

Literaturverzeichnis

- [1] German stopwords. <http://solariz.de/download-7>, April 2010.
- [2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
- [3] Robert S. Boyer and Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10), Oktober 1977.
- [4] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2005.
- [5] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *7th Conference on Usenix Symposium on Operating Systems Design and Implementation*, 9, 2006.
- [6] Richard Cole and Ramesh Hariharan. Tighter upper bounds on the exact complexity of string matching. *SIAM J. Comput.*, 26(3):803–856, 1997.
- [7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI, Sixth Symposium on Operating System Design and Implementation*, pages 137–150, 2004.
- [8] M. L. Fredman, R. Sedgewick, D. D. Sleator, and R. E. Tarjan. The pairing heap: a new form of self-adjusting heap. *Algorithmica*, 1(1):111–129, 1986.
- [9] Michael Fredman and Robert Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.
- [10] C.A.R. Hoare. Quicksort. *Computer Journal*, 5(1):10–15, 1962.
- [11] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. New York: Plenum, 1972.
- [12] Donald E. Knuth. *The Art of Computer Programming. Vol. 3: Sorting and Searching*. Addison-Wesley, second edition, 1998.
- [13] The Lucene Webpages. lucene.apache.org.
- [14] Fredrik Lundh. Python hash algorithms. <http://effbot.org/zone/python-hash.htm>, 2002.

- [15] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. Interpreting the data: Parallel analysis with sawzall. *Scientific Programming*, 13(4):277–298, 2005.
- [16] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6), June 1990.
- [17] Gaston H. Gonnet Ricardo A. Baeza-Yates. A new approach to text searching. *Communications of the ACM*, 35(10):74–82, Oktober 1992.
- [18] Jean Vuillemin. A data structure for manipulating priority queues. *Communications of the ACM*, 21:309–314, 1978.
- [19] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin Beedle & Associates, Dezember 2003.

Index

- O*(*n*), 2
- Ord*(*n*), 135
- P*, 4
- P*-*NP*-Problem, 6
- Γ , 314
- Ω , 309
- Ω (*n*), 2
- β , 77
- ...**-Operation, 186
- NP*, 5
- ε , 185
- deg*, 314
- k*-Opt-Heuristik, 246
- Überladung, 269
- AVLTree._balance*, 61
- AVLTree._calcHeight*, 58
- AVLTree._doubleLeft*, 62
- AVLTree._simpleLeft*, 61
- AVLTree.insert*, 59
- BTree.deleteND*, 55
- BTree.insert*, 53
- BTree.search*, 52
- BloomFilter.insert*, 87
- Grammatik._addP*, 191
- Grammatik.automaton*, 205
- Grammatik.firstCalc*, 194
- Grammatik.followCalc*, 196
- Grammatik.goto*, 204
- Grammatik.huelle*, 204
- Grammatik.parse*, 211
- Grammatik.tabCalc*, 209
- Graph.E*, 151
- Graph.V*, 151
- Graph.addEdge*, 151
- Graph.isEdge*, 151
- Index.addFile*, 111
- Index.ask*, 111
- Index.crawl*, 111
- Index.toIndex*, 111
- KMP*, 219
- OurDict._insert*, 82
- OurDict._lookup*, 81
- OurDict._resize*, 84
- Patricia.insert*, 106
- Patricia.search*, 105
- RBTree._balance*, 66
- RBTree._insert*, 66
- RBTree.insert*, 66
- SkipList.search*, 94
- SkipList.delete*, 97
- SkipList.insert*, 95
- TSPBruteForce*, 238
- Trie.insert*, 103
- Trie.search*, 102
- UF.find*, 176
- UF.union*, 176
- VerschTab*, 221
- acoCycle*, 263
- adaptGlobal*, 265
- allCrossTours*, 254
- allCrosses*, 251
- ant*, 261, 265
- bfs* (Breitensuche), 153
- boyerMoore*, 228
- buildHeap* (binärer Heap), 42
- decKey* (Fibonacci-Heap), 137
- dfs* (Tiefensuche), 156
- dijkstra*, 163
- edgeCrossOver*, 256
- extractMinND* (Pairing-Heap), 144
- extractMin* (Fibonacci-Heap), 135
- fullAddB*, 124
- getMinFH* (Fibonacci-Heap), 131
- getMin* (Pairing-Heap), 143
- hashStrSimple*, 73
- hashStr*, 75
- heapSort*, 43
- horner*, 74

- if-Ausdruck, 274
- insND(l, key)*, 17
- insND*, 18
- insertionSortRek*, 18
- insertionSort*, 19
- insert* (Binärer Heap), 37
- insert* (Fibonacci-Heap), 132
- kruskal*, 172
- makedelta1*, 223
- match* (Stringmatching), 214
- maxFlow*, 180
- meltBinTree*, 123
- mergeSort*, 34
- merge* (Binomial-Heaps), 126
- merge* (Pairing-Heap), 144
- minExtractB* (Binomial-Heaps), 127
- minExtrakt* (Binärer Heap), 38
- minHeapify* (binärer Heap), 40
- nodeCrossOver*, 255
- pairmerge* (Pairing-Heap), 144
- partitionIP*, 27, 28
- quickSortIP*, 28
- quickSortIter*, 31
- quicksort*, 24
- rabinKarp*, 231
- rollhash*, 230
- shiftOr*, 234
- topSort* (Topologische Sortierung), 160
- tsp2Opt*, 247
- tsp2_5Opt*, 249
- tspGen*, 257
- tspMelt*, 244
- tspRandomInsertion*, 243
- tsp*, 239
- vapourize*, 262
- warshall*, 166
- s-t-Schnitt*, 183
- 2-Opt-Heuristik, 246
- 2.5-Opt-Heuristik, 248
- 3-KNF, 6
- 3SAT, 6
- Ableitung, 187
- Ableitungsschritt, 187
- ACO, 258
- ACO-Zyklus, 262
- Adelson-Welski, Georgi, 57
- adjazent, 149
- Adjazenzliste, 149
- Adjazenzmatrix, 149
- Agent, 259
- Aktionstabelle, 208
- All Pairs Shortest Paths, 162
- Alphabet, 185
- Ameisen-Algorithmen, 258
- Amortisationsanalyse, 220
- Amortisierte Laufzeit, 4
- anonyme Funktion, 290
- Ant Colony Optimization, 258
- anti-symmetrisch, 304
- Anweisung, 273
- Anweisung vs. Ausdruck, 273
- Ausdruck, 273
- Ausführungszeit, 83
- Average-Case-Laufzeit, 4
- AVL-Baum, 57
 - Implementierung
 - AVLTree._balance*, 61
 - AVLTree._calcHeight*, 58
 - AVLTree._doubleLeft*, 62
 - AVLTree._simpleLeft*, 61
 - AVLTree.insert*, 59
- Backtracking, 156, 257
- Bad-Character-Heuristik, 221
- Bad-Charakter-Heuristik
 - Implementierung
 - badChar*, 223
 - makedelta1*, 223
- balancierter Baum, 63
- Baum, 315
- Belegungsgrad β einer Hash-Tabelle, 77
- Bellmannsches Optimalitätsprinzip, 238
- benannter Parameter, 51, 275, 297
- Bernoulli-Verteilung, 313
- binäre Suche, 21
- binäre Und-Verknüpfung, 229
- Binärer Heap, 116
 - Einfügen eines Elements, 36
 - Höhe, 36
 - Implementierung
 - buildHeap*, 42
 - insert*, 37, 117
 - minExtract*, 118

- minExtrakt*, 38
- minHeapify*, 40, 118
- Repräsentation, 34, 116
- Binärer Suchbaum, 49
 - Implementierung
 - BTree.deleteND*, 55
 - BTree.insert*, 53
 - BTree.search*, 52
- Binomial-Heap, 119
 - Implementierung
 - fullAddB*, 124
 - meltBinTree*, 123
 - merge*, 126
 - minExtractB*, 127
 - Ordnung, 120
- Binomialkoeffizient, 239
- Binomialverteilung, 313
- Bit-Maske, 80
- Bloomfilter, 85
 - Implementierung, 87
 - BloomFilter.elem*, 87
 - BloomFilter.insert*, 87
 - Lösch-Funktion, 88
- BloomFilter.elem, 87
- Breitensuche
 - Implementierung
 - bfs*, 153
- British Library, 47
- Brute-Force, 237
- Buchstabe, 185
- Cache-Speicher, 112
- Carry-Bit, 123
- Cache, 92
- charakteristischer Vektor, 232
- Clique-Problem, 6
- Clustering (beim einfachen Hashing), 78
- Countingfilter, 88, 89
 - Lösch-Funktion, 89
- Crawler, 109
- Cross-Over zweier Lösungen (Kreuzung), 255
- dünn besetzt, 149
- DAG, 315
- Data Mining, 185
- Datenbank, 47
- Datenmengen (Vergleich), 47
- Datenstruktur
 - AVL-Baum, 57
 - Binärer Heap, 116
 - Binärer Suchbaum, 49
 - Binomial-Heap, 119
 - Bloomfilter, 85
 - Fibonacci-Heap, 127
 - Graph, 147, 149
 - Hashtabelle, 72
 - Heap, 115
 - Pairing-Heap, 142
 - Patricia, 100
 - Rot-Schwarz-Baum, 63
 - Skip-Listen, 93
 - Trie, 100
- Datentypen, 267
- Deterministischer endlicher Automat, 214
- Deterministischer endlicher Automat (DEA), 205
- Dichte einer Zufallsvariablen, 311
- Dictionary-Operationen, 72, 283
- Dijkstra, Edsger, 162
- Dijkstra-Algorithmus, 162
 - Implementierung
 - dijkstra*, 163
- disjunkt, 174
- Divide-And-Conquer, 22
- Doppelrotation, 59
- doppeltes Hashing, 78
- dynamic dispatch, 100
- dynamisch, 83
- dynamische Typisierung, 268
- einfaches Hashing, 78
- Einfachrotation, 59
- Einrücktiefe, 270
- Elementarereignis, 309
- Emergenz, 259
- Endrekursion, 31
- Entscheidungsbaum, 21
- Ereignis, 310
- Erfüllbarkeitsproblem, 6
- Erfolgswahrscheinlichkeit, 313
- Erwartungswert, 312
- Erweiterungspfad, 180

- erzeugte Sprache, 187
- Evolution, 255
- Fakultätsfunktion, 7
- falsch-positiv, 86
- Farthest-Insertion-Heuristik, 242
- Fibonacci, 307
- Fibonacci-Baum, 128
 - Ordnung, 128
- Fibonacci-Heap, 127
 - Implementierung
 - decKey*, 137
 - extractMin*, 135
 - getMinFH*, 131
 - insert*, 132
- FIFO-Datenstruktur, 152
- first-in, first-out, 152
- Fluss in einem Netzwerk, 178
- Flusserhaltung, 179
- Ford-Fulkerson Algorithmus, 180
- Ford-Fulkerson-Algorithmus, 179
 - Implementierung
 - maxFlow*, 180
- funktionale Programmierung, 287
- Ganzzahlen (*int* in Python), 267
- Gegenwahrscheinlichkeit, 89
- Generation, 255
- Genetischer Algorithmus, 255
- Genpool, 255
- Geometrische Verteilung, 313
- Gesetz der Flusserhaltung, 179
- getrennte Verkettung, 77
- Gewichtsfunktion *w*, 161
- Gleitpunktzahlen (*float* in Python), 267
- globales Optimum, 246
- Goldener Schnitt, 308
- Good-Suffix-Heuristik, 221, 224
- Google, 47
- Grad eines Knotens, 314
- Grammatik
 - Implementierung
 - Grammatik.addP*, 191
 - Grammatik.automaton*, 205
 - Grammatik.firstCalc*, 194
 - Grammatik.followCalc*, 196
 - Grammatik.huelle*, 204
 - Grammatik.parse*, 211
 - Grammatik.tabCalc*, 209
- Graph, 147
 - Implementierung
 - Graph.E*, 151
 - Graph.V*, 151
 - Graph.addEdge*, 151
 - Graph.isEdge*, 151
 - Pfad in . . . , 314
 - Repräsentation, 149
 - Weg in . . . , 315
 - Zusammenhang, 316
 - Zusammenhangskomponente, 316
 - Zyklus in . . . , 315
- Greedy-Algorithmus, 162
- Greedy-Heuristiken, 241
- Groß-Oh-Notation, 1
- Höhe eines Knotens (in einer Skip-Liste), 93
- Halteproblem, 83
- Handlungsreisender, 237
- Hash-Funktion, 72
- Hash-Tabelle, 72
- Hashing, 72
 - doppeltes Hashing, 78
 - einfaches Hashing, 78
 - getrennte Verkettung, 77
 - Kollisionsbehandlung, 77
- Haskell (Programmiersprache), 198
- Heap, 34, 115
- Heap Sort
 - Implementierung
 - heapSort*, 43
- Heap-Eigenschaft, 34
- Heapsort, 34
- Heuristik, 241
- Heuristiken
 - k*-Opt-Heuristik, 246
 - 2-Opt-Heuristik, 246
 - 2.5-Opt-Heuristik, 248
 - Farthest-Insertion-Heuristik, 242
 - Greedy, 241
 - Kanten-Cross-Over, 255
 - Knoten-Cross-Over, 255
 - lokale Verbesserung, 246
 - Nearest-Insertion-Heuristik, 242

- Nearest-Neighbor-Heuristik, 241
- Random-Insertion-Heuristik, 242
- Tourverschmelzung, 244
- Hintereinanderausführung, 274
- Hoare, C.A.R., 27
- Horner-Schema, 74, 230, 294
 - Implementierung
 - horner2*, 75
 - horner*, 75
- horner2*, 75
- IDLE, 267
- imperative Programmierung, 287
- Implementierung
 - destruktiv, 13
 - in-place, 13
 - nicht-destruktiv, 13
 - rekursiv, 7
- Implementierungen
 - minExtract*, 38
- in-place, 19
- Index, 109
- Indexer, 109
- Induktionsanfang, 306
- Induktionshypothese, 306
- Induktionsschritt, 306
- Information Retrieval, 47, 108
- Insertion Sort, 17
 - Implementierung
 - insND*, 18
 - insertionSortRek*, 18
 - insertionSort*, 19
 - in-Place, 19
 - Laufzeit, 19
 - nicht-destruktiv, 17
- Interpreter, 267
- invertierter Index, 109
- Iteration vs. Rekursion, 7
- kürzeste Wege, 162
- Kanten, 147, 314
- Kanten eines Schnittes, 183
- Kanten-Cross-Over, 255
- kantenbewerteter Graph, 161
- Kapazität, 178
- Kapazität eines Schnittes, 183
- Kirchhoff'sches Gesetz, 179
- Klasse, 298
- Klassen
 - instanzen, 299
 - methoden, 298
 - __ini__*-Methode, 299
- Klassenattribut, 300
- Klassendefinitionen
 - AVLTree*, 58
 - BTree*, 50
 - BloomFilter*, 87
 - Grammatik*, 191
 - Graph*, 150
 - Index*, 111
 - OurDict*, 79
 - Patricia*, 104
 - RBTree*, 63
 - SLEntry*, 94
 - SkipList*, 94
 - Trie*, 102, 106
 - UF* (Union-Find), 176
 - string*, 76
- Knoten, 147, 314
- Knoten-Cross-Over, 255
- Knuth-Morris-Pratt-Algorithmus, 216
 - Implementierung
 - KMP*, 219
 - VerschTab*, 221
- Kollisionsbehandlung, 77
- Komplexitätsklasse, 4
- Komponente, 316
- Konjunktive Normalform, 6
- Konkatenation, 269
- kostengünstigster Verbindungsgraph, 169
- Kreis, 315
- Kreiseigenschaft, 171
- Kreuzprodukt, 303
- Kreuzung von Lösungen (Cross-Over), 255
- Kruskal-Algorithmus
 - Implementierung, 172
 - kruskal*, 172
 - Korrektheit, 170
- Länge eines Pfades, 314
- Länge eines Weges, 315
- Lambda-Ausdruck, 290
- Landau-Symbole, 1

- Landis, Jewgeni, 57
- lange Ganzzahlen (*long int* in Python), 267
- last-in, first-out, 154
- leere Menge, 303
- leeres Wort ϵ , 185
- Leonardo da Pisa, 307
- lexikalische Suche, 108
- LIFO-Datenstruktur, 154
- Linksrekursion, 201
- Listenkomprehension, 288
- lokale Verbesserungsstrategien, 246
- lokales Optimum, 246

- Maske, 80
- mathematische Tupel, 303
- Matrix
 - dünn besetzt, 149
- Max-Flow-Min-Cut-Theorem, 182
- Max-Heap, 34
- Max-Heap-Eigenschaft, 34
- Maximaler Fluss, 178
- Mehrdeutigkeit einer Grammatik, 188
- Membership-Test, 85
- Menge, 303
- Mengenkomprehension, 288, 303
- Mergesort, 33
- merging, 33
- Metasymbol (Nichtterminal), 186
- Methode, 298
- Min-Heap, 34
- Min-Heap-Eigenschaft, 34
- minimaler Schnitt, 183
- minimaler Spannbaum, 169
- Minimumsextraktion, 38
- Mutation, 246

- Nachbarschaft eines Knotens, 314
- Navigationssystem, 162
- NEA (=Nichtdeterministischer endlicher Automat), 232
- Nearest-Insertion-Heuristik, 242
- Nearest-Neighbor-Heuristik, 241
- Netzwerk, 178
 - Kapazität, 178
- NFA (nichtdeterministischer endlicher Automat, 214
- Nicht-Determinismus, 5
- nichtdeterministische Rechenmaschine, 5
- nichtdeterministischer endlicher Automat, 232
- Nichtdeterministischer endlicher Auuto-mat, 214
- Nichtterminal(-symbol), 186
- NoSQL, 91

- Objekt, 299
- Objektattribut, 300
- objektorientierte Programmierung, 298
- Optimalitätsprinzip, 238
- Ordnung
 - Binomial-Heap, 120
 - Fibonacci-Baum, 128

- Pairing-Heap, 142
 - Implementierung
 - extractMinND*, 144
 - getMin*, 143
 - merge*, 144
 - pairmerge*, 144
- Parsergenerator, 185, 197
- Parsing, 185
- Patricia, 100
 - Implementierung
 - Patricia.insert*, 106
 - Patricia.search*, 105
- perfekte Zahl, 272
- Permutation, 238
- Persistenz, 114
- Pfad, 314
 - Länge, 314
- Pfadkomprimierung, 177
- Pheromon, 259
- Pheromonspur, 259
- Pivot-Element, 23
- Polymorphie, 269
- polynomieller Algorithmus, 4
- Potential-Funktion, 131
- Potentialmethode (zur amortisierten Lauf-zeitanalyse), 4
- prädiktive Grammatik, 198
- prädiktives Parsen, 198
- Präfix, 216
- praktisch lösbarer Algorithmus, 4

- Prioritätswarteschlange, 35, 115
- Priority Search Queue, 35
- Problem des Handlungsreisenden, 237
- Problemgröße, 2
- Produktion, 187
- Programmstack, 8
- Proxy, 92
- Python-Referenzen, 276
- Pythondatentypen
 - complex*, 267
 - dict*, 283
 - float*, 267
 - int*, 267
 - list*, 277
 - long*, 267
 - str*, 268, 285
 - tuple*, 282
- Pythonfunktionen
 - all*, 292
 - any*, 292
 - del**, 280
 - dict.items()*, 284
 - dict.keys()*, 284
 - dict.values()*, 284
 - dir*, 278
 - enumerate*, 292
 - len*, 280
 - list.count*, 277
 - map*, 291
 - max*, 280
 - min*, 280
 - range*, 272
 - reduce*, 293
 - str.capitalize*, 285
 - str.endswith*, 285
 - str.find*, 285
 - str.join*, 285
 - str.lower*, 285
 - str.partition*, 285
 - str.replace*, 285
 - str.split*, 285
 - str.startswith*, 285
 - str.upper*, 285
 - sum*, 280
- Pythonkommandos
 - break**, 272
 - continue**, 272
 - def**, 274
 - elif**, 270
 - for**, 270
 - if**-Ausdruck, 274
 - list.append*, 277
 - list.insert*, 277
 - list.remove*, 277
 - list.reverse*, 277
 - list.sort*, 277
 - return**, 275
 - while**, 270
- Pythonmethoden
 - __cmp__*, 301
 - __getitem__*, 301
 - __ini__*, 301
 - __len__*, 301
 - __setitem__*, 301
 - __str__*, 301
- Pythonmodule
 - heapq*, 44
 - marshal*, 114
 - pickle*, 114
 - pygeodb*, 242
 - random*, 30
 - shelve*, 114
 - time*, 30
- Pythonoperatoren, 269
 - ***, 270
 - +**, 270
 - , 270
 - /**, 270
 - <<**, 270
 - <**, 270
 - ==**, 270
 - >>**, 270
 - >**, 270
 - %**, 270
 - &**, 270
 - ^**, 270
 - ~**, 270
 - and**, 270
 - in**, 270
 - is**, 270
 - not**, 270
 - or**, 270
- Pythonshell, 267
- Pythonvariablen

- lokale, 275
- Quelle (Netzwerk), 178
- Queue, 152
 - dequeue*-Operation, 152
 - enqueue*-Operation, 152
- Quicksort, 22
 - Implementierung
 - mergeSort*, 34
 - partitionIP*, 28
 - quicksortIP*, 28
 - quicksortIter*, 31
 - quicksort*, 24
 - in-Place, 27
 - Randomisiert, 29
- Random-Insertion-Heuristik, 242
- randomisierte Datenstruktur, 93
- Read-Eval-Print-Loop (REPL), 267
- Rechtsableitung, 202
- reflexiv, 304
- Rekursion, 6
 - ‘Kochrezept’, 12
- Rekursionsabbruch, 10
- Rekursionsschritt, 12
- rekursive Funktion, 6
- rekursiver Abstieg, 8
- rekursiver Aufstieg, 8
- Relation, 304
- REPL (Read-Eval-Print-Loop), 267
- Repräsentation von Datenstrukturen, 14
 - Repräsentation als Dictionary, 15
 - Repräsentation als Klasse, 15
 - Repräsentation als Liste, 15
- Restnetzwerk, 180
- Retrieval, 47
- Rollender Hash, 229
- Rot-Schwarz-Baum, 63
 - Einfügen eines Knotens, 64
 - Implementierung
 - RBTree.balance*, 66
 - RBTree.insert*, 66
 - Löschen eines Knotens, 69
- Rotation, 59
- Routenplanung, 162
- Routing-Tabelle, 100
- Rucksack-Problem, 6
- Satzform, 188
- Schlüssel, 49
- Schleife
 - Python:**for**, 270
 - Python:**while**, 270
- Schleifenabbruch, 272
- Schleifeninvariante, 43
- Schleifenkopf, 273
- Schnitt
 - s-t*-Schnitt, 183
 - minimaler Schnitt, 183
 - Kanten eines Schnittes, 183
 - Kapazität eines Schnittes, 183
- Schnitt in einem Graphen, 182
- Schnitteigenschaft, 172
- schwach zusammenhängend, 316
- Schwarm-Intelligenz, 259
- semantische Suche, 108
- Semaphore, 162
- Senke, 178
- Sequenzoperationen (in Python), 280
- Shift-Or-Algorithmus, 232
- Sierpinski-Dreieck, 12
- Skelettautomat, 215
- Skip-Liste, 93
 - Höhe, 93
 - Höhe eines Knotens, 93
 - Implementierung
 - SkipList.delete*, 97
 - SkipList.insert*, 95
 - SkipList.search*, 95
 - Vorwärtszeiger, 93
- Slicing (in Python), 279
- Sortieren, 17
- Spannbaum, 169, 316
- Sprache, 186
- Springerproblem, 157
- Sprungtabelle, 208
- Stack, 8
 - pop*-Operation, 154
 - push*-Operation, 154
- Stack Overflow, 8
- Stackframe, 30
- Stapelspeicher, 154
- stark zusammenhängend, 316
- Startsymbol (einer Grammatik), 186
- statisch, 83

- statische Typisierung, 268
- Stemming, 109
- Stoppwort, 114
- Stringmatching, 213
- Strings in Python, 268, 285
 - """...""", 268
 - "... ", 268
 - '... ', 268
 - '''...''', 268
- Suchmaschine, 108
 - Aufbau, 108
 - Implementierung, 108
- symmetrisch, 304
- Syntaxanalyse, 185
- Syntaxanalysetabelle, 208
- Syntaxbaum, 188
- Syntaxbeschreibungsfomalismen, 270
 - [...], 270
 - [...]*, 270
- tail recursion, 31
- Terminal(-symbol), 186
- Tiefensuche, 154
 - Implementierung
 - dfs*, 156
- Top-Down-Parser, 197
- Topologische Sortierung, 159
 - Implementierung
 - topSort*, 160
- Tourverschmelzung, 244
- transitiv, 304
- transitive Hülle, 167, 305
- Travelling-Salesman-Problem, 6, 237
- Trie, 100
 - Implementierung
 - Trie.insert*, 103
 - Trie.search*, 102
- Tupel (in der Mathematik), 303
- Tupel (in Python), 277, 282
- Turingmaschine, 5
 - nicht-deterministisch, 5
- Typ-2-Grammatik, 186
- Typ-2-Sprache, 185, 188
- Typ-3-Sprache, 185
- Typisierung
 - dynamisch, 268
 - statisch, 268
- Unabhängigkeit von Zufallseignissen, 310
- Und-Verknüpfung, 229
- Union-Find-Operationen, 174
 - Balancierung, 176
 - Implementierung
 - UF.find*, 176, 177
 - UF.union*, 176
 - Pfadkomprimierung, 177
- Usability, 112
- Variable (Nichtterminal), 186
- Vereinigungs-Suche, 174
- Vererbung, 300
- Verschiebetabelle, 217
- Verteilung einer Zufallsvariablen, 311
- Vollständige Induktion, 306
- Vorwärtszeiger einer Skip-Liste, 93
- Wahrscheinlichkeitsraum, 309
- Warshall-Algorithmus, 165
 - Implementierung
 - warshall*, 166
- Wartbarkeit von Programmen, 273
- Warteschlange, 152
- Web-Cache, 92
- Web-Proxy, 92
- Weg, 315
 - Länge, 315
- Worst-Case-Laufzeit, 4
- Wort, 186
- Wurzelbaum, 315
- Yacc, 185, 197
- Zufallsvariable, 311
- zusammengesetzte Datentypen, 277
- zusammenhängender Graph, 316
- Zusammenhangskomponente, 316
- Zyklus, 315