

# Klausur

## Algorithmen und Datenstrukturen

### Sommersemester 2021

### 7. September 2021

Ihre Matrikelnummer:

**Achtung:** Geben Sie keine weiteren persönlichen Daten außer der Matrikelnummer an!

#### Generelle Hinweise:

- Schreiben Sie **lesbar** (lesbar für die *Korrektoren*, nicht nur lesbar für *Sie*!).
- Schreiben Sie zu Ihrer Sicherheit die Matrikelnummer auf jedem Blatt in das jeweils dafür vorgesehene Feld.
- Geben Sie in Ihrer Lösung zu jeder Aufgabe an, welcher Text zu welcher Teilaufgabe gehört (wenn Sie Ihre Lösungen zu den Teilaufgaben strikt nacheinander, also ohne Hin- und Hersprünge zwischen Teilaufgaben verfassen, reicht es natürlich, die jeweilige Teilaufgabe jeweils zu Beginn anzugeben).
- Sie können selbstverständlich auch Zwischen- und Nebenergebnisse in Ihre Abgabe schreiben, um leichter zum Ergebnis zu kommen. Stellen Sie dann aber die eigentliche Lösung klar heraus.

#### Überblick:

Aufgabe	1	2	3	4	5	6	Summe
Max.	15	19	17	10	27	17	105

## Aufgabe 1(a): 5 Punkte

### Lösen Sie die folgende GENEX-Aufgabe:

Aus der Vorlesung kennen Sie den Algorithmus *String Matching basierend auf endlichem Automaten*.

Gegeben sei der Stringmatcher  $T = sdv$  und der String  $S = sdsdv s$ .

Diese wurden generiert über dem *Alphabet* =  $(z, a, s, w, v, e, n, t, d, p)$

Daraus resultiert die folgende Tabelle die den endlichen Automaten beschreibt:

State	z	a	s	w	v	e	n	t	d	p
0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	2	0
2	0	0	1	0	3	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0

In der Tabelle findet sich für jeden State der Nachfolgestate abhängig vom nächsten in  $S$  gesehenen Buchstaben.

Die Variable  $q$  beschreibt die Länge des zur Zeit längsten gültigen Kandidaten. Geben Sie  $q$  als einzelne natürliche Zahl an.

Die Sequenz  $R$  beinhaltet die Startindizes aller vollständigen Matches. Geben Sie  $R$  folgendermaßen an:  $n, m, \dots$

Geben Sie die Zahl  $q$  und die geordnete Sequenz  $R$  des Algorithmus nach dem Induktionsschritt  $i = 2$  an.


**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

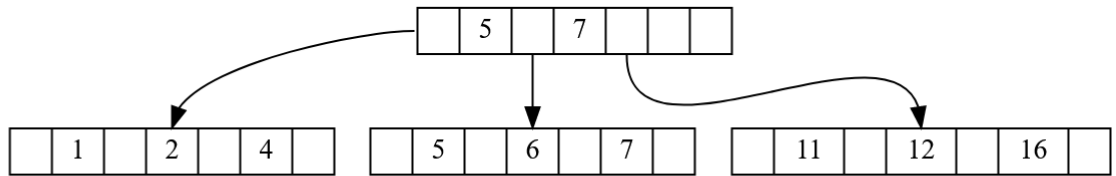
## Aufgabe 1(b): 5 Punkte

### Lösen Sie die folgende GENEX-Aufgabe:

Aus der Vorlesung kennen Sie den Algorithmus *B-Tree: Insert*.

Gegeben sei nun folgender B-Baum der Ordnung  $M = 2$ :

 Zoom



Geben Sie den resultierenden B-Baum nach dem Einfügen des Schlüssels 13 an.

**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

## Lösen Sie die folgende GENEX-Aufgabe:

Kruskal - Nabla

<https://nabla.algo.informatik.tu-darmstadt.de/catalog/2/genex/kruskal/pa...>

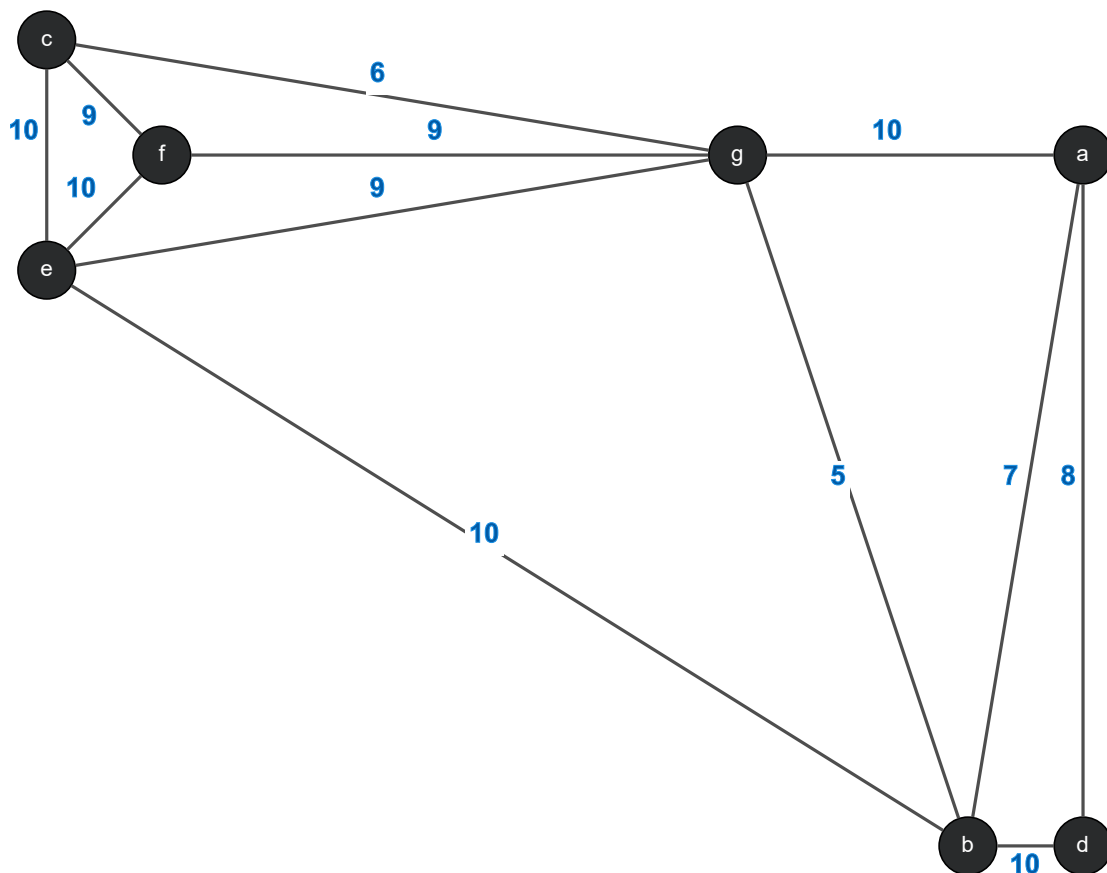
[Aufgabenkatalog](#) / [Graphen](#) / Kruskal

## Kruskal

### Aufgabe

Aus der Vorlesung kennen Sie den Algorithmus *Kruskal*.

Gegeben sei der folgende Graph:



Es seien die folgenden sortierten Kanten gegeben:

$S := (e, c), (f, e), (d, b), (g, a), (b, e), (f, c), (g, f), (g, e), (d, a), (a, b), (g, c), (g, b)$

Geben Sie die folgenden Informationen zum Induktionsschritt  $i = 6$  an:

1. Die Menge von Mengen  $UF$
2. Die Menge von Kanten  $E$

[← Zurück](#)

**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

## Aufgabe 2: 19 Punkte

Aus Vorlesung und Übungen kennen Sie folgende Klasse:

```
public class ListItem <T> {  
    public T key;  
    public ListItem<T> next;  
}
```

Schreiben Sie alles, was in folgender Methodendefinition ausgelassen und durch ... angedeutet ist, also die Anweisungen in:

```
public <T> ListItem<ListItem<T>> makeListOfLimitedListsFromArray  
    ( T[] a, Comparator<T> cmp, Adder<T> adder, T limit ) { ... }
```

Methode `makeListOfLimitedListsFromArray` soll eine korrekt gebildete Liste von Listen zurückliefern (*Terminologie*: die Elemente der *Hauptliste* sind die *Einzellisten*). Jede Komponente von `a` kommt exakt einmal vor (einfach durch „=" aus `a` kopiert). Die Einzellisten sind maximal lang, so dass die Summe in jeder Einzelliste höchstens `limit` ist.

Die Methode darf ohne Nachprüfung von `a != null` und von `a[i] ≤ limit` für alle  $i \in \{0, \dots, a.length - 1\}$  ausgehen, aber `a.length == 0` ist möglich. Sie soll `head` nach `a.length-1` vielen Schleifendurchläufen zurückliefern.

**Verbindliche Schleifeninvariante:** Nach  $h \geq 0$  Iterationen gilt:

1. `head` verweist auf eine korrekte gebildete Liste von Listen ( $L_h$  bezeichne die Länge der Hauptliste).
2. Für  $k \in \{0, \dots, L_h - 1\}$  bezeichnen `a[ $\ell_k$ ]`, ..., `a[ $r_k$ ]` im Folgenden die Elemente der Einzelliste, die an Position  $k$  der Hauptliste ist.
  - (a) Die Intervalle  $[\ell_k \dots r_k]$  partitionieren  $[0 \dots h]$ . Das heißt, es ist  $\ell_k = 0$ ,  $r_{L_h-1} = h$  und  $\ell_{k+1} = r_k + 1$  für alle  $k \in \{0, \dots, L_h - 2\}$ ;
  - (b) gemäß `Adder` und `Comparator`: `a[ $\ell_k$ ] + ... + a[ $r_k$ ] ≤ limit` für alle  $k \in \{0, \dots, L_h - 1\}$  und `a[ $\ell_k$ ] + ... + a[ $\ell_{k+1}$ ] > limit` für alle  $k \in \{0, \dots, L_h - 2\}$ .

**Verbindliche Anforderung:** Die asymptotische Komplexität im Worst Case ist linear in der Länge des Arrays `a`.

**Hinweise:** `Adder<T>` hat eine Methode `add` mit zwei Parametern vom Typ `T` und Rückgabetyt `T`. Methode `compare` von `Comparator<T>` liefert bekanntlich genau dann einen Wert kleiner 0, wenn der erste Parameter kleiner als der zweite ist.

**Ihre Lösung schreiben Sie auf die folgenden drei Seiten:**



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

### Aufgabe 3: 17 Punkte

Folgende Klasse sei gegeben:

```
public class BinaryTreeNode <T> {  
    public T key;  
    public BinaryTreeNode<T> left;  
    public BinaryTreeNode<T> right;  
}
```

Schreiben Sie alles, was in folgender Methodendefinition ausgelassen und durch ... angedeutet ist, also die Anweisungen in:

```
public <T> BinaryTreeNode<T> removeIf  
    ( BinaryTreeNode<T> root, Predicate<T> pred ) { ... }
```

**Erinnerung:** Methode `test` von `Predicate<T>` ist boolesch und hat einen Parameter vom Typ `T`.

**Konkrete Aufgabe:** Methode `removeIf` darf ohne Nachprüfung davon ausgehen, dass `root` auf die Wurzel eines korrekt gebildeten binären Baumes verweist (der auch leer sein kann, d.h. `root==null` ist möglich). Sie entfernt genau die `key`-Werte aus dem Baum, bei denen Methode `test` von `pred` den Wert `true` zurückliefert, und entfernt auch genauso viele Knoten (aber nicht unbedingt die, in denen diese `key`-Werte sind), so dass die Rückgabe ein Verweis auf die Wurzel eines korrekt gebildeten binären Baumes auf den verbliebenen `key`-Werten ist.

#### **Verbindliche Anforderungen:**

- Methode `removeIf` ist rein rekursiv, das heißt, Schleifen sind weder in `removeIf` noch in davon direkt oder indirekt aufgerufenen Methoden erlaubt.
- Es werden keine neuen Objekte mit Operator `new` oder anders erzeugt.
- Die Reihenfolge der `key`-Werte wird beibehalten: Falls der Baum unmittelbar vor dem Aufruf von `removeIf` Suchbaumeigenschaft gemäß irgendeiner Vergleichsoperation auf `T` hat, dann auch unmittelbar nach diesem Aufruf.

**Unverbindlicher Hinweis:** Wenn der `key`-Wert in einem Knoten zu entfernen ist, machen Sie eine Fallunterscheidung: beide Nachfolger nichtexistent, nur linker existent, nur rechter existent oder beide existent. Für den vierten Fall schreiben Sie eine – selbstverständlich rein rekursive – Hilfsmethode `removeRightmostDescendantAndReturnItsKey` (Sie dürfen auch einen anderen Namen wählen).

**Ihre Lösung schreiben Sie auf die folgenden drei Seiten:**

**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

## Aufgabe 4: 10 Punkte

Folgende Klassen seien gegeben:

```
public class Arc <T> {
    public Node<T> head;
}

public class Node <T> {
    public List<Arc<T>> outgoingArcs;
    public Collection<Node<T>> reachedFrom;
}
```

Die Elemente von `node.outgoingArcs` sind die aus einem Knoten `node` herauszeigenden Kanten, und `arc.head` ist der Zielknoten einer Kante `arc`.

Schreiben Sie alles, was in folgender Methodendefinition ausgelassen und durch ... angedeutet ist, also die Anweisungen in:

```
public <T> void setReachedFrom ( <Node<T>> currNodes ) { ... }
```

**Konkrete Aufgabe:** Die Methode `setReachedFrom` darf ohne Nachprüfung davon ausgehen, dass `currNodes` nicht leer ist und einen oder mehrere Knoten eines korrekt gebildeten gerichteten Graphen enthält. Vorausgesetzt werden kann ebenfalls, dass `node.reachedFrom` unmittelbar vor dem Aufruf von `setReachedFrom` für alle Knoten `node` leer ist. Für jeden Knoten `node` in diesem Graphen, der von mindestens einem der Knoten in `currNodes` aus erreicht werden kann, und für jede Kante `arc` in `node.outgoingArcs` soll `node` hinterher in `arc.head.reachedFrom` enthalten sein.

**Verbindliche Anforderung:** Bei jedem Knoten sollen die Kanten in `outgoingArcs` maximal einmal durchlaufen werden (also am Besten wenn zum ersten Mal erreicht).

**Unverbindlicher Hinweis:** Um sicherzustellen, dass Sie wirklich alle erreichbaren Knoten prozessiert haben, verwenden Sie `currNodes` in `setReachedFrom` einfach weiter, und zwar als Zwischenspeicher für genau die Knoten, die Sie schon erreicht, deren herauszeigende Kanten Sie aber noch nicht weiterverfolgt haben.

**Erinnerung:** Methode `add` von `Collection` hat einen Parameter vom formalen Typ `T` und kann als `void`-Methode verwendet werden. Methode `remove` entfernt das Element, dessen Index der aktuelle Parameter ist, und liefert dieses Element zurück. Sie können die verkürzte `for`-Schleife, die Sie u.a. von Arrays kennen, auch bei Referenzen von `List` anwenden; alternativ können Sie sich von der Liste einen `Iterator<T>` mit der parameterlosen Methode `iterator` holen und diese Liste mit den parameterlosen Methoden `hasNext` (boolesch) und `next` (Rückgabetyp `T`) durchlaufen.

**Ihre Lösung schreiben Sie auf die folgenden drei Seiten:**



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

<b>Aufgabe 5(a): 13 Punkte</b>
--------------------------------

Formulieren Sie das Sortierproblem, das durch Bucketsort gelöst wird, nach dem Schema (i) was ist eine zulässige Eingabe, (ii) was ist eine zulässige Ausgabe. Verwenden und erläutern Sie bei (ii) den Begriff „lexikographisch“.

(iii) Formulieren Sie für Bucketsort die Schleifeninvariante.

(iv) Wodurch wird bei Bucketsort gewährleistet, dass Präfixe korrekt einsortiert werden?

**Ihre Lösung schreiben Sie auf diese und die nächste Seite:**

**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

**Aufgabe 5(b): 14 Punkte**

Formulieren Sie die Problemstellung *Minimal spannender Baum* (*MST*) nach dem Schema (i) was ist eine zulässige Eingabe, (ii) was ist eine zulässige Ausgabe, (iii) was ist das Optimierungsziel. Verwenden und erläutern Sie bei (ii) auch den Begriff „spannender Baum“.

(iv) Formulieren Sie die Schleifeninvariante für den Algorithmus von Prim. Gehen Sie dabei auch auf Hilfsdatenstrukturen in maximaler Allgemeinheit ein (also abstrakte Datenstrukturen, nicht deren konkrete Implementationen).

(vi) Wie müssen alle Datenstrukturen also initialisiert werden, damit die Invariante vor dem ersten Schleifendurchlauf erfüllt ist?

**Ihre Lösung schreiben Sie auf diese und die nächste Seite:**

**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

### Aufgabe 6(a): 6 Punkte

Gegeben sei eine Subroutine (in Java: Methode), die bei einer Liste der Länge  $n$  prüft, ob sie ein Palindrom ist, das heißt, das erste Element ist gleich dem letzten, das zweite gleich dem vorletzten und so weiter. Sei  $m$  die Länge der Liste. Wir unterscheiden im Folgenden:

I: die Liste ist wie üblich einfach verkettet.

II: die Liste ist doppelt verkettet, das heißt, neben dem Verweis **head** auf das erste Element gibt es auch einen Verweis **tail** auf das letzte Element als zweiten Einstiegspunkt, und neben dem Verweis **next** von jedem Listenelement auf das jeweils nächste (bzw. **null** beim letzten Listenelement) gibt es auch einen Verweis **back** von jedem Listenelement auf das jeweils vorhergehende (bzw. **null** beim ersten Listenelement).

**Konkrete Aufgaben:** Die Subroutine sei möglichst effizient implementiert und benutze nur konstant viel zusätzlichen Speicher.

(i) Wie sieht der Best Case bei I und II aus, (ii) wie sieht der Worst Case bei I und II aus?

(iii) Was ist die asymptotische Komplexität im Best Case bei I und II, (iv) was ist die asymptotische Komplexität im Worst Case bei I und II? Verwenden Sie keine O-Notation, sondern Begriffe wie konstant, logarithmisch, linear, quadratisch usw.

**Ihre Lösungen schreiben Sie auf diese und die nächste Seite:**



**Ihre Matrikelnummer (zu Ihrer Sicherheit):**\_\_\_\_\_

**Aufgabe 6(b): 7 Punkte**

Betrachten Sie einen generischen Algorithmus  $A$ , der ein Array  $a1$  der Länge  $n1$  mit generischem Komponententyp  $T1$  und ein Array  $a2$  der Länge  $n2$  mit generischem Komponententyp  $T2$  verarbeitet. Die asymptotische Komplexität von  $A$  sei dominiert durch  $n2$  viele Sortierprobleme auf  $a1$  plus  $n1$  viele Sortierprobleme auf  $a2$ . Aufgrund der Generizität ist nur Sortieren durch paarweisen Vergleich möglich. Die Worst-Case-Komplexität des für  $T1$  verwendeten Vergleichs sei  $C1$ , die des für  $T2$  verwendeten Vergleichs sei  $C2$ . Welche asymptotische Komplexität  $\Theta(\dots)$  hat  $A$  im Worst Case, wenn ein Sortieralgorithmus verwendet wird, dessen asymptotische Komplexität im Worst Case bestmöglich ist?

**Konkrete Aufgabe:** Formulieren Sie eine Formel für „...“ in  $\Theta(\dots)$ . (Verwenden Sie auf der Tastatur  $*$  für Multiplikation und  $\log(x)$  für den Logarithmus von  $x$ .)

Ihre Matrikelnummer (zu Ihrer Sicherheit):\_\_\_\_\_

**Aufgabe 6(c): 4 Punkte**

Begründen Sie mathematisch: Die Funktion  $f_1 : n \rightarrow n \log_2^3 n$  wächst asymptotisch echt schneller als die Funktion  $f_2 : n \rightarrow n \log_2^2 n$ . Sie können in Ihrer Antwort die Basis des Logarithmus auslassen und einfach „log(n)“ schreiben.

***Unverbindlicher Hinweis:*** Verwenden Sie ***nicht*** die Regel von L'Hopital, wie Sie es in Altklausuren gesehen haben, sondern argumentieren Sie direkter, auf Basis der Eigenschaften der Logarithmus-Funktionen.

**Diese Seite ist nur von den Korrektoren auszufüllen!**

<b>1</b>	(a)	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX																				
	(b)	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX																				
	(c)	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX																				
<b>2</b>		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S		
<b>3</b>		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q				
<b>4</b>		A	B	C	D	E	F	G	H	I	J											
<b>5</b>	(a)	A	B	C	D	E	F	G	H	I	J	K	L	M								
	(b)	A	B	C	D	E	F	G	H	I	J	K	L	M	N							
<b>6</b>	(a)	A	B	C	D	E	F															
	(b)	A	B	C	D	E	F	G														
	(c)	A	B	C	D																	

Letzte Spalte: Summe Punkte Aufgabe

Unterstes Feld: Gesamtpunktzahl aus allen Aufgaben

--