verwaltung benötigen, um zu funktionieren. [a]

[a] `https://de.wikipedia.org/wiki/Peer-to-Peer`

(iv) P2P kann auch für eine Rechner-Rechner-Verbindung stehen.

Lösungsvorschlag

richtig

(v) Es gibt strukturierte und unstrukturierte P2P-Systeme. In unstrukturierten P2P-Systemen wird zum Auffinden von Peers eine verteilte Hashtabelle verwendet (DHT).

Lösungsvorschlag

richtig

(vi) In einem P2P-System sind theoretisch alle Peers gleichberechtigt, praktisch gibt es jedoch leistungsabhängige Gruppierungen.

Lösungsvorschlag

richtig

(vii) Ein Peer kann sowohl ein Client wie auch ein Server für einen anderen Peer sein.

Lösungsvorschlag

richtig

(b) Wählen Sie zwei falsche Aussagen aus der vorherigen Tabelle aus und berichtigen Sie diese in jeweils einem Satz.

Lösungsvorschlag

Sie oben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-12.tex`

Examensaufgabe „Client-Server-Modell“ (66116-2021-F.T1-TA1-A8)

Client-Server-Modell

Das Client-Server-Modell ist ein Architekturmuster. Nennen Sie zwei Vorteile einer nach diesem Muster gestalteten Architektur.

Lösungsvorschlag

(a) Einfache Integration weiterer Clients

(b) Prinzipiell uneingeschränkte Anzahl der Clients [a]

(c) Es muss nur ein Server gewartet werden. Dies gilt z. B. für Updates, die einmalig und zentral auf dem Server durchgeführt werden und danach für alle Clients verfügbar sind. [b]

[a] https://www.karteikarte.com/card/164928/vorteile-und-nachteile-des-client-server-modells

[b] https://www.eoda.de/wissen/blog/client-server-architekturen-performance-und-agilitaet-fuer-data-s

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-8.tex

Examensaufgabe „Client-Server-Technologien“ (66116-2021-F.T1-TA1-A9)

Client-Server-Modell

Betrachten Sie die folgende Liste von Technologien:

- Nodejs
- PHP
- CSS
- AJAX
- Python
- Java

Welche dieser Technologien laufen in einem Client-Server-System üblicherweise auf der Seite des Klienten und welche auf der Seite des Servers? Nehmen Sie hierzu an, dass der Client ein Browser ist.

Übertragen Sie die im Folgenden gegebene Tabelle in Ihren Bearbeitungsbogen und ordnen Sie die aufgelisteten Technologien anhand der Buchstaben in die Tabelle ein. Fortsetzung nächste Seite!

Hinweis: Mehrfachzuordnungen sind möglich.

Lösungsvorschlag

Client-seitige Technologien	Server-seitige Technologien
CSS	Nodejs
AJAX	PHP
Python	Python
Java	Java

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-9.tex`

Testen

Examensaufgabe „Methode function: Formale Verifikation - Induktionsbeweis“ (46115-2015-H.T2-A4)

Gegeben sei die folgende Methode `function`:

```
double function(int n) {
  if (n == 1)
    return 0.5 * n;
  else
    return 1.0 / (n * (n + 1)) + function(n - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/Induktion.java`

Beweisen Sie folgenden Zusammenhang mittels vollständiger Induktion:

$$\forall n \geq 1\colon \text{function}(n) = f(n) \text{ mit } f(n) := 1 - \frac{1}{n+1}$$

Hinweis: Eventuelle Rechenungenauigkeiten, wie z. B. in Java, bei der Behandlung von Fließkommazahlen (z. B. `double`) sollen beim Beweis nicht berücksichtigt werden - Sie dürfen also annehmen, Fließkommazahlen würden mathematische Genauigkeit aufweisen.

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$f(1) := 1 - \frac{1}{1+1} = 1 - \frac{1}{2} = \frac{1}{2}$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$f(n) := 1 - \frac{1}{n+1}$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss. ——

zu zeigen:

$f(n+1) := 1 - \frac{1}{(n+1)+1} = f(n)$

Vorarbeiten (Java in Mathe umwandeln):

$\text{function}(n) = \frac{1}{n \cdot (n+1)} + f(n-1)$

$$
\begin{aligned}
f(n+1) &= \frac{1}{(n+1)\cdot((n+1)+1)} + f((n+1)-1) && n+1 \text{ eingesetzt} \\
&= \frac{1}{(n+1)\cdot(n+2)} + f(\mathrm{n}) && \text{vereinfacht} \\
&= \frac{1}{(n+1)\cdot(n+2)} + 1 - \frac{1}{n+1} && \text{für } f(n) \text{ Formel eingesetzt} \\
&= 1 + \frac{1}{(n+1)\cdot(n+2)} - \frac{1}{n+1} && \text{1. Bruch an 2. Stelle geschrieben} \\
&= 1 + \frac{1}{(n+1)\cdot(n+2)} - \frac{1\cdot(n+2)}{(n+1)\cdot(n+2)} && \text{2. Bruch mit } (n+2) \text{ erweitert} \\
&= 1 + \frac{1-(n+2)}{(n+1)\cdot(n+2)} && \text{die 2 Brüche subtrahiert} \\
&= 1 + \frac{1-n-2}{(n+1)\cdot(n+2)} && -+2=-2 \\
&= 1 + \frac{-1-n}{(n+1)\cdot(n+2)} && 1-2=-1 \\
&= 1 + \frac{-1\cdot(1+n)}{(n+1)\cdot(n+2)} && (n+1) \text{ ausgeklammert} \\
&= 1 + \left(-1\cdot\frac{(1+n)}{(n+1)\cdot(n+2)}\right) && \text{minus vor den Bruch bringen} \\
&= 1 - \frac{(1+n)}{(n+1)\cdot(n+2)} && \text{plus minus ist minus} \\
&= 1 - \frac{1}{n+2} && (n+1) \text{ gekürzt} \\
&= 1 - \frac{1}{(n+1)+1} && \text{Umformen zur Verdeutlichung}
\end{aligned}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2015/09/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Hanoi“ (46116-2014-F.T2-TA1-A1)

Gegeben sei folgende Methode zur Berechnung der Anzahl der notwendigen Züge beim Spiel *„Die Türme von Hanoi“*:

```
int hanoi(int nr, char from, char to) {
  char free = (char) ('A' + 'B' + 'C' - from - to);
  if (nr > 0) {
    int moves = 1;
    moves += hanoi(nr - 1, from, free);
    System.out.println("Move piece nr. " + nr + " from " + from + " to " + to);
    moves += hanoi(nr - 1, free, to);
    return moves;
  } else {
    return 0;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2014/fruehjahr/Hanoi.java`

(a) Beweisen Sie formal mittels vollständiger Induktion, dass zum Umlegen von k Scheiben (z. B. vom Turm A zum Turm C) insgesamt $2^k - 1$ Schritte notwendig sind, also dass für $k \geq 0$ folgender Zusammenhang gilt:

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Lösungsvorschlag

Zu zeigen:

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$k = 0$

$$\text{hanoi}(0, \text{'A'}, \text{'C'}) = 0$$

$$2^0 - 1 = 1 - 1 = 0$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Terminierungsfunktion

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss.

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 1 + \text{hanoi}(k-1, \text{'A'}, \text{'B'}) + \text{hanoi}(k-1, \text{'B'}, \text{'C'})$$

$k \to k+1$

$$\begin{aligned}
\text{hanoi}(k+1, \text{'A'}, \text{'C'}) &= 1 + \text{hanoi}((k+1)-1, \text{'A'}, \text{'B'}) + \\
&\quad \text{hanoi}((k+1)-1, \text{'B'}, \text{'C'}) \\
&= 1 + \text{hanoi}(k, \text{'A'}, \text{'B'}) + \\
&\quad \text{hanoi}(k, \text{'B'}, \text{'C'}) && k+1-1 = k \\
&= 1 + 2^k - 1 + 2^k - 1 && \text{Formeln eingesetzt} \\
&= 2^k + 2^k - 1 && 1-1-1 = -1 \\
&= 2 \cdot 2^k - 1 && 2^k + 2^k = 2 \cdot 2^k \\
&= 2^{k+1} - 1 && 2 \cdot 2^k = 2^{k+1}
\end{aligned}$$

(b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie kurz Ihre Wahl!

Lösungsvorschlag

Betrachte die Argumentenfolge $k, k-1, k-2, \ldots, 0$. Die Terminierungsfunktion ist offenbar $T(k) = k$. $T(k)$ ist bei jedem Rekursionsschritt auf der Folge der Argumente streng monoton fallend. Bei der impliziten Annahme k ist ganzzahlig und $k \geq 0$ ist $T(k)$ nach unten durch 0 beschränkt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2014/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

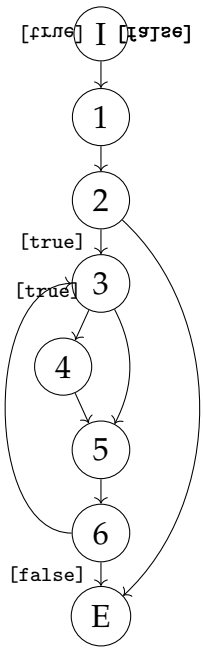
Examensaufgabe „Methode „isPalindrom()““ (46116-2015-H.T2-TA1-A2)

Kontrollflussgraph

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:
Abkürzungen: I = Import, E = Export

```java
boolean isPalindrom(String s) {
  boolean yesItIs = true;
  if (s != null && s.length() > 1) {
    do {
      if (s.charAt(0) != s.charAt(s.length() -
↪ 1)) {
        yesItIs = false;
      }
      s = s.substring(1, s.length() - 1);

    } while (yesItIs && s.length() > 1);
  }
  return yesItIs;
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java`

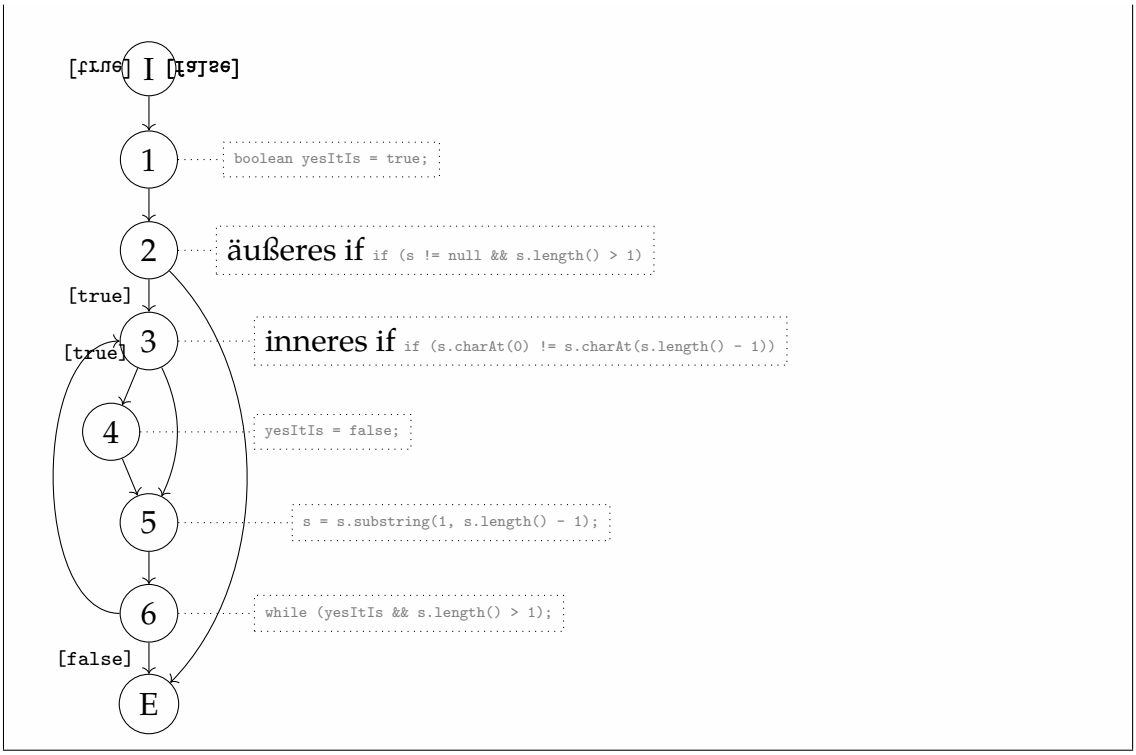


(a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

Lösungsvorschlag

Bemerkung: In der Aufgabenstellung steht *„Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, [...] “*. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!

C1-Test Zweigüberdeckung (Branch Coverage)
C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)



(i) Verzweigungsüberdeckung (Branch-Coverage, C_1)

Lösungsvorschlag

Pfad 1 (p1) (I) - (1) - (2) - (E)
(äußere `if`-Bedingung `false`)

Pfad 2 (p2) (I) - (1) - (2) - (3) - (5) - (6) - (3) - (4) - (5) - (6) - (E)
(äußere `if`-Bedingung `true`, innere `if`-Bedingung `false`, Wiederholung, innere `if`-Bedingung `true`, keine Wiederholung)

(ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

Lösungsvorschlag

ohne Ausführung der Wiederholung (äußere Pfade): p1 (siehe oben)
(I) - (1) - (2) - (E)

Boundary-Test: (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

interior-Test: (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)
innere `if`-Bedingung `true`: (3) - (4) - (5) - (6)
innere `if`-Bedingung `false`: (3) - (5) - (6)

p5 (I) - (1) - (2) - (3) - (4) - (5) - (6) - (3) - (4) - (5) - (6) - (E)
(innere `if`-Bedingung `true`, innere `if`-Bedingung `true`)

p2 (siehe oben) (I) - (1) - (2) - (3) - (5) - (6) - (3) - (4) - (5) - (6) - (E)
(innere `if`-Bedingung `false`, innere `if`-Bedingung `true`)

p6 (I) - (1) - (2) - (3) - (4) - (5) - (6) - (3) - (4) - (5) - (6) - (E)

(innere `if`-Bedingung `true`, innere `if`-Bedingung `false`)

p7 (I) - (1) - (2) - (3) - (5) - (6) - (3) - (5) - (6) - (E)
(innere `if`-Bedingung `false`, innere `if`-Bedingung `false`)

Zyklomatische Komplexität nach Mc-Cabe
C2a Vollständige Pfadüberdeckung (Full Path Coverage)

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

(b) Welche der vorangehend ermittelten Pfade für die $C_{\infty,2}$-Überdeckung sind mittels Testfällen tatsächlich überdeckbar (*„feasible“*)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

p1 `s = "a";`

p2 `s = "abaa";`

p3 `s = "ab";`

p4 `s = "aa";`

p5 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p6 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p7 `s = "abba";`

(c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

$M = b + p = 3 + 1 = 4$

(b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)

Alternativ

$M = e - n + 2p = 10 - 8 + 2 = 4$

(e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Lösungsvorschlag

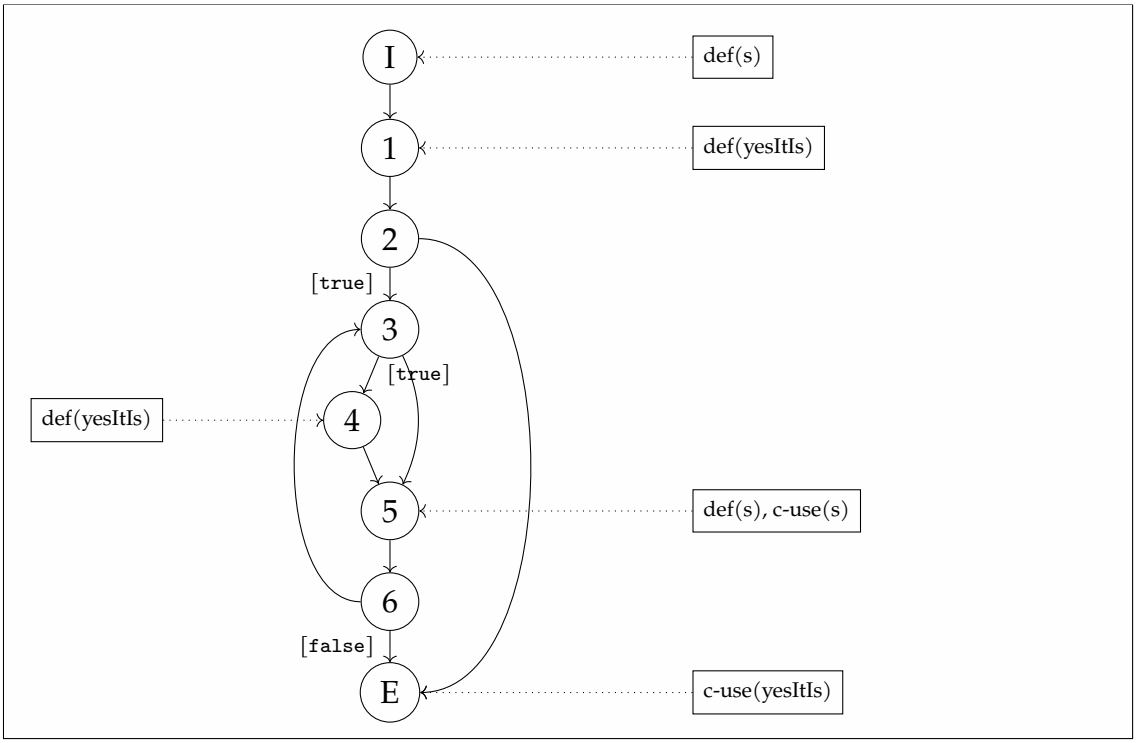
Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wor-

tes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

Datenfluss-annotierter Kontrollflussgraph

(e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (`c-use`) oder prädikativ (`p-use`) sind.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/09/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „ASCII“ (46116-2015-H.T2-TA1-A3)

Sei wp(A, Q) die schwächste Vorbedingung (weakest precondition) eines Programmfragments A bei gegebener Nachbedingung Q so, dass A alle Eingaben, die wp(A,Q) erfüllen, auf gültige Ausgaben abbildet, die Q erfüllen.

Bestimmen Sie schrittweise und formal (mittels Floyd-Hoare-Kalkül) jeweils wp(A, Q) für folgende Code-Fragmente A und Nachbedingungen Q und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.

Die Variablen x, y und z in folgenden Pseudo-Codes seien ganzzahlig (vom Typ int). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Sequenz:

```
x = -2 * (x + 2 * y);
y += 2 * x + y + z;
z -= x - y - z;
```

$Q :\equiv x = y + z$

Lösungsvorschlag

Code umformulieren:

```
x = -2 * (x + 2 * y);
y = y + 2 * x + y + z;
z = z - (x - y - z);
```

$\text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;z=z-(x-y-z);"}, x = y + z)$

z eingesetzen

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (z - (x - y - z)))$

Innere Klammer auflösen

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (-x + y - 2z))$

Klammer auflösen

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = -x + 2y + 2z)$

$-x$ auf beiden Seiten

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -2x + 2y + 2z)$

$\div 2$ auf beiden Seiten

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -x + y + z)$

y einsetzen

$\equiv \quad \text{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = -x + (2y + 2x + z) + z)$

Term vereinfachen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = x + 2y + 2z)$$

x einsetzen

$$\equiv \quad \mathrm{wp}(\texttt{""}, 0 = (-2(x + 2y)) + 2y + 2z)$$

wp weglassen

$$\equiv \quad 0 = (-2(x + 2y)) + 2y + 2z$$

ausmultiplizieren

$$\equiv \quad 0 = (-2x - 4y) + 2y + 2z$$

Klammer auflösen, vereinfachen

$$\equiv \quad 0 = -2x - 2y + 2z$$

$\div 2$ auf beiden Seiten

$$\equiv \quad 0 = -x - y + z$$

x nach links holen mit $+x$ auf beiden Seiten

$$\equiv \quad x = -y + z$$

y ganz nach links schreiben

$$\equiv \quad x = z - y$$

$$x = -2 \cdot (x + 2 \cdot y)$$

(b) Verzweigung:

```
if (x < y) {
  x = y + z;
} else if (y > 0) {
  z = y - 1;
} else {
  x -= y -= z;
}
```

$Q :\equiv x > z$

Lösungsvorschlag

1. Fall: $x < y$

2. Fall: $x \geq y \land y > 0$

3. Fall: $x \geq y \land y \leq 0$

Code umformulieren:

```
if (x < y) {
  x = y + z;
} else if (x >= y && y > 0) {
  z = y - 1;
} else {
  y = y - z;
  x = x - y;
}
```

$\mathrm{wp}(\texttt{"if(x<y){x=y+z;}else if(x>=y\&\&y>0){z=y-1;}else{y=y-z;x=x-y;}"}, x > z)$

$\equiv$ (In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}&\Big((x < y) \wedge \mathrm{wp}(\texttt{"x=y+z;"}, x > z)\Big) \vee \\ &\Big((x \geq y \wedge y > 0) \wedge \mathrm{wp}(\texttt{"z=y-1;"}, x > z)\Big) \vee \\ &\Big((x \geq y \wedge y \leq 0) \wedge \mathrm{wp}(\texttt{"y=y-z;x=x-y;"}, x > z)\Big)\end{aligned}$$

$\equiv$ (Code einsetzen)

$$\begin{aligned}&\Big((x < y) \wedge \mathrm{wp}(\texttt{""}, y + z > z)\Big) \vee \\ &\Big((x \geq y \wedge y > 0) \wedge \mathrm{wp}(\texttt{""}, x > y - 1)\Big) \vee \\ &\Big((x \geq y \wedge y \leq 0) \wedge \mathrm{wp}(\texttt{"y=y-z;"}, x - y > z)\Big)\end{aligned}$$

$\equiv$ (wp-Kalkül-Schreibweise weg lassen, Code weiter einsetzen)

$$\begin{aligned}&\Big((x < y) \wedge y + z > z\Big) \vee \\ &\Big((x \geq y \wedge y > 0) \wedge x > y - 1\Big) \vee \\ &\Big((x \geq y \wedge y \leq 0) \wedge \mathrm{wp}(\texttt{""}, x - (y - z) > z)\Big)\end{aligned}$$

$\equiv$ (Terme vereinfachen, wp-Kalkül-Schreibweise weg lassen)

$$\Big(x < y \wedge y > 0\Big) \vee$$
$$\Big(x \geq y^{a} \wedge y > 0\Big) \vee$$
$$\Big((x \geq y \wedge y \leq 0) \wedge x - (y - z) > z\Big)$$

$\equiv$ (letzten Term vereinfachen)

$$\Big(x < y \wedge y > 0\Big) \vee$$
$$\Big(x \geq y \wedge y > 0\Big) \vee$$
$$\Big((x \geq y \wedge y \leq 0) \wedge x - y > 0\Big)$$

$\equiv$ (ein $\wedge$ eleminieren)

$$\Big(x < y \wedge y > 0\Big) \vee$$
$$\Big(x \geq y \wedge y > 0\Big) \vee$$
$$\Big(y \leq 0 \wedge x > y\Big)$$

[a] $x > y - 1 \wedge x \geq y$ ergibt $x \geq y$

[a] $x > y - 1 \wedge x \geq y$ ergibt $x \geq y$

(c) Mehrfachauswahl:

```
switch (z) {
  case "x":
    y = "x";
  case "y":
    y = --z;
    break;
  default:
    y = 0x39 + "?";
}
```

$Q :\equiv$ `'x'` $= y$

Hinweis zu den ASCII-Codes

- `'x'` $= 120_{(10)}$
- `'y'` $= 121_{(10)}$
- `0x39` $= 57_{(10)}$
- `'?'` $= 63_{(10)}$

Lösungsvorschlag

Mehrfachauswahl in Bedingte Anweisungen umschreiben. Dabei beachten, dass bei fehlendem `break` die Anweisungen im folgenden Fall bzw. ggf. in den folgenden Fällen ausgeführt werden:

```
if (z == "x") {
  y = "x";
  y = z - 1;
} else if (z == "y") {
  y = z - 1;
} else {
  y = 0x39 + "?";
}
```

Da kein `break` im Fall `z == "x"`. `--z` bedeutet, dass die Variable erst um eins verringert und dann zugewiesen wird.

```
if (z == 120) {
  y = 120;
  y = 120 - 1;
} else if (z == 121) {
  y = 121 - 1;
} else {
  y = 57 + 63;
}
```

Vereinfachung / Zusammenfassung:

```
if (z == 120) {
  y = 120;
  y = 119;
} else if (z == 121) {
  y = 120;
} else {
  y = 120;
}
```

$$\text{wp}(\texttt{"if(z==120){y=120;y=119;}else if(z==121){y=120;}else{y=120;}"}, 120 = y)$$

$$\equiv$$ (In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}
&\Big((z = 120) \land \mathrm{wp}(\texttt{"y=120;y=119;"}, 120 = y)\Big) \lor \\
&\Big(((z \neq 120) \land (z = 121)) \land \mathrm{wp}(\texttt{"y=120;"}, 120 = y)\Big) \lor \\
&\Big(((z \neq 120) \land (z \neq 121)) \land \mathrm{wp}(\texttt{"y=120;"}, 120 = y)\Big)
\end{aligned}$$

$\equiv$ (Code einsetzen)

$$\begin{aligned}
&\Big((z = 120) \land \mathrm{wp}(\texttt{"y=120;"}, 120 = 119)\Big) \lor \\
&\Big(((z \neq 120) \land (z = 121)) \land \mathrm{wp}(\texttt{""}, 120 = 120)\Big) \lor \\
&\Big(((z \neq 120) \land (z \neq 121)) \land \mathrm{wp}(\texttt{""}, 120 = 120)\Big)
\end{aligned}$$

$\equiv$ (vereinfachen)

$$\begin{aligned}
&\mathrm{false} \lor \\
&\Big((z = 121) \land \mathrm{true}\Big) \lor \\
&\Big(((z \neq 120) \land (z \neq 121)) \land \mathrm{true}\Big)
\end{aligned}$$

$\equiv \mathrm{false} \lor (z = 121) \lor ((z \neq 120) \land (z \neq 121))$
$\equiv (z = 121) \lor (z \neq 121)$
$\equiv z \neq 121$
Alle Zahlen außer 120 sind möglich bzw. alle Zeichen außer 'x'.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2015/09/Thema-2/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Catalan-Zahl“ (46116-2016-H.T2-TA1-A4)

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte $n \geq 0$ aufgerufen wird.

```
int cn(int n) {
  if (n == 0)
    return 1;
  else
    return (4 * (n - 1) + 2) * cn(n - 1) / (n + 1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, ðdass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

(a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes $n \geq 0$ die n-te Catalan-Zahl C_n berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Exkurs: Fakultät

Für alle natürlichen Zahlen n ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^{n} k$$

als das Produkt der natürlichen Zahlen von 1 bis n definiert. Da das leere Produkt stets 1 ist, gilt

$$0! = 1$$

Die Fakultät lässt sich auch rekursiv definieren:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

Fakultäten für negative oder nicht ganze Zahlen sind nicht definiert. Es gibt aber eine Erweiterung der Fakultät auf solche Argumente [a]

[a] `https://de.wikipedia.org/wiki/Fakultät_(Mathematik)`

Exkurs: Catalan-Zahl

Die Catalan-Zahlen bilden eine Folge natürlicher Zahlen, die in vielen Problemen der Kombinatorik auftritt. Sie sind nach dem belgischen Mathematiker Eugène Charles Catalan benannt.

> Die Folge der Catalan-Zahlen $C_0, C_1, C_2, C_3, \ldots$ beginnt mit $1, 1, 2, 5, 14, 42, 132, \ldots$ [a]
>
> [a] `https://de.wikipedia.org/wiki/Catalan-Zahl`

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

(i) $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$

(ii) $(a+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$\begin{aligned} C_0 &= \frac{(2 \cdot 0)!}{(0+1)! \cdot 0!} \\ &= \frac{0!}{1! \cdot 0!} \\ &= \frac{1}{1 \cdot 1} \\ &= \frac{1}{1} \\ &= 1 \end{aligned}$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss.

Vom Code ausgehend

$$
\begin{aligned}
C_{n+1} &= \frac{(4 \cdot (n+1-1)+2) \cdot \mathrm{cn}(n+1-1)}{n+1+1} && \text{Java nach Mathe} \\
&= \frac{(4n+2) \cdot \mathrm{cn}(n)}{n+2} && \text{addiert, subtrahiert} \\
&= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && \text{für cn(n) Formel eingesetzt} \\
&= \frac{(4n+2) \cdot (2n)! \cdot (n+1)}{(n+2) \cdot (n+1)! \cdot n! \cdot (n+1)} && (n+1) \text{ multipliziert} \\
&= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot (n+1) \cdot n!} && \text{umsortiert} \\
&= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{Hilfsgleichungen verwendet} \\
&= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

Mathematische Herangehensweise

$$
\begin{aligned}
C_{n+1} &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && n+1 \text{ in } C_n \text{ eingesetzt} \\
&= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{addiert} \\
&= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1) \cdot (n+1)! \cdot n!} && \text{Hilfsgleichungen verwendet} \\
&= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && (n+1) \text{ gekürzt} \\
&= \frac{4n+2}{n+2} \cdot C_n && \text{Catalan-Formel ersetzt} \\
&= \frac{4((n+1)-1)+2}{(n+1)+1} \cdot C_{(n+1)-1} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

(b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf `cn(n)` für jedes $n \geq 0$ terminiert.

Lösungsvorschlag

$T(n) = n$. Diese Funktion verringert sich bei jedem Rekursionsschritt um eins. Sie ist monoton fallend und für $T(0) = 0$ definiert. Damit ist sie eine Terminierungsfunktion für `cn(n)` .

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46116/2016/09/Thema-2/Teilaufgabe-1/Aufgabe-4.tex

Examensaufgabe „drei hoch“ (66112-2003-H.T2-A5)

Zeigen Sie mit Hilfe vollständiger Induktion, dass das folgende Programm bzgl. der Vorbedingung $x > 0$ und der Nachbedingung drei_hoch $x = 3^x$ partiell korrekt ist!

```
(define (drei_hoch x)
  (cond ((= x 0) 1)
    (else (* 3 (drei_hoch (- x 1))))
  )
)
```

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist.

$$\text{drei\_hoch}\,1 = 3 \cdot (\text{drei\_hoch}\,0) = 3 \cdot 1 = 3$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$.

für alle $x < x0$ gilt drei_hoch $x = 3x$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

x->x+1

$$\begin{aligned}\text{drei\_hoch}\,(x+1) &= 3 \cdot \text{drei\_hoch}\,(-(x+1)1)) \\ &= 3 \cdot (\text{drei\_hoch}\,x) \\ &= 3 \cdot 3^x \\ &= 3^{x+1}\end{aligned}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66112/2003/09/Thema-2/Aufgabe-5.tex

Examensaufgabe „Methode „sumOfSquares()““ (66115-2017-F.T1-A4)

Vollständige Induktion

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei folgende rekursive Methode für $n \geq 0$:

```
long sumOfSquares (long n) {
  if (n == 0)
    return 0;
  else
    return n * n + sumOfSquares(n - 1);
}
```

(a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \in \mathbb{N} : \texttt{sumOfSquares(n)} = \frac{n(n+1)(2n+1)}{6}$$

Lösungsvorschlag

Sei $f(n) : \frac{n(n+1)(2n+1)}{6}$

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

Für $n = 0$ gilt:

$\texttt{sumOfSquares(0)} \overset{\texttt{if}}{=} 0 = f(0)$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

Für ein festes $n \in \mathbb{N}$ gelte:

$\texttt{sumOfSquares(n)} = f(n)$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

$n \to n + 1$

$$
\begin{aligned}
f(n+1) &= \texttt{sumOfSquares(n+1)} && \text{Java-Methode eingesetzt}\\
&\overset{\texttt{else}}{=} \texttt{(n+1)*(n+1) + sumOfSquares(n)} && \text{Java-Code der else-Verzweigung verwendet}\\
&\overset{\text{I.H.}}{=} (n+1)(n+1)+f(n) && \text{mathematisch notiert}\\
&= (n+1)(n+1)+\frac{n(n+1)(2n+1)}{6} && \text{Formel eingesetzt}\\
&= (n+1)^2+\frac{n(n+1)(2n+1)}{6} && \text{potenziert}\\
&= \frac{6(n+1)^2}{6}+\frac{n(n+1)(2n+1)}{6} && (n+1)^2 \text{ in Bruch umgewandelt}\\
&= \frac{6(n+1)^2+n(n+1)(2n+1)}{6} && \text{Addition gleichnamiger Brüche}\\
&= \frac{(n+1)6(n+1)+(n+1)n(2n+1)}{6} && n+1 \text{ ausklammern vorbereitet}\\
&= \frac{(n+1)(6(n+1)+n(2n+1))}{6} && n+1 \text{ ausgeklammert}\\
&= \frac{(n+1)(6n+6+2n^2+n))}{6} && \text{Klammern ausmultipliziert / aufgelöst}\\
&= \frac{(n+1)(2n^2+7n+6)}{6} && \text{umsortiert, addiert } 6n+n=7n\\
&= \frac{(n+1)(2n^2+3n+4n+6)}{6} && \text{Ausklammern vorbereitet}\\
&= \frac{(n+1)(n+2)(2n+3)}{6} && (n+2) \text{ ausgeklammert}\\
&= \frac{(n+1)((n+1)+1)(2(n+1)+1))}{6} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

[a]

[a] https://mathcs.org/analysis/reals/infinity/answers/sm_sq_cb.html

(b) Beweisen Sie die Terminierung von `sumOfSquares(n)` für alle $n \geq 0$.

Lösungsvorschlag

Sei $T(n) = n$. Die Funktion $T(n)$ ist offenbar ganzzahlig. In jedem Rekursionsschritt wird n um eins verringert, somit ist $T(n)$ streng monoton fallend. Durch die Abbruchbedingung `n==0` ist $T(n)$ insbesondere nach unten beschränkt. Somit ist T eine gültige Terminierungsfunktion.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/03/Thema-1/Aufgabe-4.tex

Examensaufgabe „Methode „specialSums()““ (66116-2014-H.T2-TA2-A3)

Kontrollflussorientieres Testen
Kontrollflussgraph

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl `until` die Summe aller Zahlen bildet, die kleiner als `until` und Vielfache von 4 oder 6 sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$\texttt{until} > 0 \Rightarrow \texttt{specialSums(until)} = \sum\{y \mid 0 < y < \texttt{until} \wedge (y\%4 = 0 \vee y\%6 = 0)\}$
$\texttt{until} \leq 0 \Rightarrow \texttt{specialSums(until)} = 0$

wobei % den Modulo-Operator bezeichnet.

```java
public static long specialSums(int until) {
  long sum = 0; // 0
  if (until > 0) { // 1
    for (int i = 1; i <= until; i++) { // 2 // 5
      if (i % 4 == 0 || i % 6 == 0) { // 3
        sum += i; // 4
      }
    }
  }
  return sum; // 6
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2014/herbst/SpecialSum.java`

Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife. Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

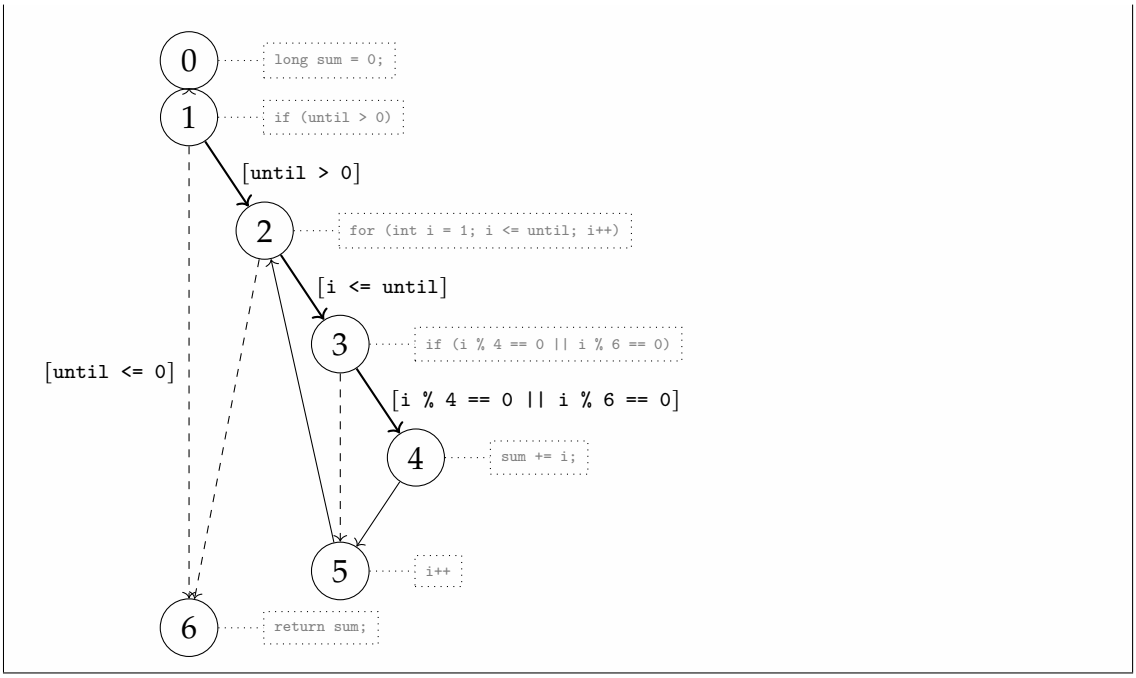
$$i \leq \texttt{until} \text{ in } i < \texttt{until}$$

geändert würde.

(a) Zeichnen Sie das zum Programm gehörige Ablaufdiagramm.

Lösungsvorschlag

C0-Test Anweisungsüberdeckung (Statement Coverage)
C1-Test Zweigüberdeckung (Branch Coverage)



(b) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Der Fehler fällt nur dann auf, wenn `until` durch 4 oder 6 ohne Rest teilbar ist. `until = 0%4` oder `until = 0%6`. Wähle daher den Testfall $\{(1,0)\}$. Alternativ kann für die Eingabe auch 2, 3, 5, 7, 9, 10, 11, 13, ... gewählt werden.

(c) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Betrachte den Testfall $\{(0,0),(5,4)\}$.

erste if-Bedingung Das erste Tupel mit until $= 0$ stellt sicher, dass die erste `if`-Bedingung `false` wird.

Bedingung der for-Schleife Für die zweite Eingabe until $= 5$ werden für i die Werte $1,2,3,4,5,6$ angenommen. Wobei für $i = 6$ die Bedingung der for-Schleife `false` ist.

Innere if-Bedingung Für $i = 1,2,3,5$ wird die innere if-Bedingung jeweils `false`, für $i = 4$ wird sie `true`.

(d) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungs- überdeckung und Zweigüberdeckung ihres Testfalls.

Lösungsvorschlag

Anweisungsüberdeckung Wähle $\{(4,0)\}$. Durch die fehlerhafte Bedingung in der for-Schleife wird der Wert $i = 4$ akzeptiert. Da alle Anweisungen ausgeführt werden, wird eine Anweisungsüberdeckung mit 100%

erreicht.

Verzweigungsüberdeckung Da die erste Verzweigung nur zur Hälfte überdeckt wird und die anderen beiden vollständig, gilt für die Verzweigungsüberdeckung:
$\frac{1+2+2}{2+2+2} = \frac{5}{6}$

(e) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an, der nicht erreichbar ist. Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar (I, O) die Eingabe I und die zu dieser erwartete Ausgabe O darstellt.

Lösungsvorschlag

Ein gültiger Pfad im Kontrollflussgraphen wäre ⓪ - ① - ② - ③ - ④ - ⑤ - ② - ⑥. Der Übergang von ③ auf ④ ist hier aber nicht möglich, da beim ersten Durchlaufen der for-Schleife (②) i immer 1 ist und 1 weder durch 4 noch durch 6 teilbar ist. Somit kann die Bedingung des inneren ifs (③) beim ersten Durchlauf nie *wahr* sein, womit immer der Übergang ③ - ⑤ zu Beginn genommen werden muss. Alle Pfade, die zu Beginn ③ - ④ enthalten, sind somit nicht überdeckbar.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2014/09/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Methode „doubleFac()“: wp-Kalkül und Schleifeninvariante“ (66116-2015-H.T2-TA2-A3)

wp-Kalkül
Invariante

Gegeben Sei folgendes Programm:

```
long doubleFac (long n) {
  /* P */ long df = 1;
  for (long x = n; x > 1; x -= 2) {
    df *= x;
  } /* Q */
  return df;
}
```

sowie die Vorbedingung $P \equiv n \geq 0$ und Nachbedingung $Q \equiv (\text{df} = n!!)$ wobei gilt

$$n!! := \begin{cases} 2^k \cdot k! & n \text{ gerade}, k := \frac{n}{2} \\ \frac{(2k)!}{2^k \cdot k!} & n \text{ ungerade}, k := \frac{n+1}{2} \end{cases}$$

Exkurs: Fakultät
Die Fakultät ist eine Funktion, die einer natürlichen Zahl das Produkt aller natürlichen Zahlen (ohne Null) kleiner und gleich dieser Zahl zuordnet. Für alle natürlichen Zahlen n ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^{n} k$$

Exkurs: Doppelfakultät
Die seltener verwendete „Doppelfakultät“ oder „doppelte Fakultät“ ist für gerade n das Produkt aller geraden Zahlen kleiner gleich n. Für ungerade n ist es das Produkt aller ungeraden Zahlen kleiner gleich n. Sie ist definiert als:

$$n!! = \begin{cases} n \cdot (n-2) \cdot (n-4) \cdots 2 & \text{für } n \text{ gerade und } n > 0, \\ n \cdot (n-2) \cdot (n-4) \cdots 1 & \text{für } n \text{ ungerade und } n > 0, \\ 1 & \text{für } n \in \{-1, 0\} \end{cases}$$

Häufig werden anstelle der Doppelfakultät Ausdrücke mit der gewöhnlichen Fakultät verwendet. Es gilt $(2k)!! = 2^k k!$ und $(2k-1)!! = \frac{(2k)!}{2^k k!}$

Zur Vereinfachung nehmen Sie im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels wp-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

(i) $\text{df} = n!! - x!! \wedge x \geq 1$
(ii) $\text{df} = (n-x)!! \wedge x \geq 1$
(iii) $\text{df} \cdot x!! = n!! \wedge x \geq 0$
(iv) $(\text{df} + x)!! = n!! \wedge x \geq 0$

Lösungsvorschlag

Zunächst wird der Code in einen äquivalenten Code mit while-Schleife umgewandelt:

```
long doubleFac (long n) {
  /* P */ long df = 1;
  long x = n ;
  while (x > 1) {
    df = df * x;
    x = x - 2;
  } /* Q */
  return df;
}
```

(i) $\mathrm{df} = n!! - x!! \wedge x \geq 1$

(ii) $\mathrm{df} = (n - x)!! \wedge x \geq 1$

Die ersten beiden Bedingungen sind unmöglich, da z. B. für $n = 2$ nach der Schleife $x = 0$ gilt und daher $x \geq 1$ verletzt wäre.

(iii) $\mathrm{df} \cdot x!! = n!! \wedge x \geq 0$

Nach dem Ausschlussprinzip ist es daher die dritte Bedingung: $I \equiv (\mathrm{df} + x)!! = n!! \wedge x \geq 0$.

(iv) $(\mathrm{df} + x)!! = n!! \wedge x \geq 0$

Die letzte kann es auch nicht sein, da vor der Schleife $\mathrm{df} = 1$ und $x = n$ gilt, ð$(\mathrm{df} + x)!! = (1 + n)!!$. Jedoch ist offenbar $(1 + n)!! \neq n!!$.

$\Rightarrow$ Die Schleifeninvariante lautet: $\mathrm{df} \cdot x!! = n!! \wedge x \geq 0$

(b) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung P gilt.

Lösungsvorschlag

Zu zeigen $P \Rightarrow \mathrm{wp}(\texttt{"Code vor der Schleife"}, I)$

$$\begin{aligned}
\mathrm{wp}(\texttt{"Code vor der Schleife"}, I) &\equiv \mathrm{wp}(\texttt{"df = 1; x = n;"}, (\mathrm{df} \cdot x)!! = n!! \land x \geq 0) \\
&\equiv \mathrm{wp}(\texttt{"df = 1;"}, (\mathrm{df} \cdot n)!! = n!! \land n \geq 0) \\
&\equiv \mathrm{wp}(\texttt{""}, (1 \cdot n)!! = n!! \land n \geq 0) \\
&\equiv n!! = n!! \land n \geq 0 \\
&\equiv n \geq 0 \\
&\equiv P
\end{aligned}$$

Insbesondere folgt damit die Behauptung.

(c) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

Lösungsvorschlag

zu zeigen: $I \wedge \text{Schleifenbedingung} \Rightarrow \text{wp}(\texttt{"Code in der Schleife"}, I)$

Bevor wir dies beweisen, zeigen wir erst $x \cdot (x-2)!! = x!!$.

- Fall x ist gerade ($n!! = 2^k \cdot k!$ für $k := \frac{n}{2}$):
  $x \cdot (x-2)!! = x \cdot 2^{\frac{x-2}{2}} \cdot (\frac{x-2}{2})! = x \cdot \frac{1}{2} \cdot 2^{\frac{x}{2}} \cdot (\frac{x}{2}-1)! = 2^{\frac{x}{2}} \cdot (\frac{x}{2})! = x!!$
  Nebenrechnung (Division mit gleicher Basis: $x^{a-b} = \frac{x^a}{x^b}$):
  $2^{\frac{x-2}{2}} = 2^{(\frac{x}{2}-\frac{2}{2})} = \frac{2^{\frac{x}{2}}}{2^{\frac{2}{2}}} = \frac{2^{\frac{x}{2}}}{2^1} = \frac{2^{\frac{x}{2}}}{2} = \frac{1}{2} \cdot 2^{\frac{x}{2}}$
  Nebenrechnung ($n! = (n-1)! \cdot n$):
  $x \cdot \frac{1}{2} \cdot (\frac{x}{2}-1)! = \frac{x}{2} \cdot (\frac{x}{2}-1)! = \frac{x}{2}!$
- Fall x ist ungerade:

Dies benutzen wir nun, um den eigentlichen Beweis zu führen:

$$
\begin{aligned}
\text{wp}(\texttt{"Code vor der Schleife"}, I) &\equiv \text{wp}(\texttt{"df = df * x; x = x - 2;"}, (\text{df} \cdot x)!! = n!! \wedge x \geq 0) \\
&\equiv \text{wp}(\texttt{"df = df * x;"}, (\text{df} \cdot (x-2)))!! = n!! \wedge x - 2 \geq 0) \\
&\equiv \text{wp}(\texttt{""}, (\text{df} \cdot x \cdot (x-2)))!! = n!! \wedge x - 2 \geq 0) \\
&\equiv (\text{df} \cdot x)!! = n!! \wedge x \geq 2 \\
&\equiv (\text{df} \cdot x)!! = n!! \wedge x > 1 \\
&\equiv I \wedge x > 1 \\
&\equiv I \wedge \text{Schleifenbedingung}
\end{aligned}
$$

(d) Zeigen Sie formal mittels wp-Kalkül, dass am Ende der Methode die Nachbedingung Q erfüllt wird.

Lösungsvorschlag

z.z. $I \wedge \neg\text{Schleifenbedingung} \Rightarrow \text{wp}(\texttt{"Code nach der Schleife"}, Q)$

Wir vereinfachen den Ausdruck $I \wedge \neg$Schleifenbedingung:

$I \wedge \neg\text{Schleifenbedingung} \equiv I \wedge (x \leq 1) \equiv I \wedge ((x = 0) \vee (x = 1)) \equiv (I \wedge (x = 0)) \vee (I \wedge (x = 1)) \equiv (df \cdot 1 = n!!) \vee (df \cdot 1 = n!!) \equiv df = n!!$

Damit gilt:

$\text{wp}(\texttt{"Code nach der Schleife"}, Q) \equiv \text{wp}(\texttt{""}, df = n!!) \equiv df = n!! \equiv I \wedge \neg\text{Schleifenbedingung}$

(e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz ihre Wahl. Terminierungsfunktion

Lösungsvorschlag

Sei $T(x) := x$. T ist offenbar ganzzahlig. Da x in jedem Schleifendurchlauf um 2 verringert wird, ist T streng monoton fallend. Aus der Schleifeninvariante folgt $x \geq 0$ und daher ist x auch nach unten beschränkt. Damit folgt $I \Rightarrow T \geq 0$ und T ist eine gültige Terminierungsfunktion.

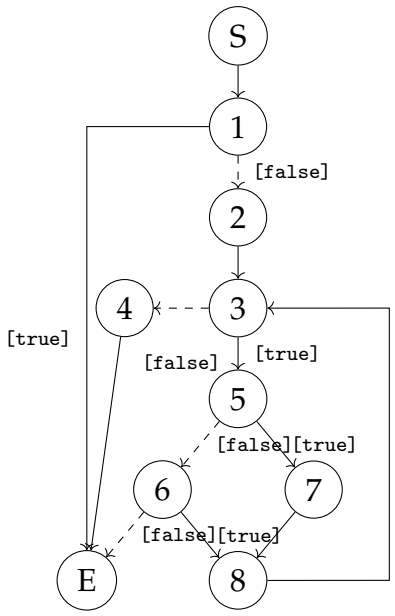
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2015/09/Thema-2/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Methode „binToInt()“ und Kontrollflussgraph“ (66116-2017-F.T2-TA2-A1)

Kontrollflussgraph

Gegeben Sei folgende Methode und ihr Kontrollflussgraph:

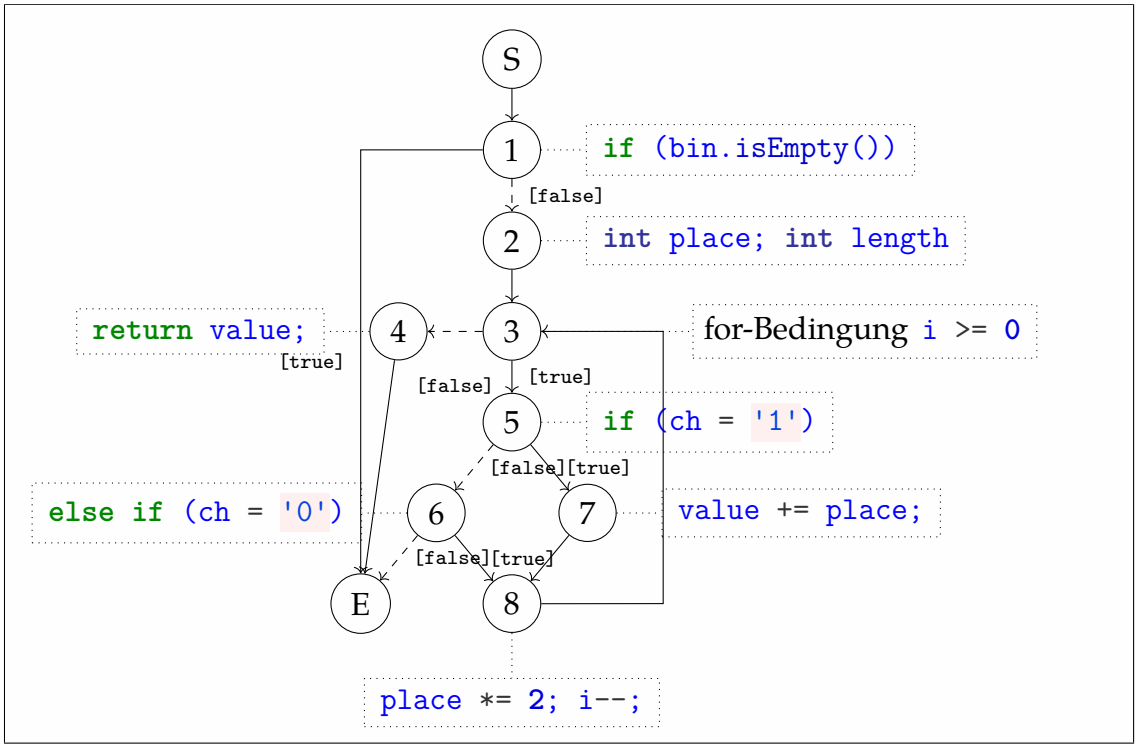
```
int binToInt(String bin) {
  if (bin.isEmpty())
    return -1;
  int place = 1, value = 0;
  int length = bin.length() -
↪ 1;
  for (int i = length; i >= 0;
↪ --i) {
    char ch = bin.charAt(i);
    if (ch == '1') {
      value += place;
    } else if (ch == '0') {
      // do nothing
    } else {
      return -1;
    }
    place *= 2;
  }
  return value;
}
```



Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java`

(a) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen

Lösungsvorschlag



(i) Verzweigungsüberdeckung

Lösungsvorschlag

p1 (Pfad 1) S 1 E
p2 S 1 2 3 4 E
p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E

(ii) Schleife-Inneres-Überdeckung

Lösungsvorschlag

Äußere Pfade (äußere Pfade): (ohne Ausführung der Wiederholung)

p1 S 1 E
p2 S 1 2 3 4 E

Grenzpfade (boundary test) (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

p4 S 1 2 3 5 6 E

Innere Pfade (interior test) (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

p5 S 1 2 3 5 7 8 3 5 7 8 3 4 E
p6 S 1 2 3 5 7 8 3 5 6 8 3 4 E
p7 S 1 2 3 5 6 8 3 5 6 8 3 4 E
p8 S 1 2 3 5 6 8 3 5 7 8 3 4 E
p9 S 1 2 3 5 7 8 3 5 7 8 3 5 6 E
p10 = p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E
p11 S 1 2 3 5 6 8 3 5 6 8 3 5 6 E
p12 S 1 2 3 5 6 8 3 5 7 8 3 5 6 E

mit minimaler Testfallanzahl genügen würden.

(b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall an, andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

Erweitere Methode, die die Knotennamen ausgibt:

```
public static final String RESET = "\u001B[0m";
public static final String ROT = "\u001B[31m";
public static final String GRÜN = "\u001B[32m";

static int binToIntLog(String bin) {
  System.out.println("\nInput: " + bin);
  System.out.print("S");
  System.out.print(1);
  if (bin.isEmpty()) {
```

```
      System.out.print("E");
      System.out.println("\nOutput: " + -1);
      return -1;
    }
    System.out.print(2);
    int place = 1, value = 0;
    int length = bin.length() - 1;
    System.out.print(3);
    for (int i = length; i >= 0; --i) {
      char ch = bin.charAt(i);
      System.out.print(5);
      if (ch == '1') {
        System.out.print(ROT + 7 + RESET);
        value += place;
      } else {
        System.out.print(GRÜN + 6 + RESET);
        if (ch == '0') {
          // do nothing
        } else {
          System.out.print("E");
          System.out.println("\nOutput: " + -1);
          return -1;
        }
      }
      System.out.print(8);
      place *= 2;
      System.out.print(3);
    }
    System.out.print(4);
    System.out.print("E");
    System.out.println("\nOutput: " + value);
    return value;
  }

  public static void main(String[] args) {
    binToIntLog(""); // p1
    binToIntLog("??"); // p2 not feasible
    binToIntLog("x01"); // p3
    binToIntLog("x"); // p4

    binToIntLog("11"); // p5
    binToIntLog("01"); // p6
    binToIntLog("00"); // p7
    binToIntLog("10"); // p8

    binToIntLog("x11"); // p9
    binToIntLog("x01"); // p10
    binToIntLog("x00"); // p11
    binToIntLog("x10"); // p12
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java`

Alle mit Ausnahme von p2.

Zyklomatische Komplexität nach Mc-Cabe

p2 ist nicht überdeckbar. Passiert ein Wert der Variable `bin` die erste if-Verzweigung, dann hat der Wert eine Länge größer 0 und betritt deshalb die Wiederholung mit fester Anzahl.

p1	S 1 E	`binToInt("");`
p2	S 1 2 3 4 E	not feasible
p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p4	S 1 2 3 5 6 E	`binToInt("x");`
p5	S 1 2 3 5 7 8 3 5 7 8 3 4 E	`binToInt("11");`
p6	S 1 2 3 5 7 8 3 5 6 8 3 4 E	`binToInt("01");`
p7	S 1 2 3 5 6 8 3 5 6 8 3 4 E	`binToInt("00");`
p8	S 1 2 3 5 6 8 3 5 7 8 3 4 E	`binToInt("10");`
p9	S 1 2 3 5 7 8 3 5 7 8 3 5 6 E	`binToInt("x11");`
p10 = p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p11	S 1 2 3 5 6 8 3 5 6 8 3 5 6 E	`binToInt("x00");`
p12	S 1 2 3 5 6 8 3 5 7 8 3 5 6 E	`binToInt("x10");`

(c) Bestimmen Sie anhand des Kontrollflussgraphen die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach Mc-Cabe.

Lösungsvorschlag

Binärverzweigungen 4

Knoten 10

Kanten 13

Anhand der Binärverzweigungen:

$$\begin{aligned} M &= b + p \\ &= 4 + 1 \\ &= 5 \end{aligned}$$

oder durch Anzahl Kanten e und Knoten n

$$\begin{aligned} M &= e - n + 2p \\ &= 13 - 10 + 2 \cdot 1 \\ &= 5 \end{aligned}$$

(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

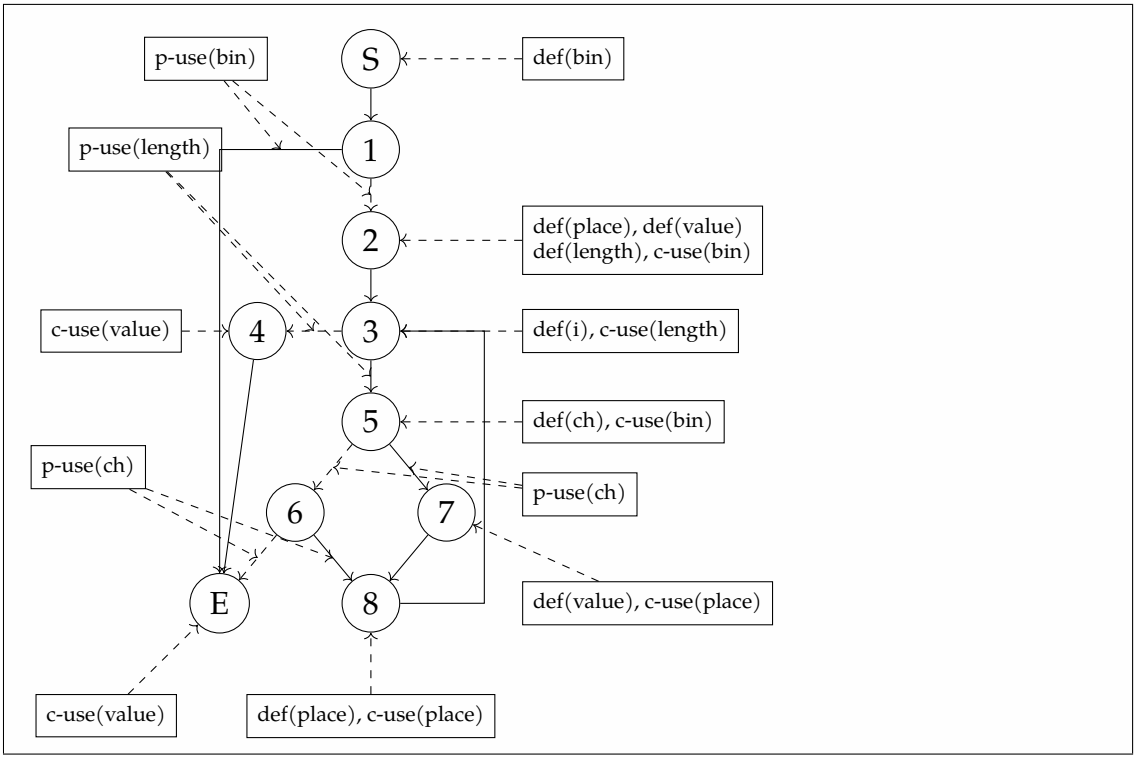
Datenfluss-annotierter Kontrollflussgraph

Lösungsvorschlag

Nein, da **p2** nicht überdeckbar ist.

(e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (`defs` bzw. `uses`) für jede betroffene Variable in der zeitlichen Reihenfolge ihres AuftreteS zur Laufzeit an.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „wp-Kalkül mit Invariante bei Methode „mul()““ (66116-2017-F.T2-TA2-A4)

Invariante

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei die folgende Methode mit Vorbedingung $P := x \geq 0 \wedge y \geq 0$ und Nachbedingung $Q := x \cdot y = z$.

```
int mul (int x , int y) {
  /* P */
  int z = 0, i = 0;
  while (i++ != x)
    z += y;
  /* Q */
  return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

- $I_1 := z + i \cdot y = x \cdot y$
- $I_2 := \text{false}$
- $I_3 := z + (x - i) \cdot y = x \cdot y$

(a) Beweisen Sie formal für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

Lösungsvorschlag

$$\begin{aligned} \text{wp}(\texttt{"Code vor der Schleife"}, I_1) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, z + i \cdot y = x \cdot y) \\ &\equiv \text{wp}(\texttt{""}, 0 + 0 \cdot y = x \cdot y) \\ &\equiv 0 = x \cdot y \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned} \text{wp}(\texttt{"Code vor der Schleife"}, I_2) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, \text{false}) \\ &\equiv \text{wp}(\texttt{""}, \text{false}) \\ &\equiv \text{false} \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned}\text{wp}(\texttt{"Code vor der Schleife"}, I_3) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, z + (x - i) \cdot y = x \cdot y) \\ &\equiv \text{wp}(\texttt{""}, 0 + (x - 0) \cdot y = x \cdot y) \\ &\equiv x \cdot y = x \cdot y \\ &\equiv \text{wahr}\end{aligned}$$

(b) Weisen Sie formal nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind oder welche nicht.

Lösungsvorschlag

Für den Nachweis muss der Code etwas umformuliert werden:

```
int mul (int x , int y) {
  /* P */
  int z = 0, i = 0;
  while (i != x) {
    i = i + 1;
    z = z + y;
  }
  /* Q */
  return z;
}
```

$$\begin{aligned}\text{wp}(\texttt{"Code Schleife"}, I_1 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, z + i \cdot y = x \cdot y \wedge i \neq x) \\ &\equiv \text{wp}(\texttt{"i = i + 1;"}, z + y + i \cdot y = x \cdot y \wedge i \neq x) \\ &\equiv \text{wp}(\texttt{""}, z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\ &\equiv z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x \\ &\equiv z + i \cdot y + 2 \cdot y = x \cdot y \wedge i + 1 \neq x \\ &\equiv \text{falsch} \wedge i + 1 \neq x \\ &\equiv \text{falsch}\end{aligned}$$

$$\begin{aligned}\text{wp}(\texttt{"Code Schleife"}, I_2 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, \text{false} \wedge i \neq x) \\ &\equiv \text{wp}(\texttt{""}, \text{false} \wedge i \neq x) \\ &\equiv \text{falsch} \wedge i \neq x \\ &\equiv \text{falsch}\end{aligned}$$

ektheit

$$\begin{aligned}
\text{wp}(\texttt{"Code Schleife"}, I_3 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, z + (x-i)\cdot y = x\cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{"i = i + 1;"}, z + y + (x-i)\cdot y = x\cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{""}, z + y + (x-i+1)\cdot y = x\cdot y \wedge i+1 \neq x) \\
&\equiv z + y + x\cdot y - i\cdot y + y = x\cdot y \wedge i+1 \neq x \\
&\equiv z + 2\cdot y + x\cdot y - i\cdot y = x\cdot y \wedge i+1 \neq x \\
&\equiv \text{wahr}
\end{aligned}$$

(c) Beweisen Sie formal, aus welchen der drei Prädikate die Nachbedingung gefolgert werden darf bzw. nicht gefolgert werden kann.

Lösungsvorschlag

$I_1 := z + i\cdot y = x\cdot y$ $I_2 := \text{false}$ $I_3 := z + (x-i)\cdot y = x\cdot y$

$$\begin{aligned}
\text{wp}(\texttt{"Code nach Schleife"}, I_1 \wedge i = x) &\equiv \text{wp}(\texttt{""}, z + i\cdot y = x\cdot y \wedge i = x) \\
&\equiv z + i\cdot y = x\cdot y \wedge i = x \\
&\equiv z + x\cdot y = x\cdot y \\
&\neq Q
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\texttt{"Code nach Schleife"}, I_2 \wedge i = x) &\equiv \text{wp}(\texttt{""}, \text{false} \wedge i = x) \\
&\equiv \text{false} \wedge i = x \\
&\equiv \text{falsch} \\
&\neq Q
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\texttt{"Code nach Schleife"}, I_3 \wedge i = x) &\equiv \text{wp}(\texttt{""}, z + (x-i)\cdot y = x\cdot y \wedge i = x) \\
&\equiv z + (x-i)\cdot y = x\cdot y \wedge i = x \\
&\equiv z + (x-x)\cdot y = x\cdot y \\
&\equiv z + 0\cdot y = x\cdot y \\
&\equiv z + 0 = x\cdot y \\
&\equiv z = x\cdot y \\
&\equiv Q
\end{aligned}$$

(d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.

Lösungsvorschlag

Aus den Teilaufgaben folgt der Beweis der partiellen Korrektheit mit Hilfe der Invariante i_3. i steigt streng monoton von 0 an so lange gilt $i \neq x$. $i = x$ ist die Abbruchbedingung für die bedingte Wiederholung. Dann terminiert die Methode. Die Methode `mul` ist also total korrekt.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2017/03/Thema-2/Teilaufgabe-2/Aufgabe-4.tex

Examensaufgabe „Roboter in einer Montagehalle“ (66116-2019-F.T1-TA2-A3)

Testen

(a) Erläutern Sie kurz, was man unter der Methode der testgetriebenen Entwicklung versteht.

Lösungsvorschlag

> Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

(b) Geben Sie für obige Aufgabenstellung (Abarbeitung der Aufträge durch den Roboter) einen Testfall für eine typische Situation an (d. h. das Einsammeln von 4 Objekten an unterschiedlichen Positionen). Spezifizieren Sie als Input alle für die Abarbeitung eines Auftrages relevanten Eingabe- und Klassen-Parameter sowie den vollständigen und korrekten Output.

Lösungsvorschlag

> Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`

Examensaufgabe „Test-getriebene Entwicklung“ (66116-2019-F.T2-TA2-A1) [Testen]

Bewerten Sie die folgenden Aussagen und nehmen Sie jeweils Stellung.

(a) Tests zuerst

In test-getriebener Softwareentwicklung wird der Test vor der zu testenden Funktion programmiert.

Lösungsvorschlag

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

(b) Komponententests

Komponententests sind immer White-Box-Tests und nie Black-Box-Tests.

Lösungsvorschlag

Falsch: Komponententests (anderes Wort für Unit- oder Modul-Tests) können beides sein. [a]

[a] `https://qastack.com.de/software/362746/what-is-black-box-unit-testing`

(c) Akzeptanztests

Akzeptanz - resp. Funktionstests sind immer Black-Box-Tests und nie White-Box-Tests.

Lösungsvorschlag

In systems engineering, it may involve black-box testing performed on a system

Acceptance testing in extreme programming: Acceptance tests are black-box system tests.

a

[a] `https://en.wikipedia.org/wiki/Acceptance_testing`

(d) Fehler

Ein fehlgeschlagener Test und ein Testausführungsfehler bezeichnen denselben Sachverhalt.

Lösungsvorschlag

Falsch:

Fehler (Error)

Der Software-Test konnte durchgeführt werden, das ermittelte Ist-Ergebnis und das vorgegebene Soll-Ergebnis weichen jedoch voneinander ab. Ein derartiger Fehler liegt z. B. dann vor, wenn ein Funktionsaufruf einen abweichenden Wert berechnet.

Fehlschlag (Failure)

Während der Testdurchführung wurde ein Systemversagen festgestellt. Ein Fehlschlag liegt z. B. dann vor, wenn das zu testende System einen Programmabsturz verursacht und hierdurch erst gar kein Ist-Ergebnis ermittelt werden konnte.

Das Misslingen kann als Ursache einen Fehler (Error) oder ein falsches Ergebnis (Failure) haben, die beide per Exception signalisiert werden. Der Unterschied zwischen den beiden Begriffen liegt darin, dass Failures erwartet werden, während Errors eher unerwartet auftreten. [a]

[a] `https://de.wikipedia.org/wiki/JUnit`

(e) Test Suiten

Tests können hierarchisch strukturiert werden, so dass mit einem Befehl das gesamte zu testende System getestet werden kann.

Lösungsvorschlag

Richtig:

test suite (englisch „Testsammlung“, aus französisch suite Folge, Verkettung) bezeichnet eine organisierte Sammlung von Werkzeugen zum Testen technischer Apparate und Vorgänge. [a]

[a] `https://de.wikipedia.org/wiki/Testsuite`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/03/Thema-2/Teilaufgabe-2/Aufgabe-1.tex`

Examensaufgabe „White-Box-Tests“ (66116-2019-H.T1-TA1-A3)

White-Box-Testing

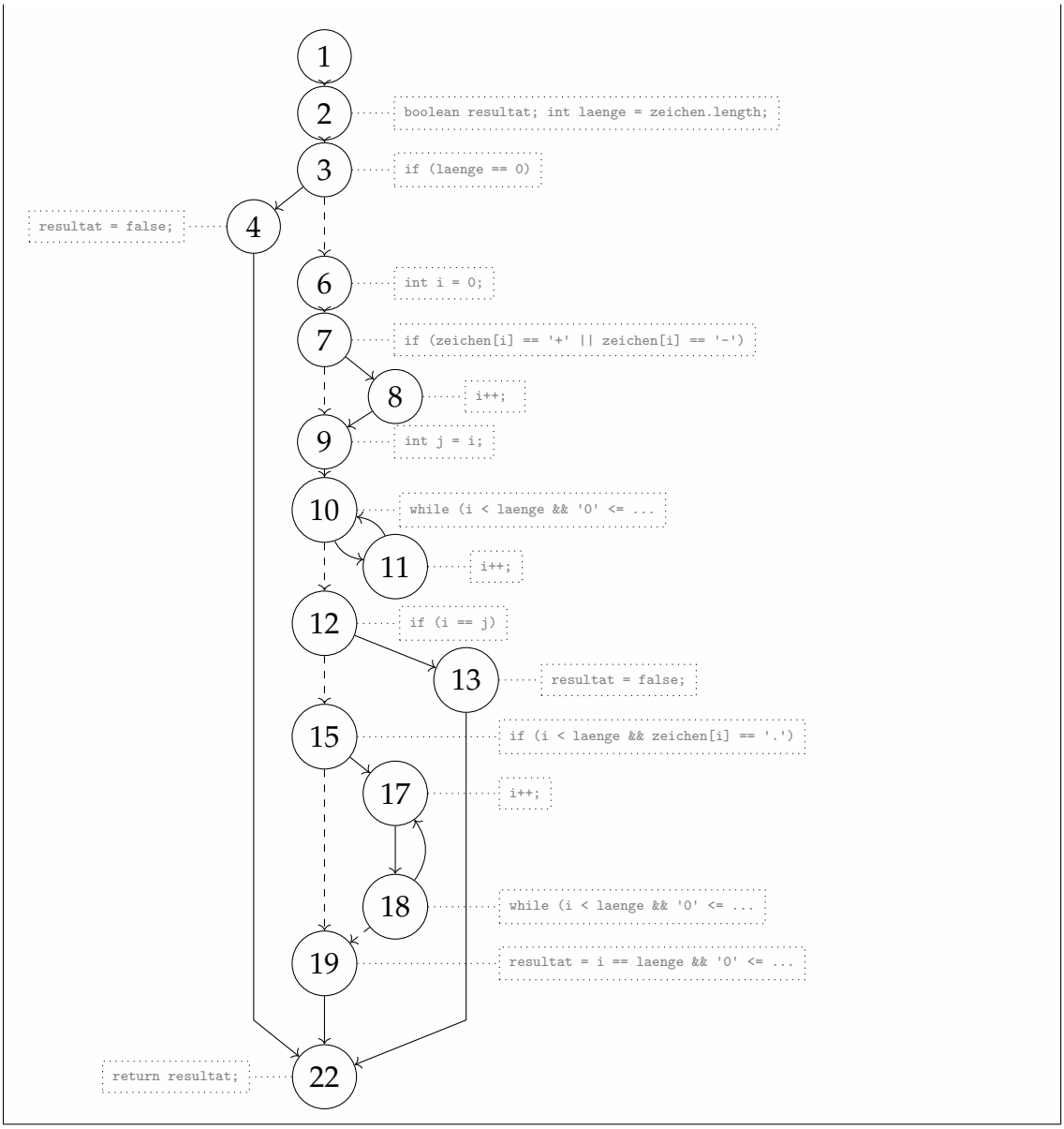
Eine Dezimalzahl hat ein optionales Vorzeichen, dem eine nichtleere Sequenz von Dezimalziffern folgt. Der anschließende gebrochene Anteil ist optional und besteht aus einem Dezimalpunkt, gefolgt von einer nichtleeren Sequenz von Dezimalziffern.

Die folgende Java-Methode erkennt, ob eine Zeichenfolge eine Dezimalzahl ist:

```
public static boolean istDezimalzahl(char[] zeichen) { // 1
  boolean resultat; int laenge = zeichen.length; // 2
  if (laenge == 0) // 3
    resultat = false; // 4
  else { // 5
    int i = 0; // 6
    if (zeichen[i] == '+' || zeichen[i] == '-') // 7
      i++; // 8
    int j = i; // 9
    while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9') // 10
      i++; // 11
    if (i == j) // 12
      resultat = false; // 13
    else { // 14
      if (i < laenge && zeichen[i] == '.') // 15
        do // 16
          i++; // 17
        while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9'); // 18
      resultat = i == laenge && '0' <= zeichen[i - 1] && zeichen[i - 1] <= '9';
↪  // 19
    } // 20
  } // 21
  return resultat; // 22
}
```

(a) Konstruieren Sie zu dieser Methode einen Kontrollflußgraphen. Markieren Sie dessen Knoten mit Zeilennummern des Quelltexts.

Lösungsvorschlag



(b) Geben Sie eine minimale Testmenge an, die das Kriterium der Knotenüberdeckung erfüllt. Geben Sie für jeden Testfall den durchlaufenen Pfad in der Notation $1 \to 2 \to \ldots$ an.

Lösungsvorschlag

- „“:
  (1) - (2) - (3) - (4) - (22)
- „1“:
  (1) - (2) - (3) - (6) - (7) - (9) - (10) - (11) - (10) - (12) - (15) - (19) - (22)
- „+1.0“:
  (1) - (2) - (3) - (6) - (7) - (8) - (9) - (10) - (11) - (10) - (12) - (15) - (17) - (18) - (19) - (22)
- „x“:
  (1) - (2) - (3) - (6) - (7) - (9) - (10) - (12) - (13) - (22)

(c) Verfahren Sie wie in b) für das Kriterium der Kantenüberdeckung.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(d) Wie stehen die Kriterien der Knoten- und Kantenüberdeckung zueinander in Beziehung? Begründen Sie Ihre Anwort.

Hinweis: Eine Testmenge ist minimal, wenn es keine andere Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität braucht nicht bewiesen zu werden.

Lösungsvorschlag

Die Kantenüberdeckung fordert, dass jede *Kante* des Kontrollflussgraphen von mindestens einem Testfall durchlaufen werden muss. Um das Kriterium zu erfüllen, müssen die Testfälle so gewählt werden, dass jede Verzweigungsbedingung mindestens *einmal wahr* und mindestens *einmal falsch* wird. Da hierdurch alle Knoten ebenfalls mindestens einmal besucht werden müssen, ist die *Anweisungsüberdeckung* in der Zweigüberdeckung *vollständig enthalten*.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2019/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Verifikation“ (66116-2020-H.T1-TA1-A1)

(a) Definieren Sie die Begriffe *„partielle Korrektheit“* und *„totale Korrektheit“* und grenzen Sie sie voneinander ab.

Lösungsvorschlag

partielle Korrektheit Ein Programmcode wird bezüglich einer Vorbedingung P und einer Nachbedingung Q partiell korrekt genannt, wenn bei einer Eingabe, die die Vorbedingung P erfüllt, jedes Ergebnis die Nachbedingung Q erfüllt. Dabei ist es noch möglich, dass das Programm nicht für jede Eingabe ein Ergebnis liefert, also nicht für jede Eingabe terminiert.

totale Korrektheit Ein Code wird total korrekt genannt, wenn er partiell korrekt ist und zusätzlich für jede Eingabe, die die Vorbedingung P erfüllt, terminiert. Aus der Definition folgt sofort, dass total korrekte Programme auch immer partiell korrekt sind.

(b) Geben Sie die Verifikationsregel für die abweisende Schleife `while(b) { A }` an.

Lösungsvorschlag

Eine abweisende Schleife ist eine while-Schleife, da die Schleifenbedingung schon bei der ersten Prüfung falsch sein kann und somit die Schleife abgewiesen wird.

Um die schwächste Vorbedingung eines Ausdrucks der Form „`while(b) { A }`“ zu finden, verwendet man eine *Schleifeninvariante*. Sie ist ein Prädikat für das

$$\{I \wedge b\} A \{I\}$$

gilt. Die Schleifeninvariante gilt also sowohl vor, während und nach der Schleife.

(c) Erläutern Sie kurz und prägnant die Schritte zur Verifikation einer abweisenden Schleife mit Vorbedingung P und Nachbedingung Q.

Lösungsvorschlag

Schritt 0: Schleifeninvariante I finden

Schritt 1: I gilt vor Schleifenbeginn,
d.h: $P \Rightarrow \text{wp}(\texttt{"Code vor Schleife"}, I)$

Schritt 2: I gilt nach jedem Schleifendurchlauf
d.h. $I \wedge b \Rightarrow \text{wp}(\texttt{"Code in der Schleife"}, I)$

Schritt 3: Bei Terminierung der Schleife liefert Methode das gewünschte Ergebnis,
d.h. $I \wedge \neq b \Rightarrow \text{wp}(\texttt{"Code nach der Schleife"}, Q)$

(d) Wie kann man die Terminierung einer Schleife beweisen?

Lösungsvorschlag

Zum Beweis der Terminierung einer Schleife muss eine Terminierungsfunktion T angeben werden:

$$T\colon V \to \mathbb{N}$$

V ist eine Teilmenge der Ausdrücke über die Variablenwerte der Schleife

Die Terminierungsfunktion muss folgende Eigenschaften besitzen:

- Ihre Werte sind natürliche Zahlen (einschließlich 0).
- Jede Ausführung des Schleifenrumpfs verringert ihren Wert (streng monoton fallend).
- Die Schleifenbedingung ist false, wenn $T = 0$.

T ist die obere Schranke für die noch ausstehende Anzahl von Schleifendurchläufen.

Beweise für Terminierung sind nicht immer möglich!

(e) Geben Sie für das folgende Suchprogramm die nummerierten Zusicherungen an. Lassen Sie dabei jeweils die invariante Vorbedingung P des Suchprogramms weg. Schreiben Sie nicht auf dem Aufgabenblatt! wp-Kalkül

$P \equiv n > 0 \land a_0 \dots a_{n-1} \in \mathbb{Z}^n \land m \in \mathbb{Z}$

```
int i = -1;
// (1)
int j = 0;
// (2)
while (i == -1 && j < n) // (3)
{ // (4)
  if (a[j] == m) {
    // (5)
    i = j;
    // (6)
  } else {
    // (7)
    j = j + 1;
    // (8)
  }
  // (9)
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/Verfikation.java`

$Q \equiv P \land (i = -1 \land \forall\, 0 \leq k < n : a_k \neq m) \lor (i \geq 0 \land a_i = m)$

Lösungsvorschlag

In dieser Aufgabenstellung fällt die Vorbedingung mit der Invariante zusammen. Es muss kein wp-Kalkül berechnet werden, sondern „nur“ die Zuweisungen nachverfolgt werden, um zum Schluss die Nachbedingung zu erhalten.

1. $(i = -1) \land P$
2. $(i = -1) \land (j = 0) \land P$
3. $(i = -1) \land (0 \leq j < n) \land P$
4. $(i = -1) \land (0 \leq j < n) \land P$
5. $(i = -1) \land (a_j = m) \land (0 \leq j < n) \land P$
6. $(i \geq 0) \land i = j \land (a_j = m) \land (0 \leq j < n) \land P$
7. $(i = -1) \land (\forall\, 0 \leq j < n : a_j \neq m) \land P$
8. $(i = -1) \land (\forall\, 0 < j \leq n : a_j \neq m) \land P$
9. $((i = -1) \land (\forall\, 0 \leq k < j : a_k \neq m)) \lor ((i \geq 0) \land (a_i = m)) \land P$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „White-Box-Testverfahren“ (66116-2020-H.T1-TA1-A4)

White-Box-Testing
Kontrollflussgraph

Diese Aufgabe behandelt *Wortpalindrome*, also Wörter, die vorwärts und rückwärts gelesen jeweils dieselbe Zeichenkette bilden, z. B. `Otto` oder `Rentner`. Leere Wortpalindrome (also Wortpalindrome der Wortlänge 0) sind dabei nicht zulässig.

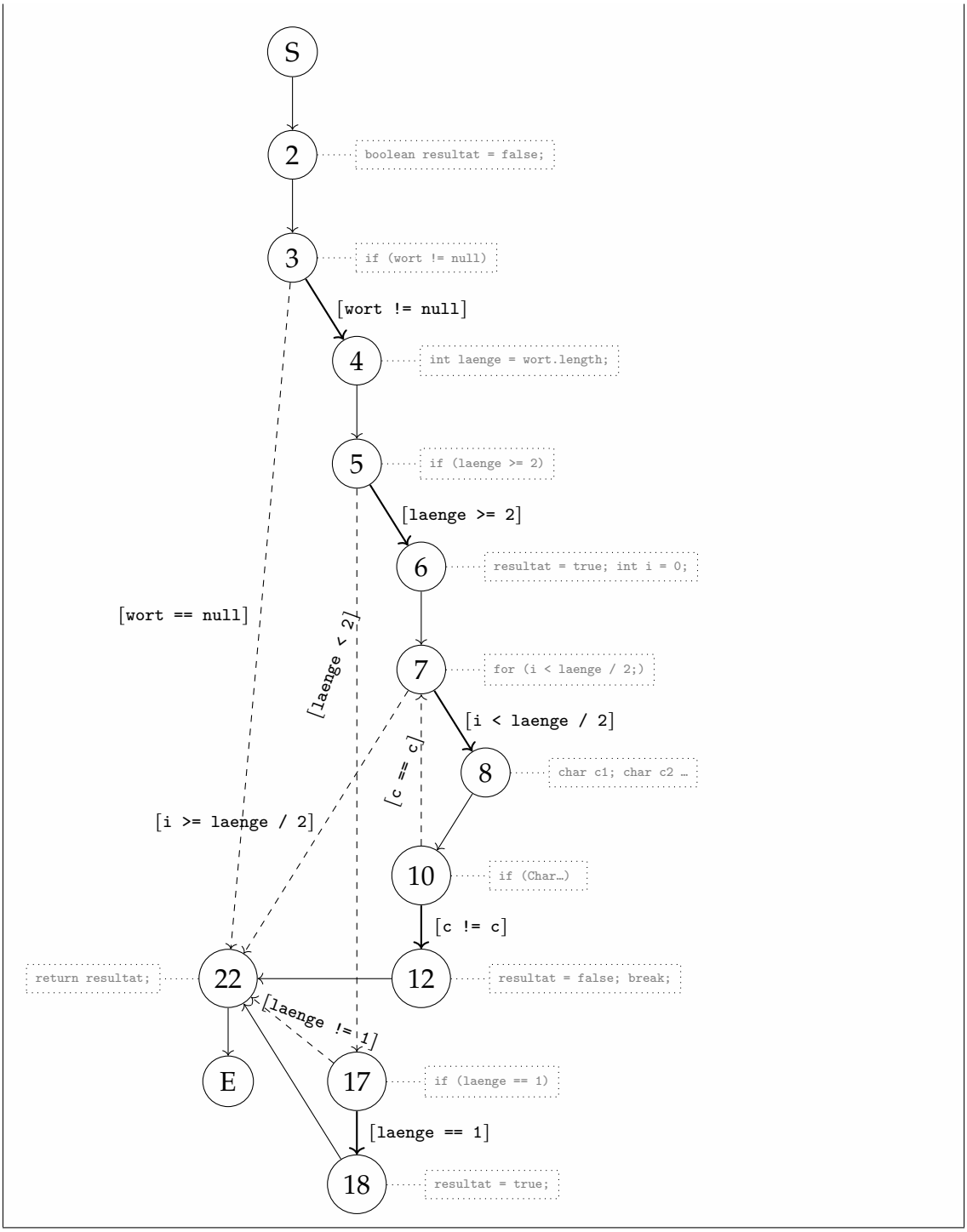
Folgende *Java-Methode* prüft, ob das übergebene Zeichen-Array ein Wortpalindrom darstellt:

```
public static boolean istWortpalindrom(char[] wort) { // 1
  boolean resultat = false; // 2
  if (wort != null) { // 3
    int laenge = wort.length; // 4
    if (laenge >= 2) { // 5
      resultat = true; // 6
      for (int i = 0; i < laenge / 2; ++i) { // 7
        char c1 = wort[i]; // 8
        char c2 = wort[laenge - 1 - i]; // 9
        if (Character.toLowerCase(c1) != Character.toLowerCase(c2)) // 10
        { // 11
          resultat = false; // 12
          break; // 13
        } // 14
      } // 15
    } else { // 16
      if (laenge == 1) { // 17
        resultat = true; // 18
      } // 19
    } // 20
  } // 21
  return resultat; // 22
} // 23
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/Palindrom.java`

(a) Geben Sie für die Methode einen *Kontrollflussgraphen* an, wobei Sie die Knoten mit den jeweiligen Zeilennummern im Quelltext beschriften.

Lösungsvorschlag

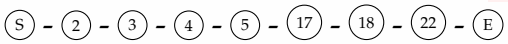


(b) Geben Sie eine *minimale Testmenge* an, die das Kriterium der Anweisungsüberdeckung erfüllt.

Hinweis: Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss *nicht* bewiesen werden.

Lösungsvorschlag

```
isWortpalindrom(new char[] { 'a' }):
```



`isWortpalindrom(new char[] { 'a', 'b' })`:
S - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 10 - 12 - 22 - E

(c) Geben Sie eine *minimale Testmenge* an, die das Kriterium der *Boundary-Interior-Pfadüberdeckung* erfüllt.

Hinweis: Das Kriterium *Boundary-Interior-Pfadüberdeckung* beschreibt einen Spezialfall der Pfadüberdeckung, wobei nur Pfade berücksichtigt werden, bei denen jede Schleife nicht mehr als zweimal durchlaufen wird.

Lösungsvorschlag

Es gibt noch ganz viele infeasible Pfade, die hier nicht aufgeführt werden.

Äußere Pfade

- `isWortpalindrom(null)`:
  S - 2 - 3 - 22 - E
- `isWortpalindrom(new char[] { })`:
  S - 2 - 3 - 4 - 5 - 17 - 22 - E
- `isWortpalindrom(new char[] { 'a' })`:
  S - 2 - 3 - 4 - 5 - 17 - 18 - 22 - E

Grenzpfade (boundary paths, boundary test) Für Schleifen fester Laufläge ist diese Testfallgruppe leer.

`isWortpalindrom(new char[] { 'a', 'a' })`:
S - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 10 - 7 - 22 - E

`isWortpalindrom(new char[] { 'a', 'b' })`:
S - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 10 - 12 - 22 - E

Innere Pfade (interior test)

- `isWortpalindrom(new char[] { 'a', 'a', 'a', 'a' })`:
  S - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 10 - 7 - 8 - 10 - 7 - 22 - E
- `isWortpalindrom(new char[] { 'a', 'b', 'a', 'a' })`:
  S - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 10 - 7 - 8 - 10 - 12 - 22 - E

(d) Im Falle des Kriteriums Pfadüberdeckung können minimale Testmengen sehr groß werden, da die Anzahl der Pfade sehr schnell zunimmt. Wie viele *mögliche Pfade* ergeben sich maximal für eine Schleife, die drei einseitig bedingte Anweisungen hintereinander enthält und bis zu zweimal durchlaufen wird? Geben Sie Ihren Rechenweg an (das Ergebnis alleine gibt keine Punkte).

Lösungsvorschlag

Pro Schleifendurchlauf: $2 \cdot 2 \cdot 2 = 2^3 = 8$

Maximal 2 Schleifendurchläufe: $8 \cdot 8 + 8 + 1 = 73$

wenn man die Fälle 0, 1 oder 2 Durchläufe getrennt betrachtet. Natürlich entstehen aus der Sicht des reinen Graphen keine neuen Pfade, so dass man auch

zur Antwort 9 neigen könnte. Aber es geht darum, dass bei späteren Durchläufen andere Dinge passieren können (Fehler), so dass man eine Art Kopie des Schleifenteils pro Durchlauf betrachtet.

(e) Könnte für das hier abgebildete Quelltext-Beispiel auch das Verfahren der *unbegrenzten Pfadüberdeckung* (also Abdeckung aller möglicher Pfade ohne Beschränkung) als Test-Kriterium gewählt werden? Begründen Sie.

Lösungsvorschlag

Nein. Der Pfad von Konten (17) zu Knoten (22) wird nie beschritten, d. h. wird Zeile 17 betreten, wird auch immer die bedingte Anweisung in Zeile 18 ausgeführt.

Statt

```
if (laenge == 1) {
  resultat = true;
}
```

könnte auch nur so geschrieben werden:

```
resultat = true;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-4.tex

Übungsaufgabe „Gaußsche Summenformel“ (Vollständige Induktion)

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `sum` nur für Werte $n \geq 0$ aufgerufen wird.

```
public static int sum(int n) {
  if (n <= 0) {
    return 0;
  }
  return n + sum(n - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/induktion/Gauss.java`

Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `sum(n)` die Summe der ersten n aufeinanderfolgenden natürlichen Zahlen für alle $n \geq 0$ berechnet, wobei gilt

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$\sum_{k=0}^{0} k = \frac{0(0+1)}{2} = \frac{0}{2} = 0$$

$$\texttt{sum(0)} = 0$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

$$\texttt{sum(n)} = n + \frac{(n-1)((n-1)+1)}{2} = n + \frac{(n-1)n}{2}$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss. ——

$$\sum_{k=0}^{n+1} k = \frac{(n+1)\big((n+1)+1\big)}{2}$$

$$\begin{aligned}
\texttt{sum(n+1)} &= (n+1) + \frac{((n+1)-1)(n+1))}{2} && n+1-1=n \\
&= (n+1) + \frac{n(n+1)}{2} && (n+1) \text{ eingesetzt} \\
&= \frac{2(n+1)}{2} + \frac{n(n+1)}{2} && (n+1) \text{ als Bruch geschrieben} \\
&= \frac{2(n+1)+n(n+1)}{2} && \text{Hauptnenner 2} \\
&= \frac{(2+n)(n+1)}{2} && (n+1) \text{ ausgeklammert} \\
&= \frac{(n+2)(n+1)}{2} && \text{Kommutativgesetz angewandt} \\
&= \frac{(n+1)(n+2)}{2} && \text{getauscht nach Kommutativgesetz} \\
&= \frac{(n+1)\big((n+1)+1\big)}{2} && \text{mit } (n+1) \text{ an der Stelle von } n
\end{aligned}$$

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class GaussTest {

  private void teste(int n, int erwartet) {
    assertEquals(Gauss.sum(n), erwartet);
  }

  @Test
  public void teste() {
    teste(0, 0);
    teste(1, 1);
    teste(2, 3);
    teste(3, 6);
    teste(4, 10);
    teste(5, 15);
    teste(6, 21);
    teste(7, 28);
    teste(8, 36);
    teste(9, 45);
    teste(10, 55);
    teste(11, 66);
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/induktion/GaussTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/20_Vollstaendige-Induktion/Aufgabe_Gausssche-Summenformel.tex`

Übungsaufgabe „Geometrische Summenformel geoSum()“ (Vollständige Induktion)

Gegeben sei folgende Methode:

```java
public class GeoSum {
  // Math.pow(q, n) == q^n
  double geoSum(int n, double q) {
    if (n == 0) {
      return 1 - q;
    } else {
      return (1 - q) * Math.pow(q, n) + geoSum(n - 1, q);
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/totale_korrektheit/GeoSum.java`

Weisen Sie mittels vollständiger Induktion nach, dass

$$\text{geoSum}(n, q) = 1 - q^{n+1}$$

Dabei können Sie davon ausgehen, dass $q > 0$, $n \in \mathbb{N}_0$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$f(0) : \text{geoSum}(0, q) = 1 - q^{0+1} = 1 - q^1 = 1 - q$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$f(n) : \text{geoSum}(n, q) = 1 - q^{n+1}$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss. ——

$$\begin{aligned}
f(n) &= \text{geoSum}(n, q) \\
&= (1 - q) \cdot q^n + \text{geoSum}(n - 1, q) && \text{Java-Code in Mathe-Formel umgewandelt} \\
&= (1 - q) \cdot q^n + 1 - q^{(n-1)+1} && \text{für rekursiven Methodenaufruf gegebene Formel eingesetzt} \\
&= (1 - q) \cdot q^n + 1 - q^n && \text{Addition im Exponent}
\end{aligned}$$

$$
\begin{aligned}
f(n+1) &= \text{geoSum}(n+1, q) \\
&= (1-q) \cdot q^{n+1} + 1 - q^{n+1} && \text{von Java konvertierte Formel verwendet und } n+1 \text{ eingesetzt} \\
&= q^{n+1} - q^{(n+1)+1} + 1 - q^{n+1} && \text{ausmultipliziert} \\
&= -q^{(n+1)+1} + 1 && q^{n+1} - q^{n+1} = 0 \\
&= 1 - q^{(n+1)+1} && \text{Kommutativgesetz der Addition} \\
&= 1 - q^{(n+1)+1} && \text{was zu zeigen war}
\end{aligned}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/20_Vollstaendige-Induktion/Aufgabe_Geometrische-Summenformel.tex`

Übungsaufgabe „Summe ungerader Zahlen (Maurolicus 1575)“ (Vollständige Induktion)

Die schrittweise Berechnung der Summe der ersten n ungeraden Zahlen legt die Vermutung nahe: Die Summe aller ungeraden Zahlen von 1 bis $2n-1$ ist gleich dem Quadrat von n:

$$1 = 1 = 1^2$$
$$1 + 3 = 4 = 2^2$$
$$1 + 3 + 5 = 9 = 3^2$$
$$1 + 3 + 5 + 7 = 16 = 4^2$$

Folgende Java-Methode berechnet die Summer aller ungeraden Zahlen:

```java
public static int oddSum(int n) {
  if (n <= 1) {
    return 1;
  }
  return 2 * n - 1 + oddSum(n - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/induktion/Maurolicus.java`

Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `oddSum(n)` die Summe aller ungeraden Zahlen von 1 bis nur n-ten ungeraden Zahl berechnet, wobei gilt:

$$\sum_{i=1}^{n}(2i-1) = n^2$$

.

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$\sum_{i=1}^{1}(2i-1) = 2 \cdot 1 - 1 = 1 = 1^2$$

$$\texttt{oddSum(1)} = 1 = 1^2$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\sum_{i=1}^{n}(2i-1) = n^2$$

$$\texttt{oddSum(n)} = 2n - 1 + (n-1)^2$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss. ——

$$
\begin{aligned}
\texttt{oddSum(n)} &= 2(n+1) - 1 + ((n+1) - 1)^2 \\
&= 2(n+1) - 1 + n^2 \\
&= 2n + 2 + n^2 - 1 && \text{ausmultiplizieren} \\
&= 2n + 1 + n^2 && 2 - 1 = 1 \\
&= n^2 + 2n + 1 && \text{Kommutativgesetz} \\
&= (n+1)^2 && \text{mit erster Binomischer Formel: } (a+b)^2 = a^2 + 2ab + b^2
\end{aligned}
$$

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class MaurolicusTest {

  private void teste(int n, int erwartet) {
    assertEquals(Maurolicus.oddSum(n), erwartet);
  }

  @Test
  public void teste() {
    teste(1, 1);
    teste(2, 4);
    teste(3, 9);
    teste(4, 16);
    teste(5, 25);
    teste(6, 36);
    teste(7, 49);
    teste(8, 64);
    teste(9, 81);
    teste(10, 100);
    teste(11, 121);
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/induktion/MaurolicusTest.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/20_Vollstaendige-Induktion/Aufgabe_Summe-ungerader-Zahlen.tex`

Übungsaufgabe „Grundwissen“ (Formale Verifikation, wp-Kalkül, Hoare-Kalkül, Partielle Korrektheit, Totale Korrektheit, Invariante, Terminierungsfunktion)

(a) Geben Sie zwei verschiedene Möglichkeiten der formalen Verifikation an.

Lösungsvorschlag

1. Möglichkeit: formale Verifikation mittels *vollständiger Induktion* (eignet sich bei *rekursiven* Programmen).

2. Möglichkeit: formale Verifikation mittels *wp-Kalkül* oder *Hoare-Kalkül* (eignet sich bei *iterativen* Programmen).

(b) Erläutern Sie den Unterschied von partieller und totaler Korrektheit.

Lösungsvorschlag

partielle Korrektheit: Das Programm verhält sich spezifikationsgemäß, *falls* es terminiert.

totale Korrektheit: Das Programm verhält sich spezifikationsgemäß und es *terminiert immer*.

(c) Gegeben sei die Anweisungssequenz A. Sei P die Vorbedingung und Q die Nachbedingung dieser Sequenz. Erläutern Sie, wie man die (partielle) Korrektheit dieses Programmes nachweisen kann.

Lösungsvorschlag

Vorgehen	Horare-Kalkül	wp-Kalkül
Wenn die Vorbedingung P zutrifft, gilt nach der Ausführung der Anweisungssequenz A die Nachbedingung Q.	$\{P\}A\{Q\}$	$P \Rightarrow \text{wp}(A, Q)$

(d) Gegeben sei nun folgendes Programm:

```
A_1
while(b):
    A_2
A_3
```

wobei A_1, A_2, A_3 Anweisungssequenzen sind. Sei P die Vorbedingung und Q die Nachbedingung des Programms. Die Schleifeninvariante der while-Schleife wird mit I bezeichnet. Erläutern Sie, wie man die (partielle) Korrektheit dieses Programmes nachweisen kann.

Lösungsvorschlag Terminierungsfunktion

Vorgehen	Horare-Kalkül	wp-Kalkül
Die Invariante I gilt vor Schleifeneintritt.	$\{P\}A_1\{I\}$	$P \Rightarrow \text{wp}(A_1, I)$
I ist invariant, d. h. I gilt nach jedem Schleifendurchlauf.	$\{I \wedge b\}A_2\{I\}$	$I \wedge b \Rightarrow \text{wp}(A_2, I)$
Die Nachbedingung Q wird erfüllt.	$\{I \wedge \neg b\}A_3\{Q\}$	$I \wedge \neg b \Rightarrow \text{wp}(A_3, I)$

(e) Beschreiben Sie, welche Vorraussetzungen eine Terminierungsfunktion erfüllen muss, damit die totale Korrektheit gezeigt werden kann.

Lösungsvorschlag

Mit einer Terminierungsfunktion T kann bewiesen werden, dass eine Wiederholung terminiert. Sie ist eine Funktion, die

- ganzzahlig,
- nach unten beschränkt (die Schleifenbedingung ist *false*, wenn $T = 0$) und
- streng monoton fallend (jede Ausführung der Wiederholung verringert ihren Wert)

ist.

Im Hoare-Kalkül muss $\{I \wedge b \wedge (T = n)\}A\{T < n\}$ gezeigt werden, im wp-Kalkül $I \Rightarrow T \geq 0$. [a]

[a] `https://osg.informatik.tu-chemnitz.de/lehre/aup/aup-07-AlgorithmenEntwurf-script_de.pdf`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/10_Formale-Verifikation/Aufgabe_Grundwissen.tex`

Übungsaufgabe „Methode „f()““ (wp-Kalkül)

wp-Kalkül

Gegeben sei folgendes Programm: wp-Kalkül

```
int f(int x, int y) {
  /* P */
  x = 2 * x + 1 + x * x;
  y += 7;
  if (x > 196) {
    y = 2 * y;
  } else {
    y -= 8;
    x *= 2;
  } /* Q */
  return x + y;
}
```

Bestimmen Sie die schwächste Vorbedingung (weakest precondition), für die die Nachbedingung $Q := (x \geq 8) \land (y\%2 = 1)$ noch zutrifft.

Lösungsvorschlag

Mit dem Distributivgesetz der Konjugation gilt:

$$\begin{aligned}
&\text{wp}(\texttt{"A; if(b) B; else C;"}, Q) \equiv \\
&\text{wp}(\texttt{"A;"}, b) \land \text{wp}(\texttt{"A;B;"}, Q) \\
&\lor \\
&\text{wp}(\texttt{"A;"}, \neg b) \land \text{wp}(\texttt{"A;C;"}, Q)
\end{aligned}$$

Der tatsächliche Programmcode wird eingesetzt:

$$\begin{aligned}
&\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;if(x>196){y=2*y;}else{y-=8;x*=2;}"}, (x \geq 8) \land (y\%2 = 1)) \equiv \\
&\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;"}, x > 196) \land \\
&\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;y=2*y;"}, (x \geq 8) \land (y\%2 = 1)) \\
&\lor \\
&\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;"}, x \leq 196) \land \\
&\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;y-=8;x*=2;"}, (x \geq 8) \land (y\%2 = 1)) \\
&=: \text{P}
\end{aligned}$$

Nebenrechnung: wp("A;", b)

$$\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;"}, x > 196)$$

Wir lassen $y+=7$ weg, weil in der Nachbedingung kein y vorkommt und setzen in den Term $x > 196$ für das x die erste Code-Zeile $2 \cdot x + 1 + x \cdot x$ ein.

$$\equiv \quad \text{wp}(\texttt{""}, 2 \cdot x + 1 + x \cdot x > 196)$$

Nach der Transmformationsregel *Nichts passiert, die Vorbedingung bleibt gleich* kann das auch so geschrieben werden:

$$\equiv \quad 2 \cdot x + 1 + x \cdot x > 196$$

Die erste binomische Formel (Plus-Formel) lautet $(a+b)^2 = a^2 + 2ab + b^2$. Man kann die Formel auch umgedreht verwenden: $a^2 + 2ab + b^2 = (a+b)^2$. Die erste Code-Zeile $2 \cdot x + 1 + x \cdot x$ kann umformuliert werden in $1 + 2 \cdot 1 \cdot x + x \cdot x = 1^2 + 2 \cdot 1 \cdot x + x^2 = (1+x)^2 = (x+1)^2$. Wir haben für a die Zahl 1 und für b den Buchstaben x eingesetzt.

$$\equiv \quad (x+1)^2 > 196$$

Nebenrechnung: wp("A;B;", Q)

$$\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;y=2*y;"}, (x \geq 8) \land (y\%2 = 1))$$

Für das x in der Nachbedingung setzen wir die erste Code-Zeile $2 \cdot x + 1 + x \cdot x$ ein. Für das y in der Nachbedingung setzen wir dritte Code-Zeile `y=2*y;` ein und dann die zweite Code-Zeile `y+=7;`. Das wp-Kalkül arbeitet den Code rückswärts ab. in $y\%2$ die dritte Anweisung $y = 2 \cdot y$ einfügen: $2 \cdot y\%2$ dann in $2 \cdot y\%2$ die zweite Anweisung $y = y + 7$ einfügen: $2 \cdot (y+7)\%2$

$$\equiv \quad (x+1)^2 \geq 8 \land 2(y+7)\%2 = 1$$

Diese Aussage ist falsch, da $2(y+7)$ immer eine gerade Zahl ergibt und der Rest von einer Division durch zwei einer geraden Zahl immer 0 ist und nicht 1.

$$\equiv \quad (x+1)^2 \geq 8 \land \text{falsch}$$

$$\equiv \quad \text{falsch}$$

Nebenrechnung: wp("A;", $\neg b$)

$$\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;"}, x \leq 196)$$

Analog zu Nebenrechnung 1

$$\equiv \quad (x+1)^2 \leq 196$$

Nebenrechnung: wp("A;C;", Q)

$$\text{wp}(\texttt{"x=2*x+1+x*x;y+=7;y-=8;x*=2;"}, (x \geq 8) \land (y\%2 = 1))$$

„`x*=2;`“: $x \cdot 2$ für x einsetzen:

$$\equiv \quad \text{wp}(\texttt{"x=2*x+1+x*x;y+=7;y-=8;"}, (2 \cdot x \geq 8) \land (y\%2 = 1))$$

„`y-=8;`“: $y - 8$ für y einsetzen:

$$\equiv \quad \text{wp}(\texttt{"x=2*x+1+x*x;y+=7;"}, (2 \cdot x \geq 8) \land ((y-8)\%2 = 1))$$

„`y+=7`“: $y + 7$ für y einsetzen:

$$\equiv \quad \text{wp}(\texttt{"x=2*x+1+x*x;"}, (2 \cdot x \geq 8) \land (((y+7)-8)\%2 = 1))$$

„`x=2*x+1+x*x;`“: $(x+1)^2$ für x einsetzen:

$$\equiv \quad \text{wp}(\texttt{""}, (2 \cdot (x+1)^2 \geq 8) \land (((y+7)-8)\%2 = 1))$$

Nur noch die Nachbedingung stehen lassen:

$$\equiv \quad (2 \cdot (x+1)^2 \geq 8) \land (((y+7)-8)\%2 = 1)$$

Subtraktion:

$$\equiv \quad (2 \cdot (x+1)^2 \geq 8) \land ((y-1)\%2 = 1)$$

Vereinfachen (links beide Seiten durch 2 teilen und rechts von beiden Seiten 1 abziehen)

$$\equiv \quad (\tfrac{2 \cdot (x+1)^2}{2} \geq \tfrac{8}{2}) \land (((y-1)\%2) - 1 = 1 - 1)$$

Zwischenergebnis:

$$\equiv \quad ((x+1)^2 \geq 4) \land y\%2 = 0$$

Zusammenführung

Die Zwischenergebnisse aus den Nebenrechnungen zusammenfügen:

$$\equiv \quad [(x+1)^2 > 196 \land \text{falsch}] \lor [(x+1)^2 \leq 196 \land (x+1)^2 \geq 4 \land y\%2 = 0]$$

„falsch“ und eine Aussage verbunden mit logischem Und „$\land$“ ist insgesamt falsch:

$$\equiv \quad \text{falsch} \lor [(x+1)^2 \leq 196 \land (x+1)^2 \geq 4 \land y\%2 = 0]$$

falsch verbunden mit oder weglassen:

$$\equiv \quad (x+1)^2 \leq 196 \land (x+1)^2 \geq 4 \land y\%2 = 0$$

Umgruppieren, sodass nur noch ein $(x+1)^2$ geschrieben werden muss:

$$\equiv \quad 4 \leq (x+1)^2 \leq 196 \land y\%2 = 0$$

$4 = 2^2$ und $196 = 14^2$

$$\equiv \quad 2^2 \leq (x+1)^2 \leq 14^2 \land y\%2 = 0$$

Hoch zwei weg lassen: Betragsklammer $|x|$ oder auch Betragsfunktion hinzufügen (Die Betragsfunktion ist festgelegt als „Abstand einer Zahl von der Zahl Null“.

$$\equiv \quad 2 \leq |x+1| \leq 14 \land y\%2 = 0$$

Auf die Gleichung der linken Aussage -1 anwenden:

$$\equiv \quad 1 \leq |x| \leq 13 \land y\%2 = 0$$

Die Betragsklammer weg lassen:

$$\equiv \quad (1 \leq x \leq 13 \lor -13 \leq x \leq -1) \land y\%2 = 0$$

$$=: P$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/10_Formale-Verifikation/Aufgabe_Methode-f.tex`

Übungsaufgabe „Vorlesungsaufgabe WP-Kalkül“ (wp-Kalkül)

wp-Kalkül

Bestimmen Sie zur Nachbedingung Q die Vorbedingung P!

Nachbedingung: $Q \equiv x + y = 17$

Programmcode:

```
// P: ?
x += 5;
y *= 2;
z = z % 4;
y--;
// Q: x + y = 17
```

Lösungsvorschlag

ist gleichbedeuted mit

```
x = x + 5;
y = y * 2;
z = z % 4;
y = y - 1;
```

$$\text{wp}(\texttt{"x += 5; y *= 2; z = z \% 4; y--;"}, x + y = 17)$$

$y - 1$ einsetzten

$$\equiv \text{wp}(\texttt{"x += 5; y *= 2; z = z \% 4;"}, x + y - 1 = 17)$$

die 1 mit $+$ nach rechts bringen

$$\equiv \text{wp}(\texttt{"x += 5; y *= 2; z = z \% 4;"}, x + y = 18)$$

Im nächsten Schritt müssten wir ein z verändern. Wir haben aber in userer Bedingung kein z, deshalb kann es wegfallen.

$$\equiv \text{wp}(\texttt{"x += 5; y *= 2;"}, x + y = 18)$$

$y \cdot 2$ einsetzen

$$\equiv \text{wp}(\texttt{"x += 5;"}, x + y \cdot 2 = 18)$$

Auf x wird 5 hinzuaddiert.

$$\equiv \text{wp}(\texttt{""}, x + 5 + y \cdot 2 = 18)$$

Wir haben keinen Programmcode mehr. Wir können wp weglassen.

$$\equiv x + 5 + y \cdot 2 = 18$$

Die 5 nach rechts bringen

$$\equiv x + y \cdot 2 = 13$$

Alle Eingaben die Vorbedingung $P \equiv x + y \cdot 2 = 13$ erfüllen, erfüllen die Nachbedingung Q $x + y = 17$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/10_Formale-Verifikation/Aufgabe_Vorlesungsaufgabe-WP-Kalkuel.tex`

Übungsaufgabe „Größter gemeinsamer Teiler“ (Datenfluss-annotierter Kontrollflussgraph, Zyklomatische Komplexität nach Mc-Cabe, C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage))

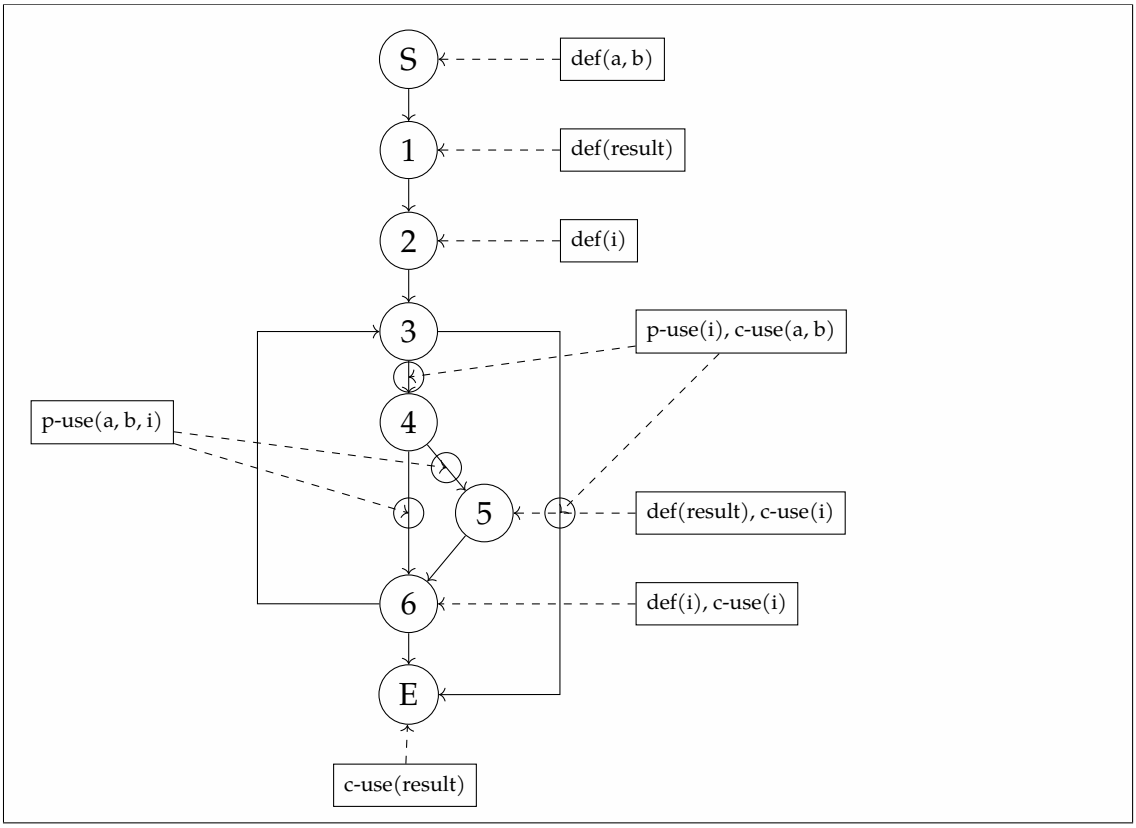
Gegeben sei folgende Methode:

```
public int ggT(int a, int b) {
  int result = 1;
  for (int i = 1; i <= Math.min(a, b); i++) {
    if ((a % i == 0) & (b % i == 0)) {
      result = i;
    }
  }
  return result;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/white_box/WhiteBox.java`

(a) Erstellen Sie den zur Methode gehörenden datenflussannotierten Kontrollflussgraphen.

Lösungsvorschlag



(b) Geben Sie die zyklomatische Komplexität M nach McCabe der Methode `ggT` an. (Nur das Ergebnis!)

Lösungsvorschlag

C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)

Berechnung durch Anzahl Binärverzweigungen b (p Anzahl der Zusammenhangskomponenten des Kontrollflussgraphen)

$$M = b + p$$

$\rightarrow M = 2 + 1 = 3$

oder durch Anzahl Kanten e und Knoten n

$$M = e - n + 2p$$

$\rightarrow M = 9 - 8 + 2 \cdot 1 = 3$

(c) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen Schleifen-Inneres-Überdeckung (Boundary-Interior-Coverage) genügen würden.

Lösungsvorschlag

Äußere Pfade

- S 1 2 3 E

Grenzpfade

- S 1 2 3 4 5 6 3 E
- S 1 2 3 4 6 3 E

Innere Pfade

- S 1 2 3 4 5 6 3 4 5 6 3 E
- S 1 2 3 4 6 3 4 6 3 E
- S 1 2 3 4 5 6 3 4 6 3 E
- S 1 2 3 4 6 3 4 5 6 3 E

(d) Geben Sie an, welche der Pfade aus der vorherigen Aufgabe nicht überdeckbar (”feasible”) sind und begründen Sie dies.

Lösungsvorschlag

Äußere Pfade

S 1 2 3 E ja, z. B. `ggT(-1, -2)`.

Grenzpfade

S 1 2 3 4 5 6 3 E ja, z. B. `ggT(10, 20)`.

S 1 2 3 4 6 3 E ja, z. B. `ggT(1, 2)`.

Innere Pfade

S 1 2 3 4 5 6 3 4 5 6 3 E ja, z. B. `ggT(2, 2)`.

S 1 2 3 4 6 3 4 6 3 E nicht feasible, da geteilt durch eins immer Modulo 0 ergibt, egal welche Zahl a oder b hat. Bei der ersten Schleifenwiederholung wird immer die innere If-Verzweigung genommen.

S 1 2 3 4 5 6 3 4 6 3 E ja, z. B. `ggT(2, 3)`.

S 1 2 3 4 6 3 4 5 6 3 E nicht feasible, da geteilt durch eins immer Modulo 0 ergibt, egal welche Zahl a oder b hat. Bei der ersten Schleifenwiederholung wird immer die innere If-Verzweigung genommen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/20_Black_White-Box-Test/Aufgabe_Groesster-gemeinsamer-Teiler.tex`

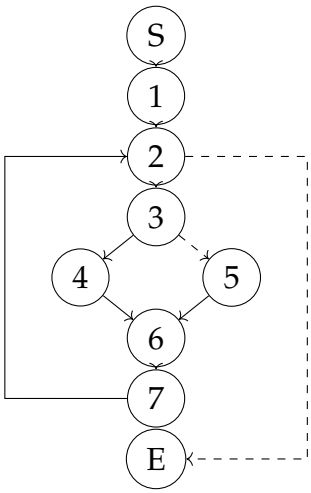
Übungsaufgabe „Methode „log()““ (Kontrollflussgraph, Überdeckbarkeit, C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage))

Kontrollflussgraph
Überdeckbarkeit

Gegeben sei folgende Methode und ihr Kontrollflussgraph:

```
int log(int a) {
  int x = a;
  int z = 0;
  while (x > 1) {
    if (x % 2 == 0) {
      z++;
      x /= 2;
    } else {
      x--;
    }
  }
  return z;
}
```

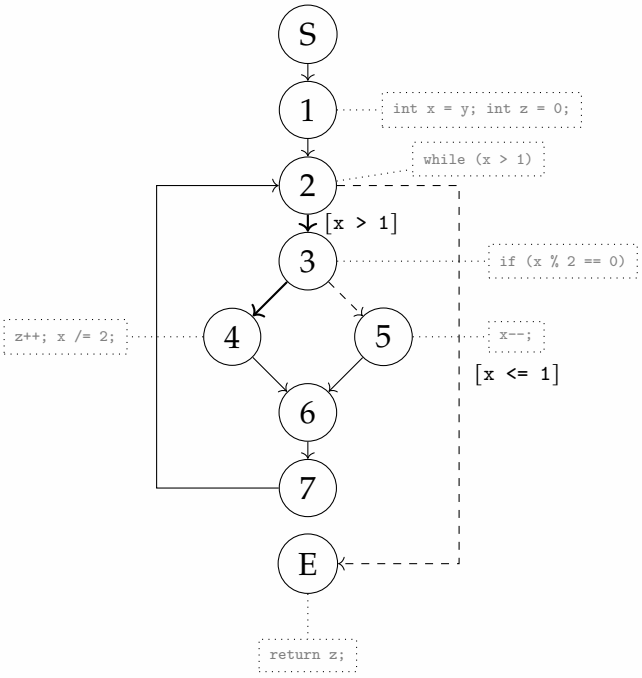
Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/aufgaben/sosy/ab_7/Aufgabe3.java`



(a) Begründen Sie, warum der Pfad Ⓢ - ① - ② - ③ - ⑤ - ⑥ - ⑦ - ② - Ⓔ infeasible (= nicht überdeckbar) ist, also weshalb es keine Eingabe gibt, unter der dieser Pfad durchlaufen werden kann.

Lösungsvorschlag

Damit dieser Pfad durchlaufen werden könnte, müsste die Eingabe a gleichzeitig 2 und ungerade sein.



(b) Geben Sie eine minimale Menge von Pfaden an, mit der eine vollständigen Schleifen-Inneres-Überdeckung erzielt werden kann, sowie gegebenenfalls zu jedem Pfad eine Eingabe, unter der dieser Pfad durchlaufen werden kann.

C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)

Lösungsvorschlag

ohne Schleifenausführung

- Ⓢ - ① - ② - Ⓔ (a=1)

boundary-Tests

- Ⓢ - ① - ② - ③ - ④ - ⑥ - ⑦ - ② - Ⓔ (a=2)
- Ⓢ - ① - ② - ③ - ⑤ - ⑥ - ⑦ - ② - Ⓔ (infeasible)

interior-Tests

- Ⓢ - ① - ② - ③ - ④ - ⑥ - ⑦ - ② - ③ - ④ - ⑥ - ⑦ - ② - Ⓔ (a=4)
- Ⓢ - ① - ② - ③ - ④ - ⑥ - ⑦ - ② - ③ - ⑤ - ⑥ - ⑦ - ② - Ⓔ (a=6)
- Ⓢ - ① - ② - ③ - ⑤ - ⑥ - ⑦ - ② - ③ - ④ - ⑥ - ⑦ - ② - Ⓔ (a=3)
- Ⓢ - ① - ② - ③ - ⑤ - ⑥ - ⑦ - ② - ③ - ⑤ - ⑥ - ⑦ - ② - Ⓔ (infeasible)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/40_SOSY/05_Testen/20_Black_White-Box-Test/Aufgabe_Methode-log.tex`

Teil IV

Theoretische Informatik (THEO)

Reguläre Sprache

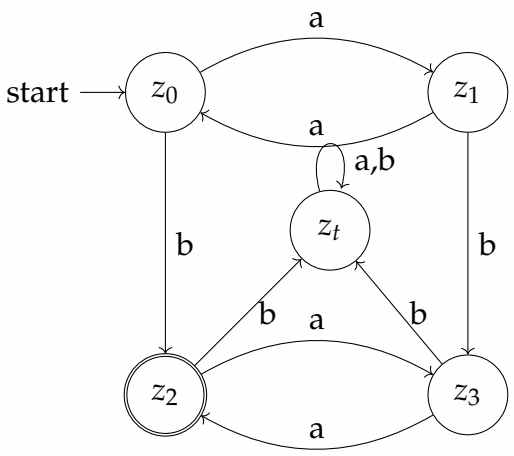
Examensaufgabe „Alphabet ab“ (46115-2010-F.T2-A1)

Aufgabe 1

(a) Gegeben ist die folgende Sprache L1 über dem Alphabet $\Sigma = \{a, b\}$:
$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und } b \text{ kommt in } w \text{ genau einmal vor}\}$.

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache $L1$ akzeptiert.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahfqe3eb7

(ii) Geben Sie einen regulären Ausdruck an, der die Sprache $L1$ beschreibt.

Lösungsvorschlag

(aa)*(b|aba)(aa)*

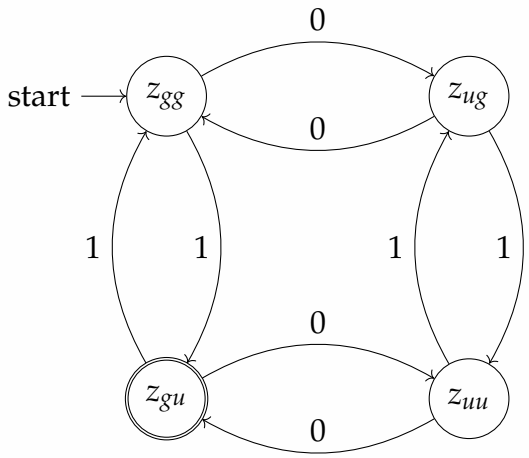
(b) Die folgende Sprache $L2$ ist eine Erweiterung von $L1$:
$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und die Anzahl der } b \text{ in } w \text{ ist ungerade}\}$.

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache $L2$ akzeptiert.

Lösungsvorschlag

a

gu = gerade Anzahl a’s, ungerade Anzahl b’s
ug = ungerade Anzahl a’s, gerade Anzahl b’s



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af0vcjys9

[a] `https://www.informatik.uni-hamburg.de/TGI/lehre/vl/SS14/FGI1/Folien/fgi1_v2_handout.pdf`

(ii) Geben Sie eine rechtslineare Grammatik an, die die Sprache $L2$ erzeugt.

Lösungsvorschlag

$$
\mathrm{P} = \left\{
\begin{aligned}
A &\to aB \mid bD \mid b \\
B &\to bC \mid aA \\
C &\to aD \mid a \mid bB \\
D &\to bA \mid aC
\end{aligned}
\right\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahfqe3eb7

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2010/03/Thema-2/Aufgabe-1.tex`

Examensaufgabe „Sprache abc“ (46115-2015-F.T1-A1)

Reguläre Sprache

Aufgabe 1

Gegeben sei die Sprache L. L besteht aus der Menge aller Worte über dem Alphabet $\{a, b, c\}$, die mit a beginnen und mit b enden und die nie zwei aufeinander folgende c's enthalten.

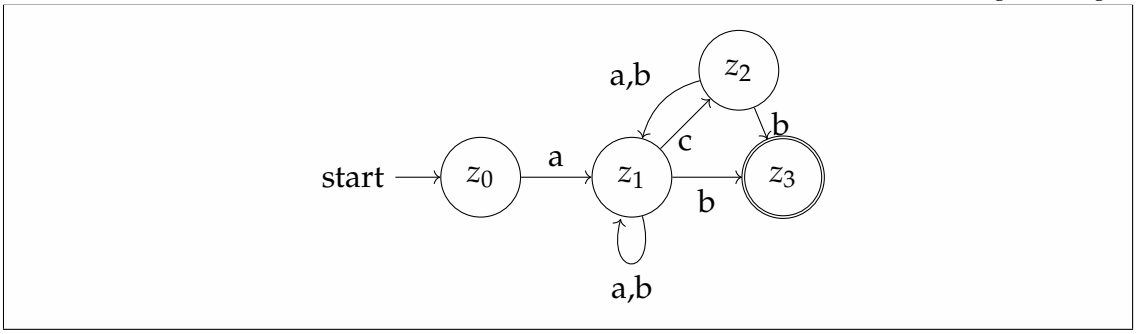
(a) Geben Sie einen regulären Ausdruck für L an.

Lösungsvorschlag

```
a(c?[ab]+)*(cb|b)
a(c(a|b)|(a|b))*(cb|b)
```

(b) Geben Sie einen vollständigen deterministischen endlichen Automaten für L an.

Lösungsvorschlag

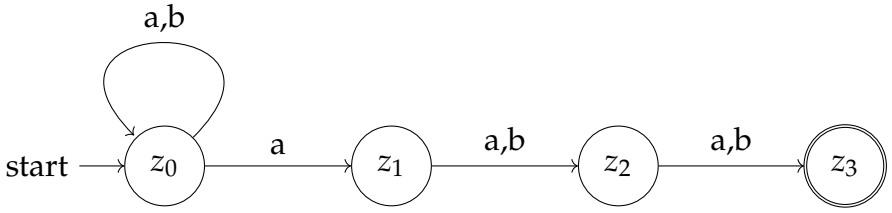


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2015/03/Thema-1/Aufgabe-1.tex

Examensaufgabe „Alphabet ab, vorvorletztes Zeichen a“ (46115-2016-H.T1-A1)

Potenzmengenalgorithmus

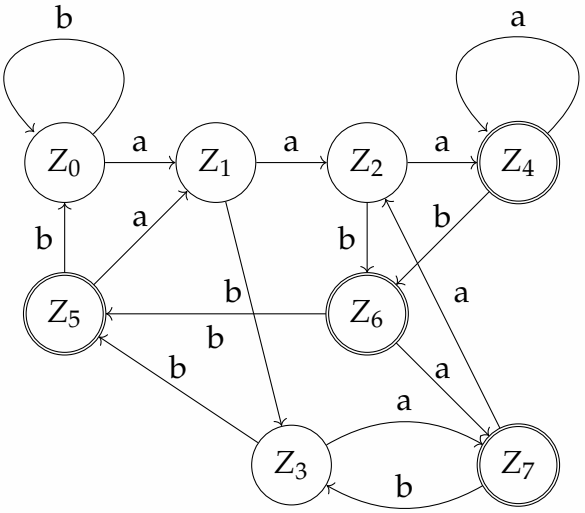
(a) Konstruieren Sie aus dem NEA mit der Potenzmengenkonstruktion einen (deterministischen) EA, der dieselbe Sprache akzeptiert.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiqxdazuw

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
Z_0	$Z_0\{z_0\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
Z_1	$Z_1\{z_0, z_1\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$
Z_2	$Z_2\{z_0, z_1, z_2\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
Z_3	$Z_3\{z_0, z_2\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
Z_4	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
Z_5	$Z_5\{z_0, z_3\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
Z_6	$Z_6\{z_0, z_2, z_3\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
Z_7	$Z_7\{z_0, z_1, z_3\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein

Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiqnk06c9

(b) Beschreiben Sie möglichst einfach, welche Sprachen von den folgenden regulären Ausdrücken beschrieben werden:

(i) $(a|b)^*a$

Lösungsvorschlag

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enden auf a.

(ii) $(a|b)^*a(a|b)^*a(a|b)^*$

Lösungsvorschlag

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enthalten mindestens 2 a's.

(iii) $(a|b)^*a(bb)^*a(a|b)^*$

Lösungsvorschlag

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enthalten mindestens 2 a's, die von einer geradzahligen Anzahl von b's getrennt sind.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/09/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Reguläre Sprachen)“ (46115-2019-H.T1-A1)

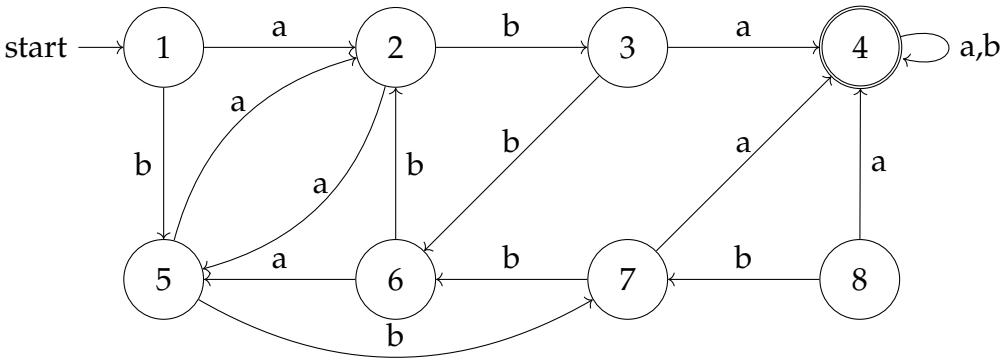
Reguläre Sprache

(a) Geben Sie einen regulären Ausdruck für die Sprache über dem Alphabet $\{a, b\}$ an, die genau alle Wörter enthält, die eine gerade Anzahl a's haben.

Lösungsvorschlag

$b^*(ab^*ab^*)^*$

(b) Sei A der folgende DEA über dem Alphabet $\{a, b\}$:



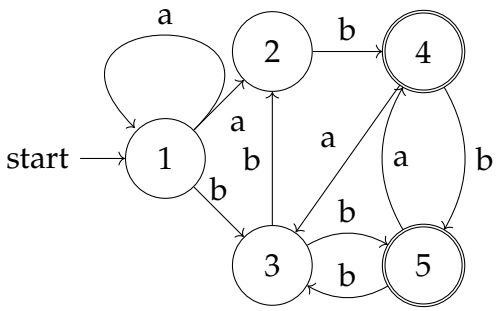
Führen Sie den Minimierungsalgorithmus für A durch und geben Sie den minimalen äquivalenten DEA für $L(A)$ als Zeichnung an.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Komplemetieren eines NEA“ (46115-2019-H.T2-A1)

Potenzmengenalgorithmus
Nichtdeterministisch endlicher Automat (NEA)
Deterministisch endlicher Automat (DEA)

Es sei der nichtdeterministische endliche Automat $A = (\{1,2,3,4,5\}, \{a,b\}, \delta, \{4,5\}, 1)$ gegeben, wobei δ durch folgenden Zeichnung beschrieben ist.

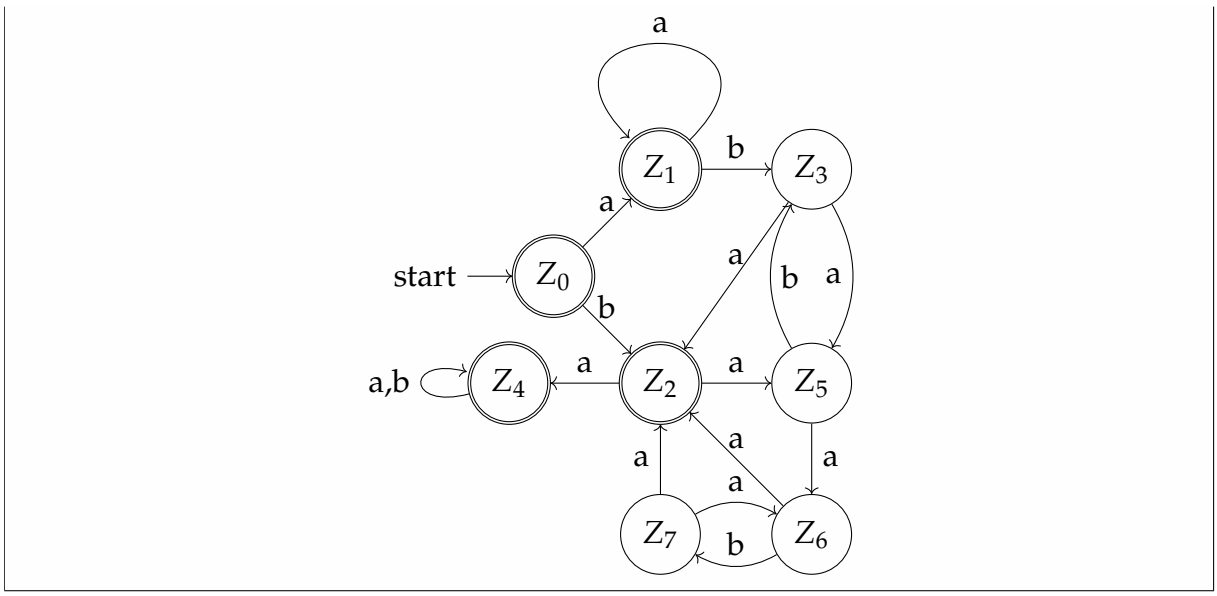


Konstruieren Sie nachvollziehbar einen deterministischen endlichen Automaten A' , der das Komplement von $L(A)$ akzeptiert!

Lösungsvorschlag

Zuerst mit Hilfe der Potenzmengenkonstruktion einen deterministischen endlichen Automaten erstellen und dann die Zustände mit den Endzuständen tauschen.

Name	Zustandsmenge	Eingabe a	Eingabe b
Z_0	$Z_0\{1\}$	$Z_1\{1,2\}$	$Z_2\{3\}$
Z_1	$Z_1\{1,2\}$	$Z_1\{1,2\}$	$Z_3\{3,4\}$
Z_2	$Z_2\{3\}$	$Z_4\{\}$	$Z_5\{2,5\}$
Z_3	$Z_3\{3,4\}$	$Z_2\{3\}$	$Z_5\{2,5\}$
Z_4	$Z_4\{\}$	$Z_4\{\}$	$Z_4\{\}$
Z_5	$Z_5\{2,5\}$	$Z_6\{4\}$	$Z_3\{3,4\}$
Z_6	$Z_6\{4\}$	$Z_2\{3\}$	$Z_7\{5\}$
Z_7	$Z_7\{5\}$	$Z_6\{4\}$	$Z_2\{3\}$



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-2/Aufgabe-1.tex`

Examensaufgabe „Rechtslineare Grammatik“ (46115-2019-H.T2-A2)

Reguläre Sprache

Gegeben ist die rechtslineare Grammatik $G = (\{a, b\}, \{S, A, B, C, D\}, S, P)$ mit

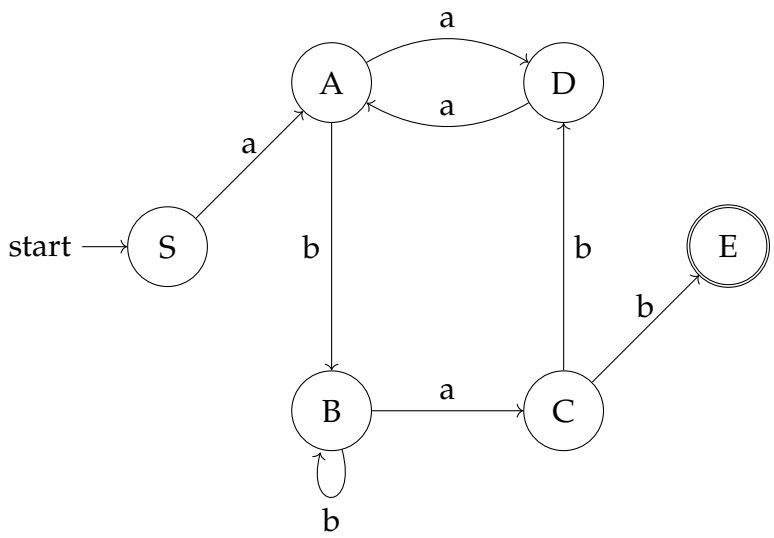
$$
\mathrm{P} = \left\{ \begin{array}{r} S \to aA \\ A \to bB \\ A \to aD \\ B \to aC \\ B \to bB \\ C \to bD \\ C \to b \\ D \to aA \end{array} \right\}
$$

. Sei L die von G erzeugte Sprache.

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpkv4ansc

(a) Zeichnen Sie einen nichtdeterministischen endlichen Automaten, der L akzeptiert!

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahh979eqz

(b) Konstruieren Sie auf nachvollziehbare Weise einen regulären Ausdruck α mit $L(\alpha) = L$!

Lösungsvorschlag

$(ab + ab|(a|b+) + ab + ab)$ (von Flaci automatisch konvertiert)

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-2/Aufgabe-2.tex`

Examensaufgabe „Reguläre Sprache“ (46115-2020-F.T1-A1)

Reguläre Sprache
Reguläre Ausdrücke

(a) Betrachten Sie die formale Sprache $L \subseteq \{0,1\}^*$ aller Wörter, die 01 oder 110 als Teilwort enthalten.

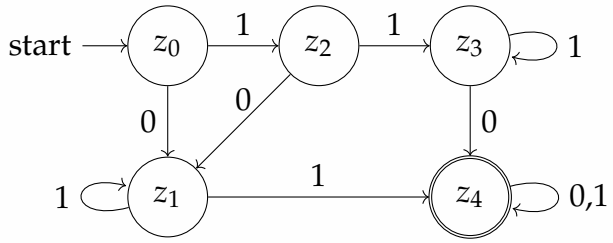
Geben Sie einen regulären Ausdruck für die Sprache L an.

Lösungsvorschlag

(0|1)*(01|110)(0|1)*

(b) Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L aus Teilaufgabe (a) akzeptiert. (Hinweis: es werden nicht mehr als 6 Zustände benötigt.)

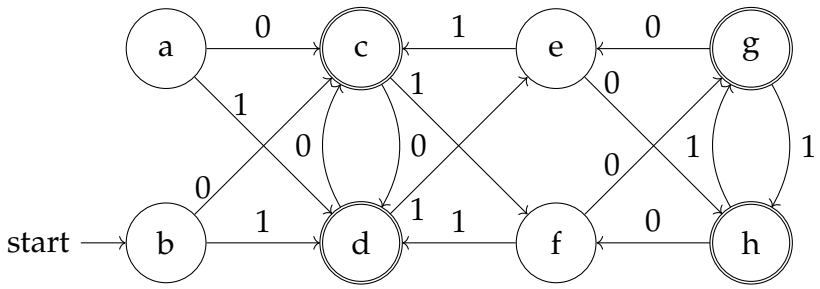
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A54gek0vz

(c) Minimieren Sie den folgenden deterministischen endlichen Automaten:

Machen Sie dabei Ihren Rechenweg deutlich!



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajpw4j73w

Lösungsvorschlag

a	∅	∅	∅	∅	∅	∅	∅	∅
b		∅	∅	∅	∅	∅	∅	∅
c	x_1	x_1	∅	∅	∅	∅	∅	∅
d	x_1	x_1		∅	∅	∅	∅	∅
e	x_2	x_2	x_1	x_1	∅	∅	∅	∅
f	x_3	x_3	x_1	x_1		∅	∅	∅
g	x_1	x_1	x_2	x_2	x_1	x_1	∅	∅
h	x_1	x_1	x_2	x_2	x_1	x_1		∅
	a	b	c	d	e	f	g	h

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

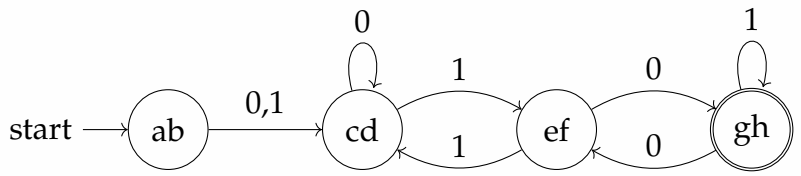
x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Die Zustandpaare werden aufsteigend sortiert notiert.

Übergangstabelle

Zustandspaar	**0**	**1**
(a, b)	(c, c)	(d, d)
(a, e)	(c, h) x_2	(c, d)
(a, f)	(c, g) x_3	(d, d)
(b, e)	(c, h) x_2	(c, d)
(b, f)	(c, g) x_3	(d, g)
(c, d)	**(c, d)**	**(e, f)**
(c, g)	(d, e) x_2	(e, f)
(c, h)	(d, f) x_2	(f, f)
(d, g)	(c, e) x_2	(e, e)
(d, h)	(c, f) x_2	(e, f)
(e, f)	**(g, h)**	**(c, d)**
(g, h)	**(e, f)**	**(g, h)**



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arzvh5kyz

(d) Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!

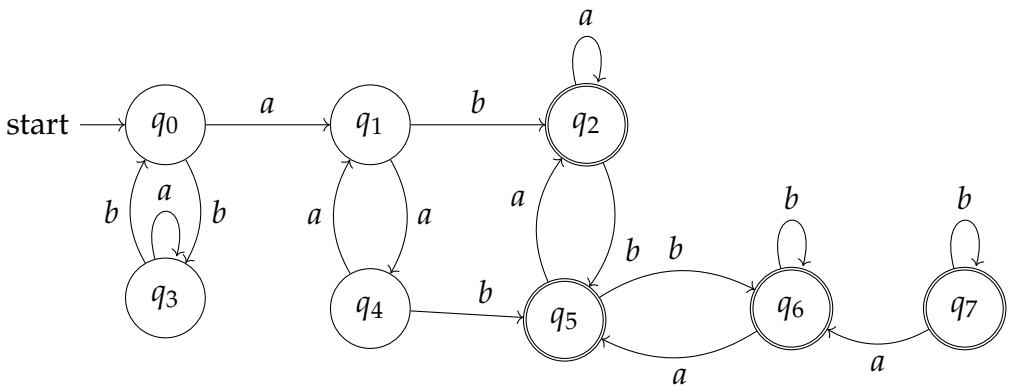
„Zu jeder regulären Sprache L über dem Alphabet Σ gibt es eine Sprache $L' \subseteq \Sigma^*$, die L enthält ($\eth L \subseteq L$) und nicht regulär ist.“

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2020/03/Thema-1/Aufgabe-1.tex`

Minimierungsalgorithmus

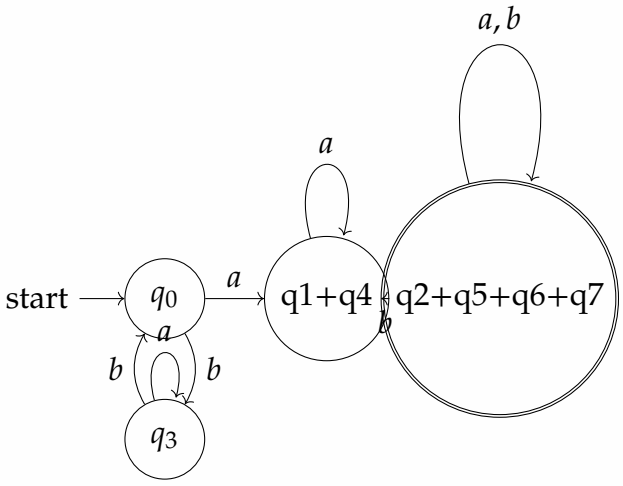
Examensaufgabe „Minimierung von Endlichen Automaten“ (46115-2021-F.T1-TA1-A3)

Betrachten Sie den unten gezeigten deterministischen endlichen Automaten, der Worte über dem Alphabet X = a,b verarbeitet. Bestimmen Sie den dazugehörigen Minimalautomaten, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine mini- male Anzahl an Zuständen benutzt. Erläutern Sie Ihre Berechnung, indem Sie z. B. eine Minimierungstabelle angeben.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5v10or9

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apkyuoo1g

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

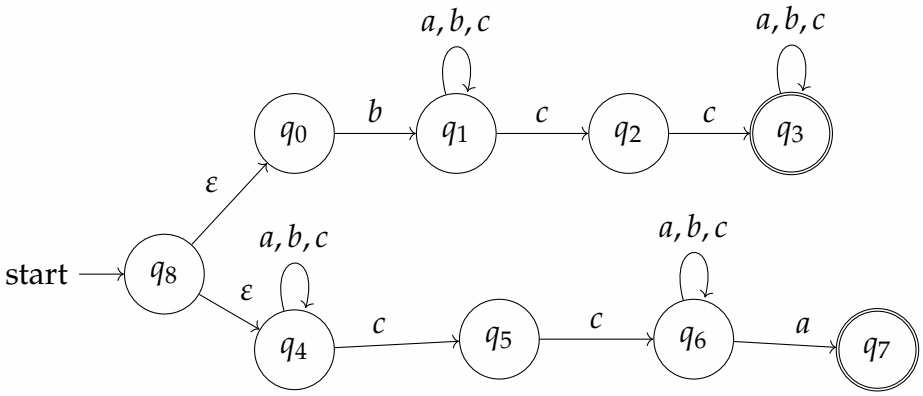
Examensaufgabe „Alphabet abc“ (46115-2021-F.T2-TA1-A1)

Reguläre Sprache

(a) Betrachten Sie die formale Sprache $L \subseteq \{a, b, c\}^*$: aller Wörter, die entweder mit b beginnen oder mit a enden (aber nicht beides gleichzeitig) und das Teilwort cc enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L akzeptiert. (Hinweis: Es werden weniger als 10 Zustände benötigt.)

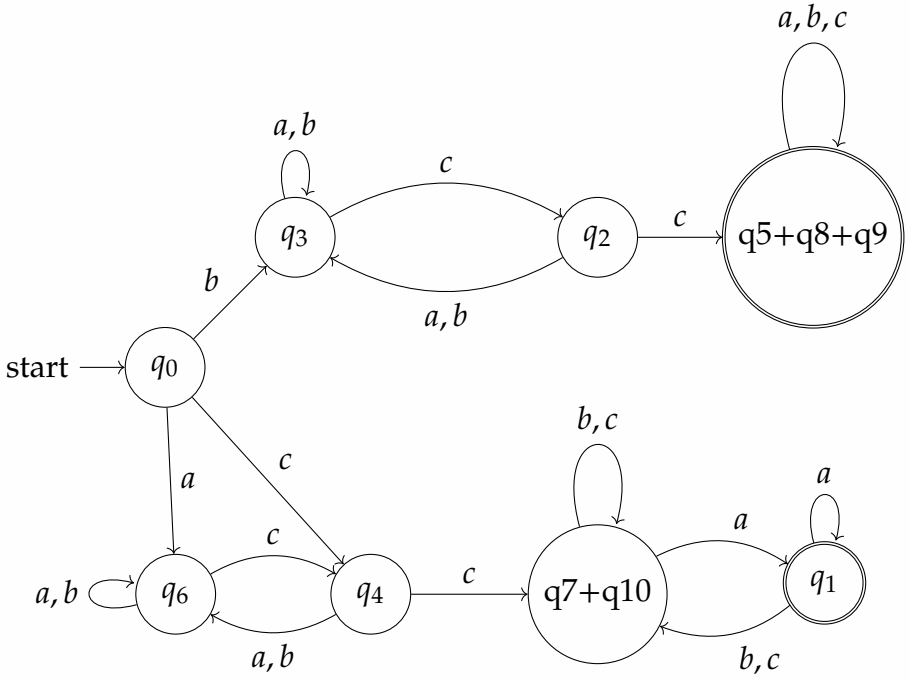
Lösungsvorschlag

NEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3pvv7ha

konvertierter DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ai89m0txw

(b) Ist die folgende Aussage richtig? Begründen Sie Ihre Antwort.

„Jede Teilsprache einer regulären Sprache ist regulär, d. h. für ein Alphabet und formale Sprachen $L' \subseteq L \subseteq \Sigma^*$ ist L' regulär, falls L regulär ist.“

Lösungsvorschlag

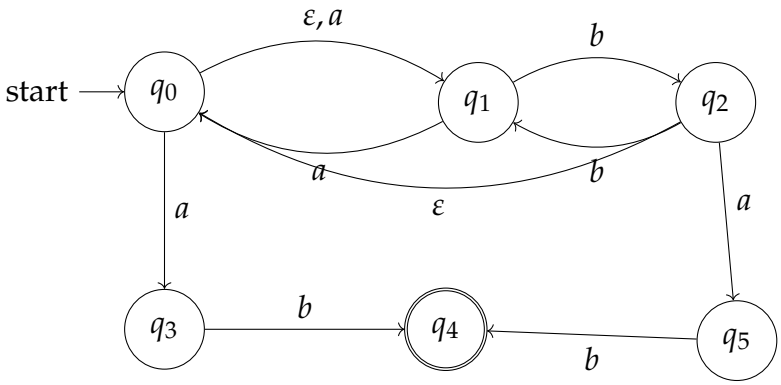
Ja. Reguläre Sprachen sind abgeschlossen unter dem Komplement und der Vereinigung.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „NEA ab“ (46115-2021-F.T2-TA1-A2)

Reguläre Sprache

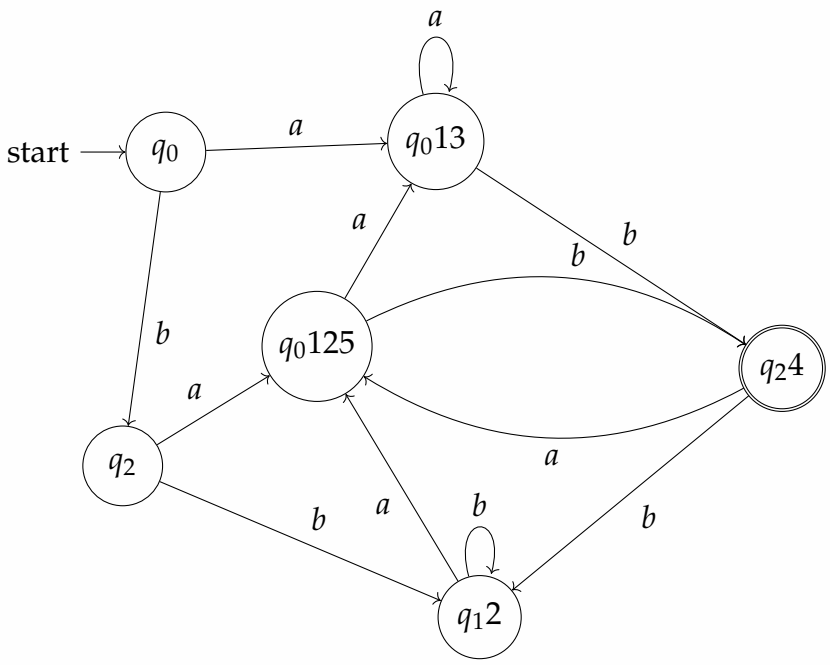
Gegeben sei der folgende e-nichtdeterministische endliche Automat A über dem Alphabet



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A54bbh2iz

(a) Konstruieren Sie einen deterministischen endlichen Automaten für A. Wenden Sie dafür die Potenzmengenkonstruktion an.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arnqcysfc

(b) Beschreiben Sie die von A akzeptierte Sprache L(A) mit eigenen Worten und so einfach wie möglich.

Lösungsvorschlag

Endet auf ab, Präfix beliebig auch leer

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „L1, L2, L3 regulär oder kontextfrei“ (46115-2021-F.T2-TA1-A3)

Reguläre Sprache
Kontextfreie Sprache

Sei $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ die Menge aller natürlichen Zahlen mit 0. Betrachten Sie die folgenden Sprachen.

(a) $L_1 = \{ a^{3n} b^{2n} a^n \mid n \in \mathbb{N}_0 \}$

Lösungsvorschlag

nicht kontextfrei

(b) $L_2 = \{ a^{3n} a^{2n} b^n \mid n \in \mathbb{N}_0 \}$

Lösungsvorschlag

kontextfrei.

Der Ausdruck lässt umformen in: $L_2 = \{ a^{5n} b^n \mid n \in \mathbb{N}_0 \}$

$$\mathrm{P} = \Big\{ \quad S \to aaaaaSb \mid \varepsilon \quad \Big\}$$

(c) $L_3 = \{ (ab)^n a (ba)^n b (ab)^n \mid n \in \mathbb{N}_0 \}$

Lösungsvorschlag

nicht kontextfrei

Geben Sie jeweils an, ob L_1, L_2 und L_3 kontextfrei und ob L_1, L_2 und L_3 regulär sind. Beweisen Sie Ihre Behauptung und ordnen Sie jede Sprache in die kleinstmögliche Klasse (regulär, kontextfrei, nicht kontextfrei) ein. Für eine Einordnung in kontextfrei zeigen Sie also, dass die Sprache kontextfrei und nicht regulär ist.

Erfolgt ein Beweis durch Angabe eines Automaten, so ist eine klare Beschreibung der Funktionsweise des Automaten und der Bedeutung der Zustände erforderlich. Erfolgt der Beweis durch Angabe eines regulären Ausdruckes, so ist eine intuitive Beschreibung erforderlich. Wird der Beweis durch die Angabe einer Grammatik geführt, so ist die Bedeutung der Variablen zu erläutern.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-3.tex

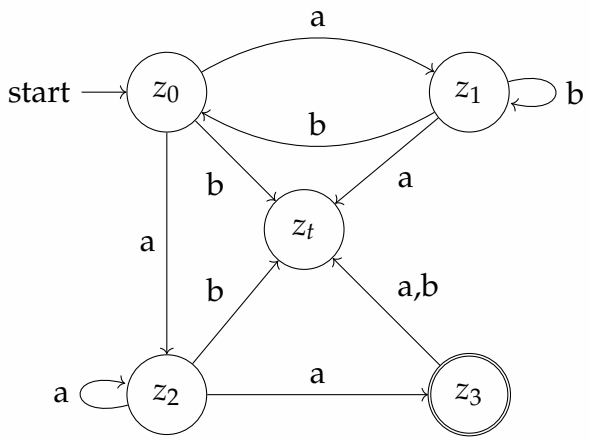
Examensaufgabe „Reguläre Sprache“ (66115-2007-H.T2-A1)

Reguläre Sprache

Gegeben sei der nichtdeterministische endliche Automat M mit dem Alphabet $\Sigma = \{a, b\}$, der Zustandsmenge $\{z_0, z_1, z_2, z_3\}$, Anfangszustand z_0, Endzustand $\{z_3\}$ und der Überführungsfunktion δ mit:

$$\begin{aligned}
&\delta(z_0, a) = \{z_1, z_2\},\\
&\delta(z_1, b) = \{z_0, z_1\},\\
&\delta(z_2, a) = \{z_2, z_3\},\\
&\delta(z_0, b) = \delta(z_1, a) = \delta(z_2, b) = \delta(z_3, a) = \delta(z_3, b) = \varnothing
\end{aligned}$$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Afybo27zc

$L(M)$ sei die von M akzeptierte Sprache.

(a) Gelten folgende Aussagen?

(i) Es gibt Zeichenreihen in $L(M)$, die genauso viele a's enthalten wie b's.

Lösungsvorschlag

Ja, zum Beispiel das Wort *abbbaa* oder *abbbbaaa*. Mit der Überführungsfunktion $\delta(z_1, b) = \{z_1\}$ können beliebig viele b's akzeptiert werden, sodass die Anzahl von a's und b's ausgeglichen werden kann.

(ii) Jede Zeichenreihe in $L(M)$, die mindestens vier b's enthält, enthält auch mindestens vier a's.

Lösungsvorschlag

Nein, z. b. das Wort *abbbbaa* wird akzeptiert. Ein Wort muss nur mindestens drei a's enthalten. Mit der Überführungsfunktion $\delta(z_1, b) = \{z_1\}$

> können aber beliebig viele b's akzeptiert werden.

Reguläre Grammatik
Reguläre Ausdrücke
Potenzmengenalgorithmus

Begründen Sie Ihre Antworten.

(b) Geben Sie eine reguläre (Typ-3-)Grammatik an, die $L(M)$ erzeugt.

(c) Beschreiben Sie $L(M)$ durch einen regulären Ausdruck.

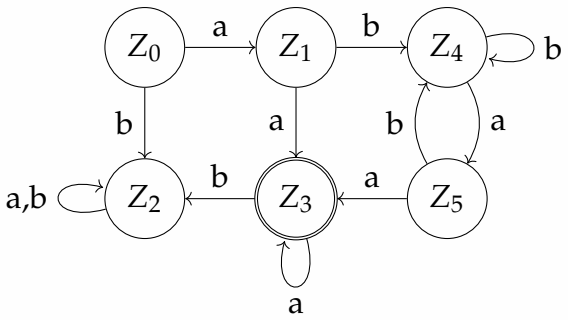
Lösungsvorschlag

(ab+)*aa+

(d) Konstruieren Sie aus M mit der Potenzmengen-Konstruktion (und entsprechender Begründung) einen deterministischen endlichen Automaten, der $L(M)$ akzeptiert.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
Z_0	Z_0 $\{z_0\}$	Z_1 $\{z_1, z_2\}$	Z_2 $\{z_t\}$
Z_1	Z_1 $\{z_1, z_2\}$	Z_3 $\{z_2, z_3, z_t\}$	Z_4 $\{z_0, z_1, z_t\}$
Z_2	Z_2 $\{z_t\}$	Z_2 $\{z_t\}$	Z_2 $\{z_t\}$
Z_3	Z_3 $\{z_2, z_3, z_t\}$	Z_3 $\{z_2, z_3, z_t\}$	Z_2 $\{z_t\}$
Z_4	Z_4 $\{z_0, z_1, z_t\}$	Z_5 $\{z_1, z_2, z_t\}$	Z_4 $\{z_0, z_1, z_t\}$
Z_5	Z_5 $\{z_1, z_2, z_t\}$	Z_3 $\{z_2, z_3, z_t\}$	Z_4 $\{z_0, z_1, z_t\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aig9i4u7z

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2007/09/Thema-2/Aufgabe-1.tex

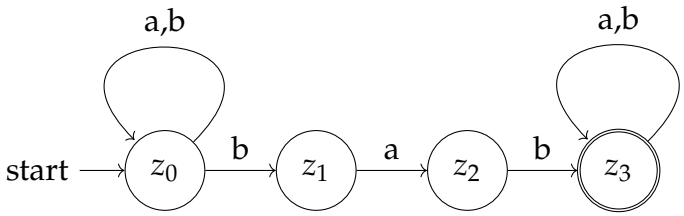
Examensaufgabe „NEA und Minimalisierung“ (66115-2012-H.T1-A1)

Potenzmengenalgorithmus

Wir fixieren das Alphabet $\Sigma = \{a, b\}$ und definieren $L \subseteq \Sigma^*$ durch

$$L = \{ w \mid \text{in } w \text{ kommt das Teilwort } \texttt{bab} \text{ vor} \}$$

z. B. ist $\texttt{babaabb} \in L$, aber $\texttt{baabaabb} \notin L$. Der folgende nichtdeterministische Automat A erkennt L:

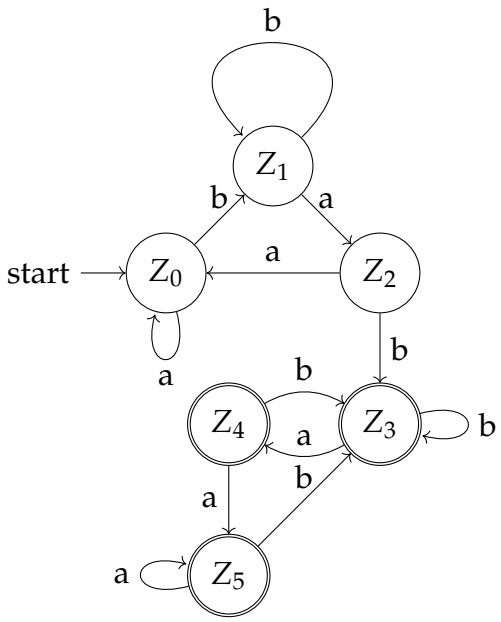


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af75jwj3r

(a) Wenden Sie die Potenzmengenkonstruktion auf den Automaten an und geben Sie den resultierenden deterministischen Automaten an. Nicht erreichbare Zustände sollen nicht dargestellt werden.

Lösungsvorschlag

Zustandsmenge	Eingabe a	Eingabe b
Z_0 $\{z_0\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$
Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_2\}$	Z_1 $\{z_0, z_1\}$
Z_2 $\{z_0, z_2\}$	Z_0 $\{z_0\}$	Z_3 $\{z_0, z_1, z_3\}$
Z_3 $\{z_0, z_1, z_3\}$	Z_4 $\{z_0, z_2, z_3\}$	Z_3 $\{z_0, z_1, z_3\}$
Z_4 $\{z_0, z_2, z_3\}$	Z_5 $\{z_0, z_3\}$	Z_3 $\{z_0, z_1, z_3\}$
Z_5 $\{z_0, z_3\}$	Z_5 $\{z_0, z_3\}$	Z_3 $\{z_0, z_1, z_3\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro483e89

(b) Konstruieren Sie aus dem so erhaltenen deterministischen Automaten den Minimalautomaten für L. Beschreiben Sie dabei die Arbeitsschritte des verwendeten Algorithmus in nachvollziehbarer Weise.

Lösungsvorschlag

z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_1	x_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_2	x_2	x_2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_3	x_1	x_1	x_1	$\varnothing$	$\varnothing$	$\varnothing$
z_4	x_1	x_1	x_1		$\varnothing$	$\varnothing$
z_5	x_1	x_1	x_1			$\varnothing$
	z_0	z_1	z_2	z_3	z_4	z_5

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

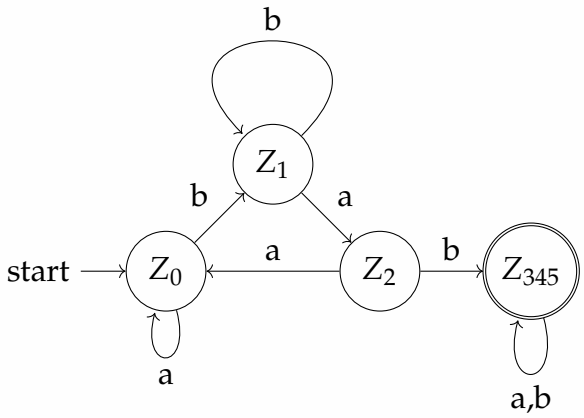
x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	**a**	**b**
(z_0, z_1)	$(z_0, z_2)\ x_3$	(z_1, z_1)
(z_0, z_2)	(z_0, z_0)	$(z_1, z_3)\ x_2$
(z_1, z_2)	$(z_2, z_0)\ x_3$	$(z_1, z_3)\ x_2$
(z_3, z_4)	(z_4, z_5)	(z_3, z_3)
(z_3, z_5)	(z_4, z_5)	(z_3, z_3)
(z_4, z_5)	(z_5, z_5)	(z_3, z_3)



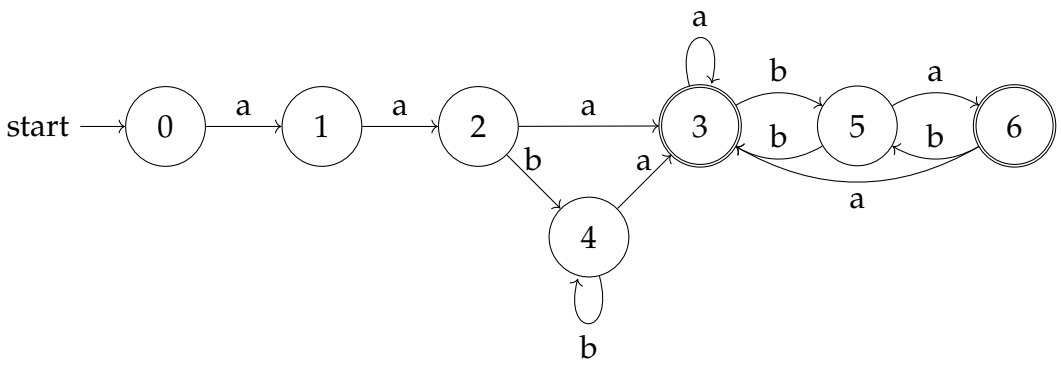
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3joif5z

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/09/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Minimierung DFA“ (66115-2013-H.T2-A3)

Minimierungsalgorithmus

Minimieren Sie den folgenden deterministischen Automaten mit den Zuständen $\{0, 1, 2, 3, 4, 5, 6\}$, dem Startzustand 0 und den Endzuständen $\{3, 6\}$. Geben Sie z. B. durch die Bezeichnung an, welche Zustände zusammengefasst wurden.



Lösungsvorschlag

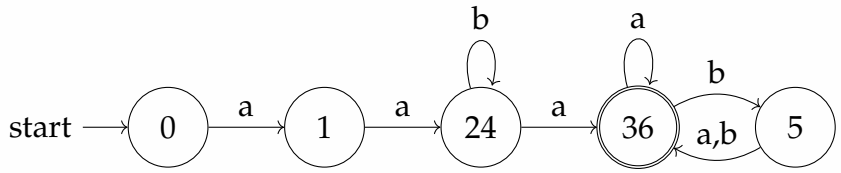
0	∅	∅	∅	∅	∅	∅	∅
1	x_3	∅	∅	∅	∅	∅	∅
2	x_2	x_2	∅	∅	∅	∅	∅
3	x_1	x_1	x_1	∅	∅	∅	∅
4	x_2	x_2		x_1	∅	∅	∅
5	x_2	x_2	x_2	x_1	x_2	∅	∅
6	x_1	x_1	x_1		x_1	x_1	∅
	0	1	2	3	4	5	6

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.
x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.
x_3 In weiteren Iterationen markierte Zustände.
x_4 ...

Übergangstabelle

Zustandspaar	**a**	**b**
(0, 1)	(1, 2) x_3	(T, T)
(0, 2)	(1, 3) x_2	(T, 4)
(0, 4)	(1, 3) x_2	(T, 4)
(0, 5)	(1, 6) x_2	(T, 3)
(1, 2)	(2, 3) x_2	(T, 4)
(1, 4)	(2, 3) x_2	(T, 4)
(1, 5)	(2, 6) x_2	(T, 3)
(2, 4)	(3, 3)	(4, 4)
(2, 5)	(3, 6)	(3, 4) x_2
(3, 6)	(3, 3)	(5, 5)
(4, 5)	(3, 6)	(3, 4) x_2

T = Trap-Zustand = Falle



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/09/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Alphabet „01“ Anzahl Unterschied höchstes 3“ (66115, 2015-F.T1-A1)

Reguläre Sprache
Pumping-Lemma (Reguläre Sprache)

Die Sprache L über den Alphabet $\Sigma = \{0, 1\}$ enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist $w \in L$ genau dann, wenn für alle u, v mit $w = uv$ gilt $||u|_0 - |u|_1| \leq 3$. Erinnerung: $|w|_a$ bezeichnet die Zahl der a's im Wort w.

(a) Sei $A = (Q, \Sigma, \delta, q_0, E)$ ein deterministischer endlicher Automat für L. Es sei $w_1 = 111$, $w_2 = 11$, $w_3 = 1$, $w_4 = \varepsilon$, $w_5 = 0$, $w_6 = 00$, $w_7 = 000$. Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihr Argumentation schlüssig und vollständig auf.

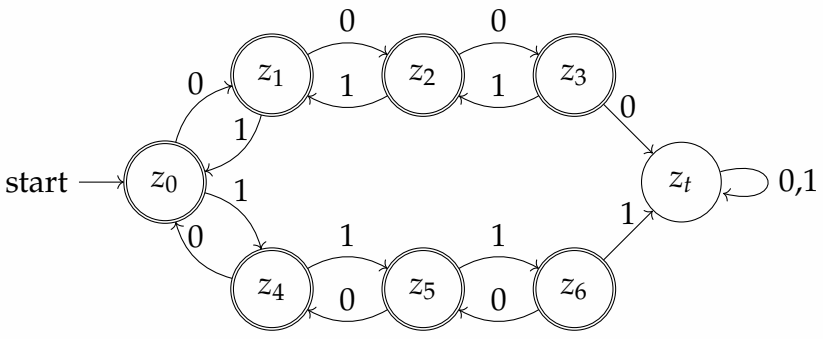
Lösungsvorschlag

Ein deterministischer endlicher Automat hat keinen zusätzlichen Speicher zur Verfügung, in dem die Anzahl der bisher vorkommenden Einsen und Nullen gespeichert werden könnte. Ein deterministischer endlicher Automat kann die von der Sprache benötigten Anzahl an Einsen und Nullen nur in Form von Zustanden speichern. Um die Anzahl von 3 Einsen bzw. 3 Nullen zu speichern, sind also 6 Zustände nötig.

Die Wörter 01 oder 0011 oder 0101 etc. haben eine Differenz von 0, wenn die Anzahl an Nullen und Einsen abzogen wird. Um auch diese Wörter darstellen zu können, ist mindestens ein weiterer Zustand nötig.

(b) Begründen Sie, dass L regulär ist.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ait6va31c

(c) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden „Beweis“ dafür an: Wäre L regulär, so sei n eine entsprechende Pumping-Zahl. Nun ist $w = (01)^n \in L$. Zerlegt man nun $w = uxv$, wobei $u = 0$, $x = 1$, $v = (01)^{n-1}$, so ist zum Beispiel $ux^5v \notin L$, denn es ist $ux^5v = 01111101010101....$ Legen Sie genau dar, an welcher Stelle dieser „Beweis“ fehlerhaft ist.

Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(i) $|v| \geq 1$ (Das Wort v ist nicht leer.)

(ii) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(iii) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^iw \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iw in der Sprache L)

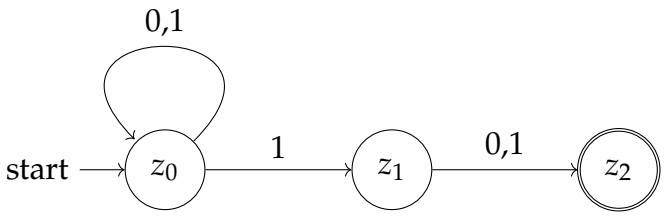
Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

Lösungsvorschlag

Das Wort $(01)^n$ wurde falsch zerlegt. Für die Pumping-Zahl $n = 3$ gibt es sehr wohl eine Zerlegung, die beim Aufpumpen regulär ist, also: $\omega = 010101$ ($u = 01$, $x = 01$ und $v = 01$). $ux^5v = 01010101010101 \in L$. Es gibt also eine Zerlegung, die beim Aufpumpen die 3 Pumping-Lemma-Eigenschaften erfüllt. Daher kann man das Pumping-Lemma so nicht widerlegt werden, indem man ein einziges Gegenbeispiel gibt.

(d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache $L_2 = \Sigma^*1\Sigma$ aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nichtdeterministischen Automaten mit nur drei Zuständen an, L_2 erkennt.

Lösungsvorschlag



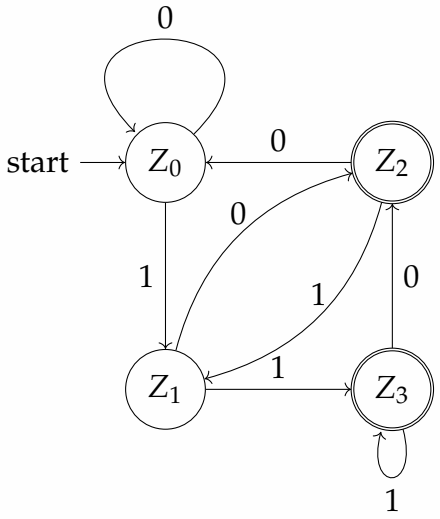
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwzjufbg

(e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für L_2 also mindestens haben?

Lösungsvorschlag

Potenzmengenkonstruktion

Name	Zustandsmenge	Eingabe 0	Eingabe 1
Z_0	$Z_0\ \{z_0\}$	$Z_0\ \{z_0\}$	$Z_1\ \{z_0, z_1\}$
Z_1	$Z_1\ \{z_0, z_1\}$	$Z_2\ \{z_0, z_2\}$	$Z_3\ \{z_0, z_1, z_2\}$
Z_2	$Z_2\ \{z_0, z_2\}$	$Z_0\ \{z_0\}$	$Z_1\ \{z_0, z_1\}$
Z_3	$Z_3\ \{z_0, z_1, z_2\}$	$Z_2\ \{z_0, z_2\}$	$Z_3\ \{z_0, z_1, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajfcofpb9

Minimierungsalgorithmus

Z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
Z_1	x_2	$\varnothing$	$\varnothing$	$\varnothing$
Z_2	x_1	x_1	$\varnothing$	$\varnothing$
Z_3	x_1	x_1	x_2	$\varnothing$
	Z_0	Z_1	Z_2	Z_3

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	**0**	**1**
(Z_0, Z_1)	(Z_0, Z_2) x_2	(Z_1, Z_3) x_2
(Z_2, Z_3)	(Z_0, Z_1)	(Z_1, Z_3) x_2

Wie aus der oben stehenden Tabelle abzulesen ist, gibt es keine äquivalenten Zustände. Der Automat kann nicht minimiert werden. Er ist bereits minimal.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Reguläre Sprachen“ (66115-2016-F.T1-A1)

Reguläre Sprache

(a) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache $L_1 = \{ a_1, a_2, \ldots, a_n \mid n \geq 3, a_i \in \{a, b\} \text{ für alle } i = 1, \ldots, n \text{ und } a_1 \geq a_n \}$ an.

Lösungsvorschlag

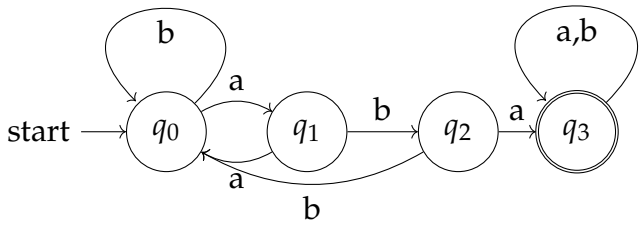
```
((a(a|b)+b)|(b(a|b)+a))
```

(b) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache $L_2 = \{ w \in \{a, b\}^* \mid w \text{ enthält genau ein } b \text{ und ist von ungerader Länge} \}$ an.

Lösungsvorschlag

```
(aa)*(b|aba)(aa)*
```

(c) Beschreiben Sie die Sprache des folgenden Automaten A_1, möglichst einfach und präzise in ihren eigenen Worten.

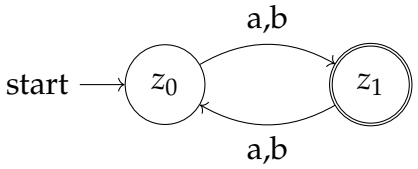


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arz003ccg

Lösungsvorschlag

Die Sprache enthält das Teilwort *aba*

(d) Betrachten Sie folgenden Automaten A_2:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ap9qbkumc

Im Original sind die Zustände mit q_x benannt. Damit wir die Schnittmenge besser bilden können, wird hier z_x verwendet.

Konstruieren Sie einen endlichen Automaten, der die Schnittmenge der Sprachen $L(A_1)$ und $L(A_2)$ akzeptiert.

Lösungsvorschlag

A_1

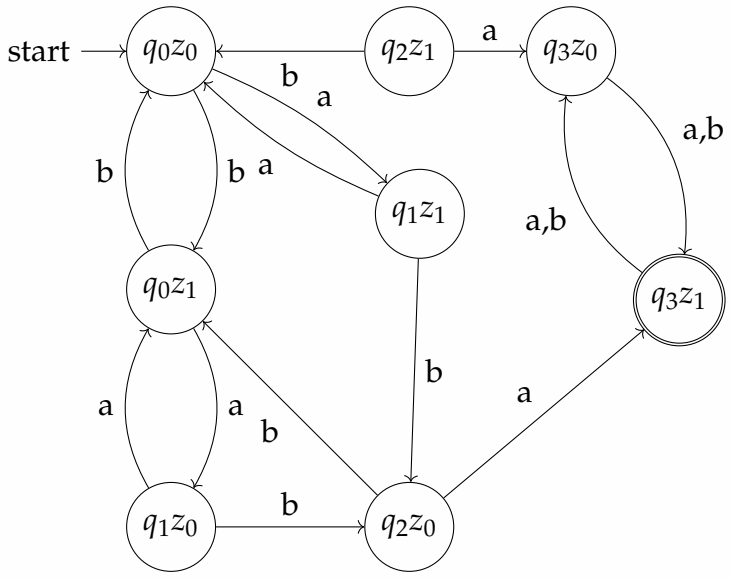
	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_0
q_3	q_3	q_3

A_2

	a	b
z_0	z_1	z_1
z_1	z_0	z_0

Neuer Endzustand: q_3z_1

	a	b
q_0z_0	q_1z_1	q_0z_1
q_1z_0	q_0z_1	q_2z_1
q_2z_0	q_3z_1	q_0z_1
q_3z_0	q_3z_1	q_3z_1
q_0z_1	q_1z_0	q_0z_0
q_1z_1	q_0z_0	q_2z_0
q_2z_1	q_3z_0	q_0z_0
q_3z_1	q_3z_0	q_3z_0



> Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3pc5rh7

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-1/Aufgabe-1.tex`

Examensaufgabe „Exponentieller Blow-Up“ (66115-2018-F.T2-A3)

Reguläre Sprache

Gesucht ist eine reguläre Sprache $C \subseteq \{a,b\}^*$, deren minimaler deterministischer endlicher Automat (DEA) mindestens 4 Zustände mehr besitzt als der minimale nichtdeterministische endliche Automat (NEA). Gehen Sie wie folgt vor:

(a) Definieren Sie $C \subseteq \{a,b\}^*$ und erklären Sie kurz, warum es bei dieser Sprache NEAs gibt, die deutlich kleiner als der minimale DEA sind.

Lösungsvorschlag

Sprache mit exponentiellem Blow-Up:

Ein NEA der Sprache

$$\begin{aligned} L_k &= \{xay \mid x,y \in \{a,b\}^* \wedge |y| = k-1\} \\ &= \{w \in \{a,b\}^* | \text{ der k-te Buchstabe von hinten ist ein } a\} \end{aligned}$$

kommt mit $k+1$ Zuständen aus.

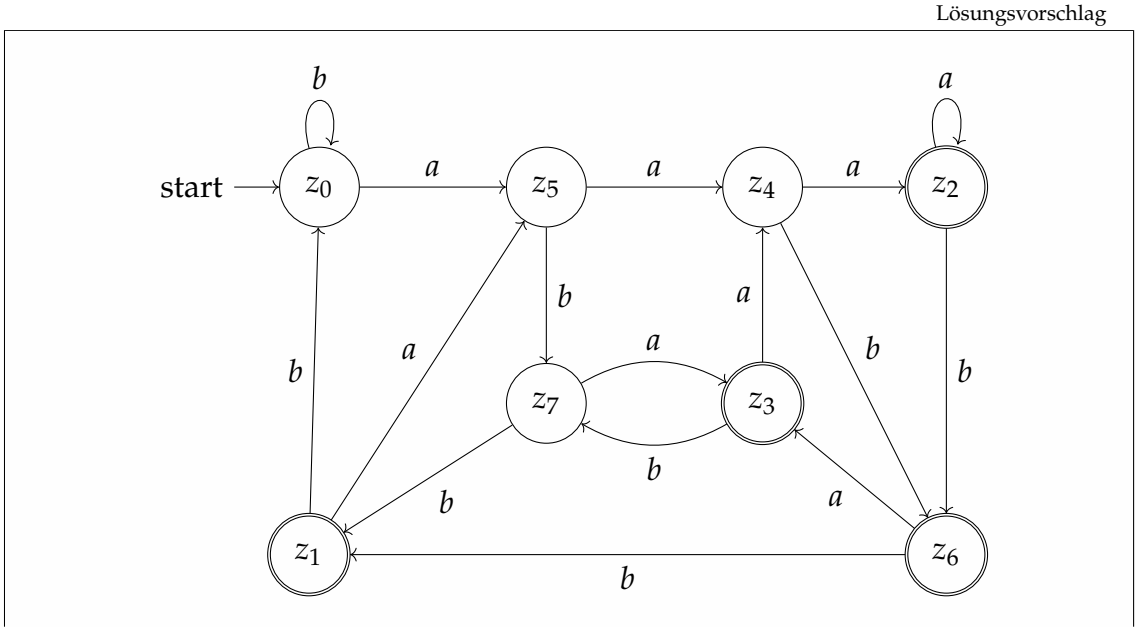
Jeder DEA M mit $L(M) = L$ hat dann mindestens 2^k Zustände. Wir wählen $k = 3$. Dann hat der zughörige NEA 4 Zustände und der zugehörige DEA mindestens 8. Sei also $L_k = \{xay | x,y \in \{a,b\}^* \wedge |y| = 2\}$ die gesuchte Sprache.

Der informelle Grund, warum ein DEA für die Sprache L_k groß sein muss, ist dass er sich immer die letzten n Symbole merken muss. [a]

[a] `https://www.tcs.ifi.lmu.de/lehre/ss-2013/timi/handouts/handout-02`

(b) Geben Sie den minimalen DEA M für C an. (Zeichnung des DEA genügt; die Minimalität muss nicht begründet werden.)

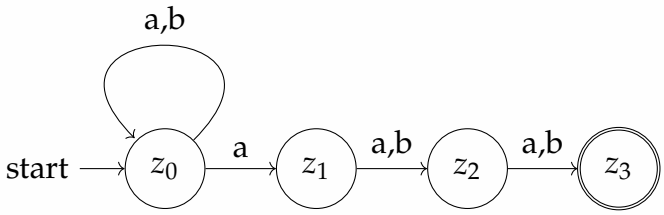
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahhefpjir

(c) Geben Sie einen NEA N für C an, der mindestens 4 Zustände weniger besitzt als M. (Zeichnung des NEA genügt)

Lösungsvorschlag



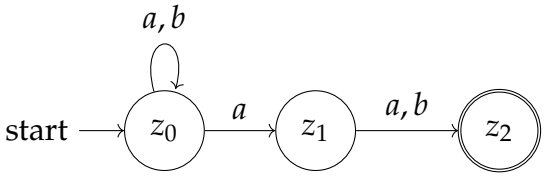
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajrz7h5r7

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/03/Thema-2/Aufgabe-3.tex`

Examensaufgabe „NEA nach DEA“ (66115-2019-F.T1-A2)

Reguläre Sprache

(a) Gegeben sei der nichtdeterministische endliche Automat A über dem Alphabet $\Sigma = \{a, b\}$ wie folgt:



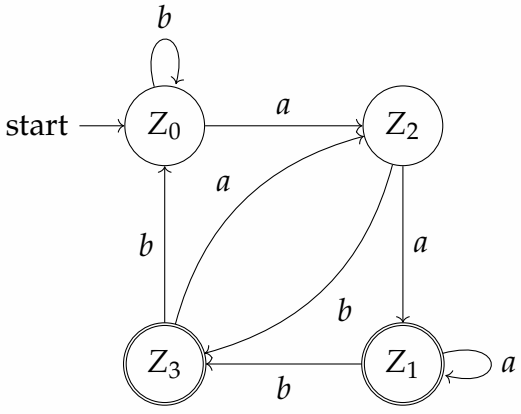
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arozq4rm2

Konstruieren Sie einen deterministischen endlichen Automaten, der das Komplement $\bar{L}(A) = \{ w \in \Sigma^* \mid w \notin L(A) \}$ der von A akzeptierten Sprache $L(A)$ akzeptiert.

Lösungsvorschlag

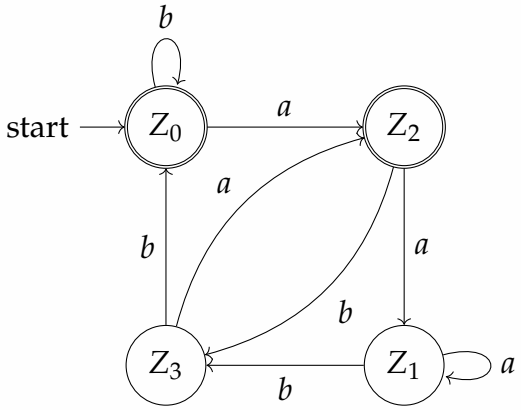
Wir konvertieren zuerst den nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten mit Hilfe des Potenzmengenalgorithmus.

Zustandsmenge	Eingabe a	Eingabe b
Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$
Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_1, z_2\}$	Z_3 $\{z_0, z_2\}$
Z_2 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_1, z_2\}$	Z_3 $\{z_0, z_2\}$
Z_3 $\{z_0, z_2\}$	Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$



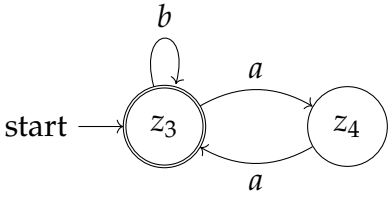
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arxujcbdg

Wir vertauschen die End- und Nicht-End-Zustände, um das Komplement zu erhalten:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5zqsonq2

(b) Gegeben sei zudem der nichtdeterministische Automat B über dem Alphabet $\Sigma = \{a, b\}$:

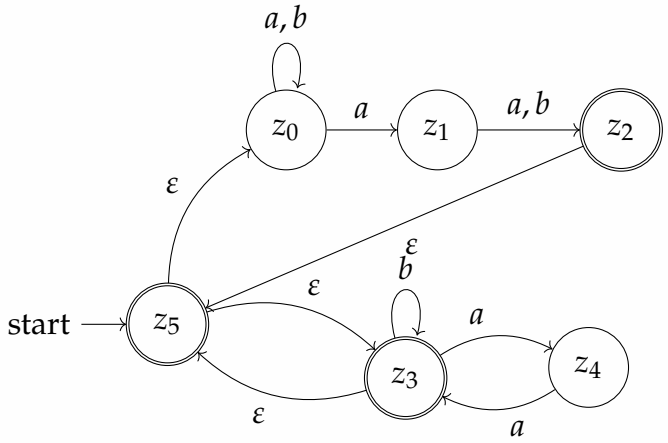


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arafgk0h2

Konstruieren Sie einen endlichen Automaten (möglicherweise mit ε-Übergängen), der die Sprache $(L(A)L(B))^* \subseteq \Sigma^*$ akzeptiert (A aus der vorigen Aufgabe). Erläutern Sie auch Ihre Konstruktionsidee.

Lösungsvorschlag

$L(A)L(B))^*$ ist die beliebige Konkatenation (Verknüpfung/Verkettung) der Sprachen $L(A)$ und $L(B)$ mit dem leeren Wort. Wir führen einen neuen Startzustand (z_5) ein, der zugleich Endzustand ist. Dadurch wird das leere Wort akzeptiert. Dieser neue Startzustand führt über ε-Übergängen zu den ehemaligen Startzuständen der Automaten A und B. Die Endzustände der Automaten A und B führen über ε-Übergängen zu z_5. Dadurch sind beliebige Konkatenationen möglich.



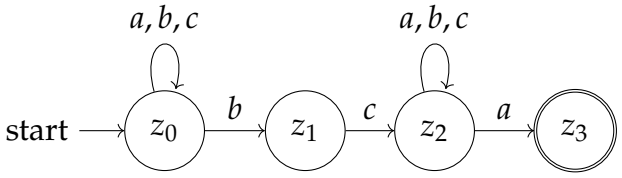
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro3uhzjz

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Automaten mit Zuständen q, r, s, t“ (66115-2020-F.T1-A2)

Reguläre Sprache
Potenzmengenalgorithmus

(a) Es sei $L \subseteq \{a, b, c\}^*$ die von dem folgenden nichtdeterministischen Automaten akzeptierte Sprache:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apmac9bwc

Beschreiben Sie (in Worten) wie die Wörter aus der Sprache L aussehen.

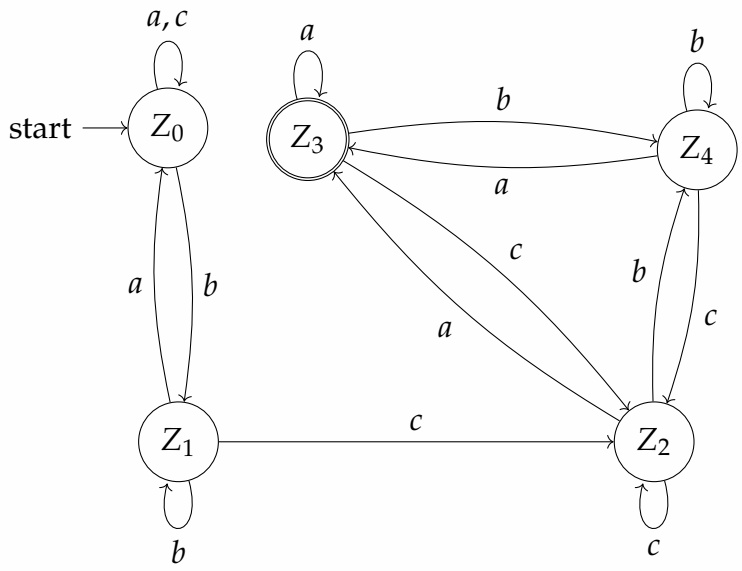
Lösungsvorschlag

Alle Wörter der Sprache L enthalten die Symbolfolge bc und enden auf a. Am Anfang der Wörter und vor dem letzten a können beliebige Kombination aus a, b, c vorkommen.

(b) Benutzen Sie die Potenzmengenkonstruktion, um einen deterministischen Automaten zu konstruieren, der zu dem Automaten aus Teil (a) äquivalent ist. (Berechnen Sie nur erreichbare Zustände.)

Lösungsvorschlag

Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
Z_0 $\{z_0\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$
Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_2\}$
Z_2 $\{z_0, z_2\}$	Z_3 $\{z_0, z_2, z_3\}$	Z_4 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_2\}$
Z_3 $\{z_0, z_2, z_3\}$	Z_3 $\{z_0, z_2, z_3\}$	Z_4 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_2\}$
Z_4 $\{z_0, z_1, z_2\}$	Z_3 $\{z_0, z_2, z_3\}$	Z_4 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_2\}$



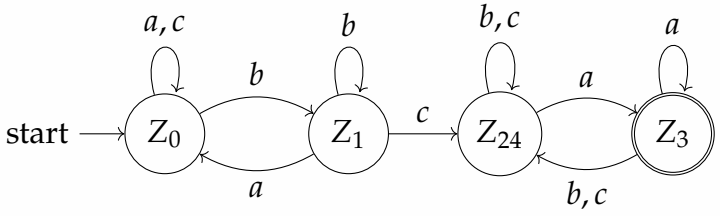
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5o6pho8c

(c) Ist der resultierende deterministische Automat schon minimal? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. $Z_2\ \{z_0, z_2\}$ und $Z_4\ \{z_0, z_1, z_2\}$ können vereinigt werden, da sie bei denselben Eingaben auf die selben Potzenmengen übergehen.

Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
$Z_2\ \{z_0, z_2\}$	$Z_3\ \{z_0, z_2, z_3\}$	$Z_4\ \{z_0, z_1, z_2\}$	$Z_2\ \{z_0, z_2\}$
$Z_3\ \{z_0, z_2, z_3\}$	$Z_3\ \{z_0, z_2, z_3\}$	$Z_4\ \{z_0, z_1, z_2\}$	$Z_2\ \{z_0, z_2\}$
$Z_4\ \{z_0, z_1, z_2\}$	$Z_3\ \{z_0, z_2, z_3\}$	$Z_4\ \{z_0, z_1, z_2\}$	$Z_2\ \{z_0, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ai1hox2b7

(d) Minimieren Sie den folgenden deterministischen Automaten:

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Vermische Fragen“ (66115-2020-H.T1-TA1-A1)

Antworten Sie mit *„Stimmt“* oder *„Stimmt nicht“*. Begründen Sie Ihr Urteil kurz.

(a) Eine Sprache ist genau dann regulär, wenn sie unendlich viele Wörter enthält.

Lösungsvorschlag

Stimmt nicht. Sprachen mit endlicher Mächtigkeit sind immer regulär. Endliche Spachen sind in obenstehender Aussage ausgeschlossen.

(b) Zu jedem nichtdeterministischen endlichen Automaten mit n Zuständen gibt es einen deterministischen endlichen Automaten, der die gleiche Sprache erkennt und höchstens n^2 Zustände hat.

Lösungsvorschlag

Stimmt nicht. Müsste 2^n heißen.

(c) Das Komplement einer kontextfreien Sprache ist wieder kontextfrei.

Lösungsvorschlag

Stimmt nicht. Kontextfreie Sprachen sind nicht abgeschlossen unter dem Komplement. Das Komplement einer kontextfreien Sprache kann regulär, kontextfrei oder kontextsensitiv sein.

(d) Wenn ein Problem unentscheidbar ist, dann ist es nicht semientscheidbar.

Lösungsvorschlag

Stimmt nicht. Semientscheidbarkeit ist eine typische Form der Unentscheidbarkeit. Unentscheidbarkeit ist das Gegenteil von Entscheidbarkeit. Unentscheidbar kann entweder völlig unentscheidbar sein oder semientscheidbar.

(e) Sei f eine totale Funktion. Dann gibt es ein WHILE-Programm, das diese berechnet.

Lösungsvorschlag

Stimmt nicht. Wir wissen nicht, ob die totale Funktion f berechenbar ist. Wenn f berechenbar ist, dann wäre die Aussage richtig.

(f) Das Halteproblem für `LOOP`-Programme ist entscheidbar.

Lösungsvorschlag

Stimmt. Alle LOOP-Programme terminieren (halten). Es gibt für jede Eingabe eine Ausgabe.

(g) Die Komplexitätsklasse $\mathcal{NP}$ enthält genau die Entscheidungsprobleme, die in nicht-polynomieller Zeit entscheidbar sind.

Lösungsvorschlag

Stimmt. Die Aussage entspricht der Definiton der Komplexitätsklasse $\mathcal{NP}$.

(h) Falls $P \geq NP$, dann gibt es keine $\mathcal{NP}$-vollständigen Probleme, die in P liegen.

Lösungsvorschlag

Stimmt. Entspricht der Definition.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Reguläre Sprache xyz“ (66115-2020-H.T1-TA1-A2)

Reguläre Sprache

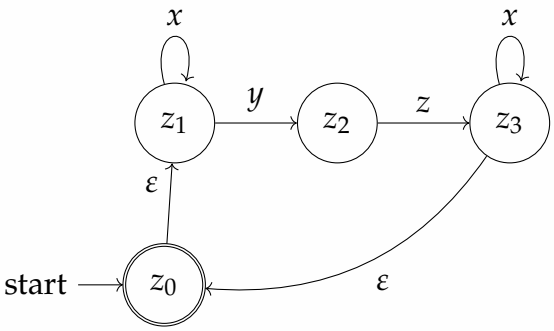
Sei $\Sigma = \{x, y, z\}$. Sei $L = (x^* yzx^*)^* \subseteq \Sigma^*$.

(a) Geben Sie einen endlichen (deterministischen oder nichtdeterministischen) Automaten A an, der L erkennt bzw. akzeptiert.

Lösungsvorschlag

Nichtdeterministischer endlicher Automat

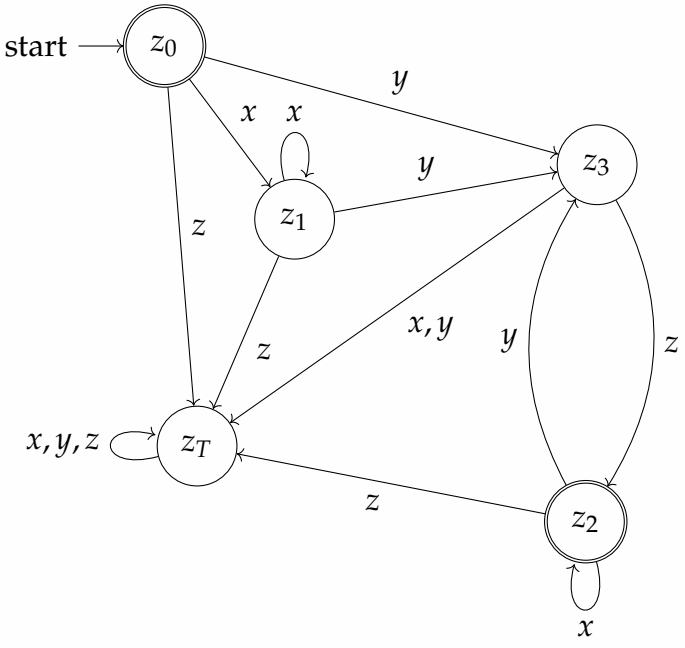
$A_{\text{NEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajpmxqvh9

Deterministischer endlicher Automat

$A_{\text{DEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5xo470g9

(b) Geben Sie eine reguläre und eindeutige Grammatik G an, die L erzeugt.

Lösungsvorschlag

$$
\begin{aligned}
P = \Big\{ & \\
& Z_0 \to xZ_1 \mid yZ_3 \mid \varepsilon \\
& Z_1 \to yZ_3 \mid xZ_1 \\
& Z_2 \to xZ_2 \mid x \mid yZ_3 \\
& Z_3 \to zZ_2 \mid z \\
& \Big\}
\end{aligned}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gjfc3c2d2

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „Palindrom über Alphabet „abc““ (66115-2020-H.T1-TA1-A3)

Kontextfreie Sprache
Pumping-Lemma (Reguläre Sprache)

Seien $\Sigma = \{a, b, c\}$ und $L = \{ wc\hat{w} \mid w \in \{a, b\}* \}$. Dabei ist $\hat{w}$ das zu w gespiegelte Wort.

(a) Zeigen Sie, dass L nicht regulär ist.

> **Exkurs: Pumping-Lemma für Reguläre Sprachen**
>
> Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:
>
> (i) $|v| \geq 1$ (Das Wort v ist nicht leer.)
>
> (ii) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)
>
> (iii) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^i w \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort $uv^i w$ in der Sprache L)
>
> Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

Lösungsvorschlag

L ist regulär. Dann gilt für L das Pumping-Lemma. Sei j die Zahl aus dem Pumping-Lemma. Dann muss sich das Wort $a^j bcba^j \in L$ aufpumpen lassen (da $|a^j bcba^j| \geq j$). $a^j bcba^j = uvw$ ist eine passende Zerlegung laut Lemma. Da $|uv| < j$, ist $u = a^x$, $v = a^y$, $w = a^z bcba^j$, wobei $y > 0$ und $x + y + z = j$. Aber dann $uv^0 w = a^{x+z} bcba^j \notin L$, da $x + z < j$. Widerspruch. [a]

[a] `https://userpages.uni-koblenz.de/~sofronie/gti-ss-2015/slides/endliche-automaten6.pdf`

(b) Zeigen Sie, dass L kontextfrei ist, indem Sie eine geeignete Grammatik angeben und anschließend begründen, dass diese die Sprache L erzeugt.

Lösungsvorschlag

$$\mathrm{P} = \Big\{ \quad S \to aSa \mid aCa \mid bSb \mid bCb \quad C \to c \quad \Big\}$$

$S \vdash aSa \vdash abCba \vdash abcba$

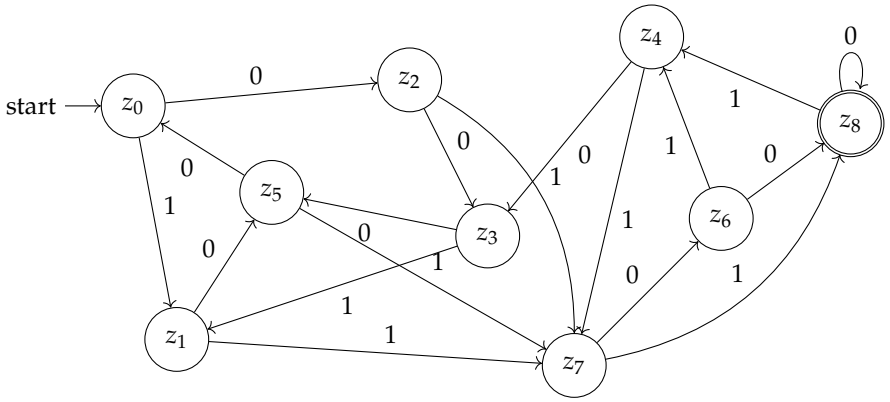
$S \vdash bSb \vdash bbSbb \vdash bbaSabb \vdash bbacabb$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-1/Teilaufgabe-1/Aufgabe-3.tex`

Examensaufgabe „Minimierungsalgorithmus“ (66115-2020-H.T2-TA1-A1)

Reguläre Sprache

(a) Geben Sie einen deterministischen endlichen Automaten (DEA) mit minimaler Anzahl an Zuständen an, der dieselbe Sprache akzeptiert wie folgender deterministischer endlicher Automat. Dokumentieren Sie Ihr Vorgehen geeignet.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj5aei652

Lösungsvorschlag

Minimierungstabelle (Table filling)

— Der Minimierungs-Algorithmus (auch Table-Filling-Algorithmus genannt) trägt in seinem Verlauf eine Markierung in alle diejenigen Zellen der Tabelle ein, die zueinander nicht äquivalente Zustände bezeichnen. Die Markierung „x_n“ in einer Tabellenzelle (i, j) bedeutet dabei, dass das Zustandspaar (i, j) in der k-ten Iteration des Algorithmus markiert wurde und die Zustände i und j somit zueinander $(k-1)$-äquivalent, aber nicht k-äquivalent und somit insbesondere nicht äquivalent sind. Bleibt eine Zelle bis zum Ende unmarkiert, sind die entsprechenden Zustände zueinander äquivalent. ———————

z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_1	x_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_2	x_3	x_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_3		x_3	x_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_4	x_3	x_4		x_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_5	x_3	x_4		x_3		$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_6	x_2	x_2	x_2	x_2	x_2	x_2	$\varnothing$	$\varnothing$	$\varnothing$
z_7	x_2	x_2	x_2	x_2	x_2	x_2	x_2	$\varnothing$	$\varnothing$
z_8	x_1	x_1	x_1	x_1	x_1	x_1	x_1	x_1	$\varnothing$
	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

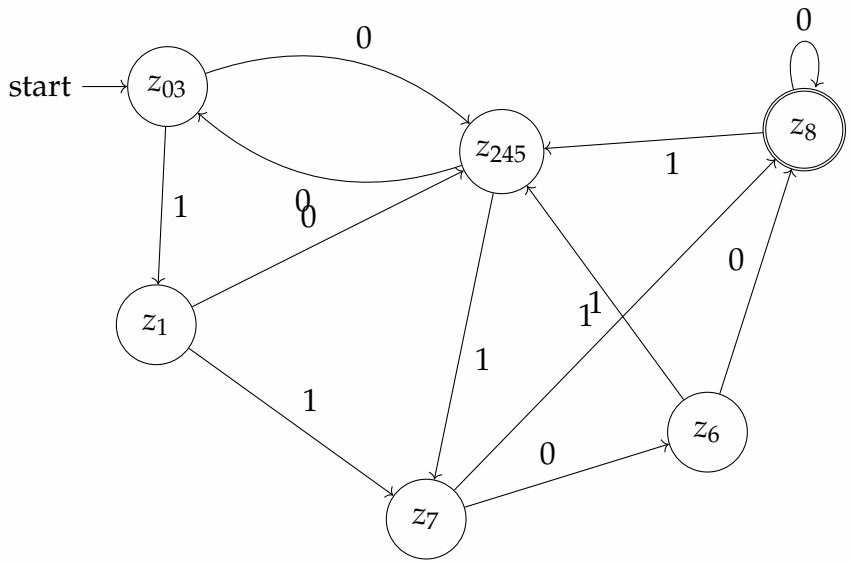
x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	0	1	
(z_0, z_1)	(z_2, z_5)	(z_1, z_7)	x_3 x_3
(z_0, z_2)	(z_2, z_3)	(z_1, z_7)	x_3
(z_0, z_3)	(z_2, z_5)	(z_1, z_1)	
(z_0, z_4)	(z_2, z_3)	(z_1, z_7)	x_3
(z_0, z_5)	(z_2, z_0)	(z_1, z_7)	x_3
(z_0, z_6)	(z_2, z_8)	(z_1, z_4)	x_2
(z_0, z_7)	(z_2, z_6)	(z_1, z_8)	x_2
(z_1, z_2)	(z_5, z_3)	(z_7, z_7)	x_4
(z_1, z_3)	(z_5, z_5)	(z_7, z_1)	x_3
(z_1, z_4)	(z_5, z_3)	(z_7, z_7)	x_4
(z_1, z_5)	(z_5, z_0)	(z_7, z_7)	x_4
(z_1, z_6)	(z_5, z_8)	(z_7, z_4)	x_2
(z_1, z_7)	(z_5, z_6)	(z_7, z_8)	x_2
(z_2, z_3)	(z_3, z_5)	(z_7, z_1)	x_3
(z_2, z_4)	(z_3, z_3)	(z_7, z_7)	
(z_2, z_5)	(z_3, z_0)	(z_7, z_7)	
(z_2, z_6)	(z_3, z_8)	(z_7, z_4)	x_2
(z_2, z_7)	(z_3, z_6)	(z_7, z_8)	x_2
(z_3, z_4)	(z_5, z_3)	(z_1, z_7)	x_3
(z_3, z_5)	(z_5, z_0)	(z_1, z_7)	x_3
(z_3, z_6)	(z_5, z_8)	(z_1, z_4)	x_2
(z_3, z_7)	(z_5, z_6)	(z_1, z_8)	x_2
(z_4, z_5)	(z_3, z_0)	(z_7, z_7)	
(z_4, z_6)	(z_3, z_8)	(z_7, z_4)	x_2
(z_4, z_7)	(z_3, z_6)	(z_7, z_8)	x_2
(z_5, z_6)	(z_0, z_8)	(z_7, z_4)	x_2
(z_5, z_7)	(z_0, z_6)	(z_7, z_8)	x_2
(z_6, z_7)	(z_8, z_6)	(z_4, z_8)	x_2



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro484bz2

(b) Beweisen oder widerlegen Sie für folgende Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$, dass sie regulär sind.

(i) $L_1 = \{ a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0 \}$

Lösungsvorschlag

Die Sprache L_1 ist regulär. Nachweis durch regulären Ausdruck:

$$a^* c (a|b)^* b^* (a|b)^* a c^*$$

(ii) $L_2 = \{ a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0 \text{ mit } k = i + j \}$

Lösungsvorschlag

Die Sprache L_2 ist nicht regulär. Widerlegung durch das Pumping-Lemma.
TODO

(c) Sei L eine reguläre Sprache über dem Alphabet Σ. Für ein festes Element $a \in \Sigma$ betrachten wir die Sprache $L_a = \{ aw \mid w \in \Sigma^*, wa \in L \}$. Zeigen Sie, dass L_a regulär ist.

Lösungsvorschlag

Die regulären Sprachen sind unter dem Komplement abgeschlossen.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Reguläre Sprachen Automaten zuordnen“ (66115-2021-F.T1-TA1-A1)

Reguläre Sprache

Im Folgenden bezeichnet $a^i = a \dots a$ und ε steht für das leere Wort (ðinsbesondere $a^i = \varepsilon$).

Die Menge $\mathbb{N}_0 = \{0, 1, 2, \dots\}$ ist die Menge aller nicht-negativer Ganzzahlen.

Die Sprachen $L_1, \dots, L_{12}$ seien definiert als:

(a) Ordnen Sie jedem der folgenden nichtdeterministischen endlichen Automaten $N_j, j = 1, \dots, 6$, (die alle über dem Alphabet $\Sigma = \{a\}$ arbeiten) **jeweils eine** der Sprachen $L_i \in \{L_1, \dots, L_{12}\}$ zu, sodass L_i, genau die von N_i, **akzeptierte Sprache** ist.

Lösungsvorschlag

- $N_1 = L_6$ (mindestens ein a)
- $N_2 = L_8$ (ungerade Anzahl an a's: $1, 5, 7, \dots$)
- $N_3 = L_2$ (gerade Anzahl an a's: $2, 4, 6, \dots$)
- $N_4 = L_{12}$ (leeres Wort)
- $N_5 = L_8$ (ungerade Anzahl an a's: $1, 5, 7, \dots$)
- $N_6 = L_{11}$ (die Sprache akzeptiert nicht)

(b) Zeigen Sie für eine der Sprachen $L_1, \dots, L_{12}$ dass diese **nicht regulär** ist.

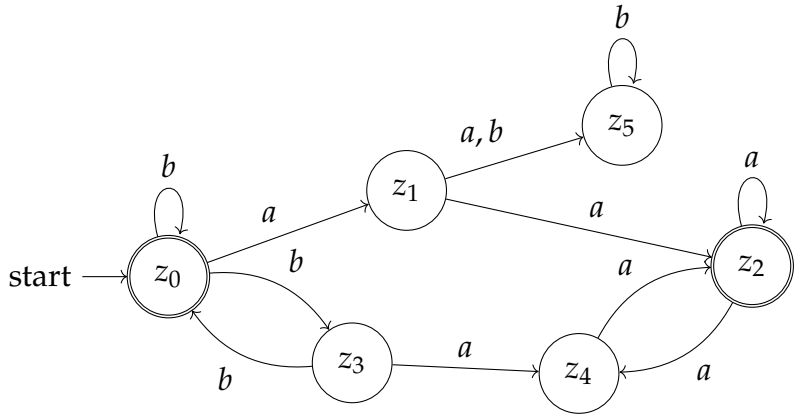
Lösungsvorschlag

$L_10 = \{ a^n \mid n \in \mathbb{N}_0, n \text{ ist Primzahl} \}$

ist nicht regulär, da sich sonst jede Primzahl p einer bestimmten Mindestgröße j als Summe von natürlichen Zahlen $u + v + w$ darstellen ließe, so dass $v \geq 1$ und für alle $i \geq 0$ auch $u + iv + w = p + (i1)v$ prim ist. Dies ist jedoch für $i = p + 1$ wegen $p + (p + 11)v = p(1 + v)$ nicht der Fall.[a]

[a] `https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws13/einftheo/einftheo-skript.pdf`

(c) Konstruieren Sie für den folgenden nichtdeterministischen endlichen Automaten (der Worte über dem Alphabet $\Sigma = \{a, b\}$ verarbeitet) einen äquivalenten deterministischen endlichen Automaten mithilfe der Potenzmengenkonstruktion. Zeichnen Sie dabei nur die vom Startzustand erreichbaren Zustände. Erläutern Sie Ihr Vorgehen.

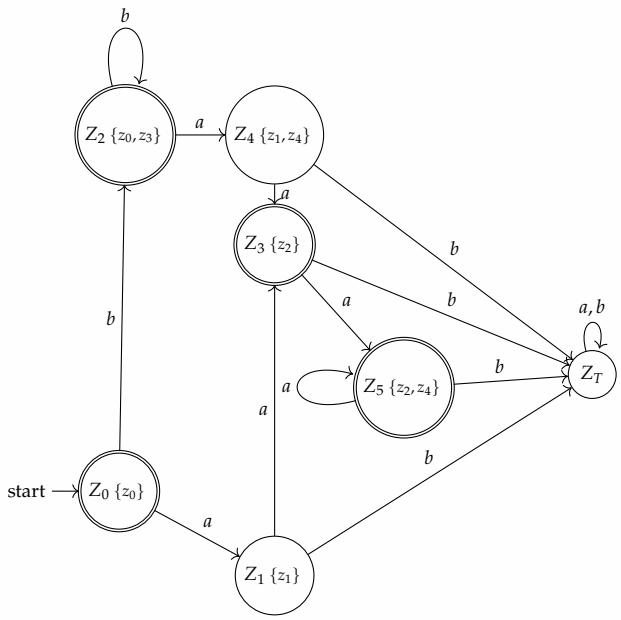


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af7iooyca

Lösungsvorschlag

Zustandsmenge	Eingabe a	Eingabe b
Z_0 $\{z_0\}$	Z_1 $\{z_1\}$	Z_2 $\{z_0, z_3\}$
Z_1 $\{z_1\}$	Z_3 $\{z_2\}$	Z_T
Z_2 $\{z_0, z_3\}$	Z_4 $\{z_1, z_4\}$	Z_2 $\{z_0, z_3\}$
Z_3 $\{z_2\}$	Z_5 $\{z_2, z_4\}$	Z_T
Z_4 $\{z_1, z_4\}$	Z_3 $\{z_2\}$	Z_T
Z_5 $\{z_2, z_4\}$	Z_5 $\{z_2, z_4\}$	Z_T

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apkyuo4ja

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-1.tex`

Examensaufgabe „Reguläre Sprachen“ (66115-2021-F.T2-TA1-A1)

Reguläre Sprache

(a) Sei

$L_1 = \{ w \in \{a, b, c\}^* \mid w$ enthält genau zweimal den Buchstaben a und der vorletzte Buchstabe ist

Geben Sie einen regulären Ausdruck für die Sprache L_1 an.

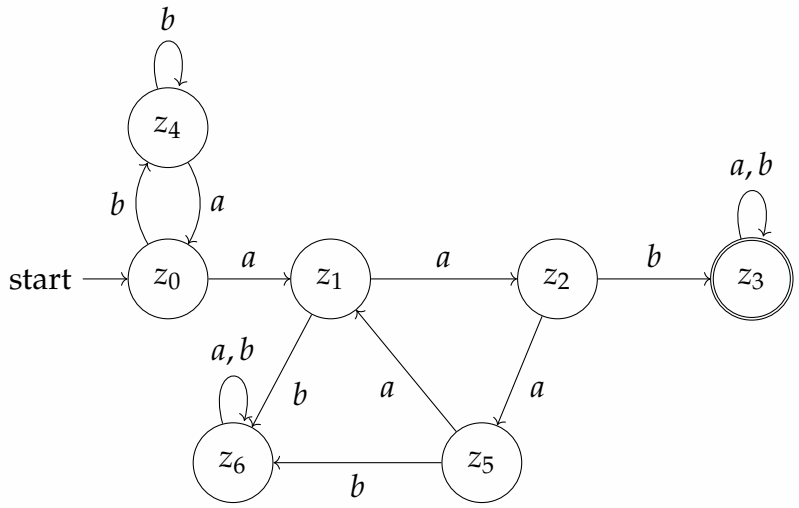
Lösungsvorschlag

```
(
  ((b|c)* a (b|c)* a (b|c)* c (b|c))
  |
  ((b|c)* a (b|c)* c a)
)
```

(b) Konstruieren Sie einen deterministischen endlichen Automaten für die Sprache L_2:

$L_2 = \{ w \in \{a, b\}^* \mid w$ enthält genau einmal das Teilwort aab $\}$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahf2oduri

(c) Sei $\mathbb{N} = \{1, 2, 3, \ldots\}$ die Menge der strikt positiven natürlichen Zahlen. Sei

$L_3 = \{ \#a^{i_1}\#a^{i_1}\# \cdots a^{i_{n-1}}\#a^{i_n}\# \mid n, i_1, \ldots, i_n \in \mathbb{N}$ und es existiert $j \in \mathbb{N}$ mit $i_j = n + 1 \}$

eine Sprache über Alphabet $\{\#, a\}$.

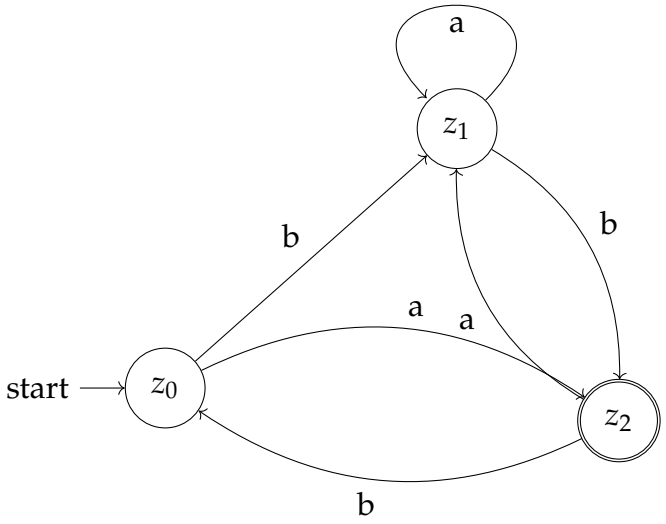
So ist z. B. $\#a\#aaa\# \in L_3$ (da das Teilwort $a^3 = aaa$ vorkommt) und $\#a\#a\#a\#a\# \in L_3$ (da das Teilwort $a^5 = aaaaa$ nicht vorkommt). Beweisen Sie, dass L_3 nicht regulär ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-1.tex

Übungsaufgabe „Grammatik aus Automat“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA), Reguläre Grammatik)

Reguläre Sprache
Deterministisch endlicher Automat (DEA)
Reguläre Grammatik

Sei $A = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, \{z_2\}, z_0)$ ein endlicher Automat. Die Übergangsfunktion sei wie in dem unten abgebildeten Diagramm definiert.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apk0iyqyg

(a) Gebe eine reguläre Grammatik G an, sodass $L(G) = L(M)$ gilt.

Lösungsvorschlag

$G = (\{Z_0, Z_1, Z_2\}, \{a, b\}, P, Z_0)$ mit folgender Produktionsmenge

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} Z_0 &\to bZ_1 \mid aZ_2 \\ Z_1 &\to aZ_1 \mid bZ_2 \\ Z_2 &\to bZ_0 \mid aZ_1 \mid \varepsilon \end{aligned}$$

$$\Big\}$$

(b) Überlegen Sie sich ein systematisches Verfahren, um einen deterministischen endlichen Automaten in eine reguläre Grammatik umzuwandeln.

Lösungsvorschlag

Analog zu obigem Beispiel folgender Algorithmus benutzt werden:

- Setze $V = \{Z_0, Z_1, \ldots Z_n\}$ und S auf den Startzustand des Automaten.
- Für jeden Übergang $\delta(Z_i, a) = Z_j$ füge die Produktion $\{Z_i \to aZ_j\}$ zu P hinzu.

- Für jeden Zustand $Z_i \in Z$ füge die Produktion $\{Z_i \rightarrow \varepsilon\}$ zu P dazu.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Aufgabe_Grammatik-aus-Automat.tex

Übungsaufgabe „NEA-DEA-Aequivalenzklassen“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA), Minimierungsalgorithmus, Reguläre Ausdrücke, Äquivalenzklassen)

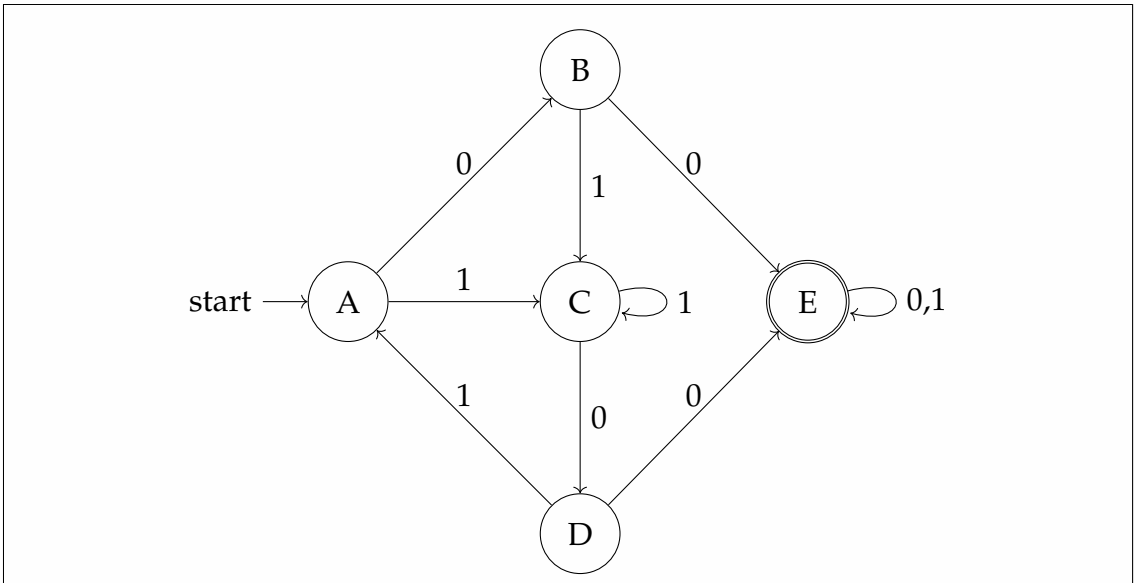
Reguläre Sprache
Deterministisch endlicher Automat (DEA)
Minimierungsalgorithmus

Gegeben ist der deterministische endliche Automat $A = (\{A, B, C, D, E\}, \{0, 1\}, \delta, \{E\}, A)$.

δ	0	1
A	B	C
B	E	C
C	D	C
D	E	A
E	E	E

(a) Minimieren Sie den Automaten mit dem bekannten Minimierungsalgorithmus. Dokumentieren Sie die Schritte geeignet.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5amu40wc

A	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
B	x_2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
C		x_2	$\varnothing$	$\varnothing$	$\varnothing$
D	x_2		x_2	$\varnothing$	$\varnothing$
E	x_1	x_1	x_1	x_1	$\varnothing$
	A	B	C	D	E

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

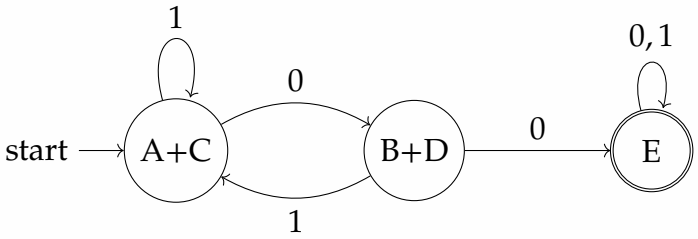
x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	**0**	**1**
(A, B)	(B, E) x_2	(C, C)
(A, C)	(B, D)	(C, C)
(A, D)	(B, E) x_2	(C, A)
(B, C)	(E, D) x_2	(C, C)
(B, D)	(E, E)	(C, A)
(C, D)	(D, E) x_2	(C, A)

Minimiert



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ara57j4oa

(b) Geben Sie einen regulären Ausdruck für die erkannte Sprache an.

Lösungsvorschlag Äquivalenzklassen

$r = (0|1)^*00(0|1)^*$

(c) Geben Sie die Äquivalenzklassen der Myhill-Nerode-Äquivalenz der Sprache durch reguläre Ausdrücke an.

Lösungsvorschlag

Die Äquivalenzklassen lauten: $[A, C], [B, D], [E]$

$$r_A = (1^*(01)^*)^*$$
$$r_B = (1^*(01)^*)^*0$$
$$r_C = r$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Aufgabe_NEA-DEA-Aequivalenzklassen.tex

Übungsaufgabe „Noten“ (Reguläre Sprache)

Reguläre Sprache

Der Viervierteltakt ist vor allem in der Unterhaltungsmusik die weitaus häufigste Taktart. Viervierteltakt bedeutet, dass ein Takt eine Länge von vier Vierteln hat. Das Symbol 𝅘𝅥 bedeutet eine Viertelnote. Eine halbe Note 𝅗𝅥 hat den Wert von zwei Vierteln, eine punktierte halbe Note 𝅗𝅥. den Wert von drei Vierteln und eine ganze Note 𝅝 den Wert von vier Vierteln. Andere Notenwerte sollen hier nicht vorkommen.

Die Prüfsoftware eines Notenverlages stellt eine Methode bereit, die testet, ob die Notenwerte eines Taktes tatsächlich vier Viertel ergeben.

Durch das Alphabet

$$\Sigma = \{\, 𝅝, 𝅗𝅥., 𝅗𝅥, 𝅘𝅥 \,\},$$

die Menge der Nichtterminale

$$\mathrm{V} = \{\, \text{<Takt>}; \text{<3/4>}; \text{<1/2>} \,\},$$

das Startsymbol <Takt> sowie die Produktionsregeln

$$R_1\text{: <Takt>} \rightarrow 𝅝 \mid \text{<3/4>}\ 𝅘𝅥$$
$$R_2\text{: <3/4>} \rightarrow 𝅗𝅥. \mid \text{<1/2>}\ 𝅘𝅥$$
$$R_3\text{: <1/2>} \rightarrow 𝅗𝅥 \mid 𝅘𝅥\,𝅘𝅥$$

ist eine Grammatik für eine formale Sprache S gegeben.

(a) Entscheide begründet, ob die Zeichenketten 𝅘𝅥𝅘𝅥𝅘𝅥𝅘𝅥 und 𝅗𝅥𝅗𝅥𝅘𝅥 zu S gehören. Gebe eine weitere Zeichenkette an, die zwar einen Viervierteltakt darstellt, aber nicht zu S gehört.

(b) Ergänze die oben angegebenen Produktionsregeln so, dass jeder mit dem Alphabet Σ mögliche Viervierteltakt dargestellt werden kann.

(c) Mit T_4 wird die formale Sprache bezeichnet, die genau alle Viervierteltakte enthält, die mit den Zeichen aus Σ gebildet werden können. Verwende für die weiteren Teilaufgaben anstelle der Notensymbole Buchstaben gemäß folgender Tabelle:

Zeichne das Zustandsübergangsdiagramm eines erkennenden endlichen Automaten, der genau T_4 akzeptiert.

(d) Entwerfe eine Implementierung des Automaten aus Teilaufgabe c in Java. Dabei soll es u. a. eine Methode istViervierteltakt(eingabe) geben, die überprüft, ob die übergebene Zeichenkette eingabe den Vorgaben für einen Viervierteltakt entspricht, und einen entsprechenden Wahrheitswert zurückgibt. Dazu ruft sie für jedes Zeichen der Eingabe jeweils die Methode zustandWechseln() auf.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Aufgabe_Noten.tex`

Übungsaufgabe „Vorlesungsaufgaben“ (Reguläre Grammatik)

Reguläre Grammatik

Gegeben ist eine Sprache $L \subset \Sigma^*$ mit $\Sigma = \{a, b\}$. Zu der Sprache L gehören alle Wörter, die die Zeichenfolge *abba* beinhalten.

(a) Gib eine Grammatik an, die diese Sprache erzeugt.

Lösungsvorschlag

$G = (V, \Sigma, P, S)$ mit $\Sigma = \{a, b\}$, $S = S$, $V = \{S, A, B, C, D\}$

Tipp: Die Produktionsregeln so entwerfen, dass zuerst das Wort „abba“ erkannt wird, dann die Regeln entwerfen.

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to aA \mid aS \mid bS \\
A &\to bB \\
B &\to bC \\
C &\to aD \\
D &\to aD \mid bD \mid \varepsilon
\end{aligned} \right\}
$$

Andere Möglichkeit:

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to aA \mid aS \mid bS \\
A &\to bB \\
B &\to bC \\
C &\to aD \mid a \\
D &\to aD \mid bD \mid a \mid b
\end{aligned} \right\}
$$

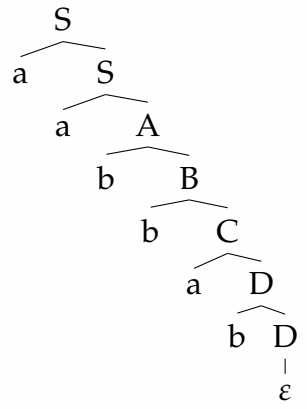
Nicht erlaubt in regulärer Grammtik:

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to abbaA
\end{aligned} \right\}
$$

(b) Gib eine Ableitung/Syntaxbaum zu deiner Grammatik für das Wort `aabbab` an.

Lösungsvorschlag

$S \rightarrow aS \rightarrow aaA \rightarrow aabB \rightarrow aabbC \rightarrow aabbaD \rightarrow aabbabD \rightarrow aabbab$



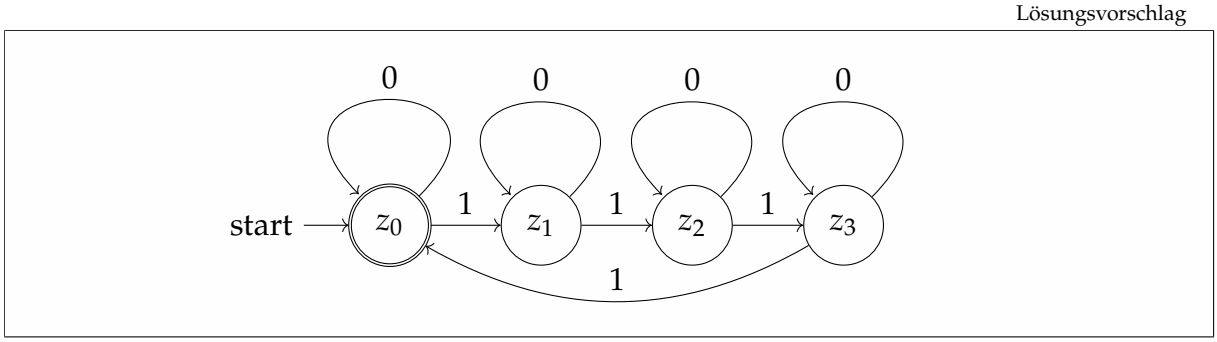
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Aufgabe_Vorlesungsaufgaben-Regulaere-Grammatiken.tex

Übungsaufgabe „Deterministischer endlicher Automat“ (Reguläre Sprache, Deterministisch endlicher Automat (DEA))

Reguläre Sprache
Deterministisch endlicher Automat (DEA)

Geben Sie einen DFA über dem Alphabet $\Sigma = \{0, 1\}$ an, der die folgende Sprache erkennt: Die Menge aller Zeichenketten, die eine durch 4 teilbare Anzahl von Einsen besitzt.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Endliche-Automaten/Aufgabe_Deterministischer-endlicher-Automat.tex`

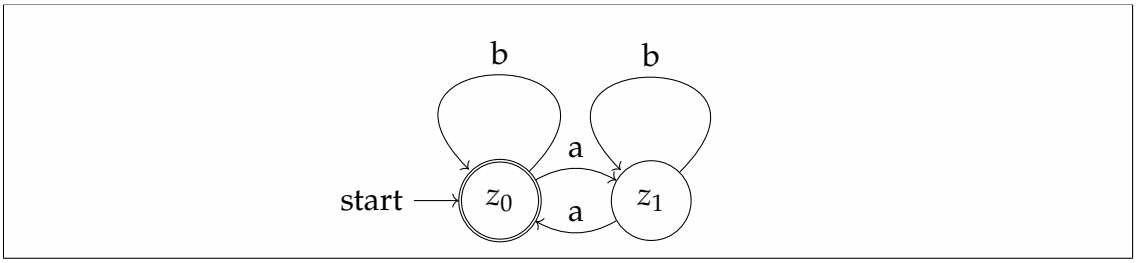
Übungsaufgabe „Vorlesungsaufgaben“ (Deterministisch endlicher Automat (DEA))

Deterministisch endlicher Automat (DEA)

Stellen Sie einen Automaten zu den folgenden Sprachen ($\Sigma = \{a, b\}$) auf:

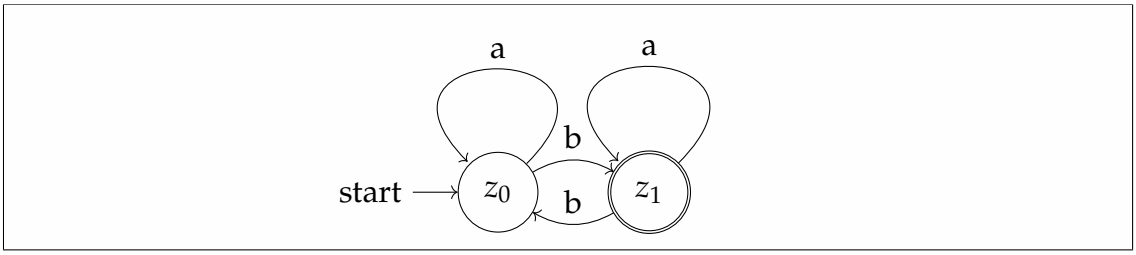
(a) $L_1 = \{ x \mid x \text{ beinhaltet eine gerade Anzahl von } a \}$

Lösungsvorschlag



(b) $L_2 = \{ x \mid x \text{ beinhaltet eine ungerade Anzahl von } b \}$

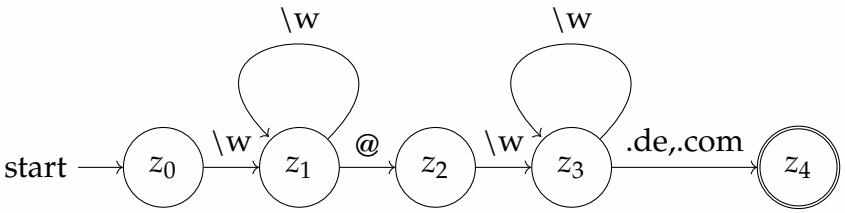
Lösungsvorschlag



(c) Geben Sie einen DEA an, der eine syntaktisch gültige E-Mail-Adresse erkennt. (mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) vor dem `@`; mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) nach dem `@`; alle E-Mail-Adressen sollen auf `.de` oder `.com` enden.

Lösungsvorschlag

Es müsste noch der Trap-Zustand hinzugefügt werden.



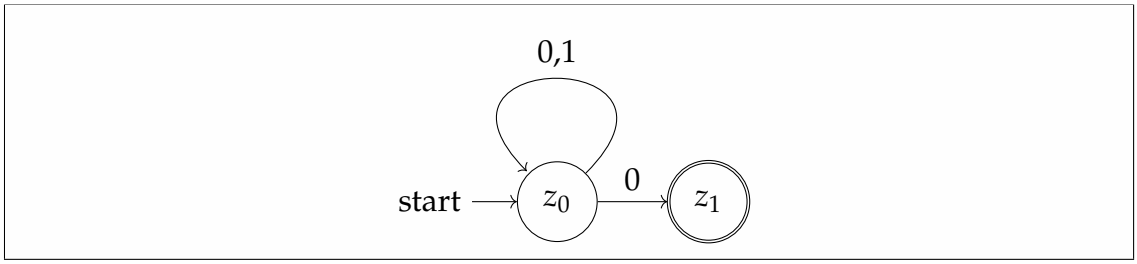
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Endliche-Automaten/Aufgabe_Vorlesungsaufgaben-DEA.tex`

Übungsaufgabe „Vorlesungsaufgaben“ (Nichtdeterministisch endlicher Automat (NEA))

Nichtdeterministisch endlicher Automat (NEA)

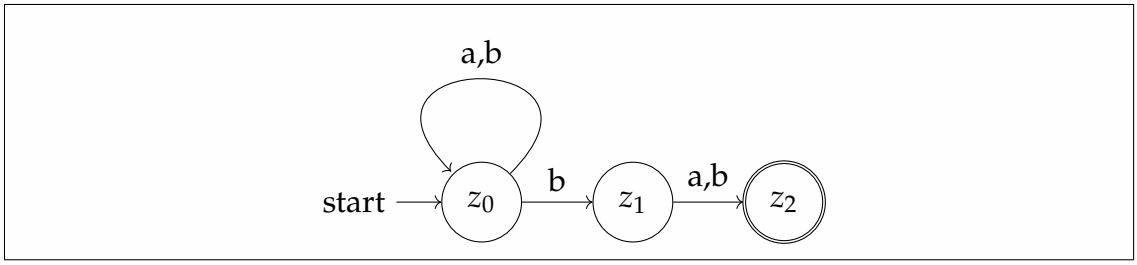
(a) Stellen Sie einen nichtdeterministischen endlichen Automaten auf, der alle durch 2 teilbaren Binärzahlen (letztes Wort ist 0) akzeptiert.

Lösungsvorschlag



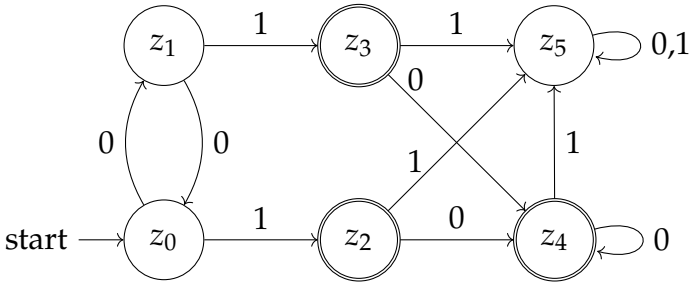
(b) Stellen Sie einen NEA auf, der alle Wörter über einem Alphabet $\Sigma = \{a, b\}$ akzeptiert, die als vorletztes Zeichen ein b besitzen.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Endliche-Automaten/Aufgabe_Vorlesungsaufgaben-NEA.tex`

Übungsaufgabe „Studiflix-Minimierung“ (Minimierungsalgorithmus)



Lösungsvorschlag

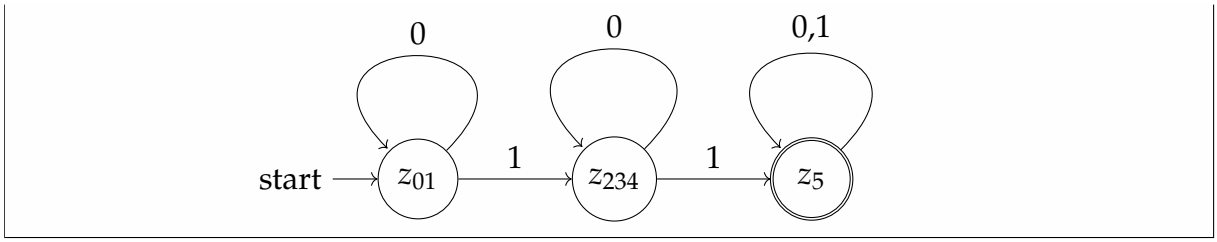
z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_1		$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_2	x_1	x_1	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_3	x_1	x_1		$\varnothing$	$\varnothing$	$\varnothing$
z_4	x_1	x_1			$\varnothing$	$\varnothing$
z_5	x_2	x_2	x_1	x_1	x_1	$\varnothing$
	z_0	z_1	z_2	z_3	z_4	z_5

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.
x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.
x_3 In weiteren Iterationen markierte Zustände.
x_4 ...

Übergangstabelle

Zustandspaar	0	1
(z_0, z_1)	(z_1, z_0)	(z_2, z_3)
(z_0, z_5)	(z_1, z_5)	(z_2, z_5) x_2
(z_1, z_5)	(z_0, z_5)	(z_3, z_5) x_2
(z_2, z_3)	(z_4, z_4)	(z_5, z_5)
(z_2, z_4)	(z_4, z_4)	(z_5, z_5)
(z_3, z_4)	(z_4, z_4)	(z_5, z_5)

(z_2, z_3), (z_2, z_4) und (z_3, z_4) können zu einem Zustand verschmolzen werden, weil sie alle drei bei der Eingabe von 0 zu (z_4, z_4) und bei 1 zu (z_5, z_5) werden. z_5 kann nicht verschmolzen werden, weil er in der Tabelle markiert ist.



1

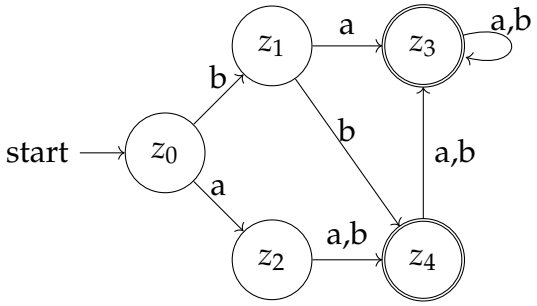
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Minimierungsalgorithmus/Aufgabe_Studiflix-Minimierung.tex

[1] https://studyflix.de/informatik/dea-minimieren-1212

Übungsaufgabe „Minimalisierung“ (Minimierungsalgorithmus)

Minimierungsalgorithmus

Minimalisiere den gegebenen DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apm4e9nk7

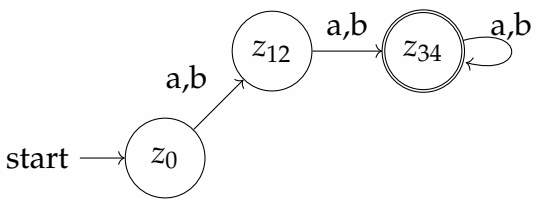
Lösungsvorschlag

z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_1	x_2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
z_2	x_2		$\varnothing$	$\varnothing$	$\varnothing$
z_3	x_1	x_1	x_1	$\varnothing$	$\varnothing$
z_4	x_1	x_1	x_1		$\varnothing$
	z_0	z_1	z_2	z_3	z_4

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.
x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.
x_3 In weiteren Iterationen markierte Zustände.
x_4 ...

Übergangstabelle

Zustandspaar	a	b
(z_0, z_1)	(z_2, z_3) x_2	(z_1, z_4)
(z_0, z_2)	(z_2, z_4) x_2	(z_1, z_4)
(z_1, z_2)	(z_3, z_4)	(z_4, z_4)
(z_3, z_4)	(z_3, z_3)	(z_3, z_3)

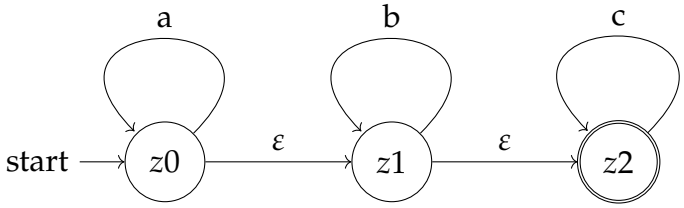


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aib87m3wc

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Minimierungsalgorithmus/Aufgabe_Vorlesungsaufgaben-Minimalisierung.tex`

Übungsaufgabe „NEA: z012, Alphabet: abc“ (Erweiteter Potenzmengenalgorithmus)

Erweiteter Potenzmengenalgorithmus



(a) Welche Sprache akzeptiert dieser Automat? Beschreiben Sie in Worten und stellen Sie einen regulären Ausdruck sowie eine Grammatik hierfür auf.

Lösungsvorschlag

in Worten Das Alphabet besteht aus a, b, c. Am Anfang stehen 0 oder beliebig viele a's, dann kommen 0 oder beliebig viele b's und dann 0 oder beliebig viele c's.

Regulärer Ausdruck $a^*b^*c^*$

Grammatik

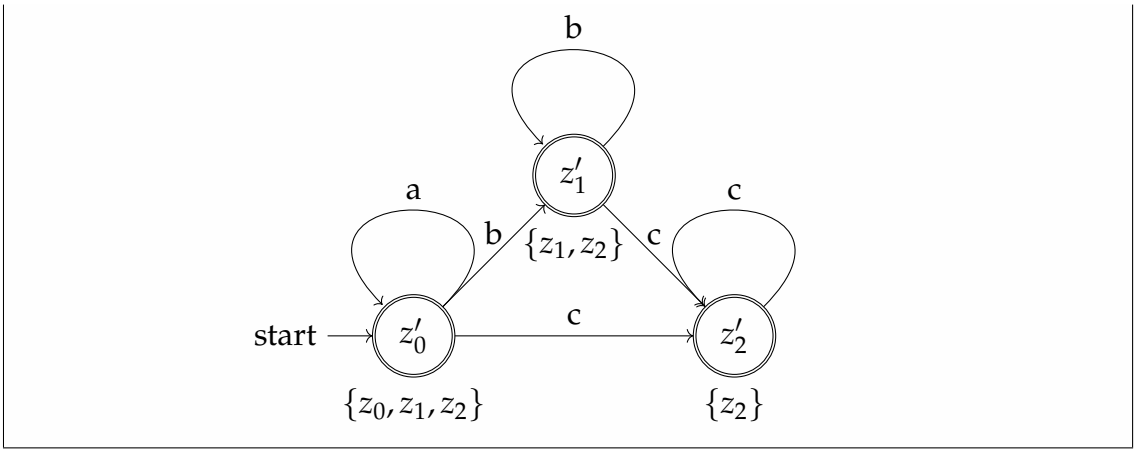
$$\mathrm{P} = \Big\{ \quad S \to aS \mid bA \mid cB \mid \varepsilon \quad A \to bA \mid cB \mid \varepsilon \quad B \to cB \mid \varepsilon \quad \Big\}$$

(b) Wandeln Sie den ε-NEA zum einem DEA mit Hilfe des erweiterter Potenzmengenalgorithmus um.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
Z_0	$\{z_0, z_1, z_2\}$	$\{z_0, z_1, z_2\}$	$\{z_1, z_2\}$	$\{z_2\}$
Z_1	$\{z_1, z_2\}$	$\{\}$	$\{z_1, z_2\}$	$\{z_2\}$
Z_2	$\{z_2\}$	$\{\}$	$\{\}$	$\{z_2\}$

Trap-Übergänge werden aus Übersichtsgründen weg gelassen.

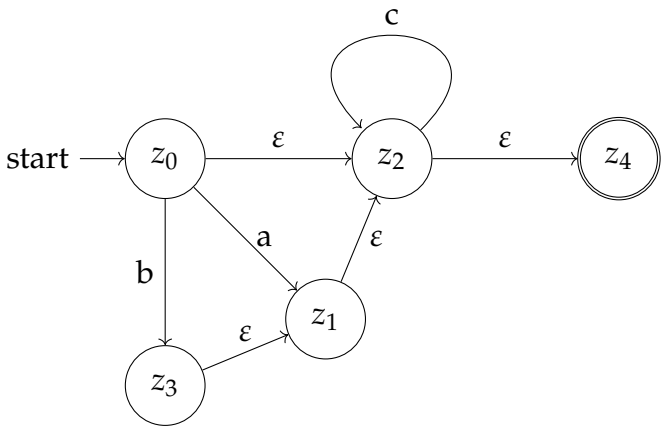


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Potenzmengenalgorithmus/Aufgabe_Erweiteter-Potenzmengenalgorithmus.tex`

Übungsaufgabe „NEA: z01234, Alphabet: ab“ (Erweiteter Potenzmengenalgorithmus)

Erweiteter Potenzmengenalgorithmus

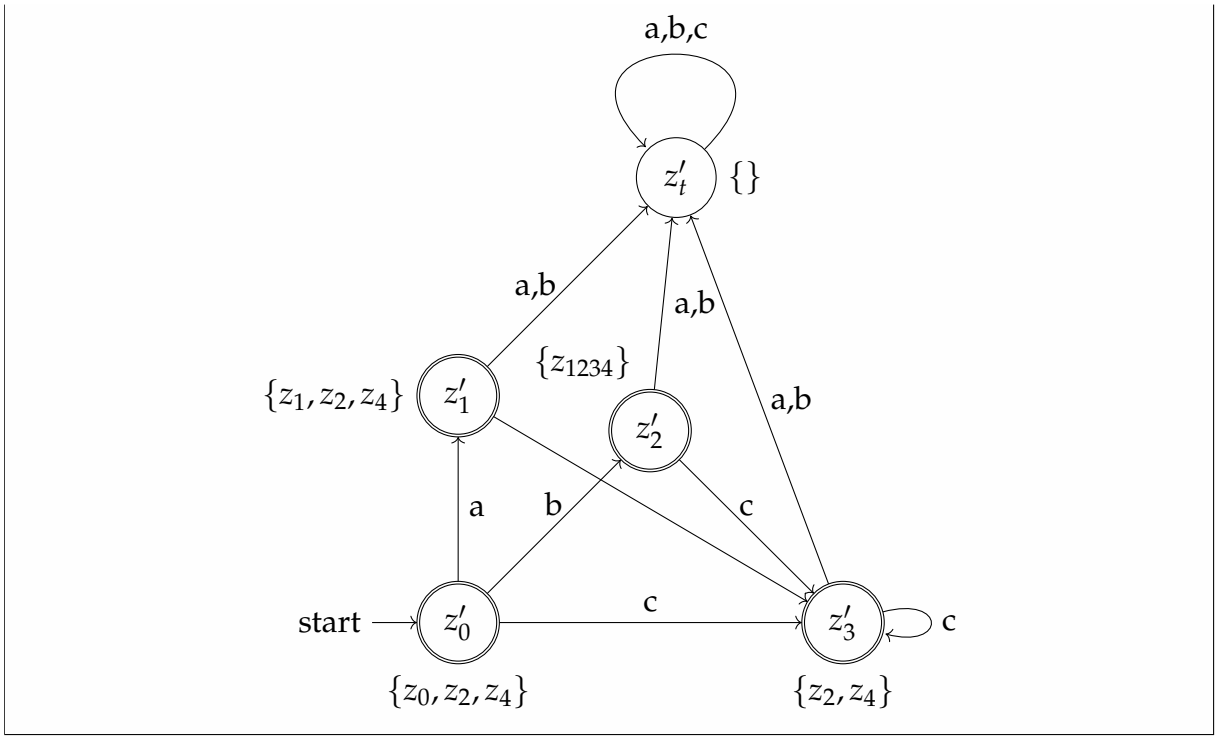
Gegeben ist der folgende ε-NEA:



Wandlen Sie den gegebenen Automaten in eine ε-freien DEA um.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
z'_0	$\{z_0, z_2, z_4\}$	$\{z_1, z_2, z_4\}$	$\{z_1, z_2, z_3, z_4\}$	$\{z_2, z_4\}$
z'_1	$\{z_1, z_2, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
z'_2	$\{z_1, z_2, z_3, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
z'_3	$\{z_2, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
z'_t	$\{\}$	$\{\}$	$\{\}$	$\{\}$

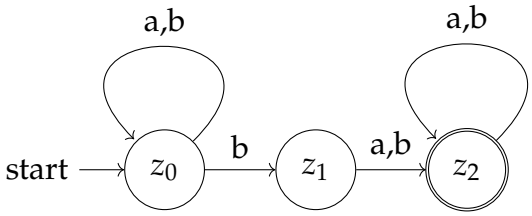


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Potenzmengenalgorithmus/Aufgabe_Vorlesungsaufgaben-Erweiteter-Potenzmengenalgorithmus.tex`

Übungsaufgabe „Vorlesungsaufgaben“ (Potenzmengenalgorithmus)

Potenzmengenalgorithmus

Gegeben ist der folgende NEA:



(a) Welche Sprache akzeptiert dieser Automat? Beschreiben Sie in Worten und stellen Sie einen regulären Ausdruck hierfür auf.

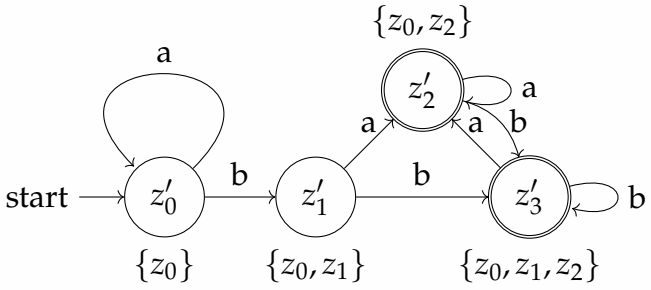
Lösungsvorschlag

$(a|b)^*b(a|b)(a|b)^*$

(b) Überführe den gegebenen NEA mit dem Potenzmengenalgorithmus in einen DEA.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
z'_0	$\{z_0\}$	$\{z_0\}$	$\{z_0, z_1\}$
z'_1	$\{z_0, z_1\}$	$\{z_0, z_2\}$	$\{z_0, z_1, z_2\}$
z'_2	$\{z_0, z_2\}$	$\{z_0, z_2\}$	$\{z_0, z_1, z_2\}$
z'_3	$\{z_0, z_1, z_2\}$	$\{z_0, z_2\}$	$\{z_0, z_1, z_2\}$

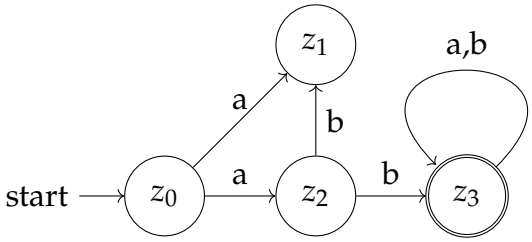


Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Potenzmengenalgorithmus/Aufgabe_Vorlesungsaufgaben-Potenzmengenalgorithmus-erstes-Beispiel.tex`

Übungsaufgabe „Vorlesungsaufgaben“ (Potenzmengenalgorithmus)

Potenzmengenalgorithmus

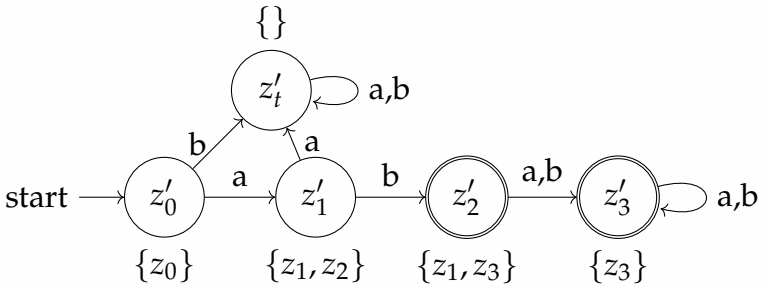
Gegeben ist der folgende nichtdeterministische endliche Automat:



Überführenn Sie den gegebenen nichtdeterministischen endlichen Automaten mit dem Potenzmengenalgorithmus in einen deterministischen endlichen Automaten.

Lösungsvorschlag

neuer Name	Zustandsmenge	Eingabe a	Eingabe b
z'_0	$\{z_0\}$	$\{z_1, z_2\}$	$\{\}$
z'_1	$\{z_1, z_2\}$	$\{\}$	$\{z_1, z_3\}$
z'_2	$\{z_1, z_3\}$	$\{z_3\}$	$\{z_3\}$
z'_3	$\{z_3\}$	$\{z_3\}$	$\{z_3\}$
z'_t	$\{\}$	$\{\}$	$\{\}$



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Potenzmengenalgorithmus/Aufgabe_Vorlesungsaufgaben-Potenzmengenalgorithmus.tex`

Übungsaufgabe „„w w““ (Pumping-Lemma (Reguläre Sprache))

Pumping-Lemma (Reguläre Sprache)

Zeigen oder widerlegen Sie: Die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$ sind regulär. [2]

Exkurs: Pumping-Lemma für Reguläre Sprachen
Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|v| \geq 1$ (Das Wort v ist nicht leer.)

(b) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(c) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^i w \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort $uv^i w$ in der Sprache L)

Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

$$L_1 = \{ww | w \in \{a, b\}^*\}$$

Lösungsvorschlag

Angenommen L_1 sei regulär, dann müsste L_1 die Bedingungen der stärkeren Variante des Pumping-Lemmas erfüllen.

Beweis durch Widerspruch:

Sei $j \in \mathbb{N}$ die Konstante aus dem Pumping-Lemma und $\omega = a^j b a^j b$ ein Wort aus L_1 ($|\omega| > j$ gilt offensichtlich).

Dann müsste ω nach dem Pumping-Lemma zerlegbar sein in $\omega = uvw$ mit $|v| \geq 1$ und $|uv| < j$. uv kann wegen $|uv| < j$ kein b enthalten und liegt komplett im ersten a^j.
Also:

$$a^j b a^j b = uvw \text{ mit } u = a^x, v = a^y, w = a^{n-x-y} b a^j b (n \geq x + y, x > 0)$$

Dann gilt

$$uv^0 w = a^x a^{j-x-y} b a^j b = a^{j-y} b a^j b \notin L_1$$

Wir haben gezeigt, dass es keine gültige Zerlegung für ω gibt. Also gilt für L_1 die stärkere Variante des Pumping-Lemmas nicht. Somit kann L_1 nicht regulär sein.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Pumping-Lemma/Aufgabe_Koblenz.tex`

[2] `https://userpages.uni-koblenz.de/~dpeuter/teaching/17ss_gti/blatt04_loesung.pdf`

Übungsaufgabe „„wn2“ „an bm cn““ (Pumping-Lemma (Reguläre Sprache))

Pumping-Lemma (Reguläre Sprache)

Begründe jeweils, ob die folgenden Sprachen regulär sind oder nicht. [3]

(a) $L_1 = \{ w \in \{a,b\}^* \mid$ auf ein a folgt immer ein $b \}$

Lösungsvorschlag

$L_1 = L(b^*(ab)^*b^*)$ und damit regulär.

(b) $L_2 = \{ w \in \{1\}^* \mid \exists n \in \mathbb{N}$ mit $|w| = n^2 \}$

Lösungsvorschlag

L_2 ist nicht regulär.

Pumping-Lemma:

j sei eine Quadratzahl: Somit ist $1^j \in L_2$. Es gilt $|uv| \leq j$ und $|v| \geq 1$. Daraus folgt, dass in v mindestens eine 1 existiert. Somit wird immer ein $i \in \mathbb{N}$ existieren, sodass $uv^iw \notin L$, weil die Quadratzahlen nicht linear darstellbar sind.

Begründung über die Zahlentheorie:

Angenommen, L_2 sei regulär, sei m die kleinste Zahl mit $m^2 > j$. Dann ist $x = 1^{m^2} \in L_2$. Für eine Zerlegung $x = uvw$ nach dem Pumping-Lemma muss dann ein k existieren mit $v = 1^k$ und $m^2 - l + k^l$ ist eine Quadratzahl für jedes $l \geq 0$. Das kann offenbar zahlentheoretisch nicht sein, und somit haben wir einen Widerspruch zur Annahme.

(c) $L_3 = \{ a^n b^m c^n \mid m,n \in \mathbb{N}_0 \}$

Lösungsvorschlag

$L_3 = \{ a^n b^m c^n \mid m,n \in \mathbb{N}_0 \}$ ist nicht regulär.
$a^j b^j c^j \in L_3$:
$|uv| \leq j$ und $|v| \geq 1$

$\rightarrow$ in uv sind nur a's und in v ist mindestens ein a
$\rightarrow$ $uv^2w \notin L_3$, weil dann mehr a's als c's in diesem Wort vorkommen

(d) $L_4 = \{ w \in \{a\}^* \mid \mod_3(|w|) = 0 \}$

[3] `https://www.uni-muenster.de/Informatik/u/lammers//EDU/ws08/AutomatenFormaleSprachen/Loesungen/Loesung05.pdf`

Lösungsvorschlag

$L_4 = ((aaa)^*)$ und damit regulär.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Pumping-Lemma/Aufgabe_Pumping-Lemma.tex

Übungsaufgabe „w c wR“ (Pumping-Lemma (Reguläre Sprache))

4

Exkurs: Pumping-Lemma für Reguläre Sprachen
Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|v| \geq 1$ (Das Wort v ist nicht leer.)

(b) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(c) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^iw \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iw in der Sprache L)

Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

$$L_1 = \{wcw^R | w \in \{a, b\}^*\}$$

Erläuterung: w^R ist die Spiegelung von w, ðes enthält die Zeichen von w in umgekehrter Reihenfolge. Worte von L_1 sind also z. B. *c*, *abcba*, *bbbaabacabaabbb*

Lösungsvorschlag

L_1 ist kontexfrei.

Beweis, dass L_1 nicht regulär ist, durch das Pumping Lemma:

Wir nehmen an L_1 wäre regulär. Dann gibt es einen endlichen Automaten, der L_1 erkennt. Die Anzahl der Zustände dieses Automaten sei j. Wir wählen jetzt das Wort $\omega = a^jca^j$. ω liegt in L_1, und ist offensichtlich länger als j. Dieses Wort muss irgendwo eine Schleife, also einen aufpumpbaren Teil enthalten, ðman kann es so in uvw zerlegen, dass für jede natürliche Zahl i auch uv^iw zu L_1 gehört. Wo könnte dieser aufpumpbare Teil liegen?

Fall 1: Der aufpumbare Teil v liegt komplett im Bereich des ersten a^j-Blocks. Dann würde aber $uv^2w = a^{j+|v|}ca^j$ mehr a's im ersten Teil als im zweiten Teil enthalten und läge nicht mehr in L_1.

Fall 2: v enthält das c. Dann würde aber uv^2w zwei c's enthalten und läge damit nicht mehr in L_1.

Fall 3: Der aufpumpbare Teil liegt komplett im Bereich des zweiten a^j-Blocks. Dann liegt analog zu Fall 1 uv^2w nicht mehr in L_1. Unser Wort lässt sich also nicht so zerlegen, dass man den Mittelteil aufpumpen kann, also ist die Annahme, dass L_1 regulär ist, falsch.

Beweis, dass L_1 kontextfrei ist, durch Angabe einer kontexfreien Grammatik:

[4] `http://www.coli.uni-saarland.de/courses/I2CL-10/material/Uebungsblaetter/Musterloesung4.4.pdf`

$$
\mathrm{P} = \left\{
\begin{array}{l}
S \to aSa \\
S \to bSb \\
S \to c
\end{array}
\right\}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Pumping-Lemma/Aufgabe_Saarland-Pinkal.tex`

Übungsaufgabe „a n b m“ (Pumping-Lemma (Reguläre Sprache))

Pumping-Lemma (Reguläre Sprache)

Exkurs: Pumping-Lemma für Reguläre Sprachen
Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|v| \geq 1$ (Das Wort v ist nicht leer.)

(b) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(c) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^i w \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort $uv^i w$ in der Sprache L)

Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

(a) Zeigen Sie, dass die Sprache $L = \{ a^n b^m \mid n \geq m \geq 1 \}$ nicht regulär ist.

Lösungsvorschlag

$|a^j b^j| \geq j$
$a^j b^j = uvw$ mit $|uv| \leq j$ und $|v| \geq 1$
$\Rightarrow$ in v nur a's
$\Rightarrow uv^0 w \notin L$

(b) Zeige, dass die Sprache $L = \{ a^n b^m \mid n > m \geq 1 \}$ nicht regulär ist.

Lösungsvorschlag

$|a^{j+1} b^j| \geq j$
$a^{j+1} b^j = uvw$ mit $|uv| \leq j$ und $|v| \geq 1$
$\Rightarrow$ in v nur a's
$\Rightarrow uv^0 w \notin L$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Pumping-Lemma/Aufgabe_Vorlesungsaufgaben-Pumping-Lemma.tex

Übungsaufgabe „Arztpraxis und Autohauskette“ (Reguläre Ausdrücke)

Abitur 2018 Inf2. IV. 1. und Abitur 2015 Inf2. III. 2.

(a) Die Software einer Arztpraxis ermöglicht unter anderem die Erstellung von Rechnungen für die Patienten. Dabei muss eine Rechnungskennzahl angegeben werden, die wie folgt aufgebaut ist:

- zwei Buchstaben für die Initialen des Patienten (erster Buchstabe des Nachnamens gefolgt vom ersten Buchstaben des Vornamens)
- Bindestrich
- Patientennummer beliebiger Länge, die aus den Ziffern 0 bis 9 besteht, aber nicht mit 0 beginnt
- Versicherungsart: P bei Privatpatienten, G bei gesetzlich Versicherten
- nur bei gesetzlich Versicherten: zweistellige Versicherungskennzahl (Nummern im Bereich von 01 bis 12)
- Bindestrich
- fortlaufende Rechnungsnummer, die aus beliebig vielen Ziffern von 0 bis 9 besteht, aber nicht mit 0 beginnt

Beispiele:

Privatpatient Ingo Matik mit der Patientennummer 32 erhält seine 9. Rechnung. Die zugehörige Rechnungskennzahl ist: MI-32P-9.

Seine Frau Martha Matik mit der Patientennummer 1234, die gesetzlich bei einer Versicherung mit der Kennzahl 07 versichert ist, erhält ihre 12. Rechnung. Die zugehörige Rechnungskennzahl ist: MM-1234G07-12. Für die Darstellung der Rechnungskennzahl stehen das lateinische Alphabet der Großbuchstaben, die Ziffern 0 bis 9 und der Bindestrich zur Verfügung.

Stelle einen regulären Ausdruck für die Rechnungskennzahl in Java-Schreibweise auf.

Lösungsvorschlag

```
public class KrankenversicherungsNummer {

  public static void matches(String nummer) {
    System.out.print("Die Versicherungsnummer " + nummer + " ");
    if
↪   (nummer.matches("[A-Z]{2}-[1-9]\\d*(P|(G(0[1-9]|1[0-2])))-[1-9][0-9]*"))
↪   {
      System.out.print("ist eine");
    } else {
      System.out.print("ist keine");
    }
    System.out.println(" valide Nummer.");
  }
```

```
  public static void main(String[] args) {
    matches("MI-32P-9");
    matches("MM-1234G07-12");
    matches("MM-1234G17-12");
  }
}
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/KrankenversicherungsNummer.java

(b) Um in die zentrale Personalabteilung der Autohauskette zu gelangen, muss man vor der Sicherheitstür ein aus genau drei Zeichen bestehendes Passwort eingeben. Dieses wird jährlich gemäß den folgenden Vorgaben der Firma festgelegt:

- Das Passwort muss mindestens einen Kleinbuchstaben und eine Ziffer enthalten;
- es darf nicht mit einem der 32 Sonderzeichen (z. B. *, §, ...) beginnen;
- Großbuchstaben sind an keiner Stelle zugelassen.

  Stelle einen regulären Ausdruck für die Rechnungskennzahl in Java-Schreibweise auf.

Lösungsvorschlag

```
public class PasswortTuer {

  public static String b = "[a-z]";
  public static String Z = "\\d";
  public static String S =
↪  "[!\"#$%&\'\\(\\)\\*\\+,-\\./:;<=>\\?@\\[\\\\]\\^`\\{\\|\\}~]";

  public static void matches(String passwort) {
    System.out.print("Das Passwort " + passwort + " ");
    // "(b(S|b)Z)|(bZ(S|Z|b))|(Zb(S|Z|b))|(Z(S|Z)b)"
    if (passwort.matches(String.format("%s|%s|%s|%s",
        String.format("(%s(%s|%s)%s)", b, S, b, Z),
        String.format("(%s%s(%s|%s|%s))", b, Z, S, Z, b),
        String.format("(%s%s(%s|%s|%s))", Z, b, S, Z, b),
        String.format("(%s(%s|%s)%s)", Z, S, Z, b)))) {
      System.out.print("ist ein");
    } else {
      System.out.print("ist kein");
    }
    System.out.println(" valides Passwort.");
  }

  public static void main(String[] args) {
    matches("ab1");
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/PasswortTuer.java

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Regulaere-Ausdruecke/Aufgabe_Arztpraxis.tex

Übungsaufgabe „Reguläre Grammatik, reguläre Ausdrücke und DEA“ (Reguläre Sprache, Reguläre Grammatik, Ableitung (Reguläre Sprache), Reguläre Ausdrücke, Deterministisch endlicher Automat (DEA))

Reguläre Sprache
Reguläre Grammatik

Gegeben sind die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b\}$:

- $L_0 = \{ w \mid w \text{ enthält mindestens ein } bb \}$
- $L_1 = \{ w \mid w \text{ endet auf höchstens ein } b \}$
- $L_2 = \{ w \mid w \text{ fängt mit } aa \text{ an oder hört mit } bb \text{ auf} \}$

(a) Geben Sie zu allen Sprachen eine reguläre Grammatik an.

Lösungsvorschlag

$G_0 = (V, \Sigma, P, S)$ mit $V = \{S, A, B\}$, $\Sigma = \{a, b\}$, $S = S$ und mit

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aS \mid bA \\ A \to aS \mid bB \mid b \\ B \to aB \mid a \mid bB \mid b \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gjp92ri0w

$G_1 = (V, \Sigma, P, S)$ mit $V = \{S, A, B\}$, $\Sigma = \{a, b\}$, $S = S$ und mit

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aS \mid bS \mid b \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gfdn0xhwg

$G_2 = (V, \Sigma, P, S)$ mit $V = \{S, A, B, C, D, E\}$, $\Sigma = \{a, b\}$, $S = S$ und mit

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aA \mid bC \\ A \to aB \mid a \mid bC \\ B \to aB \mid a \mid bB \mid b \\ C \to aD \mid bE \mid b \\ D \to bC \mid aD \\ E \to bE \mid b \mid aD \end{array} \right.$$

Ableitung (Reguläre Sprache)
Reguläre Ausdrücke
Deterministisch endlicher Automat (DEA)

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gib1z1cwi

(b) Geben Sie zu den folgenden Wörtern eine Ableitung bzw. einen Syntaxbaum anhand der erstellten Grammatiken aus der Teilaufgabe a) an:

(i) zum Wort *abba* aus der Sprache L_0.

Lösungsvorschlag

$S \vdash aS \vdash abA \vdash abbB \vdash aabb$

(ii) zum Wort *baab* aus der Sprache L_1.

Lösungsvorschlag

$S \vdash bS \vdash baS \vdash baaS \vdash baab$

(iii) zum Wort *aabb* aus der Sprache L_2.

Lösungsvorschlag

$S \vdash aA \vdash aaB \vdash aabB \vdash aabb$

(c) Geben Sie zu allen Sprachen einen regulären Ausdruck an.

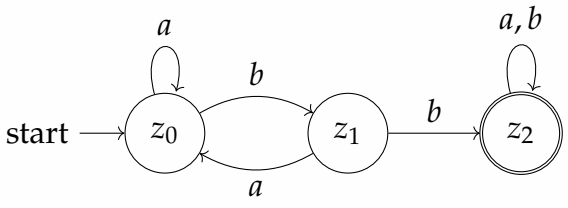
Lösungsvorschlag

$\mathbf{Reg}_0$ = `(a|b)*bb(a|b)*`
$\mathbf{Reg}_1$ = `(b|a)*b`
$\mathbf{Reg}_2$ = `(aa(a|b)*)|((a|b)*bb)`

(d) Geben Sie zu allen Sprachen einen Automaten an, der die Sprache akzeptiert.

Automat zu L_0 :

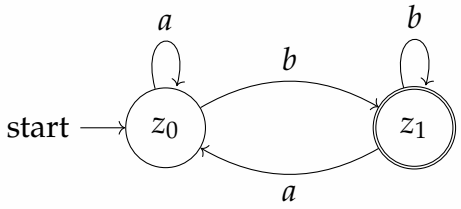
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af75ihbc7

Automat zu L_1 :

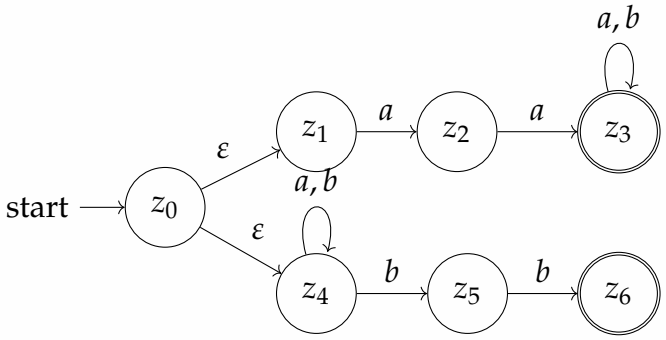
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A53w3wec9

Automat (NEA mit ε-Übergängen) zu L_2 :

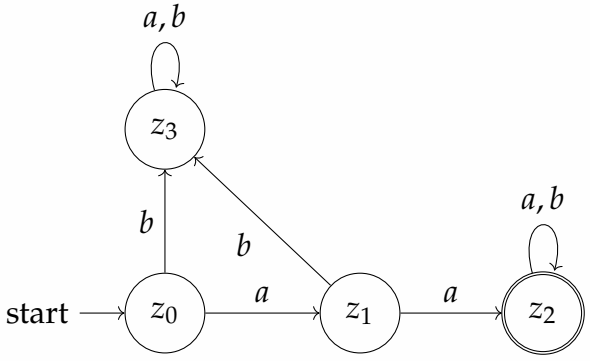
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj5awmjba

Teil-Automat (DEA Wort beginnt mit zwei a) zu L_2 :

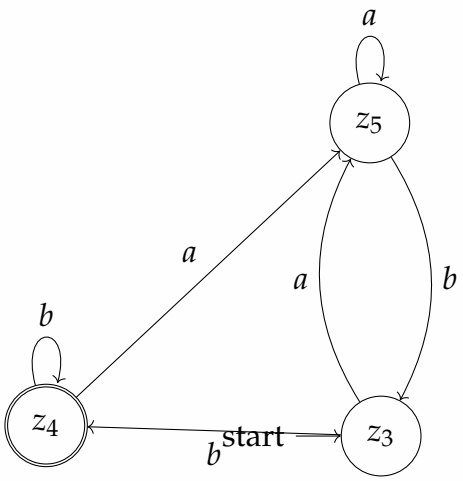
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apu1c40a9

Teil-Automat (DEA Wort endet auf zwei b) zu L_2 :

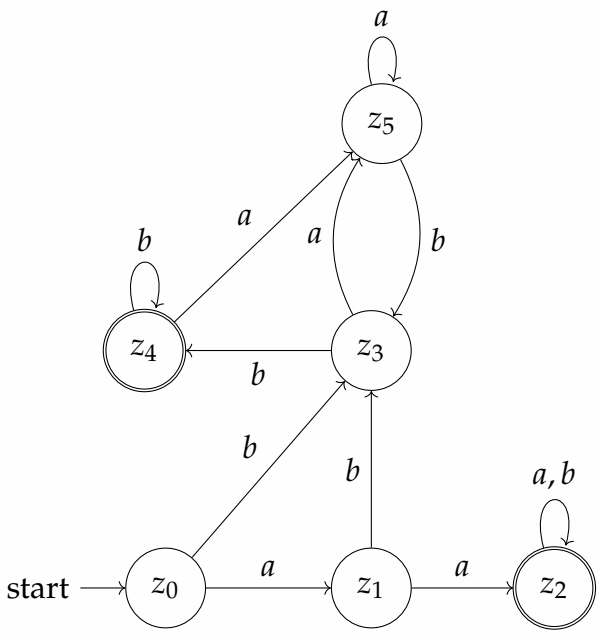
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj541j43w

Automat (DEA) zu L_2 :

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5ocw5ac2

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Regulaere-Ausdruecke/Aufgabe_Regulaere-Grammatik-regulaere-Ausdruecke-und-DEA.tex

Übungsaufgabe „Vorlesungsaufgaben“ (Reguläre Ausdrücke)

(a) Gegeben ist eine Sprache $L \subset \Sigma^*$ mit $\Sigma = \{a, b\}$. Zu der Sprache L gehören alle Wörter, die die Zeichenfolge `abba` beinhalten.

Geben Sie einen regulären Ausdruck für diese Sprache an („klassischer“ regulärer Ausdruck).

Lösungsvorschlag

$(a|b)*abba(a|b)*$

```
static String regexABBA = "(a|b)*abba(a|b)*";
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java

(b) Gebe möglichst einfache reguläre Ausdrücke für die folgenden Sprachen $L_x \subset \Sigma^*$ mit $\Sigma = \{a, b\}$ und $x \in \{1, 2, 3\}$ („klassischer“ regulärer Ausdruck).

$L_1 = \{x | x \text{ beinhaltet eine gerade Anzahl von } a\}$

Lösungsvorschlag

$b*(ab*ab*)*$

```
static String regexGeradeA = "b*(ab*ab*)*"; // Epsilon aa aaaa abba
↪  bbaa aabb bbb
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java

$L_2 = \{x | x \text{ beinhaltet eine ungerade Anzahl von } b\}$

Lösungsvorschlag

$a*ba*(ba*ba*)*$

```
static String regexUngeradeB = "a*ba*(ba*ba*)*"; // Epsilon b bbb
↪  abababa
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java

$L_3 = \{x | x \text{ beinhaltet an seinen geradzahligen Positionen ausschließlich } a\}$

Lösungsvorschlag

$((a|b)a)*(a*|b)$

```
static String regexGeradzahligA = "((a|b)a)*(a*|b)"; // aa ba aab b
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java

(c) Geben Sie einen regulären Ausdruck an, der eine syntaktisch gültige E-Mail-Adresse erkennt. (mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) vor dem `@`; mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) nach dem `@`; alle E-Mail-Adressen sollen auf `.de` oder `.com` enden.

Lösungsvorschlag

```
static String regexEMAIL = "\\w+@\\w+\\.(de|com)";
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java`

Lösungsvorschlag

```
public class TestRegularExpressions {
  // Hier bitte Lösungen der Aufgaben eintragen.
  static String regexABBA = "(a|b)*abba(a|b)*";
  static String regexGeradeA = "b*(ab*ab*)*"; // Epsilon aa aaaa abba bbaa aabb
↪  bbb
  static String regexUngeradeB = "a*ba*(ba*ba*)*"; // Epsilon b bbb abababa
  static String regexGeradzahligA = "((a|b)a)*(a*|b)"; // aa ba aab b
  static String regexEMAIL = "\\w+@\\w+\\.(de|com)";
  // Wenn die Lösungen stimmen, geben alle Tests true aus

  // Alternativen:
  // static String regexGeradzahligA = "((a|b)a)*((a|b)|)"; // aa ba aab b

  public static void main(String[] args) {
    testregexABBA();
    testregexGeradeA();
    testregexUngeradeB();
    testregexGeradzahligA();
    testregexEMAIL();
  }

  public static void testregexABBA() {
    boolean[] b = new boolean[7];
    b[0] = "abba".matches(regexABBA);
    b[1] = !"aba".matches(regexABBA);
    b[2] = "abbaabbaabba".matches(regexABBA);
    b[3] = "abababbaaabaaaabaaba".matches(regexABBA);
    b[4] = !"ab".matches(regexABBA);
    b[5] = !"bbb".matches(regexABBA);
    b[6] = !"".matches(regexABBA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
      System.out.println("Alle ABBA-Tests bestanden. Dein RegEx stimmt!");
    } else {
      for (int i = 0; i < b.length; i++) {
        if (!b[i]) {
          System.out.println("Test mit dem Index" + i +
↪  " leider nicht bestanden.");
        }
      }
    }
  }

  public static void testregexGeradeA() {
    boolean[] b = new boolean[7];
    b[0] = "aa".matches(regexGeradeA);
    b[1] = !"aaa".matches(regexGeradeA);
    b[2] = "abbaabbaabba".matches(regexGeradeA);
```

```
    b[3] = !"abababbaaabaaaabaaba".matches(regexGeradeA);
    b[4] = !"ab".matches(regexGeradeA);
    b[5] = "bbb".matches(regexGeradeA);
    b[6] = "".matches(regexGeradeA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
      System.out.println("Alle GeradeA-Tests bestanden. Dein RegEx stimmt!");
    } else {
      for (int i = 0; i < b.length; i++) {
        if (!b[i]) {
          System.out.println("Test mit dem Index" + i +
↪   " leider nicht bestanden.");
        }
      }
    }
  }

  public static void testregexUngeradeB() {
    boolean[] b = new boolean[7];
    b[0] = "b".matches(regexUngeradeB);
    b[1] = !"bb".matches(regexUngeradeB);
    b[2] = !"abbaabbaabba".matches(regexUngeradeB);
    b[3] = "abababbaaabaaaabaaba".matches(regexUngeradeB);
    b[4] = "ab".matches(regexUngeradeB);
    b[5] = "bbb".matches(regexUngeradeB);
    b[6] = !"".matches(regexUngeradeB);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
      System.out.println("Alle UngeradeB-Tests bestanden. Dein RegEx stimmt!");
    } else {
      for (int i = 0; i < b.length; i++) {
        if (!b[i]) {
          System.out.println("Test mit dem Index" + i +
↪   " leider nicht bestanden.");
        }
      }
    }
  }

  public static void testregexGeradzahligA() {
    boolean[] b = new boolean[7];
    b[0] = !"ab".matches(regexGeradzahligA);
    b[1] = "b".matches(regexGeradzahligA);
    b[2] = "babab".matches(regexGeradzahligA);
    b[3] = !"bababaab".matches(regexGeradzahligA);
    b[4] = "ba".matches(regexGeradzahligA);
    b[5] = "aaa".matches(regexGeradzahligA);
    b[6] = "".matches(regexGeradzahligA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {

↪   System.out.println("Alle GeradzahligeA-Tests bestanden. Dein RegEx stimmt!");
    } else {
      for (int i = 0; i < b.length; i++) {
        if (!b[i]) {
          System.out.println("Test mit dem Index" + i +
↪   " leider nicht bestanden.");
```

```
        }
      }
    }
  }

  public static void testregexEMAIL() {
    boolean[] b = new boolean[7];
    b[0] = "3@s.de".matches(regexEMAIL);
    b[1] = !"@0.de".matches(regexEMAIL);
    b[2] = "asdf@asdf.com".matches(regexEMAIL);
    b[3] = !"@.de".matches(regexEMAIL);
    b[4] = "s@1.com".matches(regexEMAIL);
    b[5] = !"a@a".matches(regexEMAIL);
    b[6] = !"".matches(regexEMAIL);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
      System.out.println("Alle Email-Tests bestanden. Dein RegEx stimmt!");
    } else {
      for (int i = 0; i < b.length; i++) {
        if (!b[i]) {
          System.out.println("Test mit dem Index" + i +
↪   " leider nicht bestanden.");
        }
      }
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/10_Typ-3_Regulaer/Regulaere-Ausdruecke/Aufgabe_Vorlesungsaufgaben-Regulaere-Ausdruecke.tex`

Übungsaufgabe „Reguläre Sprache in kontextfreier Sprache“ (Reguläre Sprache, Kontextfreie Sprache)

Reguläre Sprache
Kontextfreie Sprache

Zeigen Sie, dass sich eine reguläre Sprache ebenfalls als kontextfreie Sprache auffassen lässt.

Lösungsvorschlag

Im Grunde genommen kann ein DPDA einen deterministischen endlichen Automaten simulieren. Weil ein PDA einen Stack besitzen muss, erhält der PDA ein Symbol Z_0 auf seinem Stack. Der PDA ignoriert den Stack aber und arbeitet lediglich mit seinen Zuständen. Formal ausgedrückt, sei

$$A = (Q, \Sigma, \delta_A, q_0, F)$$

ein DFA. Wir konstruieren einen DPDA

$$P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F),$$

indem $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$ für alle Zustände p und q aus Q definiert wird, derart dass $\delta_A(q, a) = p$. Wir behaupten, dass $(q_0, w, Z_0) \to (p, \varepsilon, Z_0)$ genau dann, wenn $\delta_A(q_0, w) = p$. Das heißt, P simuliert A über seinen Zustand. Beide Richtungen lassen sich durch einfache Induktionsbeweise über $|w|$ zeigen. Da sowohl A als auch P akzeptieren, indem sie einen der Zustände aus F annehmen, schließen wir darauf, dass ihre Sprachen identisch sind.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Aufgabe_Regulaere-Sprache-in-kontextfreier-tex

Kontextfreie Sprache

Examensaufgabe „Sprachen L1 und L2“ (46115-2019-H.T1-A2)

(a) Betrachten Sie die folgenden Sprachen:

$L_1 = \{ a^n b^{2n} c^{2m} d^m \mid n, m \in \mathcal{N} \}$

$L_2 = \{ a^n b^{2n} c^{2n} d^n \mid n \in \mathcal{N} \}$

Zeigen Sie für Z1 und La, ob sie kontextfrei sind oder nicht. Für den Beweis von Kontext- Freiheit in dieser Frage reicht die Angabe eines Automaten oder einer Grammatik. (Beschrei-

ben Sie dann die Konstruktionsidee des Automaten oder der Grammatik.) Für den Beweis von Nicht-Kontext-Freiheit verwenden Sie eine der üblichen Methoden.

(b) Eine kontextfreie Grammatik ist in Chomsky-Normalform, falls die folgenden Bedingungen erfüllt sind:

- alle Regeln sind von der Form X — YZ oder X — o mit Nichtterminalzeichen X,Y, Z und Terminalzeichen o,
- alle Nichtterminalzeichen sind erreichbar vom Startsymbol und
- alle Nichtterminalzeichen sind erzeugend, d. h. für jedes Nichtterminalzeichen X gibt es ein Wort w über dem Terminalalphabet, so dass X =>* w.

Bringen Sie die folgende Grammatik in Chomsky-Normalform.

$$
\mathrm{P} = \left\{ \begin{array}{l}
S \to AAB \mid CD \mid abc \\
A \to AAAA \mid a \\
B \to BB \mid S \\
C \to CCC \mid CC \\
D \to d
\end{array} \right\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf7f9tp7z

Das Startsymbol der Grammatik ist S, das Terminalalphabet ist a,b,c,d und die Menge der Nichtterminalzeichen ist S,A,B,C,D.

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen. ———

$\varnothing$ Nichts zu tun

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

Eine rechte Seite in der C vorkommt, lässt sich wegen $\{C \to CCC \mid CC\}$ nicht ableiten, weil es zu einer Endlosschleife kommt. Wir entfernen die entsprechenden Regeln.

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to AAB \mid abc \\ A &\to AAAA \mid a \\ B &\to BB \mid AAB \mid abc \end{aligned} \Big\}$$

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt. ———

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to AAB \mid T_a T_b T_c \\ A &\to AAAA \mid a \\ B &\to BB \mid AAB \mid T_a T_b T_c \\ T_a &\to a \\ T_b &\to b \\ T_c &\to c \end{aligned} \Big\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. ———

$$\mathrm{P} = \Big\{$$

$$
\begin{aligned}
S &\to AS_1 \mid T_aS_2 \\
A &\to AA_1 \mid a \\
B &\to BB \mid AS_1 \mid T_aS_2 \\
T_a &\to a \\
T_b &\to b \\
T_c &\to c \\
S_1 &\to AB \\
S_2 &\to T_bT_c \\
A_1 &\to AA_2 \\
A_2 &\to AA
\end{aligned}
\Bigg\}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2019/09/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Nonterminale SABCD, Terminale ab“ (46115-2021-F.T1-TA1-A2)

Kontextfreie Sprache

(a) Verwenden Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus), um für die folgende kontextfreie Grammatik G = (V,D, P, 5) mit Variablen V=s,A,B,C,D, Terminalzeichen 3 = a,b, Produktionen

P= S -> SB | AC |a, A -> a, B -> b, C -> DD | AB, D -> AB | DC | CD

und Startsymbol S zu prüfen, ob das Wort aababababb in der durch G erzeugten Sprache liegt. Erläutern Sie dabei Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

(b) Mit a', wobei i € No = 0,1,2,..., wird das Wort bezeichnet, das aus der i-fachen Wiederholung des Zeichens a besteht (d. h. a' = € und a? = aa' !, wobei e das leere Wort ist).

Sei die Sprache L definiert über dem Alphabet a,b als

L = a"b™ | iE No, i > 1.

Zeigen Sie, dass die Sprache Z nicht vom Typ 3 der Chomsky-Hierarchie (d. h. nicht regulär) ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex

Examensaufgabe „Kontextfrei aber nicht regulär“ (66115-2012-F.T1-A3)

Kontextfreie Sprache

Beweisen Sie, dass folgende Sprache kontextfrei, aber nicht regulär ist.

$$C = \{ a^n b^m \mid n \geq m \geq 1 \}$$

Lösungsvorschlag

Nachweis Kontextfrei über Grammatik

$G = (\{S\}, \{a, b\}, P, S)$

$$\mathrm{P} = \left\{ \quad S \to aSb \mid aS \mid ab \quad \right\}$$

- Regel 1: aSb
- Regel 2: aS
- Regel 3: ab

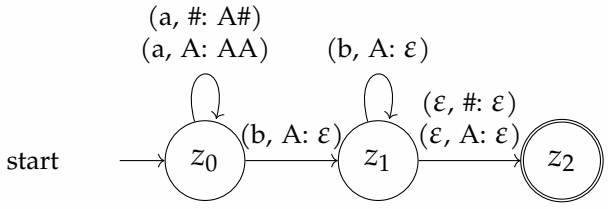
ab: $S \xrightarrow{3} ab$

$a^n b$: $S \xrightarrow[n-1]{2} a^{n-1}S \xrightarrow{3} a^{n-1}ab$

$a^n b^m$: $S \xrightarrow[m-1]{1} a^{m-1}Sb^{m-1} \xrightarrow[n-(m-1)]{2} a^{n-1}Sb^{m-1} \xrightarrow{3} a^n b^m$

$\Rightarrow L(G) = C$

Nachweis Kontextfrei über Kellerautomat



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aji151myg

Nachweis: C nicht regulär

C sei regulär
$\Rightarrow$ Pumping-Lemma für C erfüllt

j sei die Pumping-Zahl ($j \in \mathbb{N}$)
$\omega \in C$: $\omega = a^j b^j$
$\omega = uvw$
Dann gilt:

- $|v| \geq 1$
- $|uv| \leq j$
- $uv^i w \in C$ für alle $i \in \mathbb{N}_0$

In uv können nur a's vorkommen
$\Rightarrow$ In v muss mindestens ein a vorkommen
$\Rightarrow uv^0 w = a^l (a^{j-l})^0 b^j$ $((a^{j-l})^0 = \varepsilon)$
$\Rightarrow$ In ω' sind nur l viele a's, Da $l < j$, $\omega' \notin C$,
$\Rightarrow$ Widerspruch zur Annahme
$\Rightarrow$ C nicht regulär

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/03/Thema-1/Aufgabe-3.tex

Examensaufgabe „Nonterminale: SAB, Terminale: ab“ (66115-2012-F.T1-A4)

Chomsky-Normalform

Gegeben ist die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit $\Sigma = \{a, b\}, N = \{S, A, B\}$ und

$$
\mathrm{P} = \left\{ \begin{array}{l}
S \to A \\
S \to B \\
A \to aAb \\
B \to AA \\
B \to bBa \\
A \to a
\end{array} \right\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr3rgt2vg

Geben Sie eine äquivalente Grammatik in Chomsky-Normalform an.

Lösungsvorschlag

Kann auch so geschrieben werden:

$$
\mathrm{P} = \left\{ \begin{array}{l}
S \to A \mid B \\
A \to aAb \mid a \\
B \to AA \mid bBa
\end{array} \right\}
$$

(a) **Elimination der ε-Regeln**

— Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.

$\varnothing$ Nichts zu tun

(b) **Elimination von Kettenregeln**

— Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

$$
\mathrm{P} = \left\{ \begin{array}{l}
S \to aAb \mid a \mid AA \mid bBa \\
A \to aAb \mid a \\
B \to AA \mid bBa
\end{array} \right\}
$$

$$\}$$

(c) Separation von Terminalzeichen

— Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt. —

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to T_a A T_b \mid a \mid AA \mid T_b B T_a \\ A &\to T_a A T_b \mid a \\ B &\to AA \mid T_b B T_a \\ T_a &\to a \\ T_b &\to b \end{aligned}$$

$$\Big\}$$

(d) Elimination von mehrelementigen Nonterminalketten

— Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. ———————

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to T_a C \mid a \mid AA \mid T_b D \\ A &\to T_a C \mid a \\ B &\to AA \mid T_b D \\ T_a &\to a \\ T_b &\to b \\ C &\to A T_b \\ D &\to B T_a \end{aligned}$$

$$\Big\}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2012/03/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Kontextfreie Grammatiken“ (66115-2013-F.T1-A2)

Kontextfreie Sprache
CYK-Algorithmus

Gegeben sei die Grammatik $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ und

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to AB \\
S &\to CS \\
A &\to BC \\
A &\to BB \\
A &\to a \\
B &\to AC \\
B &\to b \\
C &\to AA \\
C &\to BA
\end{aligned}
\right\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr46a6j0a

$L = L(G)$ ist die von G erzeugte Sprache.

(a) Zeigen Sie, dass G mehrdeutig ist.

Lösungsvorschlag

Das Wort *baab* kann in zwei verschiedenen Ableitungen hergeleitet werden:

(i) $S \vdash AB \vdash BCB \vdash bCB \vdash bAAB \vdash baAB \vdash baaB \vdash baab$

(ii) $S \vdash CS \vdash BAS \vdash bAS \vdash baS \vdash baAB \vdash baaB \vdash baab$

(b) Entscheiden Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK), ob das Wort $w = babaaa$ zur Sprache L gehört. Begründen Sie Ihre Entscheidung.

Lösungsvorschlag

b	a	b	a	a	a
B	A	B	A	A	A
C	S	C	C	C	
-	B	A	B		
A	C	A,C			
A,C	B,C,A				
A,C,B					

$\Rightarrow babaaa \notin L(G)$

Das Startsymbol S ist nicht in der Zelle $V(1,5) = \{A, C, B\}$ enthalten.

(c) Geben Sie eine Ableitung für $w = babaaa$ an.

Lösungsvorschlag

$A \vdash BB \vdash bB \vdash bAC \vdash baC \vdash baAA \vdash baBCA \vdash babCA \vdash babAAA \vdash babaAA \vdash babaaA \vdash babaaa$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Nonterminale: SA, Terminale: 012“ (66115-2016-F.T1-A2)

Chomsky-Normalform

Betrachten Sie die folgende Grammatik $G = (\{S, A\}, \{0, 1, 2\}, P, S)$ mit

$$\mathrm{P} = \left\{ \begin{array}{l} S \to 0S0 \mid 1S1 \mid 2A2 \mid 0 \mid 1 \mid \varepsilon \\ A \to A2 \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf6scqja9

Konstruieren Sie für die Grammatik G schrittweise eine äquivalente Grammatik in Chomsky-Normalform. Geben Sie für jeden einzelnen Schritt des Verfahrens das vollständige Zwischenergebnis an und erklären Sie kurz, was in dem Schritt getan wurde.

Lösungsvorschlag

Die Regeln $\{S \to 2A2\}$ und $\{A \to A2\}$ können gelöscht werden, da es keine Regel $\{A \to \varepsilon\}$ oder $\{A \to S\}$ gibt. So erhalten wir:

$$\mathrm{P} = \left\{ \begin{array}{l} S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon \end{array} \right\}$$

(a) **Elimination der ε-Regeln**

— Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.

falls $S \to \varepsilon \in P$ neuen Startzustand S_1 einführen

$$\mathrm{P} = \left\{ \begin{array}{l} S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid 00 \mid 11 \\ S_1 \to \varepsilon \mid S \end{array} \right\}$$

(b) **Elimination von Kettenregeln**

— Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

$\varnothing$ Nichts zu tun

(c) Separation von Terminalzeichen

— Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt. —

N = Null E = Eins

$$
\mathrm{P} = \Big\{
\begin{aligned}
S &\to NSN \mid ESE \mid 0 \mid 1 \mid NN \mid EE \\
S_1 &\to \varepsilon \mid S \\
A &\to AZ \\
N &\to 0 \\
E &\to 1
\end{aligned}
\Big\}
$$

(d) Elimination von mehrelementigen Nonterminalketten

— Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1}B_n, A_{n-1} \to A_{n-2}B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. ———————

$$
\mathrm{P} = \Big\{
\begin{aligned}
S &\to NS_N \mid ES_E \mid 0 \mid 1 \mid NN \mid EE \\
S_1 &\to \varepsilon \mid S \\
S_N &\to SN \\
S_E &\to SE \\
N &\to 0 \\
E &\to 1
\end{aligned}
\Big\}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-1/Aufgabe-2.tex`

Examensaufgabe „Nonterminale: STU, Terminale: abcde“ (66115-2017-F.T2-A2)

Kontextfreie Sprache
Chomsky-Normalform

(a) Gegeben sei die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Sprache L(G), wobei $V = S, T, U$ und $\Sigma = \{a, b, c, d, e\}$. P bestehe aus den folgenden Produktionen:

$$\mathrm{P} = \left\{ \begin{array}{l} S \to U \mid SbU \\ T \to dSe \mid a \\ U \to T \mid UcT \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gib25c5oc

(i) Zeigen Sie $acdae \in L(G)$.

Lösungsvorschlag

$S \vdash U \vdash UcT \vdash TcT \vdash acT \vdash acdSe \vdash acdUe \vdash acdae$

(ii) Bringen Sie G in Chomsky-Normalform.

Lösungsvorschlag

i. **Elimination der ε-Regeln**
— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.* —

$\varnothing$ Nichts zu tun

ii. **Elimination von Kettenregeln**
— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.*

$$\mathrm{P} = \left\{ \begin{array}{l} S \to dSe \mid a \mid UcT \mid SbU \\ T \to dSe \mid a \\ U \to dSe \mid a \mid UcT \end{array} \right\}$$

iii. **Separation von Terminalzeichen**
— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* —

$$\mathrm{P} = \left\{$$

$$
\begin{aligned}
S &\to DSE \mid a \mid UCT \mid SBU \\
T &\to DSE \mid a \\
U &\to DSE \mid a \mid UCT \\
B &\to b \\
C &\to c \\
D &\to d \\
E &\to e
\end{aligned}
\Bigg\}
$$

iv. **Elimination von mehrelementigen Nonterminalketten**
— Alle Produktionen der Form $A \to B_1B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1}B_n, A_{n-1} \to A_{n-2}B_{n-1}, \dots, A_2 \to B_1B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. ———————————

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to DS_E \mid a \mid UC_T \mid SB_U \\
T &\to DS_E \mid a \\
U &\to DS_E \mid a \mid UC_T \\
B &\to b \\
C &\to c \\
D &\to d \\
E &\to e \\
S_E &\to SE \\
C_T &\to CT \\
B_U &\to BU
\end{aligned}
\Bigg\}
$$

(b) Geben Sie eine kontextfreie Grammatik für $L = \{ a^i b^k c^i | i, k \in \mathbb{N} \mid a \}$n.

Lösungsvorschlag

Wir interpretieren $\mathbb{N}$ als $\mathbb{N}_0$.

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to aSc \mid aBc \mid B \mid \varepsilon B \\
B &\to b \mid Bb
\end{aligned}
\Bigg\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghp3bfdtg

(c) Zeigen Sie, dass $L = \{ a^i b^k c^i | i, k \in \mathbb{N} \wedge i < k \mid n \}$icht kontextfrei ist, indem Sie das

Pumping-Lemma für kontextfreie Sprachen anwenden.

Pumping-Lemma (Kontextfreie Sprache)

Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei L eine kontextfreie Sprache. Dann gibt es eine Zahl j, sodass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ zerlegen lassen in $\omega = uvwxy$, sodass die folgenden Eigenschaften erfüllt sind:

(i) $|vx| \geq 1$ (Die Wörter v und x sind nicht leer.)

(ii) $|vwx| \leq j$ (Die Wörter v, w und x haben zusammen höchstens die Länge j.)

(iii) Für alle $i \in \mathbb{N}_0$ gilt $uv^iwx^iy \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iwx^iy in der Sprache L)

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/03/Thema-2/Aufgabe-2.tex`

Examensaufgabe „Kontextfreie Sprachen“ (66115-2017-H.T1-A2)

Kontextfreie Sprache

Betrachten Sie die Sprache $L_1 = L_a \cup L_b$.

- $L_a = \{ a^n bc^n \mid n \in \mathbb{N} \}$
- $L_b = \{ ab^m c^m \mid m \in \mathbb{N} \}$

(a) Geben Sie für L_1 eine kontextfreie Grammatik an.

Lösungsvorschlag

$$
\mathrm{P} = \Big\{
\begin{aligned}
S &\to S_a \mid S_b \\
S_a &\to aS_ac \mid b \\
S_b &\to a \mid aB_b \\
B_b &\to bB_bc \mid bc
\end{aligned}
\Big\}
$$

(b) Ist Ihre Grammatik aus a) eindeutig? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. Die Sprache ist nicht eindeutig. Für das Wort abc gibt es zwei Ableitungen, nämlich $S \vdash S_a \vdash aS_ac \vdash abc$ und $S \vdash S_b \vdash aB_b \vdash abc$.

(c) Betrachten Sie die Sprache $L_2 = \{ a^{2^n} \mid n \in \mathbb{N} \}$. Zeigen Sie, dass L_2 nicht kontextfrei ist.

Lösungsvorschlag

Annahme: L_2 ist kontextfrei

$\to$ Pumping-Lemma gilt für L_2

$\to j \in \mathbb{N}$ als Pumping-Zahl

$\omega \in L_2$: $|\omega| \geq j$

Konsequenz: $\omega = uvwxy$

- $|vx| \geq 1$
- $|vwx| \leq j$
- $uv^i wx^i y \in L_2$ für alle $i \in \mathbb{N}_0$

Wir wählen: $\omega = a^{2^i}$: $|\omega| \geq j$

p $a \ldots a$

r $a \ldots a$

s $a \ldots a$

t $a \dots a$

q $a \dots a$

$q + r + s + t + q = 2^j$

$\Rightarrow r + t \geq 1$

$r + s + t \leq j$

1. Fall

$r + t = 2^{j-1}$

$2^{j-1} + 2^{j-1} = 2 \cdot 2^{j-1} = 2^1 \cdot 2^{j-1} = 2^{1+j-1} = 2^j$

$\omega' = uv^2wx^2y$

$p + 2 \cdot r + s + 2 \cdot t + q$

$p + s + q + 2 \cdot (r + t)$

$2^{j-1} + 2 \cdot 2^{j-1} = 3 \cdot 2^{j-1} = 2^{j-1} + 2^i \leq 2^{j+1}$

keine Zweierpotenz

$\Rightarrow \omega \notin L_2$

$\Rightarrow$ Widerspruch zur Annahme

$\Rightarrow L_2$ nicht kontextfrei

2. Fall

$r + t \neq 2^{j-1}$

$\omega' = uv^0wx^0y$

$\Rightarrow p + s + q = 2^j - (r + t)$

$(r + t) \neq 2^{j-i}$

ist keine Zweierpotenz

$\Rightarrow \omega \notin L_2$

$\Rightarrow L_2$ nicht kontextfrei

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/09/Thema-1/Aufgabe-2.tex`

CYK-Algorithmus

Examensaufgabe „CYK mit fehlenden Zellen (T: SABC N: ab)“ (66115-2017-H.T2-A5)

Sei $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ die kontextfreie Grammatik in Chomsky-Normalform und der Menge P der Produktionen:

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to AB \mid BC \\ A &\to BA \mid a \\ C &\to AB \mid a \\ B &\to CC \mid b \end{aligned}$$

$$\Big\}$$

Sei $\omega = baaab$. Folgende Tabelle entsteht durch Anwendung des CYK-Algorithmus. Z. B. bedeutet $B \in V(3,5)$, dass aus der Variablen B das Teilwort $\omega_3\omega_4\omega_5 = aab$ hergeleitet werden kann. Drei Einträge wurden weggelassen.

(a) Bestimmen Sie die Mengen $V(1,2)$, $V(1,3)$ und $V(1,5)$.

Lösungsvorschlag

b	a	a	a	b
B	A,C	A,C	A,C	B
A,S	B	B	S,C	
-	S,C,A	B		
S,A,C	S,C			
S,C				

(b) Wie entnehmen Sie dieser Tabelle, dass $\omega \in L(G)$ ist?

Lösungsvorschlag

In der Menge $V(1,5)$ ist das Startsymbol S der Sprache $L(G)$ enthalten.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/09/Thema-2/Aufgabe-5.tex

Examensaufgabe „Kontextfreie Sprachen“ (66115-2018-H.T1-A3)

Kontextfreie Sprache
CYK-Algorithmus

(a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet $\Sigma = \{a, b, c\}$:

$$L = \{ wb^{3k}c^{2k+1}v \mid k \in \mathbb{N}, |w|_c = |u|_a \}$$

(Hierbei bezeichnet $|u|$, die Anzahl des Zeichens x in dem Wort u, und es gilt $0 \in \mathbb{N}$.) Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

(b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{S, X, Y, Z\}, \{z, y\}, P, S)$$

mit den Produktionen

$$\mathrm{P} = \left\{ \begin{array}{l} S \to ZX \mid y \\ X \to ZS \mid SS \mid x \\ Y \to SX \mid YZ \\ Z \to XX \mid XS \end{array} \right\}$$

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK) um zu zeigen, dass das Wort $xxxyx$ zu der von G erzeugten Sprache $L(G)$ gehört.

Lösungsvorschlag

x	x	x	y	x
X	X	X	S	X
Z	Z	Z	Y	
S	X	S		
Z,X	Z			
X,S,Z				

$\Rightarrow xxxyx \in L(G)$

(c) Geben Sie eine Ableitung des Wortes $xxxyx$ mit G an.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2018/09/Thema-1/Aufgabe-3.tex

Examensaufgabe „Nonterminale: STU, Terminale: ab“ (66115-2019-F.T1.A3)

Kontextfreie Sprache
Chomsky-Normalform

Gegeben sei die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Sprache $L(G)$, wobei $V = \{S, T, U\}$ und $\Sigma = \{a, b\}$. P bestehe aus den folgenden Produktionen:

$$\mathrm{P} = \left\{ \begin{aligned} S &\to TUUT \\ T &\to aT \mid \varepsilon \\ U &\to bUb \mid a \end{aligned} \right\}$$

[1]

(a) Geben Sie fünf verschiedene Wörter $w \in \Sigma^*$ mit $w \in L(G)$ an.

Lösungsvorschlag

- aa
- aaaa
- ababbaba
- aababbabaa
- abbabbbbabba

(b) Geben Sie eine explizite Beschreibung der Sprache $L(G)$ an.

Lösungsvorschlag

$L = \{ a^* b^n a b^{2n} a b^n a^* \mid n \in \mathbb{N}_0 \}$

(c) Bringen Sie G in Chomsky-Normalform und erklären Sie Ihre Vorgehensweise.

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen. —

$$\mathrm{P} = \left\{ \begin{aligned} S &\to TUUT \mid TUU \mid UUT \mid UU \\ T &\to aT \mid a \\ U &\to bUb \mid a \end{aligned} \right\}$$

(ii) **Elimination von Kettenregeln**

[1] `https://flaci.com/Gjpsin26a`

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.* —

∅ Nichts zu tun

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* ———————————————

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to TUUT \mid TUU \mid UUT \mid UU \\
T &\to AT \mid A \\
U &\to BUB \mid A \\
A &\to a \\
B &\to b
\end{aligned} \right\}
$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* ———————————

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to TS_1 \mid TS_3 \mid US_2 \mid UU \\
S_1 &\to US_2 \\
S_2 &\to UT \\
S_3 &\to UU \\
T &\to AT \mid a \\
U &\to BU_1 \mid a \\
U_1 &\to UB \\
A &\to a \\
B &\to b
\end{aligned} \right\}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/03/Thema-1/Aufgabe-3.tex`

Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-F.T1-A3)

Kontextfreie Sprache
CYK-Algorithmus

(a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet $\Sigma = \{a, b, c\}$:

$$L = \{ a^{3n+2} wvc^n \mid n \in \mathbb{N}_0, 2 \cdot |w|_b = |v|_a \}$$

(Hierbei bezeichnet $|u|_x$, die Anzahl des Zeichens x in dem Wort u.)

Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

Lösungsvorschlag

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aaaSc \mid aaaAc \\ A \to aaB \\ B \to bBaa \mid baa \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghhs1xexw

(b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{A, B, C, D\}, \{a, b, c\}, P, A)$$

mit den Produktionen

$$\mathrm{P} = \left\{ \begin{array}{l} A \to AB \mid CD \mid a \\ B \to CC \mid c \\ C \to DC \mid CB \mid b \\ D \to DB \mid a \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf7556jn2

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK), um zu zeigen, dass das Wort *abcab* zu der von G erzeugten Sprache $L(G)$ gehört.

Lösungsvorschlag

Ableitung (Kontextfreie Sprache)

a	b	c	a	b
A,D	C	B	A,D	C
C	C	-	C	
C,C	A	-		
A,A	B			
A,D,B,B				

$\Rightarrow abcab \in L(G)$

(c) Finden Sie nun ein größtmögliches Teilwort von *abcab*, dass von keinem der vier Nichtterminale von *G* ableitbar ist.

(d) Geben Sie eine Ableitung des Wortes *abcab* mit *G* an.

Lösungsvorschlag

$A \vdash AB \vdash ACC \vdash ACBC \vdash ACBDC \vdash aCBDC \vdash abBDC \vdash abcDC \vdash abcaC \vdash abcab$

(e) Beweisen Sie, dass die folgende formale Sprache über Z = a,b nicht kon- textfrei ist: , L=a”bneN.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-1/Aufgabe-3.tex`

Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-F.T2-A3)

Kontextfreie Grammatik
Kontextfreie Sprache

(a) Ist die folgende Sprache $L_1 = \{ a^{n+2}b^{2n+1} \mid n \geq 2 \}$ über dem Alphabet $\Sigma = \{a, b\}$ kontextfrei?

Falls ja, geben Sie eine kontextfreie Grammatik für L_1, an, falls nein, eine kurze Begründung (ein vollständiger Beweis ist hier nicht gefordert).

Lösungsvorschlag

L_1 ist kontextfrei

$G = (\{S, A, B\}, \{a, b\}, P, S)$

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to aAbb \\ A &\to aAbb \mid aBbb \\ B &\to aab \end{aligned}$$

$$\Big\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxk1oczg

$n = 2$ 4a 5b: aaaabbbbb

$n = 3$ 5a 7b: aaaaabbbbbbb

$n = 4$ 6a 9b: aaaaaabbbbbbbbb

(b) Geben Sie einen Kellerautomaten (PDA) formal an, der die Sprache

$L_1 = \{ w_1 w_2 w_3 \mid w_1, w_2, w_3 \in \Sigma^* \backslash \{\lambda\}$ und $w_1 = w_3^{\mathrm{rev}} \} \in CFL$ über dem Alphabet $\Sigma = \{0, 1\}$ akzeptiert.

Dabei bezeichnet λ das leere Wort und w_3^{rev} bezeichnet das Wort w_3 rückwärts gelesen. Bei Akzeptanz einer Eingabe soll sich der PDA in einem Endzustand befinden und der Keller geleert sein.

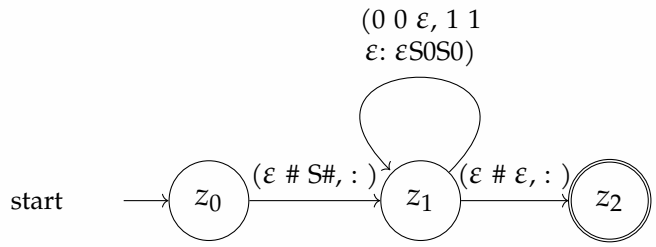
Lösungsvorschlag

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpkctmk3g

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to 0S0 \mid 1S1 \mid 0A0 \mid 1A1 \\ A &\to 0A \mid 1A \mid 0 \mid 1 \end{aligned}$$

$$\Big\}$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5z2zfkdw

(c) Beschreiben Sie in Worten die Arbeitsweise Ihres PDA aus Aufgabenteil (b).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Kontextfreie Sprachen“ (66115-2020-H.T2-TA1-A2)

Kontextfreie Sprache

(a) Sei $L = \{ 0^n 1^m 1^p 0^q \mid n + m = p + q \text{ und } n, m, p, q \in \mathbb{N}_0 \}$. Geben Sie eine kontextfreie Grammatik für L an. Sie dürfen dabei ε-Produktionen der Form $\{A \to \varepsilon\}$ verwenden.

Lösungsvorschlag

$$
\mathrm{P} = \Big\{
\begin{aligned}
S &\to 0S0 \mid 0A0 \mid 0B0 \mid \varepsilon \mid A \mid B \mid C \\
A &\to 0A1 \mid 0C1 \\
B &\to 1B0 \mid 1C0 \\
C &\to 1C1 \mid \varepsilon
\end{aligned}
\Big\}
$$

(b) Für eine Sprache L sei $L^r = \{ x^r \mid x \in L \}$ die Umkehrsprache. Dabei bezeichne x^r das Wort, das aus r entsteht, indem man die Reihenfolge der Zeichen umkehrt, beispielsweise $(abb)^r = bba$.

(i) Sei L eine kontextfreie Sprache. Zeigen Sie, dass dann auch L^r kontextfrei ist.

(ii) Geben Sie eine kontextfreie Sprache L_1, an, sodass $L_1 \cap L_1^r$ kontextfrei ist.

(iii) Geben Sie eine kontextfreie Sprache L_2, an, sodass $L_2 \cap L_2^r$ nicht kontextfrei ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „CYK mit Wort „aaaccbbb““ (66115-2021-F.T1-TA1-A2)

CYK-Algorithmus

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik mit Variablen $V = \{S, A, B, C, D\}$, Terminalzeichen $\Sigma = \{a, b, c\}$, Produktionen

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to AD \mid CC \mid c \\
A &\to a \\
B &\to b \\
C &\to CC \mid c \\
D &\to SB \mid CB
\end{aligned}
\right\}
$$

und Startsymbol S. Führen Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus) für G und das Wort *aaaccbbb* aus. Liegt *aaaccbbb* in der durch G erzeugten Sprache? Erläutern Sie Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

Lösungsvorschlag

a	a	a	c	c	b	b	b
-	-	-	S,C	D,D	-	-	
-	-	-	D,D	-	-		
-	-	S,S	-	-			
-	-	D,D	-				
-	S,S	-					
-	D,D						
S,S							

Das Wort *aaaccbbb* liegt in der Sprache.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-1/Teilaufgabe-1/Aufgabe-2.tex`

Examensaufgabe „w w1 w w2“ (66115-2021-F.T2-TA1-A2)

Kontextfreie Sprache
Pumping-Lemma (Kontextfreie Sprache)

(a) Zeigen Sie, dass die Sprache

$$L = \{\, ww_1ww_2 \mid w, w_1, w_2 \in \{a, b, c\}^* \text{ und } 2|w| \geq |w_1| + |w_2| \,\}$$

nicht kontextfrei ist.

> **Exkurs: Pumping-Lemma für Kontextfreie Sprachen**
>
> Es sei L eine kontextfreie Sprache. Dann gibt es eine Zahl j, sodass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ zerlegen lassen in $\omega = uvwxy$, sodass die folgenden Eigenschaften erfüllt sind:
>
> (i) $|vx| \geq 1$ (Die Wörter v und x sind nicht leer.)
>
> (ii) $|vwx| \leq j$ (Die Wörter v, w und x haben zusammen höchstens die Länge j.)
>
> (iii) Für alle $i \in \mathbb{N}_0$ gilt $uv^iwx^iy \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iwx^iy in der Sprache L)

Lösungsvorschlag

Es gibt eine Pumpzahl. Sie sei j. $a^jb^ja^jc^j$ ist ein Wort aus L, das sicher länger als j ist. Außerdem gilt $2|a^j| \geq |b^j| + |c^j|$. Unser gewähltes Wort ist deshalb in L.
Da $|vwx| \leq j$ und $|xv| \geq 1$ sein muss, liegt vwx entweder in w, w_1 oder w_2.

Aufteilung: vwx in w (erstes w):

u : ε

v : a

w : a^{j-2}

x : a

y : $b^ja^jc^j$

Es gilt $uv^iwx^iy \notin L$ für alle $i \in \mathbb{N}_0$, da $a^jb^ja^jc^j \notin L$ für $i = 0$, da $|a^{j-2}| + |a^j| < |b^j| + |c^j|$

Aufteilung: vwx in w (zweites w):

u : a^jb^j

v : a

w : a^{j-2}

x : a

y : c^j

Es gilt $uv^iwx^iy \notin L$ für alle $i \in \mathbb{N}_0$, da $a^jb^ja^jc^j \notin L$ für $i = 0$, da $|a^j| + |a^{j-2}| < |b^j| + |c^j|$

Aufteilung: vwx **in** w_1**:**

u : a^j

v : b

w : b^{j-2}

x : b

y : $a^j c^j$

Es gilt nicht $uv^i wx^i y \in L$ für alle $i \in \mathbb{N}_0$, da $a^j b^j a^j c^j \notin L$ für alle $i > 2$ da $2|a^j| < |b^{j-2+2i}| + |c^j|$ für alle $i > 2$

Aufteilung: vwx **in** w_2**:**

Analog zur Aufteilung vwx in w_1

$\Rightarrow L$ ist nicht kontextfrei.

(b) Betrachten Sie die Aussage

Seien $L_1, \ldots, I_n$ beliebige kontextfreie Sprachen.
Dann ist $\bigcap_{i=1}^{n}, L_i$ immer eine entscheidbare Sprache.

Entscheiden Sie, ob diese Aussage wahr ist oder nicht und begründen Sie Ihre Antwort.

Lösungsvorschlag

Diese Aussage ist falsch.

Kontextfreie Sprachen sind nicht abgeschlossen unter dem Schnitt, ðdie Schnittmenge zweier kontextfreier Sprachen kann in einer Sprache eines anderen Typs in der Chomsky Sprachen-Hierachie resultieren. Entsteht durch den Schnitt eine Typ-0-Sprache, dann ist diese nicht entscheidbar.

(c) Sei $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ die Menge der nicht negativen natürlichen Zahlen. Es ist bekannt, dass $L = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ keine kontextfreie Sprache ist. Ist die Komplementsprache $L_5 = \{a, b, c\}^* \setminus L$ kontextfrei? Begründen Sie Ihre Antwort.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex`

Übungsaufgabe „Ableitungen“ (Ableitung (Kontextfreie Sprache), Ableitungsbaum)

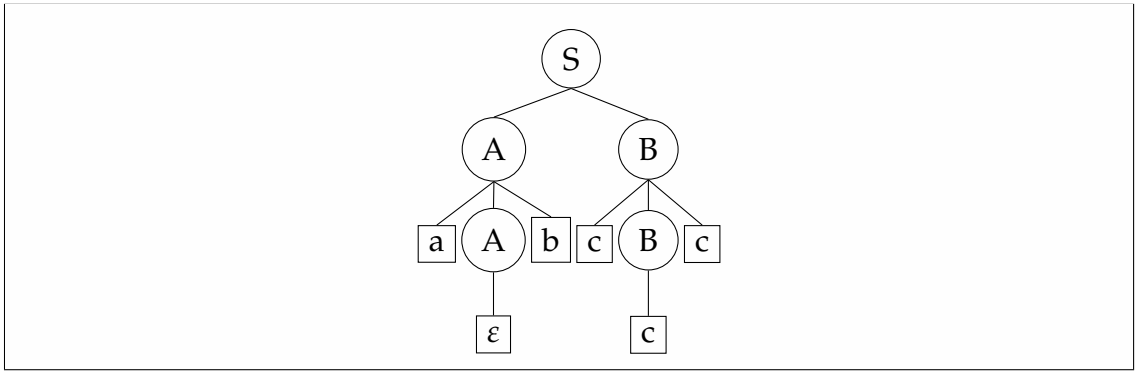
Ableitung (Kontextfreie Sprache)
Ableitungsbaum

Bestimmen Sie für die folgende Grammatik jeweils für die angegebenen Wörter w_i mehrere Ableitungen sowie Parsebäume, wenn dies nicht eindeutig möglich ist.

$$\mathrm{P} = \Big\{ \quad S \to AB \quad A \to aAb \mid \varepsilon \quad B \to cBc \mid c \quad \Big\}$$

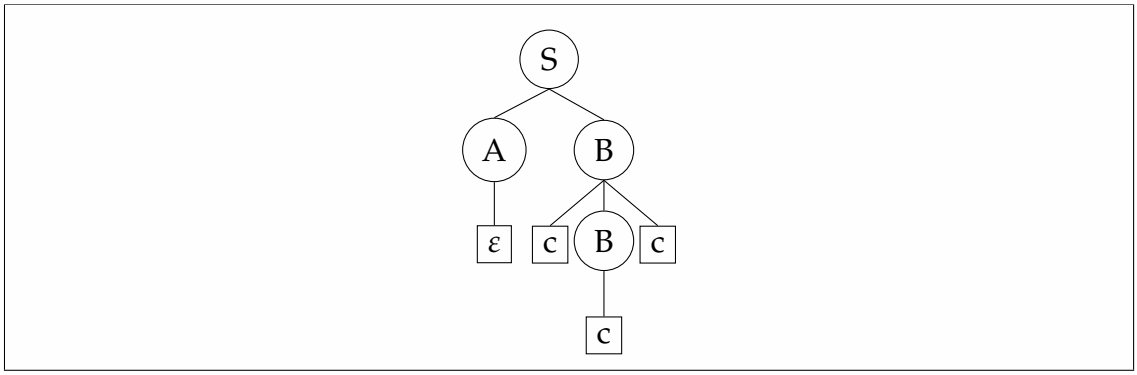
(a) $w_1 = abccc$

Lösungsvorschlag



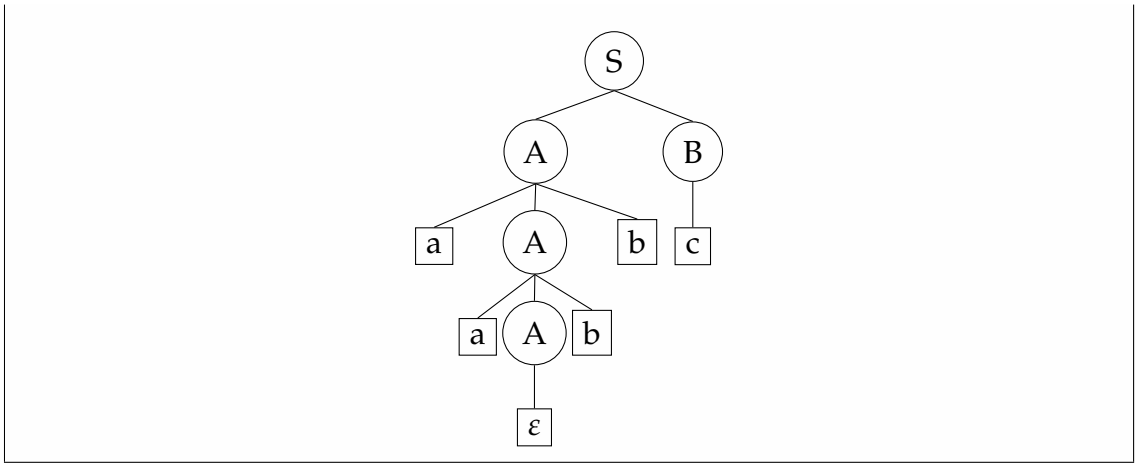
(b) $w_2 = ccc$

Lösungsvorschlag



(c) $w_3 = aabbc$

Lösungsvorschlag



Welche Sprache wird durch die Grammatik erzeugt?

Lösungsvorschlag

Die Sprache beinhaltet alle Wörter, die gleich viele *a*'s gefolgt von gleich vielen *b*'s und auf ungeradzahlig viele *c*'s endet.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Ableitung-Ableitungsbaum/Aufgabe_Ableitungen.tex`

Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache, Ableitung (Kontextfreie Sprache), Ableitungsbaum)

Kontextfreie Sprache
Ableitung (Kontextfreie Sprache)
Ableitungsbaum

(a) Erstelle eine Ableitung und einen Parsebaum für die folgende Grammatik für das Wort

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf6s60uxg

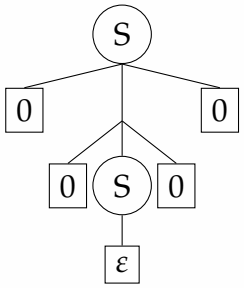
$G = (\{P\}, \{0,1\}, P, S)$

$$\mathrm{P} = \left\{ S \to \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1 \right\}$$

- 0000

Lösungsvorschlag

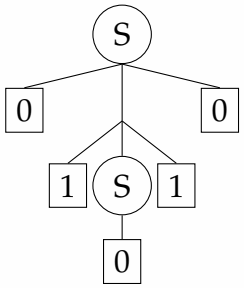
$S \vdash 0S0 \vdash 00S00 \vdash 0000$



- 01010

Lösungsvorschlag

$S \vdash 0S0 \vdash 01S10 \vdash 01010$



(b) Erstelle eine Ableitung und einen Parsebaum für die nebenstehende Grammatik für das Wort

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpmohr81a

$V = \{S, A, B\}$

$\Sigma = \{0, 1\}$

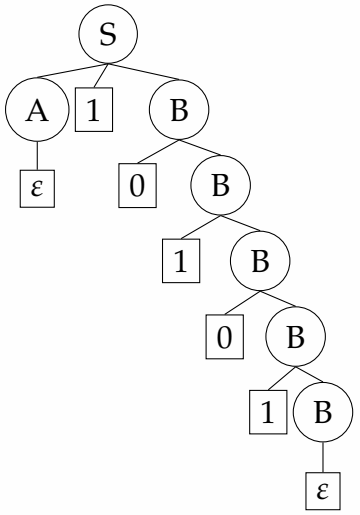
$$P = \Big\{ \begin{array}{l} S \to A1B \\ A \to 0A \mid \varepsilon \\ B \to 0B \mid 1B \mid \varepsilon \end{array} \Big\}$$

$S = S$

- 10101

Lösungsvorschlag

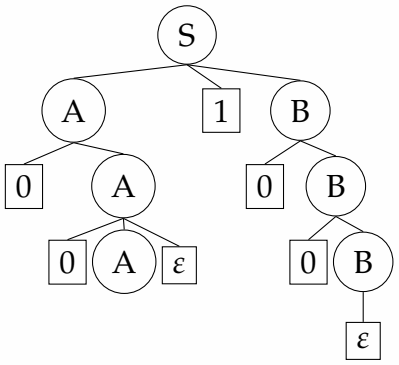
$S \vdash A1B \vdash 1B \vdash 10B \vdash 101B \vdash 1010B \vdash 10101B \vdash 10101$



- 00100

Lösungsvorschlag

$S \vdash A1B \vdash 0A1B \vdash 00A1B \vdash 001B \vdash 0010B \vdash 00100B \vdash 00100$



(c) Sind die Parsebäume eindeutig?

Lösungsvorschlag

Ja, die Parsebäume sind eindeutig.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Ableitung-Ableitungsbaum/Aufgabe_Vorlesungsaufgabe.tex

Übungsaufgabe „Klammerausdrücke“ (Kontextfreie Sprache)

Kontextfreie Sprache

In Programmierumgebungen kommen Abfolgen von Klammern, runde, eckige und geschweifte, vor. Diese müssen in der richtigen Abfolge auf- bzw. geschlossen werden. Eine korrekte Abfolge von Klammern wäre zum Beispiel:

```
{[](){(){[](())}}}{}
```

(a) Entwerfen Sie eine Grammatik, die die korrekte Abfolge solcher Klammerfolgen beschreibt.

Lösungsvorschlag

```
S -> { S } | ( S ) | [ S ] | S S | EPSILON
```

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghjeb39xr

(b) Geben Sie eine Ableitung für den oben angegebenen Klammerausdruck an.

Lösungsvorschlag

Der oben angegebene Klammerausdruck mit Einrückungen

```
01 {
02   []
03   ()
04   {
05     ()
06     {
07       []
08       (
09         []
10       )
11     }
12   }
13 }
14 {}
```

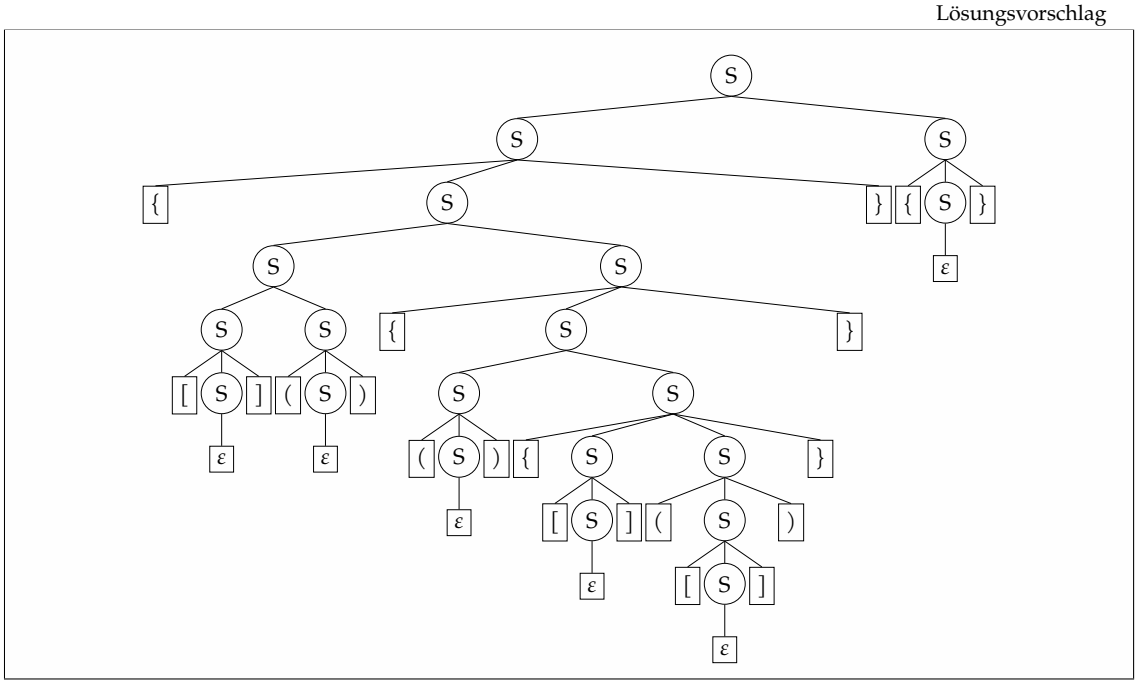
Die Ableitung:

```
S ->
SS ->
{S}S ->                    siehe Zeile 01 13
{SS}S ->
{[S]S}S ->                 siehe Zeile 02
{[]S}S ->
{[]SS}S ->
{[](S)S}S ->               siehe Zeile 03
{[](){S}}S ->              siehe Zeile 04 12
{[](){SS}}S ->
{[](){(S)S}}S ->
{[](){()S}}S ->            siehe Zeile 05
{[](){(){S}}}S ->          siehe Zeile 06 11
{[](){(){SS}}}S ->
{[](){(){[S]S}}}S ->       siehe Zeile 07
```

```
{[](){(){[]S}}}S ->
{[](){(){[](S)}}}S ->      siehe Zeile 08 10
{[](){(){[]([S])}}}S ->    siehe Zeile 09
{[](){(){[]([])}}}S ->
{[](){(){[]([])}}}{S} ->   siehe Zeile 14
{[](){(){[]([])}}}{}
```

(c) Zeichnen Sie eine Ableitungsbaum für den oben angegebenen Klammerausdruck.

Lösungsvorschlag



Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Aufgabe_Klammerausdruecke.tex`

Übungsaufgabe „Kontextfreie-Grammatik“ (Kontextfreie Grammatik) [Kontextfreie Grammatik]

Geben Sie für die folgenden Sprachen eine kontextfreie Grammatik an:

- Alle Wörter der Sprachen bestehen aus a's, gefolgt von gleich vielen b's.

Lösungsvorschlag

$$\mathrm{P} = \left\{ \quad S \to aSb \mid \varepsilon \quad \right\}$$

- Die Wörter der Sprache bestehen aus gleich vielen x wie y.

Lösungsvorschlag

$$\mathrm{P} = \left\{ \begin{array}{l} S \to xSY \mid ySX \mid \varepsilon \\ X \to xS \\ Y \to yS \end{array} \right\}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Aufgabe_Kontextfreie-Grammatik.tex`

Übungsaufgabe „(an bn)m“ (Kontextfreie Sprache, Kellerautomat)

Kontextfreie Sprache

Gegeben ist die Grammatik $G = (\{S, A, B\}, \{a, b\}, P, S)$ und den Produktionen

$$
\mathrm{P} = \left\{ \begin{aligned} S &\to SAB \mid \varepsilon \\ BA &\to AB \\ AA &\to aa \\ BB &\to bb \end{aligned} \right\}
$$

(a) Geben Sie einen Ausdruck an, der die Wörter der Sprache beschreibt.

Lösungsvorschlag

$L = \{(a^n b^n)^m \mid m \in \mathbb{N}_0 \text{ und } n \in \text{ gerade Zahlen}\}$

Einige Testableitungen um die Grammatik in Erfahrung zu bringen:

„.“ nur als optische Stütze nach 4 Zeichen eingefügt.

Mit 4 Buchstaben

$S \vdash SAB \vdash SABAB \vdash ABAB \vdash AABB \vdash aabb$

Mit 6 Buchstaben

$S \vdash ... \vdash ABAB.AB \vdash AABB.AB \vdash AABA.BB \vdash AAAB.BB \vdash \varnothing$

Mit 8 Buchstaben

$S \vdash ... \vdash ABAB.ABAB \vdash ... \vdash aabb.aabb$

$S \vdash ... \vdash ABAB.ABAB \vdash ... \vdash AABB.AABB \vdash AABA.BABB \vdash AABA.ABBB \vdash AAAB.ABBB \vdash AAAA.BBBB \vdash aaaa.bbbb$

Mit 12 Buchstaben

$S \vdash ... \vdash ABAB.ABAB.ABAB \vdash ... \vdash aabb.aabb.aabb$

$S \vdash ... \vdash ABAB.ABAB.ABAB \vdash AAAA.BBBB.AABB \vdash aaaa.bbbb.aabb$

$S \vdash ... \vdash ABAB.ABAB.ABAB \vdash AABB.ABAB.ABAB \vdash AABA.BBAB.ABAB \vdash AAAB.BBAB.ABAB... \vdash aaaa.aabb.bbbb$

(b) Geben Sie eine kontextfreie Grammatik G' an, für die gilt: $L(G') = L(G)$

Lösungsvorschlag

$$P = \left\{ \quad S \rightarrow aaSbb \mid SS \mid \varepsilon \quad \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grn19rt8w

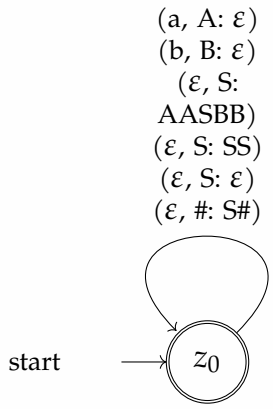
(c) Geben Sie einen Kellerautomaten an, der die Sprache akzeptiert.

Kellerautomat

Lösungsvorschlag

1. Kellerautomat (aus der Grammtik abgeleitet)

$K = (\{z_0\}, \{a, b\}, \{\#, S, A, B\}, \delta, z_0, \#, \{z_0\})$

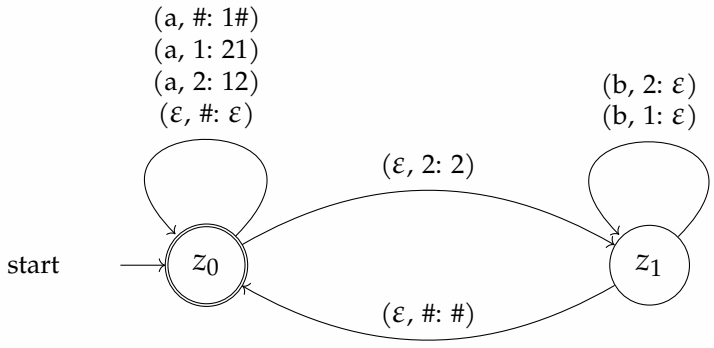


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Araj960s2

2. Kellerautomat

$K = (\{z_0, z_1\}, \{a, b\}, \{\#, 1, 2\}, \delta, z_0, \#, \{z_0\})$

Bemerkung zum Kelleralphabet: 1 steht für $1A$, also ein a befindet sich im Keller, und 2 steht für $2A$, also zwei a befinden sich im Keller.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahfqseouz

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Aufgabe_Kontextfreie-Sprache.tex

Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache)

Kontextfreie Sprache

(a) Erstelle eine (deterministische) Grammatik für Palindrome, für die ein DPDA existiert.

$L = \{ w\$w^R \mid w \in (a|b)^* \}$

(b) Wandle diese Grammatik in einen DPDA um.

Überführe die folgenden kontextfreien Grammatiken in CNF

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to ABC \\
A &\to aCD \\
B &\to bCD \\
C &\to D \mid \varepsilon \\
D &\to C
\end{aligned}
\right\}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Aufgabe_Vorlesungsaufgabe.tex

Übungsaufgabe „AB5“ (CYK-Algorithmus)

CYK-Algorithmus

Teste jeweils mit dem CYK-Algorithmus, ob das angegebene Wort zur Sprache der Grammatik gehört.

$$
\mathrm{P} = \Big\{
\begin{aligned}
S &\to SS \mid R_aA \mid R_bB \mid R_cC \\
A &\to R_bR_c \mid R_cR_b \\
B &\to R_aR_c \mid R_cR_a \\
C &\to R_aR_b \mid R_bR_a \\
R_a &\to a \\
R_b &\to b \\
R_c &\to c
\end{aligned}
\Big\}
$$

(a) $\omega_1 = acbcab$

Lösungsvorschlag

a	c	b	c	a	b
R_a	R_c	R_b	R_c	R_a	R_b
B	A	A	B	C	
S	-	S	S		
-	-	-			
-	-				
S					

$\Rightarrow \omega_1 \in L(G)$

(b) $\omega_2 = cabb$

Lösungsvorschlag

c	a	b	b
R_c	R_a	R_b	R_b
B	C	-	
S	-		
-			

$\Rightarrow \omega_2 \notin L(G)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/CYK-Algorithmus/Aufgabe_AB5.tex

Übungsaufgabe „CYK-Algorithmus“ (CYK-Algorithmus)

CYK-Algorithmus

Teste jeweils mit dem CYK-Algorithmus, ob das angegebene Wort zur Sprache der Grammatik gehört.

(a) Grammatik G_1:

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to SS \mid R_aA \mid R_bB \mid R_cC \\
A &\to R_bR_c \mid R_cR_b \\
B &\to R_aR_c \mid R_cR_a \\
C &\to R_aR_b \mid R_bR_a \\
R_a &\to a \\
R_b &\to b \\
R_c &\to c
\end{aligned}
\right\}
$$

(i) $w_1 = abccab$

(ii) $w_2 = abcba$

(b) Grammatik G_2:

$$
\mathrm{P} = \left\{
\begin{aligned}
Z &\to XD \mid XA \mid \varepsilon \\
S &\to XD \mid XA \\
A &\to V_aD \mid V_aA \mid AB \mid c \\
B &\to BB \mid CC \mid c \mid a \\
C &\to CC \mid c \\
D &\to AB \\
X &\to V_aD \mid V_aA \mid AB \mid b \mid c \\
V_a &\to a
\end{aligned}
\right\}
$$

(i) $w_3 = bacac$

Lösungsvorschlag

b	a	c	a	c
X	B,V_a	A,B,C,X	B,V_a	A,B,C,X
-	B,A,X	A,D,X,B	B,A,X	
Z,S	A,B,D,X	B,A,X,D,S,Z		
Z,S	A,B,D,S,X,Z			
Z,S				

(ii) $w_4 = baca$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/CYK-Algorithmus/Aufgabe_CYK-Algorithmus.tex

Übungsaufgabe „Foliensatz“ (CYK-Algorithmus)

CYK-Algorithmus

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to AB \mid BT \\
A &\to BA \mid a \\
B &\to TT \mid b \\
T &\to AB \mid a
\end{aligned}
\right\}
$$

Lösungsvorschlag

b	a	a	a	b
B	A,T	A,T	A,T	B
A,S	B	B	S,T	
-	S,T,A	B		
S,A,T	S,T			
S,T				

$\Rightarrow baaab \in L(G)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/CYK-Algorithmus/Aufgabe_Foliensatz.tex`

Übungsaufgabe „Youtube-Video Karsten-Morisse“ (CYK-Algorithmus)

CYK-Algorithmus

$G = (\{S, A, B, C\}, \{a, b\}, P, S)$ [2]

$$\mathrm{P} = \Big\{$$

$$\begin{aligned} S &\to AB \mid BC \\ A &\to BA \mid a \\ B &\to CC \mid b \\ C &\to AB \mid a \end{aligned}$$

$$\Big\}$$

Lösungsvorschlag

a	b	a	a	b
A,C	B	A,C	A,C	B
S,C	A,S	B	S,C	
B	-	B		
S,A	-			
S,S,C				

$\Rightarrow abaab \in L(G)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/CYK-Algorithmus/Aufgabe_Youtube-Video-Karsten-Morisse.tex`

[2] `https://www.youtube.com/watch?v=Q5TvCyu4RUo`

Übungsaufgabe „Drei Grammatiken (SABCX, ST, SAB)“ (Chomsky-Normalform)

Chomsky-Normalform

Überführen Sie jeweils die angegebene kontextfreie Grammatik in Chomsky-Normalform.

(a) $G = (\{S, A, B, C, X\}, \{a, b, c\}, P, S)$ mit P:

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to XAB \mid \varepsilon \\ A &\to aAB \mid AB \mid c \\ B &\to BB \mid C \mid a \\ C &\to CC \mid c \mid \varepsilon \\ X &\to A \mid b \end{aligned} \Big\}$$

(b) $G = (\{S, T\}, \{a, b, c\}, P, S)$ mit P:

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to aSbS \mid T \\ T &\to cT \mid c \end{aligned} \Big\}$$

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.* —

$\varnothing$ Nichts zu tun

(ii) **Elimination von Kettenregeln**

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.* —

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to aSbS \mid cT \mid c \\ T &\to cT \mid c \end{aligned} \Big\}$$

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* —

$$\mathrm{P} = \Big\{$$
$$S \to ASAS \mid CT \mid c$$
$$T \to CT \mid c$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$\Big\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1}B_n, A_{n-1} \to A_{n-2}B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* ————————

$$\mathrm{P} = \Big\{$$
$$S \to AU \mid CT \mid c$$
$$T \to CT \mid c$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$U \to SVV \qquad \to AS$$
$$\Big\}$$

(c) $G = (\{S, A, B\}, \{a, b, c\}, P, S)$ mit P:

$$\mathrm{P} = \Big\{$$
$$S \to AB$$
$$A \to aAA \mid \varepsilon$$
$$B \to bBB \mid \varepsilon$$
$$\Big\}$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Chomsky-Normalform/Aufgabe_Chomsky-Normalform.tex`

Übungsaufgabe „Vorlesungsaufgabe (S, SAB, SABCD)“ (Chomsky-Normalform)

Chomsky-Normalform

Überführen Sie die folgenden kontextfreien Grammatiken in CNF (Chomsky-Normalform).

(a) $\mathrm{P} = \left\{ S \to 0S1 \mid \varepsilon \right\}$

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.* —

$\mathrm{P} = \left\{ S \to 0S1 \mid 01 \right\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghje1ygz9

(ii) **Elimination von Kettenregeln**

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.* —

$\varnothing$ Nichts zu tun

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* —

N = Null, E = Eins

$$\mathrm{P} = \left\{ \begin{aligned} S &\to NSE \mid NE \\ N &\to 0 \\ E &\to 1 \end{aligned} \right\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* —

$$\mathrm{P} = \Big\{ \quad S \to NR \mid NE \quad R \to SE \quad N \to 0 \quad E \to 1 \quad \Big\}$$

(b) $\mathrm{P} = \Big\{ \quad S \to a \mid aA \mid B \quad A \to aBB \mid \varepsilon \quad B \to Aa \mid b \quad \Big\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/G54gubr9i

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen. —

$$\mathrm{P} = \Big\{ \quad S \to a \mid aA \mid B \quad A \to aBB \quad B \to Aa \mid b \mid a \quad \Big\}$$

Das leere Wort ist nicht in der Sprache ($\varepsilon \notin L(G)$). In der Sprache sind immer Wörter mit mindestens einem Buchstaben. In der ersten Produktionsregel wird aus $aA \to a\varepsilon$ nur das a. Das ist aber bereits in der ersten Regel enthalten. In der zweiten Regel wird das leere Wort weg gelassen. In der dritten Regel wird noch ein a hinzugefügt, das aus $Aa \to \varepsilon a \to a$ entstanden ist.

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

$$\mathrm{P} = \Big\{$$

$$
\begin{aligned}
S &\to a \mid aA \mid Aa \mid b \\
A &\to aBB \\
B &\to Aa \mid b \mid a
\end{aligned}
\Bigg\}
$$

Wir schreiben die Regel, die keine einzelnes Nonterminal auf der rechten Seite enthalten, ab. In der ersten Regel wird B mit $Aa|b|a$ ersetzte, wobei das letzte a, dann weggelassen werden kann, da es bereits am Anfang der rechten Seite vorkommt. Die B-Regel kann nicht weggelassen werden, weil sie in der A-Regel vorkommt.

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* ———————————————

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to a \mid VA \mid AV \mid b \\
A &\to VBB \\
B &\to AV \mid b \mid a \\
V &\to a
\end{aligned}
\Bigg\}
$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1}B_n, A_{n-1} \to A_{n-2}B_{n-1}, \dots, A_2 \to B_1B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* ———————————————

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to a \mid VA \mid AV \mid b \\
A &\to VC \\
B &\to AV \mid b \mid a \\
V &\to a \\
C &\to BB
\end{aligned}
\Bigg\}
$$

(c) $\mathrm{P} = \Big\{$

$$
\begin{aligned}
S &\to ABC \\
A &\to aCD \\
B &\to bCD \\
C &\to D \mid \varepsilon \\
D &\to C
\end{aligned}
\Bigg\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxwcync2

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.* —

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to ABC \mid AB \\
A &\to aCD \mid aD \\
B &\to bCD \mid bD \\
C &\to D \\
D &\to C \mid \varepsilon
\end{aligned}
\Bigg\}
$$

In der letzten Regel entsteht ein neues ε. Es muss in der nächsten Iteration entfernt werden.

$$
\mathrm{P} = \Bigg\{
\begin{aligned}
S &\to ABC \mid AB \\
A &\to aCD \mid aD \mid aC \mid a \\
B &\to bCD \mid bD \mid bC \mid b \\
C &\to D \\
D &\to C
\end{aligned}
\Bigg\}
$$

(ii) **Elimination von Kettenregeln**

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.* —

$$
\mathrm{P} = \Bigg\{
$$

$$\left.\begin{aligned} S &\to AB \\ A &\to a \\ B &\to b \end{aligned}\right\}$$

C und D sind nicht produktiv. $C \to D$ und $D \to C$ können gestrichen werden.

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* ———————————————

∅ Nichts zu tun

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* ———————————

∅ Nichts zu tun

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Chomsky-Normalform/Aufgabe_Vorlesungsaufgabe.tex`

Übungsaufgabe „0-1“ (Kontextfreie Grammatik)

Geben Sie für die folgenden Sprachen eine kontextfreie Grammatik an:

Die Wörter bestehen aus beliebig vielen Nullen, einer 1 sowie weiteren beliebig vielen Nullen oder Einsen.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \to A1A \\ A \to 0A \mid 1A \mid \varepsilon \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/G539t1rgc

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Grammatik/Aufgabe_0-1.tex`

Übungsaufgabe „Vorlesungsaufgabe“ (Kontextfreie Sprache, Ableitung (Kontextfreie Sprache), Kontextfreie Grammatik)

Kontextfreie Sprache
Ableitung (Kontextfreie Sprache)
Kontextfreie Grammatik

(a) Erstellen Sie eine Ableitung für die Wörter der Sprache zur vorgegeben Grammatik

$G = (\{S, A, B\}, \{0, 1\}, P, S)$

$$\mathrm{P} = \left\{ \begin{array}{l} S \to A1B \\ A \to 0A \mid \varepsilon \\ B \to 0B \mid 1B \mid \varepsilon \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gi1rgpemg

- 00101

Lösungsvorschlag

$S \vdash A1B \vdash 0A1B \vdash 00A1B \vdash 001B \vdash 0010B \vdash 00101B \vdash 00101$

- 1001

Lösungsvorschlag

$S \vdash A1B \vdash 1B \vdash 10B \vdash 100B \vdash 1001B \vdash 1001$

(b) Erstellen Sie eine kontextfreie Grammatik, die alle Wörter mit gleich vielen 1‘s, gefolgt von gleich vielen 0‘s enthält.

Lösungsvorschlag

$$\mathrm{P} = \left\{ \begin{array}{l} S \to 1S0 \mid \varepsilon \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxmyw2ia

(c) Erstellen Sie eine kontextfreie Grammatik, die alle regulären Ausdrücke über den Zeichen $0, 1$ darstellt. (Beispiel: `01*(1+0)0` für einen möglichen regulären Ausdruck (Das +-Zeichen ist hier anstelle des Oder-Zeichens (|)))

Lösungsvorschlag

$$G = (\{S\}, \{1; 0; (;); +; *\}, P, S)$$

$$\mathrm{P} = \Big\{$$

$$S \to \varepsilon \mid 0 \mid 1 \mid S* \mid (S) \mid SS \mid S + S$$

$$\Big\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghfgrv027

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Grammatik/Aufgabe_Vorlesungsaufgabe.tex

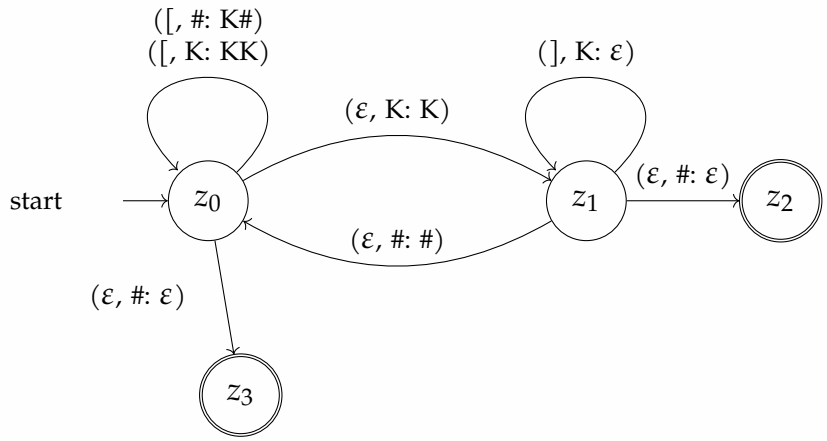
Übungsaufgabe „Balancierte Klammern“ (Kellerautomat)

Kellerautomat

Erstellen Sie einen Kellerautomaten der nur balancierte Klammerausdrücke mit eckigen Klammern akzeptiert. [3]

Lösungsvorschlag

$$K = (\{z_0, z_1, z_2, z_3\}, \{[,]\}, \{\#, K\}, \delta, z_0, \#, \{z_2, z_3\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwobf482

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Balancierte-Klammern.tex`

[3] `https://eecs.wsu.edu/~ananth/CptS317/Lectures/PDA.pdf` (Seite 9)

Übungsaufgabe „an bn“ (Kellerautomat)

Kellerautomat

Erstellen Sie einen Kellerautomaten, der folgende Sprache

$$L = \{ a^n b^n \mid n \in \mathbb{N} \}$$

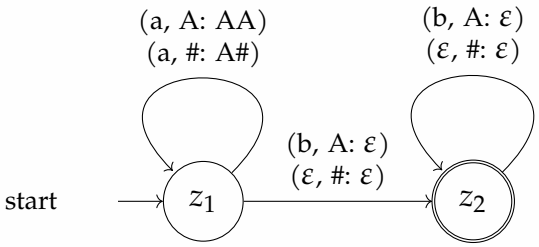
mit folgender Grammatik

$$G = (\{S\}, \{a, b\}, \{S \to aSb \mid ab\}, S)$$

erkennt.

Lösungsvorschlag

$$K = (\{z_1, z_2\}, \{a, b\}, \{\#, A\}, \delta, z_1, \#, \{z_2\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5v17t52

Aktueller Zustand	Eingabe	Keller	Folgezustand	Keller
z_1	a	#	z_1	A#
z_1	a	A	z_1	AA
z_1	b	A	z_2	ε
z_2	b	A	z_2	ε
z_2	ε	#	z_2	#

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Foliensatz.tex`

Übungsaufgabe „zu drei Grammatiken“ (Kellerautomat)

Kellerautomat

Geben Sie für die folgenden Grammatiken G_i jeweils einen Kellerautomaten P_i an, der dieselbe Sprache besitzt wie die Grammatik: $L(G_i) = L(P_i)$

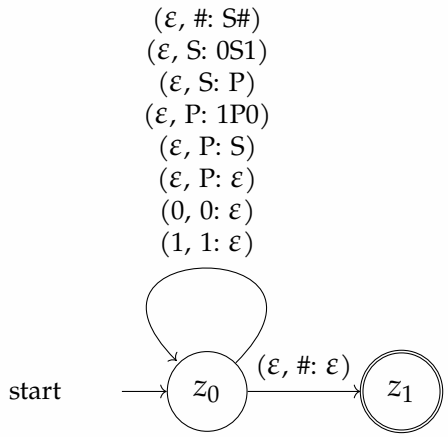
(a) $P_1 = \Big\{$

$$S \to 0S1 \mid P$$
$$P \to 1P0 \mid S \mid \varepsilon$$

$\Big\}$

Lösungsvorschlag

$K = (\{z_0, z_1\}, \{0,1\}, \{\#, S, P, 0, 1\}, \delta, z_0, \#, \{z_1\})$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5ceyrrz

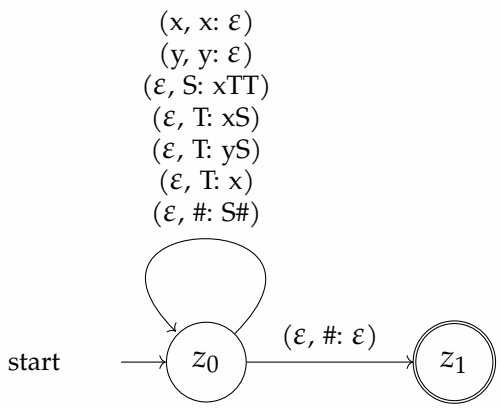
(b) $P_2 = \Big\{$

$$S \to xTT$$
$$T \to xS \mid yS \mid x$$

$\Big\}$

Lösungsvorschlag

$K = (\{z_0, z_1\}, \{x, y\}, \{\#, T, S, x, y\}, \delta, z_0, \#, \{z_1\})$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiq4r0162

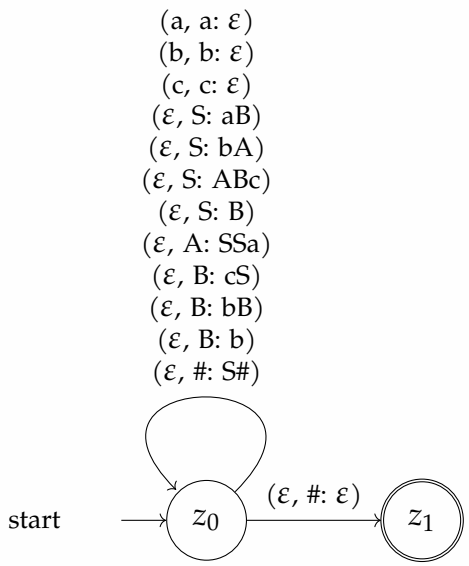
(c) $P_3 = \Big\{$

$$\begin{aligned} S &\to aB \mid bA \mid ABc \mid B \\ A &\to SSa \\ B &\to cS \mid bB \mid b \end{aligned}$$

$\Big\}$

Lösungsvorschlag

$K = (\{z_0, z_1\}, \{a, b\}, \{\#, A, B, a, b\}, \delta, z_0, \#, \{z_1\})$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt

der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajh5y0s5r

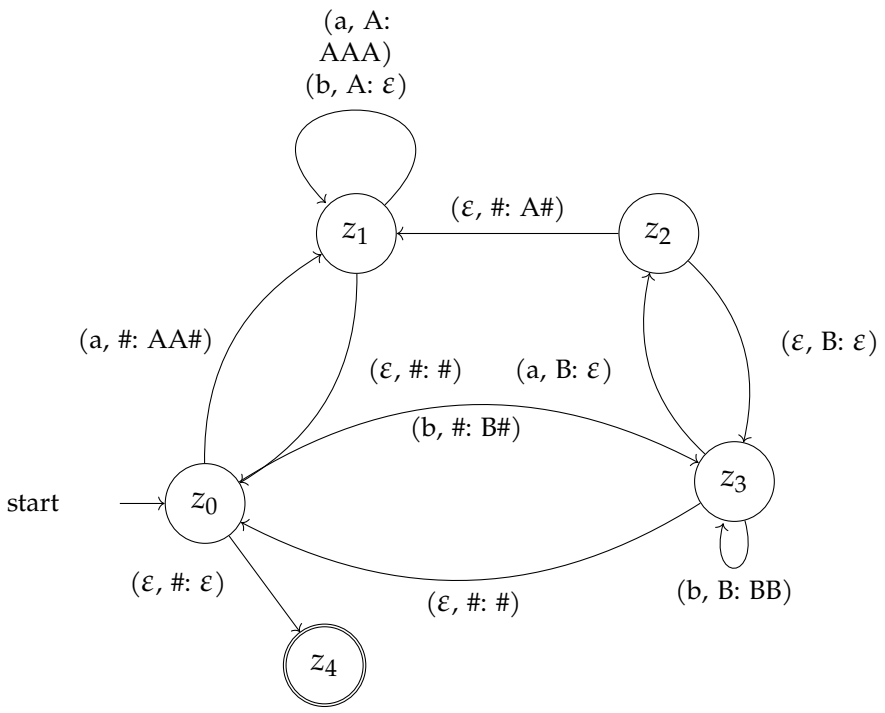
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Kellerautomat.tex

Übungsaufgabe „Konfigurationsfolge doppelt so viele b's wie a's“ (Kellerautomat)

Kellerautomat

Gegeben ist der folgende nichtdeterministische Kellerautomat mit

$$K = (\{z_0, z_1, z_2, z_3, z_4\}, \{a, b\}, \{\#, A, B\}, \delta, z_0, \#, \{z_4\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apk0ic3s9

(a) Geben Sie für die folgenden Wörter, die in der Sprache enthalten sind, eine Berechnung (Folge von Konfigurationen) des Kellerautomaten an:

(i) $w_1 = \texttt{bab}$

Lösungsvorschlag

$(z_0, bab, \#) \vdash (z_3, ab, B\#) \vdash (z_2, b, \#) \vdash (z_1, b, A\#) \vdash (z_1, \varepsilon, \#) \vdash (z_0, \varepsilon, \#) \vdash (z_4, \varepsilon, \varepsilon)$

(ii) $w_2 = \texttt{abb}$

Lösungsvorschlag

$(z_0, abb, \#) \vdash (z_1, bb, AA\#) \vdash (z_1, b, A\#) \vdash (z_1, \varepsilon, \#) \vdash (z_0, \varepsilon, \#) \vdash (z_4, \varepsilon, \varepsilon)$

(iii) $w_3 = \texttt{abababbbb}$

Lösungsvorschlag

$(z_0, ababababbb, \#) \vdash (z_1, bababbbb, AA\#) \vdash (z_1, abababbb, A\#) \vdash (z_1, babbbb, AAA\#) \vdash$
$(z_1, abbbb, AA\#) \vdash (z_1, bbbb, AAAA\#) \vdash (z_1, bbb, AAA\#) \vdash (z_1, bb, AA\#) \vdash$
$(z_1, b, A\#) \vdash (z_1, \varepsilon, \#) \vdash (z_0, \varepsilon, \#) \vdash (z_4, \varepsilon, \varepsilon)$

(b) Charakterisiere die Wörter der Sprache in eigenen Worten.

Lösungsvorschlag

$L = \{ w \mid w \text{ enthält genau doppelt so viele } b\text{'s wie } a\text{'s} \}$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Konfigurationsfolge-Kellerautomat.tex

Übungsaufgabe „0n 1n, gleich viele ab, kein Präfix mehr Einsen“ (Kellerautomat)

Kellerautomat

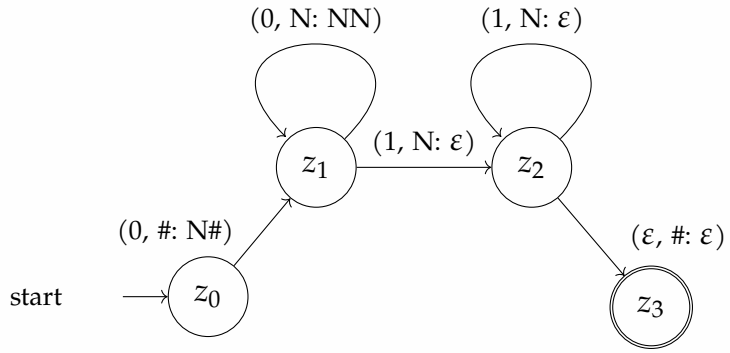
Erstellen Sie jeweils einen PDA, der die angegebenen Sprachen akzeptiert.

(a) $L = \{ 0^n 1^n \mid n \in \mathbb{N} \}$

Lösungsvorschlag

$K = (\{z_0, z_1, z_2, z_3\}, \{0, 1\}, \{\#, N\}, \delta, z_0, \#, \{z_3\})$

N = Null

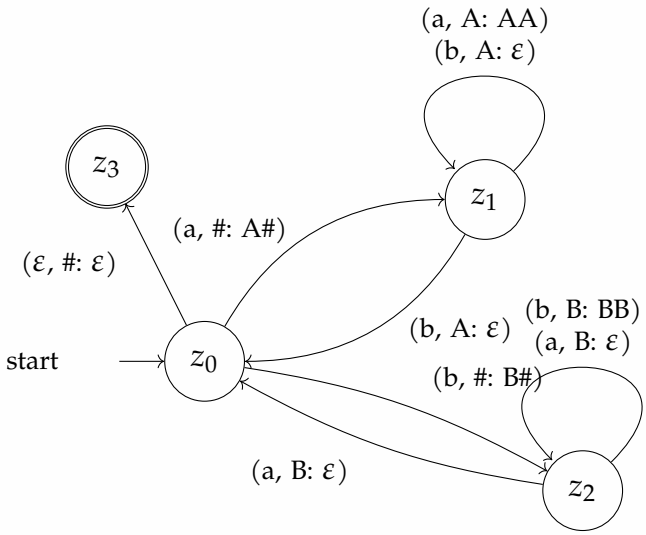


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aji1r8mf7

(b) Alle Wörter, die gleich viele a wie b enthalten.

Lösungsvorschlag

$K = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \{\#, A, B\}, \delta, z_0, \#, \{z_3\})$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt

der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arar6fos7

(c) Alle Wörter, bei denen kein Präfix mehr Einsen wie Nullen hat.

Lösungsvorschlag

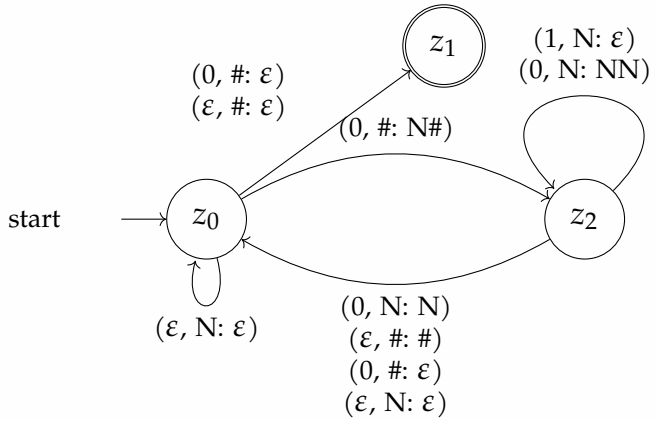
$K = (\{z_0, z_1, z_2\}, \{0, 1\}, \{\#, N\}, \delta, z_0, \#, \{z_1\})$

akzeptiert:

- 000
- 00101
- 01

nicht akzeptiert:

- 011
- 111
- 1



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af7rfyqqg

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_PDA.tex

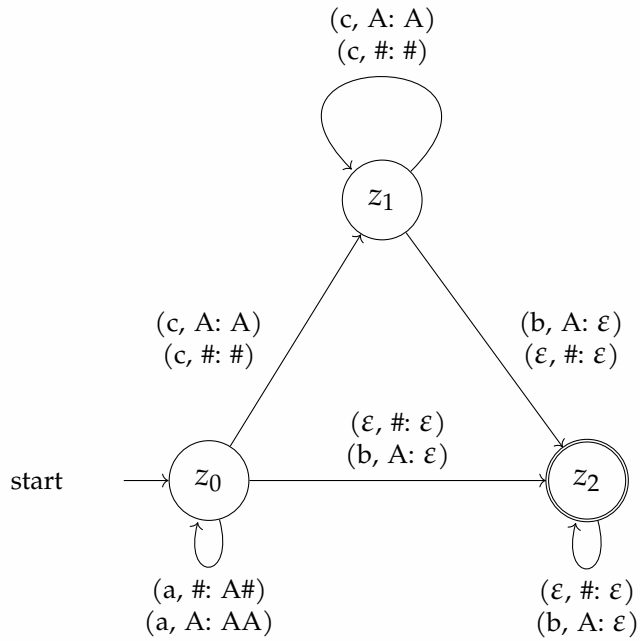
Übungsaufgabe „a hoch n c hoch i b hoch n“ (Kontextfreie Sprache, Kellerautomat, Kontextfreie Grammatik, Konfigurationsfolge)

Kontextfreie Sprache
Kellerautomat

(a) Geben Sie einen Kellerautomaten an, der die folgende Sprache erkennt:
$L = \{ a^n c^i b^n \mid n, i \in \mathbb{N}_0 \}$

Lösungsvorschlag

$K = (\{z_0, z_1, z_2\}, \{a, b, c\}, \{\#, A\}, \delta, z_0, \#, \{z_2\})$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apky9znog

Tabellenform:

Kontextfreie Grammatik
Konfigurationsfolge

Aktueller Zustand	Eingabe	Keller	Folgezustand	Keller
z_0	a	#	z_0	A#
z_0	a	A	z_0	AA
z_0	c	#	z_1	#
z_0	c	A	z_1	A
z_0	ε	#	z_2	ε
z_0	b	A	z_2	ε
z_1	c	#	z_1	#
z_1	c	A	z_1	A
z_1	ε	#	z_2	ε
z_1	b	A	z_2	ε
z_2	ε	#	z_2	ε
z_2	b	A	z_2	ε

(b) Geben Sie eine Grammatik für diese Sprache an.

Lösungsvorschlag

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aSb \mid \varepsilon \mid c \mid cC \\ C \to cC \mid \varepsilon \end{array} \right\}$$

alternativ:

$$\mathrm{P} = \left\{ \begin{array}{l} S \to aSb \mid \varepsilon \mid C \\ C \to cC \mid \varepsilon \end{array} \right\}$$

(c) Geben Sie Konfigurationsfolgen für die Erzeugung des Wortes an

- aacbb

Lösungsvorschlag

$(z_0, \text{aacbb}, \#) \vdash (z_0, \text{acbb}, \text{A}\#) \vdash (z_0, \text{cbb}, \text{AA}\#) \vdash (z_1, \text{bb}, \text{AA}\#) \vdash (z_2, \text{b}, \text{A}\#) \vdash (z_2, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$

- accb

Lösungsvorschlag

$(z_0, \text{accb}, \#) \vdash (z_0, \text{ccb}, \text{A}\#) \vdash (z_1, \text{cb}, \text{A}\#) \vdash (z_2, \text{b}, \text{A}\#) \vdash (z_2, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Vorlesungsaufgabe-1.tex

Übungsaufgabe „Nonterminal: P, Terminale: 01“ (Kellerautomat)

Kellerautomat

Erstellen Sie einen Kellerautomaten zu der Grammatik $G = (\{S\}, \{0,1\}, P, S)$ mit den folgenden Produktionsregeln

(a) $\mathrm{P} = \Big\{$

$$S \to \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$$

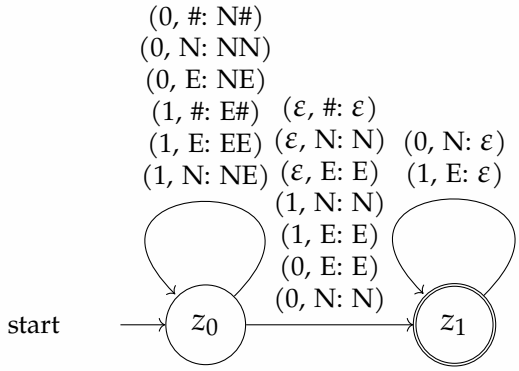
$\Big\}$

Lösungsvorschlag

$K = (\{z_0, z_1\}, \{0,1\}, \{\#, N, E\}, \delta, z_0, \#, \{z_1\})$

N = Null

E = Eins



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahij8jnn7

(b) $\mathrm{P} = \Big\{$

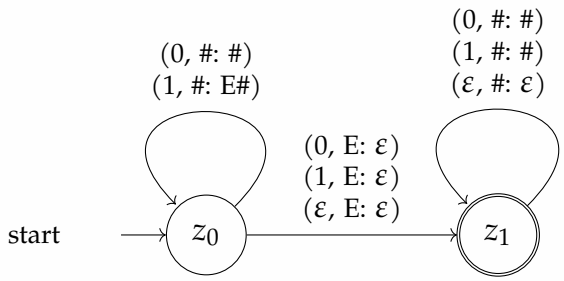
$$S \to A1B$$
$$A \to 0A \mid \varepsilon$$
$$B \to 0B \mid 1B \mid \varepsilon$$

$\Big\}$

Lösungsvorschlag

$K = (\{z_0, z_1\}, \{0,1\}, \{\#, E\}, \delta, z_0, \#, \{z_1\})$

E = Eins ist gesetzt



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3imp8a7

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Kellerautomat/Aufgabe_Vorlesungsaufgabe-2.tex`

Übungsaufgabe „an bn cn“ (Pumping-Lemma (Kontextfreie Sprache))

Pumping-Lemma (Kontextfreie Sprache)

Gegeben sei die Sprachen

$$L = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$$

Weisen Sie nach, dass L nicht kontextfrei ist.

Exkurs: Pumping-Lemma für Kontextfreie Sprachen
Es sei L eine kontextfreie Sprache. Dann gibt es eine Zahl j, sodass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ zerlegen lassen in $\omega = uvwxy$, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|vx| \geq 1$ (Die Wörter v und x sind nicht leer.)

(b) $|vwx| \leq j$ (Die Wörter v, w und x haben zusammen höchstens die Länge j.)

(c) Für alle $i \in \mathbb{N}_0$ gilt $uv^i wx^i y \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort $uv^i wx^i y$ in der Sprache L)

Lösungsvorschlag

Also gibt es eine Pumpzahl. Sie sei j. (Wähle geschickt ein „langes“ Wort...) $a^j b^j c^j$ ist ein Wort aus L, das sicher länger als j ist.

Da L kontextfrei ist, muss es nach dem Pumping-Lemma auch für dieses Wort eine beliebige Zerlegung geben:

$a^j b^j c^j = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq j$

Weil vwx höchstens j lang ist, kann es nie a‘s und c‘s zugleich enthalten (es stehen j b‘s dazwischen!).

Andererseits enthält vx mindestens ein Zeichen. Das Wort $\omega = uv^0 wx^0 y = uwy$ enthält dann nicht mehr gleich viele a‘s, b‘s und c‘s. (Widerspruch)!

Die Behauptung ist falsch.

$\Rightarrow L$ ist nicht kontextfrei!

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Pumpling-Lemma/Aufgabe_Foliensatz.tex`

Übungsaufgabe „„w w“ und „ak bl cm““ (Pumping-Lemma (Kontextfreie Sprache))

Pumping-Lemma (Kontextfreie Sprache)

Zeigen Sie jeweils, dass die angegebene Sprache nicht kontextfrei ist:

Exkurs: Pumping-Lemma für Kontextfreie Sprachen
Es sei L eine kontextfreie Sprache. Dann gibt es eine Zahl j, sodass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ zerlegen lassen in $\omega = uvwxy$, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|vx| \geq 1$ (Die Wörter v und x sind nicht leer.)

(b) $|vwx| \leq j$ (Die Wörter v, w und x haben zusammen höchstens die Länge j.)

(c) Für alle $i \in \mathbb{N}_0$ gilt $uv^iwx^iy \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iwx^iy in der Sprache L)

(a) $L_1 = \{ ww \mid w \in \{a,b\}^* \}$

Lösungsvorschlag

Sei L_1 kontextfrei. Dann existiert nach dem Pumping-Lemma eine Zahl j, so dass für jedes Wort $\omega \in L_1$ mit $|\omega| \geq j$ eine Zerlegung $\omega = uvwxy$ existiert, für die gilt: $|vx| > 0$, $|vwx| \leq n$ und für jedes $i \in \mathbb{N}$ ist $uv^ixy^iz \in L_1$.

Wähle $\omega = a^nb^na^nb^n$. Dann gibt es für jede Zerlegung $\omega = uvxyz$ mit den obigen Bedingungen zwei Möglichkeiten:

- vwx besteht aus a^jb^k mit $j + k > 0$.
- vwx besteht aus b^ja^k mit $j + k > 0$.

Dann ist in beiden Fällen $uv^0xy^0z \notin L_1$.

(b) $L_2 = \{ a^kb^lc^m \mid k > l > m; k,l,m \in N \}$

Lösungsvorschlag

Sei L_2 kontextfrei. Dann existiert nach dem Pumping-Lemma eine Zahl j, so dass für jedes Wort $\omega \in L_2$ mit $|\omega| \geq j$ eine Zerlegung $\omega = uvwxy$ existiert, für die gilt: $|vx| > 0$, $|vwx| \leq j$ und für jedes $i \in \mathbb{N}$ ist $uv^iwx^iy \in L_2$.

Wähle $\omega = a^nb^{n-1}c^{n-2}$. Dann gibt es für jede Zerlegung $\omega = uvwxy$ mit den obigen Bedingungen zwei Möglichkeiten:

- vwx enthält kein a. Dann ist $uv^2wx^2y \notin L_2$.
- vwx enthält mindestens ein a. Dann ist $uv^0wx^0y \notin L_2$.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Pumpling-Lemma/Aufgabe_Pumping-Lemma.tex`

Übungsaufgabe „„an bn“ „c2n“ und „an bn2““ (Pumping-Lemma (Kontextfreie Sprache))

Pumping-Lemma (Kontextfreie Sprache)

Zeigen Sie, dass die folgenden Sprachen nicht kontextfrei sind:

Exkurs: Pumping-Lemma für Kontextfreie Sprachen
Es sei L eine kontextfreie Sprache. Dann gibt es eine Zahl j, sodass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ zerlegen lassen in $\omega = uvwxy$, sodass die folgenden Eigenschaften erfüllt sind:

(a) $|vx| \geq 1$ (Die Wörter v und x sind nicht leer.)

(b) $|vwx| \leq j$ (Die Wörter v, w und x haben zusammen höchstens die Länge j.)

(c) Für alle $i \in \mathbb{N}_0$ gilt $uv^iwx^iy \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iwx^iy in der Sprache L)

- $L = \{ a^n b^n c^{2n} \mid n \in \mathbb{N} \}$

Lösungsvorschlag

Annahme: L ist kontextfrei.

$\forall \omega \in L$: $\omega = uvwxy$

$j \in \mathbb{N}$: $|\omega| \geq j$

$\omega = a^j b^j c^{2j}$: $|\omega| = 4j > j$

Damit gilt: $|vwx| \leq j$, $|vx| \geq 1$

Zu zeigen: Keine Möglichkeit der Zerlegung, damit $\omega' \in L$

1. Fall vwv enthält nur a's

o. E. d. A. (ohne Einschränkung der Allgemeinheit) stecken alle a's in der Zerlegung vwx, ðu ist leer

$u : \varepsilon$

$v : a^l$

$w : a^{j-(l+m)}$

$x : a^m$

$y : b \ldots bc \ldots c$

v^2wx^2y

$a^{2l}a^{j-(l+m)}a^{2m}b^jc^{2j} =$

Nebenrechnung: $2l + j - (l + m) + 2m = j + l + m > j$, da $|vx| \geq 1 \rightarrow l + m \geq 1$

$\Rightarrow \omega' = uv^2wx^2y \notin L$

2. Fall vwv enthalten a's und b's

o. E. d. A. $|v|_a = |x|_b$

u: a^p v: a^l w: $a^{j-(p+l)}b^{j-(l+r)}$ x: b^l y: b^rc^{2j}

$\Rightarrow uv^0wx^0v$

Nebenrechnung:
a's: $p + j - (l + p) = j - l$
b's: $j - (l + r) = j - l$
ist falsch, da $j - l$ echt kleiner ist, da $|vx| \geq 1 \rightarrow l \geq 1$
$\Rightarrow \omega' \notin L$

3. Fall vwx enthält nur b's
analog zu Fall 1

4. Fall vwx enthält nur b's und c's
analog zu Fall 2

5. Fall vwx enthält nur c's
analog zu Fall 1

$\Rightarrow$ Es gibt keine Zerlegung, sodass $\forall i \in \mathbb{N}_0$
$\Rightarrow$ Annahme ist falsch
$\Rightarrow L$ ist nicht kontextfrei

- $L = \{ a^n b^{n^2} \mid n \in \mathbb{N} \}$

Lösungsvorschlag

Annahme: L kontextfrei

$\Rightarrow$ Pumping-Lemma: $j \in \mathbb{N}$: $|w| \geq j$

$\omega = a^j b^{j^2}$

$j + j^2 > j$

1. Fall vwx enthält nur a's

$\Rightarrow$ ungleich viele a's wie b's als Quadrat

$\Rightarrow \omega' \notin E$

2. Fall vwx enthält nur b's

$\Rightarrow$ analog zu Fall 1

$\Rightarrow \omega' \notin E$

3. Fall vwx enthält a's und b's

o. E. d. A. v nur a's ; x nur b's

u**:** $a^{j-(l+m)}$

v**:** a^l

w**:** $a^m b^n$

x**:** b^{l^2}

y**:** $b^{j^2-(n+l^2)}$

$\Rightarrow uv^0wx^0y = \omega'$

a: $j - (l+m) + 0 \cdot l + m = j - l$

b: $n - 0 \cdot l^2 + j^2 - (n + l^2) = j^2 - l^2 = (j - l)(j + l) \neq (j - l)(j - l)$
$\Rightarrow \in L$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/20_Typ-2_Kontextfrei/Pumpling-Lemma/Aufgabe_Vorlesungsaufgaben.tex

Kontextsensitive Sprache

Übungsaufgabe „Kontextsensitive Grammatik“ (Kontextsensitive Grammatik)

Sei $L = \{ a^n b^m c^n \mid n, m \geq 0, m \leq n \}$.

(a) Geben Sie eine kontextsensitive Grammatik für L an.

Lösungsvorschlag

$G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$ mit Produktionen P wie folgt:

$$
\mathrm{P} = \left\{ \begin{aligned}
S &\to \varepsilon \mid aSc \mid aSBC \\
CB &\to BC \\
bC &\to bc \\
aB &\to ab \\
bB &\to bb \\
cC &\to cc
\end{aligned} \right\}
$$

(b) Geben Sie eine Ableitung des Wortes $a^3b^2c^3$ mittels der in Teilaufgabe a) erstellten Grammatik an.

Lösungsvorschlag

$S \vdash aSc \vdash aaSBCc \vdash aaaSBCBCc \vdash aaaBCBCc \vdash aaaBBCCc \vdash aaabBCCc \vdash aaabbCCc \vdash aaabbcCc \vdash aaabbccc$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Kontextsensitive-Grammatik/Aufgabe_Kontextsensitive-Grammatik.tex

Übungsaufgabe „Vorlesungsaufgaben kontextsensitive Grammatiken“ (Kontextsensitive Grammatik)

Kontextsensitive Grammatik

Gegen sei folgende Grammatik: [Seite 8]theo:fs:3
$G = (V, \Sigma, P, S)$ mit $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$, $S = S$ und

$$
\mathrm{P} = \left\{
\begin{aligned}
S &\to aSBC \mid aBC \\
CB &\to BC \\
aB &\to ab \\
bB &\to bb \\
bC &\to bc \\
cC &\to cc
\end{aligned}
\right\}
$$

(a) Geben Sie die Sprache an, die die folgende Grammatik erzeugt:

Lösungsvorschlag

$L = \{ a^n b^b c^n \mid n \in \mathbb{N} \}$

(b) Gib Sie eine Ableitung mit der folgenden Grammatik für das Wort aaabbbccc an.

Lösungsvorschlag

$aSBC \vdash aaSBCBC \vdash aaaBCBCBC \vdash aaaBBCCBC \vdash aaaBBCBCC \vdash aaaBBBCCC \vdash aaabBBCCC \vdash aaabbBCCC \vdash aaabbbCCC \vdash aaabbbcCC \vdash aaabbbccC \vdash aaabbbccc$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Kontextsensitive-Grammatik/Aufgabe_Vorlesungsaufgaben.tex`

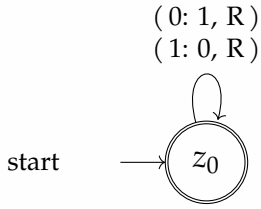
Übungsaufgabe „Vorlesungsaufgaben Komplement der Binärzahl“ (Kontextsensitive Sprache)

Kontextsensitive Sprache

Gegeben ist eine Binärzahl auf dem Band einer Turingmaschine.

(a) Definieren Sie vollständig eine TM, die das Komplement der Binärzahl (0110 $\rightarrow$ 1001) berechnet. Die Überführungsfunktion kann als Tabelle oder als Graph angegeben werden.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ap9qtjgg7

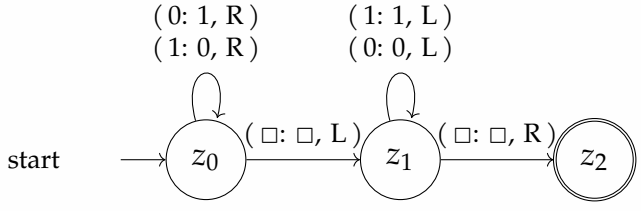
```
name: Komplement Binärzahl
init: z0
accept: z0

z0,0
z0,1,>

z0,1
z0,0,>
```

a

[a] `http://turingmachinesimulator.com/shared/lqpsawxdeh`

(b) Erweiteren Sie Ihre Maschine aus Aufgabe a) so, dass der Schreib-/Lesekopf auf dem ersten Zeichen der Eingabe terminiert.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5o7tug5r

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Vorlesungsaufgaben-Komplement-Binaerzahl.tex`

Unbeschränkte Sprache

Examensaufgabe „Turingmaschinen“ (46115-2013-F.T1-A4)

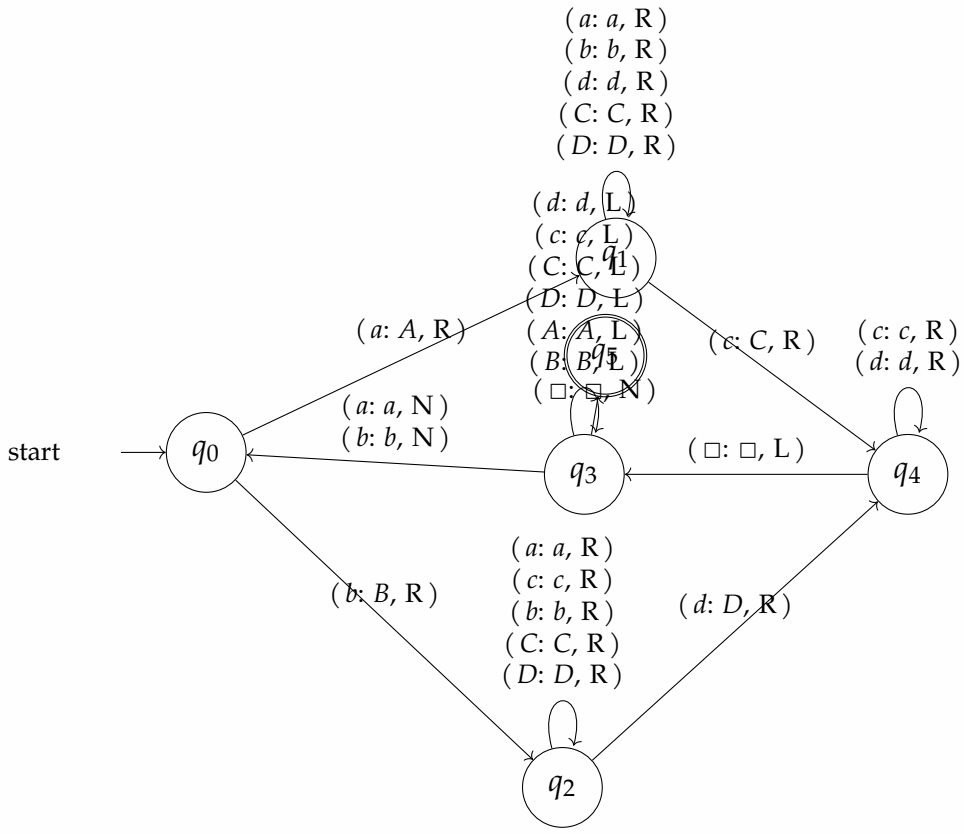
Aufgabe 4

Sei $L = \{\, uv \mid u \in \{a,b\}^*, v \in \{c,d\}^*, \#_a(u) = \#_c(v) \text{ und } \#_b(u) = \#_d(v)) \,\}$ wobei $\#_a(u)$ die Anzahl der in u vorkommenden a's ist.

(a) Geben Sie eine Turingmaschine M an, die L erkennt. Beschreiben Sie in Worten, wie Ihre Turingmaschine arbeitet.

Lösungsvorschlag

noch nicht fertig ... akzeptiert auch abcd!d



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajhi5w0ha

(b) Welche Laufzeit (Zeitkomplexität) hat Ihre Turingmaschine (in O-Notation). Begründen Sie Ihre Angabe auf der Grundlage der Beschreibung.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2013/03/Thema-1/Aufgabe-4.tex

Examensaufgabe „Berechen- und Entscheidbarkeit“ (66115-2017-F.T2-A3)

Berechenbarkeit
Turing-Maschine

(a) Primitiv rekursive Funktionen

(i) Zeigen Sie, dass die folgendermaßen definierte Funktion if: $\mathbb{N} \times \mathbb{N} \times \mathbb{N}\mathbb{N}$ primitiv rekursiv ist.
sonst

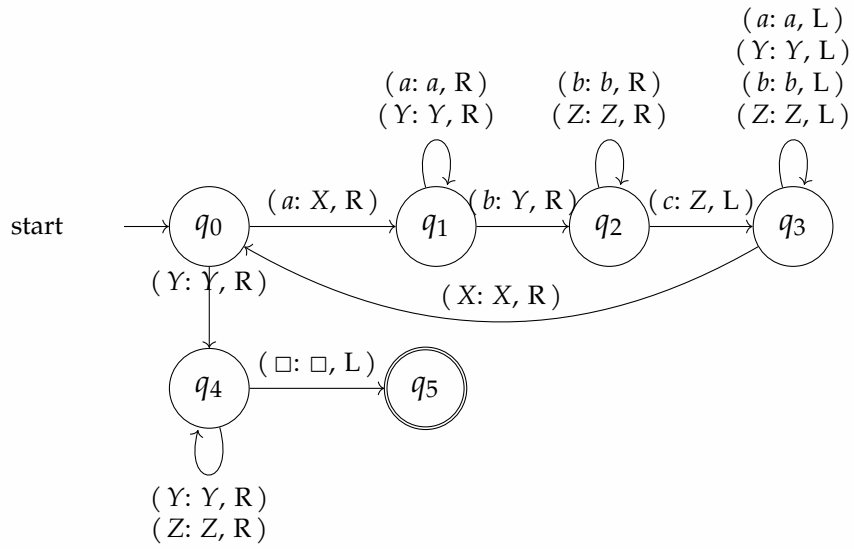
(ii) Wir nehmen eine primitiv rekursive Funktionp: NN an und definieren g(n) als die Funktion, welche die größte Zahl i < n zurückliefert, für die p(/) = 0 gilt. Falls kein solches i existiert, soll g(n) = 0 gelten:
a(n) = max (i <n |p) = 0 U 0)
if (b, x, y) = (falls b=0

Zeigen Sie, dass g: N > N primitiv rekursiv ist. (Sie dürfen obige Funktion if als primitiv rekursiv voraussetzen.)

(b) Sei $\Sigma = \{a, b, c\}$ und $L \subseteq \Sigma^*$ mit $L = \{ a^i b^i c^i \mid i \in N \}$.

(i) Beschreiben Sie eine Turingmaschine, welche die Sprache Z entscheidet. Eine textuelle Beschreibung der Konstruktionsidee ist ausreichend.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apew1n7g9 [a]

[a] https://scanftree.com/automata/turing-machine-for-a-to-power-n-b-to-power-n-c-to-power

(ii) Geben Sie Zeit- und Speicherkomplexität (abhängig von der Länge der Eingabe) Ihrer Turingmaschine an.

Lösungsvorschlag

Speicherkomplexität n (Das Eingabewort wird einmal überschrieben)

Zeitkomplexität the turing machine time complexity is the number of transition execution will executed is call time complexity of the turing machine. first we start we main loop execution is (n/3)-1. transition(a,x,R) from state 1 to 2= 1. transition (a,a,R) and (y,y,R) on state 2 is = (n/3)-1. transition (b,y,R) from state 2 to 3=1. on state 3 (b,b,R) and (z,z,R)=(n/3)-1. transition (c,z,L) from state 3 to 4=1. on state 4 (y,y,L),(b,b,L),(z,z,L) and state 5 (a,a,L)= (n/3)-1. transition (a,a,L) form state 4 to 5 =1. transition (x,x,R) from 5 to1 =1 total(n+2) following transition will executed transition(a,x,R) from state 1 to 2= 1. transition (y,y,R) on state 2 is = (n/3)-1. transition (b,y,R) from state 2 to 3=1. transition (z,z,R) on state 3=(n/3)-1 transition (c,z,L) from state 3 to 4=1. on state 4 (y,y,L) ,(z,z,L) and state (n/3)-1. transition (x,x,R) from state 54 to 6 =1 transition on state 6 (y,y,R),(z,z,R)= (n/3) transition (d,d,R) from state 6 to 7 =1 total =(4n/3)+2 over alti time complexity (n+2)(n/3)-1+ (4n/3)+2 [a]

[a] `https://www.youtube.com/watch?v=vwnz9e_Lrfo`

(c) Sei $\Sigma = \{0, 1\}$. Jedes $w \in \Sigma^*$ kodiert eine Turingmaschine M_w. Die von M_w berechnete Funktion bezeichnen wir mit $\varphi_w(x)$.

(i) Warum ist $L = \{ w \in \Sigma^* \mid \exists x \colon \varphi_w(x) = xx \}$ nicht entscheidbar?

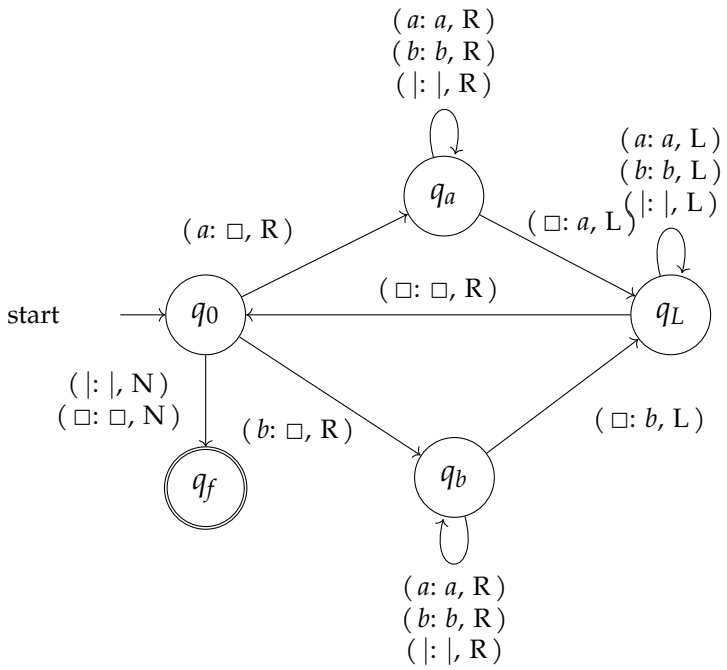
(ii) Warum ist $L = \{ w \in \Sigma^* \mid \exists x \colon w = xx \}$ entscheidbar?

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2017/03/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Turingmaschine Konfigurationsfolge“ (66115-2019-F.T1-A4)

Turing-Maschine

Wir betrachten die Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$. Hierbei ist die Zustandsmenge $Q = \{q_0, q_a, q_b, q_L, q_f\}$ mit Startzustand q_0 und akzeptierenden Zuständen $F = \{q_f\}$. Das Eingabealphabet ist $\Sigma = \{a, b, |\}$[1] das Bandalphabet ist $\Gamma = \Sigma \cup \{\square\}$ mit Blank-Zeichen $\square$ für leeres Feld. Die Übergangsfunktion $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$, wobei der Schreib-Lese-Kopf mit L nach links, mit N nicht und mit R nach rechts bewegt wird, ist durch folgende Tabelle gegeben (bspw. ist $\delta(q_0, a) = (q_a, \square, R)$):



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj54q4rd9

(a) Die Notation (v, q, aw) beschreibt eine Konfiguration der Turingmaschine: der interne Zustand ist q, der Schreib-Lesekopf steht auf einem Feld mit $a \in \Gamma$, rechts vom Schreib-Lesekopf steht $w \in \Gamma^*$, links vom Schreib-Lesekopf steht $v \in \Gamma^*$.

Vervollständigen Sie die Folge von Konfigurationen, die die Turingmaschine bei Eingabe $ab|$ bis zum Erreichen des Zustands q_f durchläuft. Sie können auch Ihre eigene Notation zur Darstellung von Konfigurationen verwenden.

$$
\begin{aligned}
&(\square, q_0, ab|) \vdash \\
&(\square, q_a, \square b|) \vdash \\
&(\square, q_a, b|) \vdash \\
&\dots
\end{aligned}
$$

[1] In der Angabe ist das Trennzeichen ein „$“. Wir verwenden stattdessen ein „|“, denn „$“ ist eine Tex-Sonderzeichen und müsste deshalb ständig besonders behandelt werden.

Lösungsvorschlag

$(\square, q_0, ab|) \vdash$
$(\square, q_a, \square b|) \vdash$
$(\square, q_a, b|) \vdash$
$(b, q_a, |) \vdash$
$(b|, q_L, a) \vdash$
$(b, q_L, |a) \vdash$
$(\square, q_L, \square b|a) \vdash$
$(\square, q_b, \square b|a) \vdash$
$(\square, q_b, \square|a) \vdash$
$(\square, q_b, |a) \vdash$
$(|, q_b, a) \vdash$
$(|a, q_L, b) \vdash$
$(|, q_L, ab) \vdash$
$(\square, q_L, |ab) \vdash$
$(\square, q_0, \square|ab) \vdash$
$(\square, q_f, |ab)$

(b) Sei $w \in \{a, b\}^*$ beliebig. Mit welchem Bandinhalt terminiert die Turingmaschine bei Eingabe von $w|$? Geben Sie auch eine kurze Begründung an.

Lösungsvorschlag

Die Turingmaschine terminiert bei alle möglichen Wörtern $w \in \{a, b\}^*$, auch bei dem leeren Wort vor $|$. Die Turing-Maschine verschiebt alle a's und b's vor dem Trennzeichen $|$ nach rechts. Ist das Trennzeichen $|$ schließlich das erste Zeichen von links gesehen, dann terminiert die Maschine.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/03/Thema-1/Aufgabe-4.tex`

Examensaufgabe „Multiplikation mit 3“ (66115-2019-H.T2-A1)

Turing-Maschine

Gesucht ist eine Turing-Maschine mit genau einem beidseitig unendlichen Band, die die Funktion $f : \mathbb{N} \to \mathbb{N}$ mit $f(x) = 3x$ berechnet. Zu Beginn der Berechnungsteht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

(a) Beschreiben Sie zunächst in Worten die Arbeitsweise Ihrer Maschine.

Lösungsvorschlag

$13 \cdot 3 = 0b1101 \cdot 0b11 = 39 = 0b100111$:

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ & & & 1 & 1 & 0 & 1 \\ \hline {\scriptstyle 1} & {\scriptstyle 1} & & & & \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \end{array}$$

Die entworfene Turningmaschine imitierte der Vorgehensweise beim schriftlichen Multiplizieren. Die Maschine geht zunächst an das Leerzeichen am rechten Ende des Eingabewortes. Die Maschine bewegt sich nun zwei Schritte nach links und liest die Zahlen ein und addiert sie. Schließlich bewegt sich die Maschine einen Schritt nach rechts und schreibt das Ergebnis der Additium. Dabei wird das Eingabewort überschrieben allmählich überschrieben.

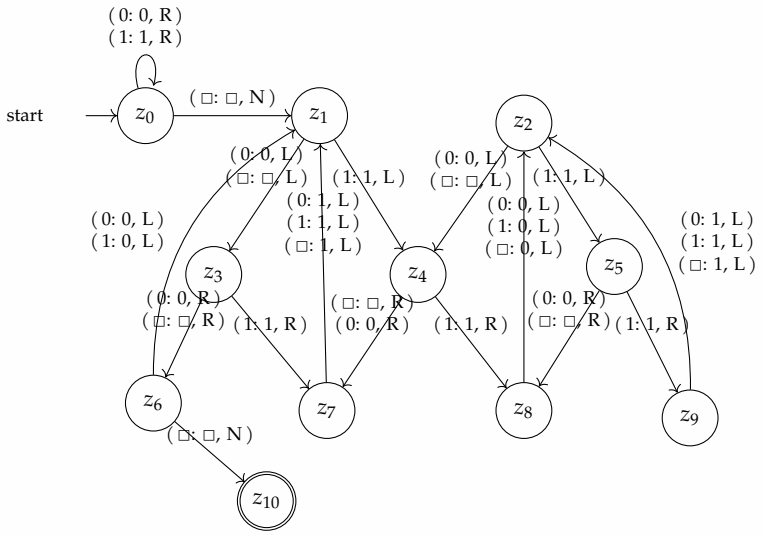
(b) Geben Sie dann das kommentierte Programm der Turing-Maschine an und erklären Sie die Bedeutung der verwendeten Zustände.

Lösungsvorschlag

z_0	An das rechte Ende des Eingabewortes gehen
z_1	Übertrag 0
z_2	Übertrag 1
z_3	1. Additionsschritt: $+0$
z_4	1. Additionsschritt: $+1$
z_5	1. Additionsschritt: $+2$
z_6	2. Additionsschritt: $+0$
z_7	2. Additionsschritt: $+1$
z_8	2. Additionsschritt: $+2$
z_9	2. Additionsschritt: $+3$
z_{10}	Endzustand

Nicht sehr übersichtlich hier: Im Anhang findet sich die JSON-Datei für fla-

ci.com



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahjkw50kg

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-2/Aufgabe-1.tex`

Übungsaufgabe „Binärzahl dekrementieren“ (Turing-Maschine)

Turing-Maschine

Sei $\Sigma = \{0, 1\}$ und $\Gamma = \{0, 1, \square\}$. Konstruiere eine Turingmaschine M, die eine in Binärform gegebene, natürliche Zahl $(\neq 0)$ um 1 dekrementiert (und wieder in Binärform ausgibt). Der Schreib-/Lesekopf steht zu Beginn der Berechnung auf dem ersten Leerzeichen links von der Eingabe und soll auch am Ende wieder dort stehen. Beachte, dass führende Nullen in der Eingabe/Ausgabe nicht vorkommen dürfen.

Lösungsvorschlag

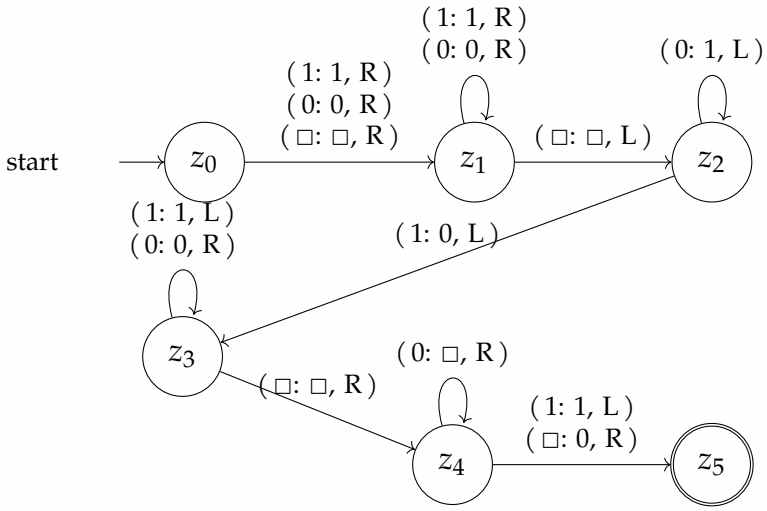
dezimal	binär
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

Die Maschine geht zunächst ans rechte Ende des Wortes, dann invertiert sie alle 0 Bits, bis sie auf eine 1 trifft. Diese wird durch 0 ersetzt. Damit ist der Dekrementierungsvorgang beendet. Nun sucht Sie das linke Ende des Wortes und löscht eventuell entstandene führende Nullen. Trifft Sie dabei auf das Leerzeichen, so war die Ausgabe die Zahl 0 und diese wird wieder aufs Band geschrieben. Insgesamt ergibt sich

$$\text{TM} = (\{z_0, z_1, z_2, z_3, z_4, z_5\}, \Sigma, \Gamma, \delta, z_0, \square, \{z_5\})$$

mit unten angegebener Übergangsfunktion:

δ	0	1	d	Kommentar
z0	$\emptyset$	$\emptyset$	(z_1: □, R)	Gehe auf erstes Zeichen des Wortes
z1	(z_1: 0, R)	(z_1: 1, R)	(z_2: □, L)	Gehe ans rechte Ende des Wortes
z2	(z_2: 1, L)	(z_3: 0, L)	$\emptyset$	Flippe alle 0 Bits bis die erste 1 erreicht wird, setzte die
z3	(z_3: 0, L)	(z_3: 1, L)	(z_4: □, R)	suche linkes Ende des Wortes
z4	(z_4: □, R)	(z_5: 1, L)	(z_5: 0, L)	lösche führende Nullen , schreibe evtl. 0 aufs Band
z5	$\emptyset$	$\emptyset$	$\emptyset$	Endzustand



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahifz611c

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Binaerzahl-dekrementieren.tex`

Übungsaufgabe „Turing-Maschine Multiplikation“ (Turing-Maschine)[Turing-Maschine]

Gesucht ist eine Turing-Maschine, die die Funktion $f : \mathbf{N} \to \mathbf{N}$ mit $f(x) = 3x$ berechnet. Zu Beginn der Berechnung steht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

(a) Überlege, was bei der Multiplikation mit 3 im Binären tatsächlich passiert. Leite hieraus die Arbeitsweise des Algorithmus für die Turingmaschine ab und beschreibe diese. Tipp: Die schriftliche Multiplikation mit Binärzahlen funktioniert genauso wie die schriftliche Multiplikation mit Dezimalzahlen!

Lösungsvorschlag

$13 \cdot 3 = 0b1101 \cdot 0b11 = 39 = 0b100111$:

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ & & & 1 & 1 & 0 & 1 \\ \hline {\scriptstyle 1} & {\scriptstyle 1} & & & & \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \end{array}$$

(b) Erstelle dazu eine TM mit 3 Bändern. Die ersten beiden Bänder sollen für die Berechnung herangezogen werden. Auf dem dritten Band soll das Ergebnis stehen. (analog zur schriftlichen Multiplikation)

Lösungsvorschlag

- z_0 Versetzung der Zeiger von Band 2 und 3 im Vergleich zu 1.
- z_1 kopiert das Wort von Band 1 auf Band 2 (mit Versetzung).
- z_2 Binäre Addition ohne Übertrag.
- z_3 Binäre Addition mit Übertrag.
- z_4 Übertrag am Ende der Addition falls sich das Wort vergrößert
- z_f fertig

```
name: Multiplikation mit 3 (3-Band)
init: z0
accept: zf

z0, 0,_,_
z1, 0,_,_, -,>,>
z0, 1,_,_
z1, 1,_,_, -,>,>

z1, 0,_,_
z1, 0,0,_, >,>,>
z1, 1,_,_
```

```
z1, 1,1,_, >,>,>

z1, _,_,_
z2, _,_,_, -,<,<

z2, _,0,_
z2, _,0,0, <,<,<
z2, _,1,_
z2, _,1,1, <,<,<
z2, 0,0,_
z2, 0,0,0, <,<,<
z2, 0,1,_
z2, 0,1,1, <,<,<
z2, 1,0,_
z2, 1,0,1, <,<,<

z2, 1,_,_
zf, 1,_,1, -,-,-

z2, 1,1,_
z3, 1,1,0, <,<,<

z3, 0,1,_
z3, 0,1,0, <,<,<
z3, 1,0,_
z3, 1,0,0, <,<,<
z3, 1,1,_
z3, 1,1,1, <,<,<

z3, 0,0,_
z2, 0,0,1, <,<,<

z3, 1,_,_
z4, 1,_,0, <,<,<

z4, _,_,_
zf, _,_,1, -,-,-
```

a

[a] http://turingmachinesimulator.com/shared/tgaybidewo

(c) Erstelle dazu eine TM mit 2 Bändern. Auf dem ersten Band steht die Eingabe und auf dem zweiten Band soll das Ergebnis stehen.

Lösungsvorschlag

z_0	Die Binärzahl überlaufen
z_1	Zur nächsten Ziffer muss 0 addiert werden.
z_2	Zur nächsten Ziffer muss 1 addiert werden.
z_3	Zur nächsten Ziffer muss 2 addiert werden.
z_f	fertig

```
name: Multiplikation mit 3 (2-Band)
init: z0
accept: zf

z0, 0,_
z0, 0,_, >,-
z0, 1,_
z0, 1,_, >,-

z0, _,_
z1, _,_, <,-

z1, 0,_
z1, 0,0, <,<

z1, 1,_
z2, 1,1, <,<

z2, 1,_
z3, 1,0, <,<

z3, 0,_
z2, 0,0, <,<
z3, _,_
z2, _,0, -,<

z2, 0,_
z1, 0,1, <,<
z2, _,_
z1, _,1, -,<

z3, 1,_
z3, 1,1, <,<

z1, _,_
zf, _,_, -,-
```

a

[a] http://turingmachinesimulator.com/shared/caoxpsgrgl

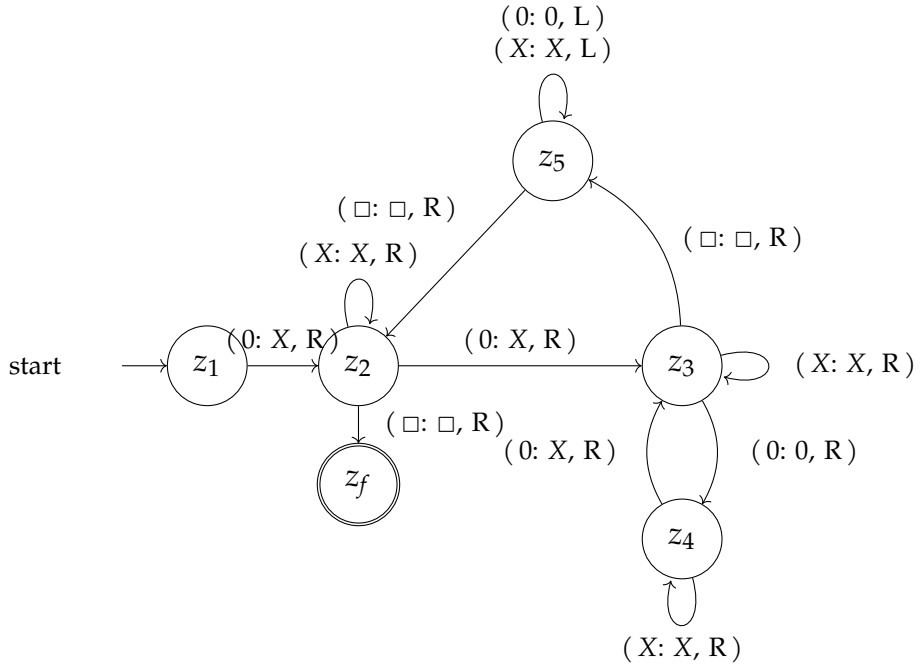
Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Turing-Maschine-Multiplikation.tex

Übungsaufgabe „Übergangsfunktion“ (Turing-Maschine)

Turing-Maschine

Gegeben sei eine TM mit folgender Übergangsfunktion:

	z_1	z_2	z_3	z_4	z_5
0	(z_2: □, R)	(z_3: X, R)	(z_4: 0, R)	(z_3: X, R)	(z_5: 0, L)
X	-	(z_2: X, R)	(z_3: X, R)	(z_4: X, R)	(z_5: X, L)
□	-	(z_f: □, R)	(z_5: □, L)	-	(z_2: □, R)



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apew8cea2

Erreicht die TM den Zustand z_f (final), so hält sie an und bearbeitet keine weitere Eingabe. Zu Beginn der Berechnung soll die TM auf dem ersten Symbol der Eingabe (links) stehen.

(a) Gebe für die folgenden Eingaben die Konfigurationsfolgen der Berechnung an:

- 00000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$$
\begin{aligned}
z_1\,00000 &\rightarrow \\
&\rightarrow\ \square\ z_2\,0000 \\
&\rightarrow\ \square\, X\, z_3\,000 \\
&\rightarrow\ \square\, X0\, z_4\,00 \\
&\rightarrow\ \square\, X0X\, z_3\,0 \\
&\rightarrow\ \square\, X0X0\, z_4
\end{aligned}
$$

- 000000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$$
\begin{aligned}
z_1\,000000 &\rightarrow \\
&\rightarrow\ \square\ z_2\,00000 \\
&\rightarrow\ \square\, X\, z_3\,0000 \\
&\rightarrow\ \square\, X0\, z_4\,000 \\
&\rightarrow\ \square\, X0X\, z_3\,00 \\
&\rightarrow\ \square\, X0X0\, z_4\,0 \\
&\rightarrow\ \square\, X0X0X\, z_3\ \square \\
&\rightarrow\ \square\, X0X0\, z_5\, X\,\square \\
&\rightarrow\ \square\, X0X\, z_5\, 0X\,\square \\
&\rightarrow\ \square\, X0\, z_5\, X0X\,\square \\
&\rightarrow\ \square\, X\, z_5\, 0X0X\,\square \\
&\rightarrow\ \square\ z_5\, X0X0X\,\square \\
&\rightarrow\ z_5\ \square\, X0X0X\,\square \\
&\rightarrow\ \square\ z_2\, X0X0X\,\square \\
&\rightarrow\ \square\, X\, z_2\, 0X0X\,\square \\
&\rightarrow\ \square\, XX\, z_3\, X0X\,\square \\
&\rightarrow\ \square\, XXX\, z_3\, 0X\,\square \\
&\rightarrow\ \square\, XXX0\, z_4\, X\,\square \\
&\rightarrow\ \square\, XXX0X\, z_4\ \square
\end{aligned}
$$

- 0000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$$
\begin{aligned}
z_1\,000 &\rightarrow \\
&\rightarrow\ \square\ z_1\,000 \\
&\rightarrow\ \square\ X\,z_3\,00 \\
&\rightarrow\ \square\ X0\,z_4\,0 \\
&\rightarrow\ \square\ X0X\,z_3\ \square \\
&\rightarrow\ \square\ X0\,z_5\,X\,\square \\
&\rightarrow\ \square\ X\,z_5\,0X\,\square \\
&\rightarrow\ \square\ z_5\,X0X\,\square \\
&\rightarrow\ z_5\ \square\,X0X\,\square \\
&\rightarrow\ \square\ z_2\,X0X\,\square \\
&\rightarrow\ \square\ X\,z_2\,0X\,\square \\
&\rightarrow\ \square\ XX\,z_3\,X\,\square \\
&\rightarrow\ \square\ XXX\,z_3\ \square \\
&\rightarrow\ \square\ XX\,z_5\,X\,\square \\
&\rightarrow\ \square\ X\,z_5\,XX\,\square \\
&\rightarrow\ \square\ z_5\,XXX\,\square \\
&\rightarrow\ z_5\ \square\,XXX\,\square \\
&\rightarrow\ \square\ z_2\,XXX\,\square \\
&\rightarrow\ \square\ X\,z_2\,XX\,\square \\
&\rightarrow\ \square\ XX\,z_2\,X\,\square \\
&\rightarrow\ \square\ XXX\,z_2\ \square \\
&\rightarrow\ \square\ XXX\,\square\ z_f
\end{aligned}
$$

(b) Gebe zwei andere Wörter über der Sprache $L \subset \{\, 0^* \,\}$ an, für die TM im Zustand z_f endet.

Lösungsvorschlag

z. B. 0 oder 00

(c) Für welche Sprache ist die TM ein Akzeptor?

Lösungsvorschlag

Die TM erkennt alle Wörter mit der Eigenschaft, dass die Anzahl der Nullen eine 2er-Potenzen ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Uebergangsfunktion.tex

Übungsaufgabe „Vorlesungsaufgaben ab-Wörter umkehren 2-Band-Turingmaschine (Turing-Maschine)

Turing-Maschine

(a) Geben Sie eine 2-Band-Turingmaschine an, die die Eingabe über dem Alphabet $\Sigma = \{a, b\}$ umkehrt.

Beispiele:

- $abb \rightarrow bba$
- $aaaaba \rightarrow abaaaa$
- $aaa \rightarrow aaa$

Tipp: Das Ergebniswort muss nicht auf dem 1. Band stehen.

Lösungsvorschlag

```
name: ab-Wörter umkehren 2-Band-Turingmaschine
init: z0
accept: z0

z0,a,_
z0,_,a,>,<

z0,b,_
z0,_,b,>,<
```

a

[a] http://turingmachinesimulator.com/shared/iqwxphngwc

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Vorlesungsaufgaben-Umkehren-2-Band.tex

Übungsaufgabe „Vorlesungsaufgaben ab-Wörter umkehren“ (Turing-Maschine)

Turing-Maschine

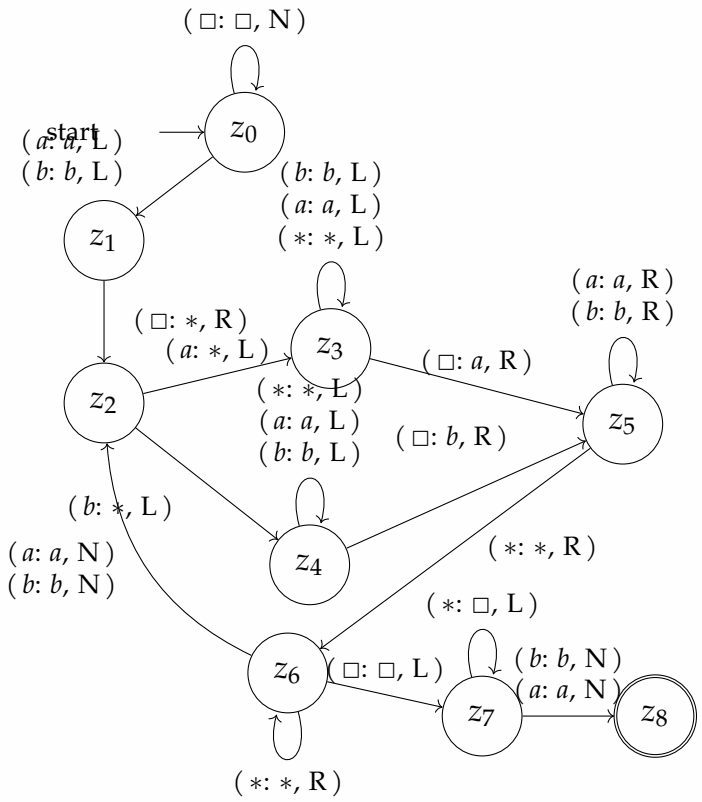
(a) Geben Sie eine Turingmaschine an, die die Eingabe über dem Alphabet $\Sigma = \{a, b\}$ umkehrt.

Beispiele:

- $abb \rightarrow bba$
- $aaaaba \rightarrow abaaaa$
- $aaa \rightarrow aaa$

Tipp: Fügen Sie ein extra Zeichen ein, welches das Eingabewort von deinem umgedrehten Wort trennt. Das Ergebniswort muss nicht an derselben Stelle wie das Eingabewort stehen.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af75rdjbc

(b) Geben Sie anschließend eine Konfigurationsfolge ihrer TM für *ab* an.

Lösungsvorschlag

$$
\begin{aligned}
z_0\,ab &\rightarrow\ z_1\ \square\, ab\\
&\rightarrow\ z_2\ *\, ab\\
&\rightarrow *\, z_3\ *\, b\\
&\rightarrow\ z_3\ **b\\
&\rightarrow\ z_3\ \square\ **b\\
&\rightarrow a\, z_5\ **b\\
&\rightarrow a*\ z_6\ *\, b\\
&\rightarrow a**\, z_6\, b\\
&\rightarrow a**\, z_2\, b\\
&\rightarrow a*\ z_4\ **\\
&\rightarrow a\, z_4\ ***\\
&\rightarrow\ z_4\, a***\\
&\rightarrow\ z_4\ \square\, a***\\
&\rightarrow b\, z_5\, a***\\
&\rightarrow ba\, z_6\ ***\\
&\rightarrow ba*\ z_6\ **\\
&\rightarrow ba**\, z_6\, *\\
&\rightarrow ba***\ z_6\ \square\\
&\rightarrow ba**\, z_7\ \square\\
&\rightarrow ba*\ z_7\ \square\\
&\rightarrow ba\, z_7\ \square\\
&\rightarrow b\, z_8\, a
\end{aligned}
$$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_Vorlesungsaufgaben-Umkehren.tex`

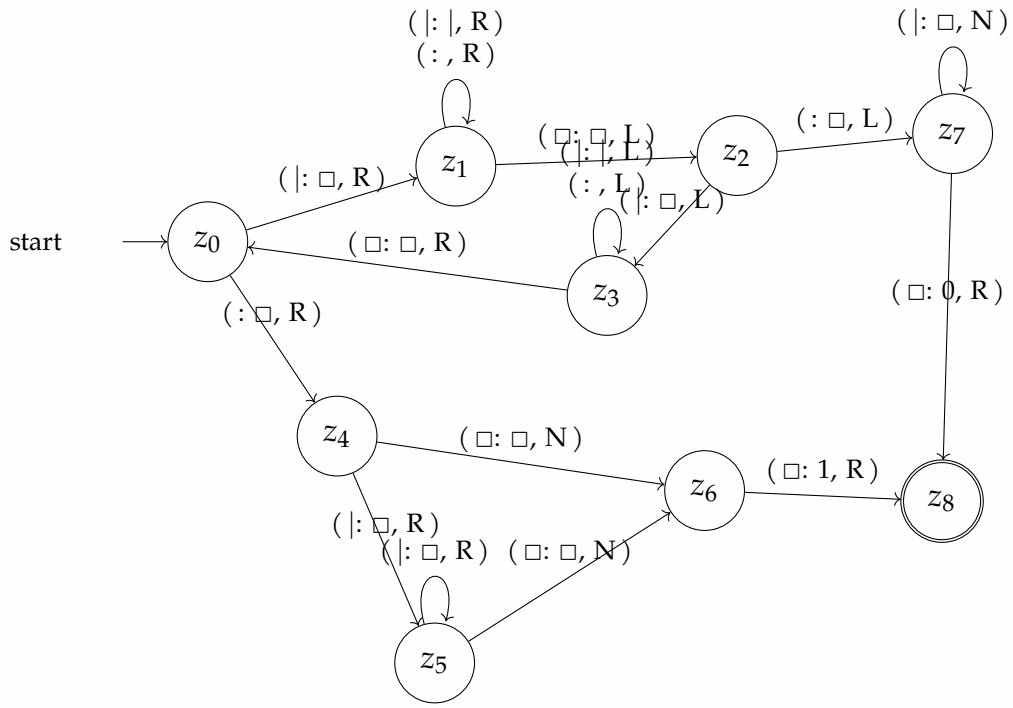
Übungsaufgabe „unäre Kodierung von n und m“ (Turing-Maschine)

Turing-Maschine

Konstruieren Sie eine Einband-Turingmaschine, die für eine Eingabe der Form $|^n \circ |^m$ mit $n, m \in \mathbb{N}$ (also die unäre Kodierung von n und m durch $\circ$ getrennt) das Prädikat $n \leq m$ berechnet.

$$f(x) = \begin{cases} 1 \text{ (akzeptiert)} & n \leq m \\ 0 \text{ (nicht akzeptiert)} & \text{sonst} \end{cases}$$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajf6mi7e7

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/30_Typ-1_Kontextsensitiv/Turing-Maschine/Aufgabe_unaere-Kodierung-n-m.tex`

Übungsaufgabe „a-2-hoch-n“ (Unbeschränkte Sprache)

Unbeschränkte Sprache

Die folgende Grammatik erzeugt die Sprache $L = \{ a^{2^n} \mid n \in \mathbb{N} \}$, nicht kontextsensitiv ist:

$$G = (\{S, L, R, D\}, \{a\}, P, S)$$

$$\mathrm{P} = \left\{ \begin{aligned} S &\to LaR \\ L &\to LD \mid \varepsilon \\ Da &\to aaD \\ DR &\to R \\ R &\to \varepsilon \end{aligned} \right\}$$

Leiten Sie das Wort a^8 ab.

Lösungsvorschlag

$LaR \vdash LDaR \vdash LDDaR \vdash LDDDaR \vdash DDDaR \vdash DDaaDR \vdash DaaDaDR \vdash aaDaDaDR \vdash aaaaDDaDR \vdash aaaaaaaaDDDR \vdash aaaaaaaaDDR \vdash aaaaaaaaR \vdash aaaaaaaa$

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/10_Formale-Sprachen/40_Typ-0_Phrasenstruktur/Aufgabe_a-2-hoch-n.tex`

Berechenbarkeit

Examensaufgabe „Gödelisierung aller Registermaschinen (RAMs)“ (66115-2015-F.T2-A4)

Sei M 0 , M 1 , . . . eine Gödelisierung aller Registermaschinen (RAMs). Geben Sie für die folgenden Mengen D 1 , D 2 , D 3 an, ob sie entscheidbar oder aufzählbar sind. Begründen Sie Ihre Behauptungen, wobei Sie die Aufzählbarkeit und Un- entscheidbarkeit des speziellen Halteproblems K 0 = x ∈ N|M x haelt bei Eingabe x verwenden dürfen. D 1 = x ∈ N|x < 9973und M x haelt bei Eingabe x D 2 = x ∈ N|x ≥ 9973und M x haelt bei Eingabe x D 3 = x ∈ N|M x haelt nicht bei Eingabe x

Lösungsvorschlag

D 1 ist eine endliche Menge und damit entscheidbar. Auch eine endliche Teilmenge des Halteproblems. Anschaulich kann man sich dies so verstellen: Man stellt dem Rechner eine Liste zur Verfügung, die alle haltenden Maschinen M x mit x < 9973 enthält. Diese Liste kann zum Beispiel vorab von einem Menschen erstellt worden sein, denn die Menge der zu prüfenden Programme ist endlich.

D 2 x ≥ 9973 entscheidbar, L halt semi-entscheidbar → semi-entscheidbar (Hier wäre auch eine Argumentation über die Cantorsche Paarungsfunktion möglich). Es ist weiterhin nicht entscheidbar. Dazu betrachten wir dei Reduktin des speziellen Halteproblems H 0 : H 0 ≤ D 2 Für alle x < 9973 lassen wir M x durch eine Turingmaschine M y simulieren, die eine höhere Nummer hat.

D 3 ist unentscheidbar, denn angenommen D 3 wäre semi-entscheidbar, dann würde sofort folgen, dass L halt entscheidbar ist, da aus der Semientscheidbarkeit von L halt und L halt die Entscheidbarkeit von L halt folgen würde

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/03/Thema-2/Aufgabe-4.tex`

Examensaufgabe „Verständnis Berechenbarkeitstheorie“ (66115-2016-F.T2-A2)

Berechenbarkeit

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

(a) Warum genügt es, sich auf Funktionen zu beschränken, die natürliche Zahlen auf natürliche Zahlen abbilden, wenn man untersuchen will, was heutige Computer im Prinzip berechnen können?

Lösungsvorschlag

Jede WHILE-berechenbare Funktion ist turing-berechenbar. Jede turing-berechenbare Funktion ist WHILE-berechenbar. Jede nichtdeterministische Turing-Maschine kann durch eine deterministische Turing-Maschine simuliert werden.

(b) Was besagt die Church-Turing-These? Könnte man sie beweisen oder widerlegen?

Lösungsvorschlag

Die Church-Turing-These besagt, dass die Klasse der turing-berechenbaren Funktionen mit der Klasse der intuitiv berechenbaren Funktionen übereinstimmt. Die These kann nicht bewiesen oder widerlegt werden, weil es sich bei dem Begriff *„intuitiv berechenbare Funktion“* um keinen mathematisch exakt definitierten Begriff handelt. Würde man ihn genau definierten, würde ein konkretes Berechnungsmodell festlegt werden, was der Kernaussage dieses Begriffes widersprechen würde.

(c) Für reelle Zahlen, wie z. B. π, lässt sich die Dezimaldarstellung durch entsprechende Programme beliebig genau approximieren. Gilt das für alle reellen Zahlen, ðlässt sich für jede reelle Zahl die Dezimaldarstellung mit entsprechenden Programmen beliebig genau approximieren?

Lösungsvorschlag

Ja mit einer Berechnungsvorschrift, ja solange der speicherplatz reicht, z. B. mittels Intervallschachtelung.

(d) Was ist für die Berechnungskraft der wesentliche Unterschied zwischen While-Berechenbarkeit und Loop-Berechenbarkeit.

Lösungsvorschlag

Alle LOOP-Programme sind mathematisch betrachtet totale Funktionen, ðsie terminieren immer. WHILE-Programme hingegen partiellen Funktionen, die nicht für alle Eingabekombinationen terminieren.

(e) Die Ackermannfunktion ist ein Beispiel einer totalen Funktion, die While-berechenbar, aber nicht Loop-berechenbar ist. Sie verallgemeinert die Idee, dass Multiplikation

die wiederholte Addition ist, Exponentiation die wiederholte Multiplikation, Hyperexponentiation die wiederholte Exponentiation usw. Die Stufe dieser hyper-hyper ... Exponentiation ist ein Parameter der Ackermannfunktion. Generieren Sie aus dieser Idee ein Argument, das illustriert, warum die Ackermannfunktion nicht Loop-berechenbar ist.

Lösungsvorschlag

Jedes LOOP-Programm benötigt zur Berechnung der Funktion $x \uparrow^n y$ mindestens $n + 2$ Schleifen.

Gäbe es ein LOOP-Programm, das $ack(n, m)$ tatsächlich für beliebige Werte von n berechnet, so müsste dieses — im Widerspruch zum endlichen Aufbau eines Loop-Programms — unendlich viele Schleifenkonstrukte enthalten.

Hyperexponentiation

- $x \uparrow^{-1} y = x + y$
- $x \uparrow^0 y = x \cdot y$
- $x \uparrow^1 y = x^y$
- $x \uparrow^2 y = x^{y^y}$

(f) Geben Sie ein Beispiel einer Menge an, die abzählbar, aber nicht rekursiv aufzählbar ist, und begründen Sie es.

Lösungsvorschlag

- Die Menge der Primzahlen. Die Primzahl sind unendlich groß und könnten abgezählt werden. Bei Primzahlen gibt es keine Berechnungsvorschrift, die z. b. die 7. Primzahl berechnet.
- Die Menge der Turningmaschine
- Diagonalsprache
- Das Komplement des Halteproblems. Die dem Halteproblem zugrundeliegende Lösungsmenge ist rekursiv aufzählbar. Wäre das Komplement des Halteproblems auch rekursiv aufzählbar, dann wäre das Halteproblem entscheidbar, was es aber nicht ist. [a]

[a] `https://www.youtube.com/watch?v=om4ZTOeQuD0`

(g) Wie ist der Zusammenhang zwischen rekursiv aufzählbar und semi-entscheidbar?

Lösungsvorschlag

Die beiden Begriffe sind äquivalent. [a]

[a] `https://www.youtube.com/watch?v=om4ZTOeQuD0`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-2/Aufgabe-2.tex`

Examensaufgabe „Registermaschinen (RAMs)“ (66115-2016-H.T2-A3)

Berechenbarkeit

Sei $M_0, M_1, \ldots$ eine Registermaschinen (RAMs). Beantworten Sie folgende Fragen zur Aufzählbarkeit und Entscheidbarkeit. Beweisen Sie Ihre Antwort.

Exkurs: Registermaschinen (RAMs)
Die Random Access Machine (kurz RAM) ist eine spezielle Art von Registermaschine. Sie hat die Fähigkeit der indirekten Adressierung der Register.

Die Random Access Machine besteht aus:

- einem Programm bestehend aus endlich vielen durchnummerierten Befehlen (beginnend mit Nummer 1)
- einem Befehlszähler b
- einem Akkumulator $c(0)$
- und einem unendlich großen Speicher aus durchnummerierten Speicherzellen (Registern) $c(1), c(2), c(3), \ldots$

Jedes Register (einschließlich b und $c(0)$) speichert eine beliebig große natürliche Zahl.

(a) Ist folgende Menge entscheidbar?

$A = \{x \in \mathbb{N} \mid x = 100 \text{ oder } M_x \text{ hält bei Eingabe } x\}$

Lösungsvorschlag

Ja, $x \geq 100$ ist entscheidbar und aufgrund des „oder“ ist die 2. Bedingung nur für $x < 100$ relevant. Da $x < 100$ eine endliche Menge darstellt, kann eine endliche Liste geführt werden und ein Experte kann für jeden Fall entscheiden, ob M_x hält oder nicht, somit ist A entscheidbar.

(b) Ist folgende Menge entscheidbar?

$B = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid M_x \text{ hält bei Eingabe } x \text{ genau dann, wenn } M_y \text{ bei Eingabe } y \text{ hält}\}$

Lösungsvorschlag

Nein. Dieses Problem entspricht der parallelen Ausführung des Halteproblems auf zwei Bändern. Das Halteproblem ist unentscheidbar, damit ist auch die parallele Ausfürhung des Halteproblems und damit B unentscheidbar.

(c) Ist folgende Menge aufzählbar?

$C = \{x \in \mathbb{N} \mid M_x \text{ hält bei Eingabe 0 mit dem Ergebnis 1}\}$

Lösungsvorschlag

Ja, die Menge ist aufzählbar, da die Menge aller Turningmaschinen aufzählbar und über natürliche Zahlen definiert ist (die wiederum aufzählbar sind).

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/09/Thema-2/Aufgabe-3.tex`

Examensaufgabe „Berechenbarkeitstheorie“ (66115-2020-F.T2-A4)

Berechenbarkeit

$A = \{ (M) \mid M$ ist Turingmaschine, die bei Eingabe 101 hält $\}$. Dabei bezeichnet (M) die Gödelnummer der Turingmaschine M.

(a) Zeigen Sie, dass A unentscheidbar ist.

Lösungsvorschlag

Reduktionsbeweis von H $0 \leq$ A: TM U

(i) Die zu $\square' \in \square 0$ passende TM $\square * \in \square$ aus A suchen mit $< \square' > = < \square * >$

(ii) 101 auf das Band schreiben

(iii) $\square *$ auf 101 starten

Damit könnte U H 0 entscheiden, was aber ein Widerspruch zu H0 semi- entscheidbar ist. Damit ist A ebenfalls semi-entscheidbar.

(b) Zeigen Sie, dass A semi-entscheidbar ist.

Lösungsvorschlag

siehe a)

(c) Ist das Komplement A^c von A entscheidbar? Ist es semi-entscheidbar? Begründen Sie Ihre Antworten.

Hinweis: Sie können die Aussagen aus Teilaufgabe a) und b) verwenden, auch wenn Sie sie nicht bewiesen haben.

Lösungsvorschlag

Wenn A unentscheidbar ist, dann kann entweder A oder A^c semi-entscheidbar sein. Wären beide semi-entscheidbar, dann wäre A aber ebenfalls entscheidbar, was aber nach Voraussetzung ausgeschlossen ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/03/Thema-2/Aufgabe-4.tex`

Übungsaufgabe „GOTO-Programme“ (GOTO-berechenbar)

GOTO-berechenbar

(a) Terminieren GOTO-Programme immer? Begründe Deine Antwort.

Lösungsvorschlag

GOTO-Programme terminieren nicht immer.

Variante 1: Die Menge der GOTO-Programme ist identisch mit der Menge der WHILE-Programme. Da WHILE-Programme partielle Funktionen beschreiben und diese nicht für alle Eingaben terminieren, terminieren GOTO-Programme ebenfalls nicht für alle Eingaben.

Variante 2: Die charakteristische Funktion einer semi-entscheidbaren Sprache ist Turing- bzw. GOTO-berechenbar, d.h. zu jeder semi-entscheidbaren Sprache gibt es eine Turing-Maschine. GOTO-Programme können Turing-Maschinen simulieren. Da hier von einer nur semi-entscheidbaren Sprache ausgegangen wird, terminiert das GOTO-Programm nicht, falls die Eingabe x kein Element der Sprache ist.

(b) Gebe ein GOTO-Programm an, dass die Summe dreier Zahlen berechnen.

Lösungsvorschlag

```
Eingabe x_1, x_2, x_3;
x_0 := x_1;
IF x_2 = 0 GOTO Z6;
  x_0 := x_0 + 1;
  x_2 := x_2 - 1;
GOTO Z2;
IF x_3 = 0 GOTO Z10;
  x_0 := x_0 + 1;
  x_3 := x_3 - 1;
GOTO Z6;
END;
Ausgabe: x_0
```

(c) Gegeben ist das GOTO-Programm:

```
x_4 := x_1;
IF x_4 = 0 GOTO Z10;
  x_5 := x_2;
IF x_5 = 0 GOTO Z8;
  x_3 := x_3 + 1;
  x_5 := x_5 - 1;
GOTO Z4;
  x_4 := x_4 - 1;
GOTO Z2;
  x_5 := x_5 - 1
```

(i) Was berechnet das Programm?

Lösungsvorschlag

$$f(n,m) = n * m$$

(ii) Übertrage das Programm in ein WHILE-Programm.

Lösungsvorschlag

```
Eingabe x_1, x_2 :
x_4 := x_1;
WHILE x_4 <> 0 DO
  x_5 := x_2;
  WHILE x_5 <> 0 DO
    x_3 := x_3 + 1;
    x_5 := x_5 - 1
  END;
x_4 := x_4 - 1
END
Ausgabe x_0

Eingabe : x_1, x_2
x_0 := mult ( x_1, x_2 );
Ausgabe : x_0
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/20_Berechenbarkeit/Aufgabe_GOTO.tex

Übungsaufgabe „LOOP-Fakultät“ (LOOP-berechenbar)

LOOP-berechenbar

(a) Geben Sie ein LOOP-Programm an, das die Funktion $f(n) = n!$ berechnet.

Lösungsvorschlag

```
x_2 := 1;
LOOP x1 DO
  x_3 := x_3 + 1;
  x_2 := x_2 * x_3;
END
x_3 := 0;
RETURN x_2;
```

(b) Beweisen Sie:

Ist $f : N \to N$ LOOP-berechenbar, so ist auch $g : N \to N$ mit $g(n) = f(i)$ LOOP-berechenbar.

Lösungsvorschlag

Bei einem LOOP-Programm der Form LOOP x_i DO P END wird das Programm P so oft ausgeführt, wie der Wert der Variablen x_i zu Beginn angibt. Beweis:

```
x_0 := 0;
i := 0;
LOOP n DO
  i := i + 1;
  y := f(i);
  x_0 := x_0 + y;
END
RETURN x_0;
```

ist LOOP-berechenbar, da $f(n)$ LOOP-berechenbar ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/20_Berechenbarkeit/Aufgabe_LOOP-Fakultaet.tex

Übungsaufgabe „Vorlesungsaufgaben“ (Berechenbarkeit)

LOOP-Implementierung

(a) Geben Sie eine LOOP-Implementierung für

(i) $add(x_i, x_j)$

Lösungsvorschlag

```
x_0 := x_i;
LOOP x_j DO
  x0 := succ(x_0);
END
```

(ii) $mult(x_i, x_j)$

Lösungsvorschlag

```
x_0 := x_i;
LOOP x_j DO
  x0 := add(x_0, x_i);
END
```

(iii) $power(x_i, x_j)$

Lösungsvorschlag

```
x_0 := succ(0);
LOOP x_j DO
  x0 := mult(x_0, x_i);
END
```

(iv) $hyper(x_i, x_j)$

Lösungsvorschlag

```
x_0 := succ(0);
LOOP x_j DO
  x_0 := power(x_i, x_0);
END
```

(v) 2^{x_i}

Lösungsvorschlag

Mit *power*

```
x_0 := power(2, x_i);
```

Mit *mult*

```
x_0 := 1;
x_2 := 2;
LOOP x_i DO
  x0 := mult(x_0, x_2);
END
```

an.

(b) Beweisen Sie, dass der größte gemeinsame Teiler zweier natürlicher Zahlen LOOP-berechenbar ist.

Lösungsvorschlag

```
ggT(x_1, x_2)

x_3 := MAX(x_1, x_2);
x_4 := MIN(x_1, x_2);

LOOP x_4 DO
  x_5 := x_3 - x_4;
  x_3 := MAX(x_4, x_5);
  x_4 := MIN(x_4, x_5);
END
x_0 := x_3;
```

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/20_Berechenbarkeit/Aufgabe_LOOP-Vorlesungsaufgaben.tex

Übungsaufgabe „Primitiv-rekursiv“ (Berechenbarkeit)

(a) Begründe, dass folgende Funktionen primitiv-rekursiv sind: 1. f (x) = x! (1 wennx > 0 2. sig(x) = 0 sonst

Lösungsvorschlag

Begründung durch eine Angabe einer Funktion: 1. f (0) = 1, f (n + 1) = mult(n + 1, f (n)) 2. sig(0) = 0, sig(n + 1) = 1

(b) Gebe eine konkrete primitiv-rekursive Implementierung für if x1 than x2 else x3 an. Wobei wie bei Programmiersprachen x1 als wahr gilt, wenn der Wert nicht Null ist.

Lösungsvorschlag

Zusätzlich werden die folgenden Funktionen festgelegt:

isZero(0) = 1 isZero(n) = isZero(n + 1) = 0 not(n) = 1 − n ite(x1, x2, x3) = isZero(x1) * x3 + not(isZero(x1)) * x2

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/20_Berechenbarkeit/Aufgabe_Primitiv-rekursiv.tex`

Übungsaufgabe „Vorlesungsaufgaben“ (Berechenbarkeit)

Geben Sie ein WHILE-Programm an, dass

- 2^{x_i}

Lösungsvorschlag

Ausnutzen der 2er-Potenzeigenschaft:

$2^1 = 1 + 1 = 2$

$2^2 = 2 + 2 = 4$

$2^3 = 4 + 4 = 8$

Hier werden nur die elementaren Bestandteile der WHILE-Sprache ausgenutzt.

```
x_2 := 1;
WHILE x_1 DO
  x_3 := x_2;
  WHILE x_3 DO
    x_2 := x_2 + 1;
    x_3 := x_3 - 1;
  END
  x_1 := x_1 - 1;
END
x_0 := x_2;
```

- $\text{ggt}(x_i, x_j)$

Lösungsvorschlag

Zusätzliche Voraussetzungen:

```
x_1 > x_2;
MOD(x_1, x_2);

x_0 := MOD(x_1, x_2);
WHILE x_0 DO
  x_1 := x_2;
  x_2 := x_0;
  x_0 := MOD(x_1, x_2);
END
```

berechnet.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/20_Berechenbarkeit/Aufgabe_WHILE-Vorlesungsaufgaben.tex

Entscheidbarkeit

Examensaufgabe „Halteproblem H m“ (66115-2014-F.T1-A5)

(a) Definieren Sie die zum Halteproblem für Turing-Maschinen bei fester Eingabe $m \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$ gehörende Menge H_m.

Lösungsvorschlag

$H_m = \{ c(M) \in \mathbb{N} \mid c(M)$ hält auf Eingabe $m \}$, wobei $c(M)$ der Codierung der Turingmaschine (Gödelnummer) entspricht.

(b) Gegeben sei das folgende Problem E:

Entscheiden Sie, ob es für die deterministische Turing-Maschine mit der Gödelnummer n eine Eingabe $w \in \mathbb{N}_0$ gibt, so dass w eine gerade Zahl ist und die Maschine n gestartet mit w hält.

Zeigen. Sie, dass E nicht entscheidbar ist. Benutzen Sie, dass H_m aus (a) für jedes $m \in \mathbb{N}_0$ nicht entscheidbar ist.

Lösungsvorschlag

Wir zeigen dies durch Reduktion $H_2 \leq E$:

- Berechenbare Funktion f: lösche Eingabe, schreibe eine 2 und starte dich selbst.
- M ist eine Turingmaschine, die E entscheidet.
- $x \in H_2$ (Quellcode der Programme, die auf die Eingabe von 2 halten)
- M_x (kompiliertes Programm, TM)
- Für alle $x \in H_2$ gilt, M_x hält auf Eingabe von 2 $\Leftrightarrow f(x) = c(M) \in E$. Denn sofern die ursprüngliche Maschine auf das Wort 2 hält, hält M auf alle Eingaben und somit auch auf Eingaben gerader Zahlen. Hält die ursprüngliche Maschine M nicht auf die Eingabe der Zahl 2, so hält M auf keine Eingabe.

(c) Zeigen Sie, dass das Problem E aus (b) partiell-entscheidbar (= rekursiv aufzählbar) ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2014/03/Thema-1/Aufgabe-5.tex`

Komplexitätstheorie

Examensaufgabe „NP“ (46115-2016-F.T1-A5)

Beschreiben Sie, was es heißt, dass ein Problem (Sprache) NP-vollständig ist. Geben Sie ein NP-vollständiges Problem Ihrer Wahl an und erläuteren Sie, dass (bzw.) warum es in NP liegt.

Lösungsvorschlag

NP-vollständig: NP-schwer und in NP

[Beliebiges Problem] liegt in NP, da der entsprechende Algorithmus dieses Problem nicht-deterministisch in Polynomialzeit löst → Algorithmus rät nichtdeterministisch Lösung, prüft sie in Polynomialzeit

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/03/Thema-1/Aufgabe-5.tex`

Examensaufgabe „SUBSET SUM, Raumausstattungsunternehmen“ (46 115. 2016-F.T2-A2)

Ein Raumausstattungsunternehmen steht immer wieder vor dem Problem, feststellen zu müssen, ob ein gegebener rechteckiger Fußboden mit rechteckigen Teppichresten ohne Verschnitt ausgelegt werden kann. Alle Längen sind hier ganzzahlige Meterbeträge. Haben sie beispielsweise zwei Reste der Größen 3×5 und einen Rest der Größe 2×5, so kann ein Fußboden der Größe 8×5 ausgelegt werden.

Das Unternehmen beauftragt eine Softwarefirma mit der Entwicklung eines Programms, welches diese Frage für beliebige Größen von Fußboden und Teppichresten entscheiden soll. Bei der Abnahme weist die Softwarefirma darauf hin, dass das Programm im wesentlichen alle Möglichkeiten durchprobiert und daher für große Eingaben schnell ineffizient wird. Auf die Frage, ob man das nicht besser machen könne, lautet die Antwort, dass das vorgelegte Problem NP-vollständig sei und daher nach derzeitigem Kenntnisstand der theoretischen Informatik nicht mehr zu erwarten sei.

> **Exkurs:**
> Das **Teilsummenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen $I = \{w_1, w_2, \ldots, w_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke c ist.

(a) Fixieren Sie ein geeignetes Format für Instanzen des Problems und geben Sie konkret an, wie die obige Beispielinstanz in diesem Format aussieht.

Lösungsvorschlag

Problem L

1. Alternative $I = \{x_1, y_1, \ldots, x_n, y_n, c_x, c_y\}$

2. Alternative $I = \{w_1, \ldots, w_n, c\}$

(b) Begründen Sie, dass das Problem in NP liegt.

Lösungsvorschlag

Es existiert ein nichtdeterministischer Algorithmus der das Problem in Polynomialzeit entscheidet:

- nichtdeterministisch Untermenge raten $(\mathcal{O}(n))$
- Prüfe: $(\mathcal{O}(n))$

1. Alternative Elementsumme der Produkte (x_i, y_i) aus Untermenge $= c$

2. Alternative Elementsumme der Untermenge $= c$

(c) Begründen Sie, dass das Problem NP-schwer ist durch Reduktion vom NP-vollständigen Problem SUBSET-SUM.

Lösungsvorschlag

$\preceq_p \mathtt{L}$

1. Alternative Die Funktion f ersetzt jedes w_i durch w_i , 1 und c durch c, 1 und startet TM für L

Berechenbarkeit: Hinzufügen von 1 für jedes Element, offensichtlich in Polynomialzeit

Korrektheit: $w \in \Leftrightarrow f(w) \in L$, offensichtlich, selbes Problem mit lediglich anders notierter Eingabe

2. Alternative Die Funktion f startet TM für L

Berechenbarkeit: Identität, offensichtlich in Polynomialzeit

Korrektheit: $w \in \Leftrightarrow f(w) \in L$ offensichtlich, selbes Problem

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/03/Thema-2/Aufgabe-2.tex

Examensaufgabe „VertexCover“ (46115-2016-H.T1-A4)

Polynomialzeitreduktion

Betrachten Sie die beiden folgenden Probleme:

VERTEX COVER

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \{1, 2, 3, \dots\}$

Frage: Gibt es eine Menge $C \subseteq V$ mit $|C| \leq k$, so dass für jede Kante (u, v) aus E mindestens einer der Knoten u und v in C ist?

VERTEX COVER 3

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \{3, 4, 5 \dots\}$.

Frage: Gibt es eine Menge $C \subseteq V$ mit $|C| \leq k$, so dass für jede Kante (u, v) aus E mindestens einer der Knoten u und v in C ist?

Geben Sie eine polynomielle Reduktion von VERTEX COVER auf VERTEX COVER 3 an und begründe anschließend, dass die Reduktion korrekt ist.

Exkurs: VERTEX COVER
Das **Knotenüberdeckungsproblem** (VERTEX COVER) fragt, ob zu einem gegebenen einfachen Graphen und einer natürlichen Zahl k eine Knotenüberdeckung der Größe von höchstens k existiert.

Das heißt, ob es eine aus maximal k Knoten bestehende Teilmenge U der Knotenmenge gibt, so dass jede Kante des Graphen mit mindestens einem Knoten aus U verbunden ist.

Lösungsvorschlag

VERTEX COVER $\preceq_p$ VERTEX COVER 3

f fügt vier neue Knoten hinzu, von denen jeweils ein Paar verbunden ist. Außerdem erhöht f k um 2.

Total: Jeder Graph kann durch f so verändert werden.

Korrektheit: Wenn VERTEX COVER für k in G existiert, dann existiert auch VERTEX COVER mit $k + 2$ Knoten in G', da für den eingefügten Teilgraphen ein VERTEX COVER mit $k = 2$ existiert.

In Polynomialzeit berechenbar: Für die Adjazenzmatrix des Graphen müssen lediglich vier neue Spalten und Zeilen eingefügt werden und $k + 2$ berechnet werden.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/46115/2016/09/Thema-1/Aufgabe-4.tex

Examensaufgabe „k-COL“ (66115-2016-F.T1-A5)

Polynomialzeitreduktion

Das Problem k-COL ist wie folgt definiert:

> **k-Col**
>
> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$.
>
> **Frage:** Kann man jedem Knoten v in V eine Zahl $z(v) \in \{1, \ldots, k\}$ zuordnen, so dass für alle Kanten $(u_1, u_2) \in E$ gilt: $z(u_1) \neq z(u_2)$?

Zeigen Sie, dass man 3-Col in polynomieller Zeit auf 4-Col reduzieren kann. Beschreiben Sie dazu die Reduktion und zeigen Sie anschließend ihre Korrektheit.

Lösungsvorschlag

Zu Zeigen:

$$\texttt{3-COL} \preceq_p \texttt{4-Col}$$

also 4-Col ist mindestens so schwer wie 3-COL Eingabeinstanz von 3-COL durch eine Funktion in eine Eingabeinstanz von 4-Col umbauen so, dass jede JA- bzw. NEIN-Instanz von 3-COL eine JA- bzw. NEIN-Instanz von 4-Col ist.

Funktion ergänzt einen beliebigen gegebenen Graphen um einen weiteren Knoten, der mit allen Knoten des ursprünglichen Graphen durch eine Kante verbunden ist.

total ja

in Polynomialzeit berechenbar ja (Begründung: z. B. Adjazenzmatrix $\rightarrow$ neue Spalte)

Korrektheit: ja

Färbe den „neuen“ Knoten mit einer Farbe. Da er mit allen anderen Knoten verbunden ist, bleiben für die übrigen Knoten nur drei Farben.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-1/Aufgabe-5.tex

Examensaufgabe „Verständnis“ (66115-2016-F.T2-A3)

Komplexitätstheorie

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

(a) In der O-Notation insbesondere für die Zeitkomplexität von Algorithmen lässt man i. A. konstante Faktoren oder kleinere Terme weg. Z. B. schreibt man anstelle $\mathcal{O}(3n^2 + 5)$ einfach nur $\mathcal{O}(n^2)$. Warum macht man das so?

Lösungsvorschlag

Das Wachstum im Unendlich ist bestimmt durch den größten Exponenten. Konstante falle bei einer asymptotischen. Analyse weg. nicht wesentlich schneller

(b) Was ist die typische Vorgehensweise, wenn man für ein neues Problem die NP-Vollständigkeit untersuchen will?

Lösungsvorschlag

Die alten Probleme werden reduziert. Das neue Problem ist größer als die alten Probleme. Das Problem muss in NP liegen.

(i) Problem *in* NP durch Angabe eines nichtdeterministischen Algorithmus in Polynomialzeit

(ii) Problem NP-schwer via Reduktion: $L_{\text{NP-vollständig}} \leq L_{\text{neues Problem}}$

(c) Was könnte man tun, um $P = NP$ zu beweisen?

Lösungsvorschlag

Es würde genügen, zu einem einzigen NP-Problem beweisen, dass es in P liegt.

Zu einem Problem einen deterministen Turingmaschin finden, die es in polynomineller Zeit löst.

(d) Sind NP-vollständige Problem mit Loop-Programmen lösbar? (Antwort mit Begründung!)

Lösungsvorschlag

nicht lösbar mit Loop-Programmen. Begründung ähnlich wie bei der Ackermann-Funktion. z. B. Passwort. Zu Passwort beliebig bräuchte man beliebige for schleifen, was dem endlichen Anzahl an Loop-Schleifen widerspricht.

(e) Wie zeigt man aus der NP-Härte des SAT-Problems die NP-Härte des 3SAT-Problems? (3SAT ist ein SAT-Problem wobei alle Klauseln maximal 3 Literale haben.)

Lösungsvorschlag

in den Lösungen enthalten

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2016/03/Thema-2/Aufgabe-3.tex

Examensaufgabe „SAT DOPPELSAT“ (66115-2019-H.T1-A4)

Polynomialzeitreduktion

Betrachten Sie die folgenden Probleme:

SAT
Gegeben: Aussagenlogische Formel F in KNF
Frage: Gibt es mindestens eine erfüllende Belegung für F?

DOPPELSAT
Gegeben: Aussagenlogische Formel F′ in KNF
Frage: Gibt es mindestens eine erfüllende Belegung für F, in der mindestens zwei Literale pro Klausel wahr sind?

(a) Führen Sie eine polynomielle Reduktion von SAT auf DOPPELSAT durch.

Lösungsvorschlag

https://courses.cs.washington.edu/courses/csep531/09wi/handouts/sol4.pdf

DOUBLE-SAT is in NP. The polynomial size certificate consists of two assignments f1 and f2. First, the verifier verifies if f1 6= f2. Then, it verifies if both assignments satisfy φ by subtituting the values for the variables and evaluate the clauses of φ. Both checks can be done in linear time. DOUBLE-SAT is NP-hard. We give a reduction from SAT. Given an instance φ of SAT which is a CNF formula of n variables x1,x2,...xn, we construct a new variable xn+1 and let ψ = φ ∧(xn+1 ∨¬xn+1) be the corresponding instance of DOUBLE-SAT. We claim that φ has a satisfying assignment iff ψ has at least two satisfying assignments. On one hand, if φ has a satisfying assignment f, we can obtain two distinct satisfying assignments of ψ by extending f with xn+1 = T and xn+1 = F respectively. On the other hand, if ψ has at least two sastisyfing assignments then the restriction of any of them to the set x1,x2,...xn is a satisfying assignment for φ. Thus, DOUBLE-SAT is NP-complete.

https://cs.stackexchange.com/questions/6371/proving-double-sat-is-np-complete

Here is one solution:

Clearly Double-SAT belongs to NP, since a NTM can decide Double-SAT as follows: On a Boolean input formula $\phi(x_1, \ldots, x_n)$, nondeterministically guess 2 assignments and verify whether both satisfy ϕ.

To show that Double-SAT is NP-Complete, we give a reduction from SAT to Double-SAT, as follows:

On input $\phi(x_1, \ldots, x_n)$:

1. Introduce a new variable y. 2. Output formula $\phi'(x_1, \ldots, x_n, y) = \phi(x_1, \ldots, x_n) \wedge (y \vee \bar{y})$.

If $\phi(x_1, \ldots, x_n)$ belongs to SAT, then ϕ has at least 1 satisfying assignment, and therefore $\phi'(x_1, \ldots, x_n, y)$ has at least 2 satisfying assignments as we can satisfy the new clause $(y \vee \bar{y})$ by assigning either $y = 1$ or $y = 0$ to the new variable y, so $\phi'(x_1, \ldots, x_n, y) \in$ Double-SAT.

On the other hand, if $\phi(x_1, \ldots, x_n) \notin$ SAT, then clearly $\phi'(x_1, \ldots, x_n, y) = \phi(x_1, \ldots, x_n) \wedge (y \vee \bar{y})$ has no satisfying assignment either, so $\phi'(x_1, \ldots, x_n, y) \notin$ Double-SAT.

Therefore, SAT $\leq_p$ Double-SAT, and hence Double-SAT is NP-Complete.

(b) Zeigen Sie, dass DOPPELSAT NP-vollständig ist.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2019/09/Thema-1/Aufgabe-4.tex`

Examensaufgabe „CLIQUE - ALMOST CLIQUE“ (66115-2021-F.T2-TA1-A4)

Polynomialzeitreduktion

Betrachten Sie die folgenden Probleme:

CLIQUE

Gegeben: Ein ungerichteter Graph $G = (V, E)$, eine Zahl $k \in \mathcal{N}$

Frage: Gibt es eine Menge $S \subseteq V$ mit $|S| = k$, sodass für alle Knoten $u \neq v \in V$ gilt, dass $\{u, v\}$ eine Kante in E ist?

ALMOST CLIQUE

Gegeben: Ein ungerichteter Graph $G = (V, E)$, eine Zahl $k \in \mathcal{N}$

Frage: Gibt es eine Menge $S \subseteq V$ mit $|S| = k$, sodass die Anzahl der Kanten zwischen Knoten in S genau $\frac{k(k-1)}{2} - 1$ ist?

Zeigen Sie, dass das Problem ALMOST CLIQUE NP-vollständig ist. Nutzen Sie dafür die NP-Vollständigkeit von CLIQUE.

Hinweis: Die Anzahl der Kanten einer k-Clique sind $\frac{k(k-1)}{2}$.

Exkurs: Cliquenproblem
Das **Cliquenproblem** fragt nach der Existenz einer Clique der Mindestgröße n in einem gegebenen Graphen. Eine Clique ist eine Teilmenge von Knoten in einem ungerichteten Graphen, bei der *jedes Knotenpaar durch eine Kante* verbunden ist.

Exkurs: `Almost Clique`
Eine Gruppe von Knoten wird ALMOST CLIQUE genannt, wenn nur eine Kante ergänzt werden muss, damit sie zu einer Clique wird.

Lösungsvorschlag

You can reduce to this from *CLIQUE*.

Given a graph $G = (V, E)$ and t, construct a new graph G^* by adding two new vertices $\{v_{n+1}, v_{n+2}\}$ and connecting them with all of G's vertices but removing the edge $\{v_{n+1}, v_{n+2}\}$, i.e. they are not neighbors in G^*. return G^* and $t + 2$.

If G has a t sized clique by adding it to the two vertices we get an $t + 2$ almost clique in G^* (by adding $\{v_{n+1}, v_{n+2}\}$).

If G^* has a $t + 2$ almost clique we can look at three cases:

1) It contains the two vertices $\{v_{n+1}, v_{n+2}\}$, then the missing edge must be $\{v_{n+1}, v_{n+2}\}$ and this implies that the other t vertices form a t clique in G.

2) It contains one of the vertices $\{v_{n+1}, v_{n+2}\}$, say w.l.o.g. v_{n+1}, then the missing edge must be inside G, say $e = \{u, v\} \in G$. If we remove u and v_{n+1} then the other t vertices, which are in G must form a clique of size t.

3) It does not contain any of the vertices $\{v_{n+1}, v_{n+2}\}$, then it is clear that this group is in G and must contain a clique of size t.

It is also clear that the reduction is in polynomial time, actually in linear time, log-space. [a]

[a] `https://cs.stackexchange.com/a/76627`

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
`https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-4.tex`

Übungsaufgabe „Reduktion-Turingmaschine“ (Polynomialzeitreduktion)

Polynomialzeitreduktion

Betrachten Sie das folgende Entscheidungsproblem:

Eingabe: eine geeignete codierte Turingmaschine M Ausgabe: entscheiden, ob die Turingmaschine M auf jedes Eingabewort nach höchstens 42 Schritten hält. Ist dieses Problem entscheidbar? Beweisen Sie Ihre Antwort.

Lösungsvorschlag

M sei TM. M liest in jedem Schritt höchstens ein Zeichen der Eingabe. ⇒Eingabe hat höchstens 42 Zeichen. ⇒Menge der zu entscheidenden Wörter ist endlich. ⇒Wir können alle Wörter der Sprache aufzählen und damit das Problem lösen.

Beweisen Sie mit Hilfe eines Reduktionsbeweises, dass das folgende Problem nicht entscheidbar ist:

Eingabe: zwei (geeignetes codierte) Turingmaschinen M 1 und M 2 sowie ein Eingabewort ω Ausgabe: entscheiden, ob M 1 auf Eingabewort ω hält und M 2 auf ω nicht hält.

Lösungsvorschlag

Das beschriebene Problem sei H N . Die TM M N , die zu H N gehört, sei wie folgt definiert: • Wir wählen eine zu ω passende TM M 0 aus dem Halteproblem H 0 aus, so dass M = (ω) hält. • Wir definieren eine TM M $\perp$, die zu keiner Eingabe hält. Dann ist für M N M 0 (w)#M $\perp$ (w) eine Möglichkeit für das Problem H N . Da aber H 0 nicht entscheidbar, so ist auch H N nicht entscheidbar.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/40_Komplexitaet/Aufgabe_Reduktion-Turingmaschine.tex

Übungsaufgabe „SAT-3SAT“ (Polynomialzeitreduktion)

Exkurs: Sat
Das **Erfüllbarkeitsproblem der Aussagenlogik** SAT und k-SAT mit $k \geq 3, k \in \mathbb{N}$ (Satz von Cook) fragt, ob eine aussagenlogische Formel erfüllbar ist. Das Erfüllbarkeitsproblem der *Aussagenlogik* ist in exponentieller Zeit in Abhängigkeit der Anzahl der Variablen mit Hilfe einer Wahrheitstabelle entscheidbar. Diese *Wahrheitstabelle* kann nicht in polynomieller Zeit aufgestellt werden.

(a) Wie zeigt man die aus der NP-Schwere des 3SAT-Problems die NP-Schwere des SAT-Problems?

Lösungsvorschlag

$$\text{3SAT} \preceq_p \text{SAT}$$

Jedes 3SAT-Problem ist auch ein SAT-Problem, weil $\text{3SAT} \subset \text{SAT}$. Damit braucht es keine Funktion (bzw. Identitäts-/Einheitsfunktion). Die Funktion ist korrekt, total und in Polynomialzeit anwendbar. Das SAT-Problem ist ebenfalls NP-schwer.

(b) Wie zeigt man die aus der NP-Schwere des SAT-Problems die NP-Schwere des 3SAT-Problems?

Lösungsvorschlag

$$\text{SAT} \preceq_p \text{3SAT}$$

Man muss eine Funktion finden, die eine allgemeine Aussagenlogik in eine Aussagenlogik mit 3 Literalen in konjunktiver Normalform umformt.

Durch die boolsche Algebra lässt sich jede logische Aussagenlogik in eine konjunktive Normalform bringen. Dies ist eine Konjunktion von Disjunktionstermen. Wir formen einen Disjunktionsterm mithilfe einer Funktion in ein 3SAT-Problem um. Diese Funktion kann auf jeden Disjunktionsterm angewendet werden und damit wird das gesamte SAT-Problem auf 3SAT reduzieren.

Die Funktion formt Formel aus SATmithilfe von Hilfsvariablen $h_1, \ldots, h_{n-2}$ derart um $(a_1 \vee \cdots \vee a_n) \rightarrow (a_1 \vee a_2 \vee h_1) \wedge (\neg h_1 \vee a_3 \vee h_2) \wedge (\neg h_2 \vee a_4 \vee h_3) \wedge \cdots \wedge (\neg h_{n-2} \vee a_n)$

total Diese Funktion ist total, denn jede in SATenthaltene Aussagenlogik kann so umgewandelt werden.

Korrektheit: Die Hilfsvariablen sind wahr, solange bis ein Literal a_x selber true ist. Ab diesem Zeitpunkt sind die Hilfsvariablen dann falsch.

JA-Instanzen: Der erste und alle mittleren Disjunktionstermen sind wahr, weil aufgrund der Nicht-Negierung und Negierung immer ein wahres Literal in den Disjunktionstermen. Somit ist dann auch der Disjunktionsterm wahr. Da es eine JA-Instanz ist, existiert ein a_x welches wahr ist. Somit sind ab diesem Zeitpunkt die Hilfvariablen falsch.

Der letzte Disjunktionsterm wird dadurch sicher wahr, weil $\neg h_{n-2}$ somit wahr ist.

NEIN-Instanz: Alle a_x sind falsch. Auch hier sind wieder der erste und alle mittleren Disjunktionsterme wahr (gleiche Begründung wie oben). Der letzte Disjunktionsterm ist allerdings falsch, weil die Hilfvariablen durchgehend wahr bleiben und alle a_x falsch sind. Durch die Konjunktion der Disjunktionsterme ist dann auch die Gesamtaussage falsch.

Polynomialzeit: Der Algorithmus, der Formeln aus SAT nach 3SAT umformt liegt in $\mathcal{O}(n)$ und somit in Polynomialzeit.

Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden:
https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/70_THEO/40_Komplexitaet/Aufgabe_SAT-3SAT.tex

Index

Ableitung (Kontextfreie Sprache), 1115, 1122, 1124, 1149
Ableitung (Reguläre Sprache), 1084
Ableitungsbaum, 1122, 1124
Abstrakte Fabrik (Abstract Factory), 790, 816
Abstrakte Klasse, 797, 809, 823
ACID, 294, 303
Adapter, 811, 816
Adjazenzliste, 564
Adjazenzmatrix, 564, 696, 707
Agile Methoden, 744, 758
Agiles Manifest, 765
Aktivitätsdiagramm, 868
Algorithmen und Datenstrukturen, 738
Algorithmische Komplexität (O-Notation), 354, 391, 407, 433, 435, 437, 442, 448, 454, 457, 461, 463, 466, 468–471
Algorithmus von Dijkstra, 390, 564, 697, 705, 707, 709, 711, 713, 720, 723, 725, 727, 729
Algorithmus von Kruskal, 701, 715
Algorithmus von Prim, 699, 702, 716, 725, 733
all uses, 422
ALTER TABLE, 159, 184, 201, 206
Anforderungsanalyse, 768
Anforderungsüberdeckung, 386
Anfrageoptimierung, 105
Anwendungsfalldiagramm, 785, 787, 788, 820, 868
Attributhülle, 288
Attributhüllen-Algorithmus, 288
AVG, 193
AVL-Baum, 388, 553, 558, 568, 582, 605, 614, 619, 626, 628, 633, 637, 643, 653, 683

B-Baum, 22, 558, 597, 627, 671, 673–676, 685, 687
Backtracking, 335, 482, 490, 506, 508
Beobachter (Observer), 790, 826
Berechenbarkeit, 1175, 1196, 1197, 1199, 1200, 1204, 1206, 1207
BETWEEN, 201
Binärbaum, 553, 568, 571, 587, 610, 616, 628, 647, 650, 665, 677
Binäre Suche, 348, 360, 362, 369, 373
Black-Box-Testing, 386, 751, 779
Blackboard-Muster, 758
Boyce-Codd-Normalform, 271
Breitensuche, 704, 719, 724, 734
Bubblesort, 377, 378, 386, 406, 421, 425, 432
Bucketsort, 402
Bäume, 626, 641

C0-Test Anweisungsüberdeckung (Statement Coverage), 950
C1-Test Zweigüberdeckung (Branch Coverage), 422, 933, 950
C2a Vollständige Pfadüberdeckung (Full Path Coverage), 387, 934
C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage), 933, 994, 997
CHECK, 201
Chomsky-Normalform, 1099, 1103, 1105, 1112, 1141, 1143
Client-Server-Modell, 922, 923, 926, 927
CONSTRAINT, 184
CPM-Netzplantechnik, 878, 886, 891, 893, 899, 910, 914, 916
CREATE TABLE, 150, 156, 157, 182, 187
CYK-Algorithmus, 1101, 1110, 1111, 1114, 1119, 1136, 1137, 1139, 1140

Datenbank, 25, 31, 39
Datenbank-Übersicht, 28, 37
Datenbankmanagementsystem, 39
Datenbanksystem, 39
Datenfluss-annotierter Kontrollflussgraph, 421, 935, 961, 993
Datenflussorientiertes Testen, 421
Datenunabhängigkeit, 20
DB, 308, 310, 312
Deadlock, 304
DELETE, 153, 160, 178, 187, 201
Delete-Anomalie, 281, 287, 291
DESC, 193

Design by Contract, 369
Deterministisch endlicher Automat (DEA), 1005, 1054, 1056, 1062, 1063, 1084
Dirty-Read, 306
Division, 122, 129
Doppelt-verkettete Liste, 344, 533
Drei-Schichten-Modell, 35
Dritte Normalform, 95, 235, 271, 292
DROP COLUMN, 201
DROP TABLE, 160, 188
Dynamische Programmierung, 319, 474, 478, 495, 504

Einbringen von Abhängigkeiten (Dependency Injection), 758, 860
Einfach-verkettete Liste, 343, 526, 536, 542, 544, 545
Einzelstück (Singleton), 759, 846
Entity-Relation-Modell, 40, 42, 44, 45, 47, 49–53, 55, 61, 63–65, 67, 69, 71–73, 75, 79
Entscheidbarkeit, 1208
Entwurfsmuster, 323, 758, 790, 846, 853, 877
Equi-Join, 28
Erbauer (Builder), 858
Erreichbarkeitsgraph, 882, 904, 906
Erweiteter Potenzmengenalgorithmus, 1069, 1071
Evolutionäre Softwaremodelle, 755, 777
EXCEPT, 29, 151, 193, 205
EXtreme Programming, 745, 751, 752, 762, 768

Fabrikmethode (Factory Method), 846
Feld (Array), 792
Formale Verifikation, 936, 987
Funktionale Abhängigkeiten, 254, 269, 287
Funktionale Anforderungen, 746, 752, 756, 771
Funktionalorienteres Testen, 779

Gantt-Diagramm, 879, 880, 884, 888, 891, 893, 897, 918
Generalisierung, 869
GOTO-berechenbar, 1201
Graphen, 696, 719, 725
Greedy-Algorithmus, 493, 499, 500
Grenzwertanalyse, 386
GROUP BY, 88, 96, 147, 148, 160, 172, 184, 193

Halde (Heap), 392, 565, 568, 581, 584, 600, 610, 626, 628, 640, 662
Hashfunktion, 689
HAVING, 88, 147, 151, 185, 193
Heapsort, 375, 393
Hoare-Kalkül, 987

Implementierung in Java, 315, 319, 323, 328, 343, 348, 391, 506, 518, 524, 529, 536, 587, 677, 791, 797, 807, 809, 818, 821, 827, 838, 870, 873
Inkrementelle Prozessmodelle, 765, 777
INSERT, 159, 172, 187
Insert-Anomalie, 287, 292
Insertionsort, 414
Interface, 809, 823
INTERSECT, 29
Invariante, 952, 962, 987
Iterative Prozessmodelle, 749, 765
Iterative Realisation, 315, 334

Kanonische Überdeckung, 276, 285, 289, 293
Kartesisches Produkt, 119
Kellerautomat, 1134, 1151–1153, 1156, 1158, 1160, 1163
Klassenadapter, 818
Klassendiagramm, 323, 523, 544, 677, 780, 782, 786, 788, 790, 791, 797, 803, 813, 816, 820, 823, 827, 835, 837, 861–863, 867–869, 873
Klassendiagramm zeichnen, 831
Kommando (Command), 759
Kommunikationsdiagramm, 868
Komplexitätstheorie, 1209, 1210, 1215
Komponentendiagramm, 759
Kompositum (Composite), 323, 544, 545, 790, 816, 834
Konfigurationsfolge, 1161
Kontextfreie Grammatik, 1116, 1131, 1148, 1149, 1161
Kontextfreie Sprache, 1017, 1045, 1092, 1093, 1096, 1097, 1101, 1105, 1108, 1111,

1112, 1114, 1116, 1118, 1120, 1124, 1127, 1132, 1135, 1149, 1160
Kontextsensitive Grammatik, 1171, 1172
Kontextsensitive Sprache, 1173
Kontinuierliche Integration (Continuous Integration), 752, 757, 762
Kontrollflussgraph, 372, 386, 932, 949, 957, 975, 996
Kontrollflussorientieres Testen, 949
Korrelierte Anfrage, 200

Lineare Suche, 354, 356, 365
Lineares Sondieren, 691
LOOP-berechenbar, 1203
Lost-Update, 306

main-Methode, 794
Master-Theorem, 438, 455, 458, 459, 464
MAX, 172
Mergesort, 375, 402, 420, 424, 426
Min-Heap, 568
Minimaler Spannbaum, 702, 731
Minimierungsalgorithmus, 1012, 1021, 1023, 1056, 1065, 1067
Model Checking, 751
Modell-Präsentation-Steuerung (Model-View-Controller), 759, 807

Natural-Join, 28
Nicht-funktionale Anforderungen, 746, 752, 756, 758, 771
Nichtdeterministisch endlicher Automat (NEA), 1005, 1064
Normalformen, 236, 239, 241, 243, 244, 251, 259, 262, 264, 267, 269
NOT NULL, 201

Objektadapter, 818
Objektdiagramm, 526, 786, 824, 837, 868
Objektorientierung, 536, 851
Offene Adressierung, 625
OR, 172

Partielle Korrektheit, 987
Peer-to-Peer-Architektur, 924
Petri-Netz, 882, 889, 904, 906, 908, 909
Pflichtenheft, 754
Physische Datenorganisation, 21
Polynomialzeitreduktion, 1210, 1212, 1214, 1217, 1220, 1222, 1223
Potenzmengenalgorithmus, 1002, 1005, 1019, 1020, 1026, 1038, 1073, 1074
Projektplanung, 760, 893
Prototyping, 746, 768
Prozessmodelle, 747, 762, 765
Pumping-Lemma (Kontextfreie Sprache), 1107, 1120, 1165–1167
Pumping-Lemma (Reguläre Sprache), 1025, 1045, 1075, 1076, 1078, 1080

Quadratisches Sondieren, 691
Quicksort, 377, 390, 395, 397, 403, 405, 409, 415, 424, 427

R-Baum, 627
Radixsort, 402
Refactoring, 751
REFERENCES, 201
Referentielle Integrität, 20
Reguläre Ausdrücke, 1009, 1019, 1057, 1081, 1084, 1087
Reguläre Grammatik, 1019, 1054, 1060, 1083
Reguläre Sprache, 999, 1001, 1004, 1007, 1009, 1013, 1015, 1017, 1018, 1025, 1029, 1032, 1035, 1038, 1040, 1042, 1046, 1050, 1053, 1054, 1056, 1059, 1062, 1083, 1092
Rekursion, 315, 319, 326, 328, 331–335, 347
Relationale Algebra, 55, 79, 82, 85, 88, 91, 94, 95, 102, 103, 106, 107, 110, 114, 122, 124, 128, 130, 137
Relationale Entwurfstheorie, 248
Relationenmodell, 41, 42, 45, 49, 53, 61, 113, 115, 117, 119, 120
RelaX - relational algebra calculator, 103, 114
Rot-Schwarz-Baum, 626

Schichtenarchitektur, 758
Schlüssel, 258
Schlüsselkandidat, 251, 268, 270, 279, 284
SCRUM, 745, 760, 763, 766, 768, 777
Selectionsort, 334, 380, 391, 406, 430
Separate Verkettung, 624, 690

Sequenzdiagramm, 759, 824, 852, 868
Serialisierbarkeitsgraph, 294
Software Engineering, 758
Softwarearchitektur, 920
Softwaremaße, 754, 762
Sortieralgorithmen, 377, 385, 402, 406, 414
Spezialisierung, 869
Spiralmodell, 758, 777
SQL, 47, 55, 79, 82, 88, 95, 99, 132, 141, 143, 145, 148, 150, 157, 164, 168, 170, 172, 174, 176, 182, 187, 189, 192, 196, 198, 203, 207, 209, 214, 220, 222, 225, 228, 232
SQL mit Übungsdatenbank, 100, 107, 133, 146, 165, 182, 190, 192, 199, 204, 207, 211, 215, 225, 229
Stapel (Stack), 443, 449, 513, 515, 518, 550–552
Streutabellen (Hashing), 566, 578, 586, 596, 608, 624, 646, 655, 659, 669, 689, 694, 695
Superschlüssel, 20, 288
Synthese-Algorithmus, 235, 240, 245, 250, 251, 255, 270, 271, 273, 276, 281

Teile-und-Herrsche (Divide-and-Conquer), 348, 473, 485, 488, 503
Terminierungsfunktion, 931, 956, 988
Testen, 751, 966, 967
Theoretische Informatik, 1040
Theta-Join, 28
Tiefensuche, 390, 513, 719, 735
Top-N-Query, 152, 190, 205, 208, 224, 226
Totale Korrektheit, 964, 987
Transaktionen, 294, 296, 300, 303, 304
Transaktionsverwaltung, 306
TRIGGER, 206
Tupel-Identifikator, 22
Tupelkalkül, 84, 94, 97, 106, 131
Turing-Maschine, 1174, 1175, 1177, 1179, 1182, 1184, 1188, 1191, 1192, 1194

UML-Diagramme, 785, 788, 820, 857, 868
Unbeschränkte Sprache, 1195
UNION, 29, 56
Unit-Test, 753, 768
UPDATE, 159, 173, 193
Update-Anomalie, 281, 287, 291

V-Modell, 745, 748, 768, 777, 779
Validation, 759
Vererbung, 797, 809, 823, 863, 869, 873
Verfeinertes Relationenmodell, 47, 116, 117
Verifikation, 759, 972
Versionsverwaltungssoftware, 756
VIEW, 56, 160, 177
Vollständige Anweisungsüberdeckung, 372
Vollständige Induktion, 319, 928, 942, 946, 947, 980, 983, 985

Warteschlange (Queue), 523, 529, 550
Wasserfallmodell, 747, 762, 768, 775, 777
White-Box-Testing, 751, 779, 969, 975
Wiederholer (Iterator), 816
WITH, 56, 152, 177
wp-Kalkül, 753, 936, 952, 974, 987, 989, 992

Zustand (State), 830, 832, 838
Zustandsdiagramm Wissen, 759, 838, 868
Zustandsdiagramm zeichnen, 786, 795, 808, 833, 844
Zwei-Phasen-Sperrprotokoll, 295
Zweite Normalform, 271, 279, 284, 291
Zyklomatische Komplexität nach Mc-Cabe, 422, 934, 960, 993

Äquivalenzklassen, 1058
Äquivalenzklassenzerlegung, 386
Überdeckbarkeit, 996