TU Wien
Institut für Logic and Computation
Algorithms and Complexity Group

186.866 Algorithmen und Datenstrukturen VU 8.0

1. Test, 2020S

26. Juni 2020

Gruppe A

Hörsaal: Platz:

Machen Sie die folgenden Angaben in deutlicher **Blockschrift**:

Nachname: Vorname:

Matrikelnummer: Unterschrift:

Sie dürfen die Lösungen nur auf die Angabeblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Tablets, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Kennzeichnen Sie bei Ankreuzfragen eindeutig, welche Kästchen Sie kreuzen. Streichen Sie Passagen, die nicht gewertet werden sollen, deutlich durch. Unleserliche Antworten werden nicht gewertet.

	A1	A2	A3	A4	A5	Summe
Erreichbare Punkte:	20	20	20	20	20	100
Erreichte Punkte:						

Viel Erfolg!

Aufgabe A1: Algorithmenanalyse **(20 Punkte)**

a) (6 Punkte) Ordnen Sie folgende Funktionen nach Dominanz ($\ll$), beginnend mit der asymptotisch am schwächsten wachsenden. Es genügt die Funktionen zu reihen, ein Beweis der Gültigkeit der Relationen ist nicht erforderlich.

$$\left(\frac{4}{5}\right)^{2n}, \quad (2n)!, \quad \frac{3n^6+5n^3}{7n^2}, \quad \log(n^n), \quad 2\cdot\sqrt{10n^7}, \quad n\cdot 100, \quad (1.01)^{3n}$$

b) (6 Punkte) Gegeben sind die folgenden Funktionen:

$$f(n) = n^2 + \log(n+10)$$
$$g_1(n) = \sqrt{n^4}\cdot \log n$$
$$g_2(n) = \frac{n^5 - 5n^3}{3n^3}$$
$$g_3(n) = \begin{cases} (2n)^3 & \text{falls } n \text{ gerade} \\ \frac{3n^2}{\sqrt{n}} & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist in	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
$g_1(n)$				
$g_2(n)$				
$g_3(n)$				

Hinweis: Setzen Sie statt dem Punkt die entsprechende Funktion (g_1, g_2 bzw. g_3) ein. Beispielsweise ist die Zelle links oben als „$f(n)$ ist in $\Theta(g_1(n))$“ zu lesen.

c) (8 Punkte) Bestimmen Sie die Laufzeiten der unten angegebenen Algorithmen in Abhängigkeit vom Eingabeparameter n in Θ-Notation. Verwenden Sie hierfür möglichst einfache Terme.

```
k ← 0
for i = 1, ..., n
    for j = 1, ..., i
        k ← k + j
    for j = i + n, ..., 3n
        k ← k − j
return k
```

Laufzeit: $\square$

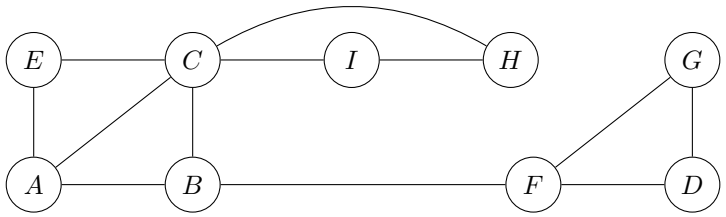
```
z ← 0
for i = 1, ..., 2n
    j ← n + 5
    while j > 1
        j ← j/2
    z ← z + j
return z
```

Laufzeit: $\square$

Aufgabe A2: Graphen **(20 Punkte)**

a) (4 Punkte) Führen Sie auf dem nachfolgenden Graphen die Breiten- und Tiefensuche entsprechend den Algorithmen in den Vorlesungsfolien durch. Geben Sie dabei jeweils die Reihenfolge an, in der die Knoten besucht werden.

Verwenden Sie jeweils F als Startknoten. Haben Sie die Wahl zwischen mehreren Knoten, gehen Sie alphabetisch vor.



BFS:

DFS:

b) (4 Punkte) Betrachten Sie die folgenden Behauptungen in Bezug auf ungerichtete und gerichtete Graphen. Kreuzen Sie an, ob diese wahr oder falsch sind.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Punkte)

- Es gibt gerichtete Graphen, die eine topologische Sortierung besitzen und einen gerichteten Kreis enthalten.

  ☐ Wahr ☐ Falsch

- Jeder zusammenhängende, ungerichtete Graph mit n Knoten und mehr als $n + 1$ Kanten enthält mindestens einen Kreis.

  ☐ Wahr ☐ Falsch

- Jeder stark zusammenhängende, gerichtete Graph mit 8 Knoten enthält mindestens einen gerichteten Kreis.

  ☐ Wahr ☐ Falsch

- Jeder ungerichtete Graph mit 10 Knoten, in dem jeder Knoten den Knotengrad 3 hat, enthält genau 15 Kanten.

  ☐ Wahr ☐ Falsch

c) (12 Punkte) Gegeben ist die folgende Beschreibung einer Ausführung des *Algorithmus von Dijkstra* auf einen gerichteten Graphen in der Implementierung mit einer Liste.

1. Discovered $= \emptyset$
   $L = \{s, a, b, c, d, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	∞	∞	∞	∞	∞	∞

2. Discovered $= \{s\}$
   $L = \{a, b, c, d, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	17	5	∞	3	∞	∞

3. Discovered $= \{s, d\}$
   $L = \{a, b, c, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	17	4	∞	3	∞	∞

4. Discovered $= \{s, d, b\}$
   $L = \{a, c, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	12	4	5	3	∞	∞

5. Discovered $= \{s, d, b, c\}$
   $L = \{a, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	∞	15

6. Discovered $= \{s, d, b, c, a\}$
   $L = \{e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

7. Discovered $= \{s, d, b, c, a, e\}$
   $L = \{u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

8. Discovered $= \{s, d, b, c, a, e, u\}$
   $L = \emptyset$

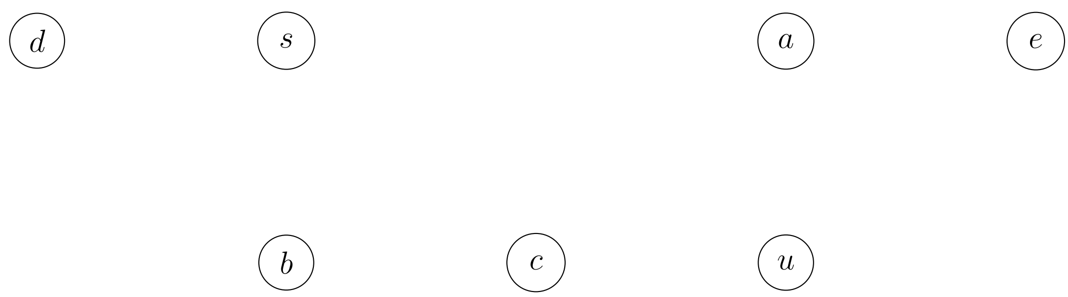
Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

(i) Extrahieren Sie aus diesem Ablauf den kürzesten Pfad von s nach u sowie seine Länge. Geben Sie den Pfad als Liste von Knoten an.

Pfad:

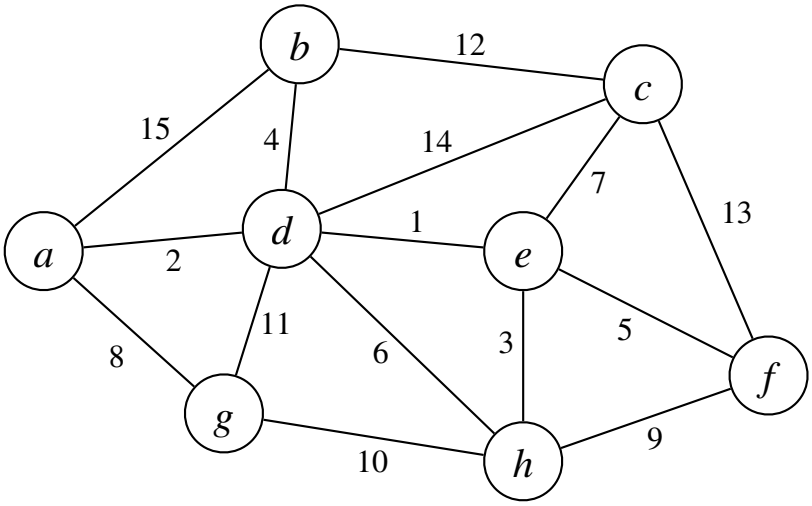
Gewicht:

(ii) Zeichnen Sie die Kanten und Kantengewichte eines Graphen der zu einem solchen Ablauf führt in der folgenden Grafik ein.



Aufgabe A3: Greedy **(20 Punkte)**

a) (10 Punkte) Ermitteln Sie mit dem Verfahren von Kruskal für den unten dargestellten gewichteten Graphen einen minimalen Spannbaum. Markieren Sie dazu die ausgewählten Kanten oder geben Sie die Kantenmenge in Form einer Liste an. Geben Sie außerdem die Reihenfolge an, in der die Kanten des Spannbaums hinzugefügt werden.



Hinweis: Es werden nicht unbedingt alle Zeilen benötigt.

Reihenfolge	Kante
1. Kante	
2. Kante	
3. Kante	
4. Kante	
5. Kante	
6. Kante	
7. Kante	
8. Kante	
9. Kante	
10. Kante	

b) (3 Punkte) Sei $G = (V, E)$ ein *nicht zusammenhängender* Graph und c Kantengewichte von E. Berechnet der Algorithmus von *Prim* mit der Eingabe (G, c) einen minimalen Spannbaum für jede Zusammenhangskomponente, wenn mit einem beliebigen Startknoten $v \in V$ begonnen wird? Begründen Sie Ihre Antwort.

☐ Ja ☐ Nein

Begründung:

c) (3 Punkte) Sei $G = (V, E)$ ein *nicht zusammenhängender* Graph und c Kantengewichte von E. Berechnet der Algorithmus von *Kruskal* mit der Eingabe (G, c) einen minimalen Spannbaum für jede Zusammenhangskomponente? Begründen Sie Ihre Antwort.

☐ Ja ☐ Nein

Begründung:

d) (4 Punkte) Kreuzen Sie an, ob folgende Aussagen wahr oder falsch sind.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Punkte)

- Das Kreislemma besagt, dass die günstigste Kante jedes in einem Graphen enthaltenen Kreises Teil des minimalen Spannbaumes dieses Graphen sein muss.

  ☐ Wahr ☐ Falsch
- Der minimale Spannbaum eines Graphen enthält zumindest eine Kante minimalen Gewichtes innerhalb jeder Kantenschnittmenge.

  ☐ Wahr ☐ Falsch
- Der Algorithmus von Prim und der Algorithmus von Kruskal berechnen gültige minimale Spannbäume und retournieren somit zwangsläufig denselben Spannbaum.

  ☐ Wahr ☐ Falsch
- Der Algorithmus von *Kruskal* ist in der Praxis auf *dünnen* Graphen gegenüber dem Algorithmus von Prim zu bevorzugen.

  ☐ Wahr ☐ Falsch

Aufgabe A4: Sortieralgorithmen **(20 Punkte)**

a) Führen Sie **Quicksort** auf dem unten gegebenen Array aus. Wählen Sie stets das letzte Element als Pivot-Element aus und implementieren Sie den Schritt des Aufteilens so, dass die Elemente einer Subfolge immer in derselben Reihenfolge angeordnet werden wie in der Originalfolge. Geben Sie, wie in der Vorlesung und Übung kennen gelernt, die Zwischenschritte an. Markieren Sie in jedem Zwischenschritt das ausgewählte Pivot-Element durch einkreisen.

8	2	4	7	1	9	3

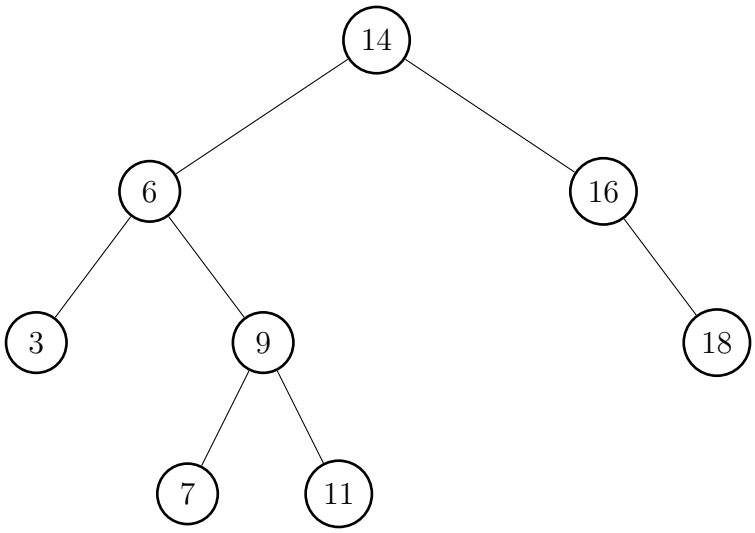
b) Sei $C(n)$ die Anzahl der Schlüsselvergleiche in Mergesort bei einer Eingabegröße n. Geben Sie die Rekursionsgleichung zur Berechnung von $C(n)$ von Mergesort an.

c) Sei die Eingabe von nachfolgenden Sortieralgorithmen ein Array mit n Zahlen im Bereich 0 bis z mit $z < n$. Ergänzen Sie folgende Laufzeiten von den in der Vorlesung kennengelernten Sortieralgorithmen in Abhängigkeit von n.

- Die **Worst-Case** Laufzeit von **Countsort** ist in $\Theta($ $\quad$ $)$
- Die **Average-Case** Laufzeit von **Quicksort** ist in $\Theta($ $\quad$ $)$
- Die **Best-Case** Laufzeit von **Mergesort** ist in $\Theta($ $\quad$ $)$

Aufgabe A5: AVL und B-Bäume **(20 Punkte)**

a) Gegeben ist folgender **AVL-Baum**:



(i) (6 Punkte) Fügen Sie die Schlüssel 13 und 8 in dieser Reihenfolge in diesen AVL-Baum ein. Falls notwendig, so rebalancieren Sie den Baum nach jedem Einfügen mit geeigneten Rotationsoperationen, um wieder einen gültigen AVL-Baum zu erhalten. Zeichnen Sie den vollständigen Baum direkt nach dem Einfügen und nach jedem Rotationsschritt.

(ii) (6 Punkte) Löschen Sie aus dem *ursprünglichen*, gegebenen AVL-Baum die Schlüssel 11 und 3 in dieser Reihenfolge. Falls notwendig rebalancieren Sie den Baum nach jedem Löschvorgang mit geeigneten Rotationsoperationen. Zeichnen Sie wiederum den vollständigen Baum direkt nach dem Löschen und nach jedem Rotationsschritt.

b) (8 Punkte) Fügen Sie die Elemente der Folge

$$\langle 18, 6, 7, 13, 14, 20, 22, 26, 35, 10 \rangle$$

in dieser Reihenfolge in einen anfangs leeren **B-Baum der Ordnung 3** ein. Zeichnen Sie den B-Baum jeweils vor und nach jeder Reorganisationsmaßnahme und geben Sie den endgültigen B-Baum an.