TU Wien
Institut für Logic and Computation
Algorithms and Complexity Group

186.866 Algorithmen und Datenstrukturen VU 8.0

1. Test, 2023S

12. Mai 2023

Gruppe A

Nachfolgende Angaben gut leserlich in Blockschrift machen.

Nachname: | Vorname:

Matrikelnummer: | Unterschrift:

Sie dürfen die Lösungen nur auf diesen Prüfungsbogen schreiben. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie dokumentenechte Stifte (keine Bleistifte, etc.).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Smartwatches, 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. Schreiben Sie leserlich, 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) (10 Punkte) Bestimmen Sie die Laufzeiten und Rückgabewerte der beiden unten angegebenen Algorithmen in Abhängigkeit von $n \in \mathbb{N}$ in Θ-Notation. Verwenden Sie möglichst einfache Terme.

(i)

```
k ← n
a ← 1
while k > 1
    for i = 1, ..., n
        a ← a + 3
    a ← a − 1
    k ← k/4
return a
```

(ii)

```
d ← n
r ← 0
while d > 1
    d ← d/2
    r ← r + 1
d ← 0
while r > 1
    r ← r/2
    d ← d + 1
return d
```

	(i)	(ii)
Laufzeit:		
Rückgabewert:		

b) (8 Punkte) Gegeben ist der folgende Algorithmus, der ein Array A und dessen Länge $n \geq 1$ als Eingabe erhält:

```
Function f(A, n):
    while A[0] = 0
        i ← n − 1
        while i > 0 and A[i] = 0
            i ← i − 1
        if i = 0 then
            return
        A[i] ← 0
        do
            i ← i − 1
        while A[i] ≠ 0
        A[i] ← 1
```

(i) Angenommen, der Algorithmus erhält das Array $A = [0, 0, 1, 1]$ der Länge $n = 4$ als Eingabe. Wie sieht das Array nach dem Durchlauf aus?

Array A:

(ii) Geben Sie die Best-Case und die Worst-Case Laufzeiten des Algorithmus in Abhängigkeit von n in Θ-Notation an. Verwenden Sie möglichst einfache Terme.

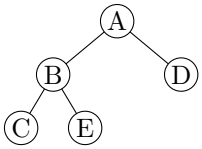
Best-Case:	
Worst-Case:	

c) (2 Punkte) Beweisen oder widerlegen Sie die folgende Aussage. Geben Sie für einen Beweis passende Konstanten c und n_0 an.

$$2^{2n+3} \in O(5^n)$$

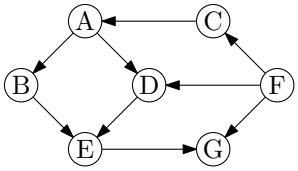
Aufgabe A2: Graphen **(20 Punkte)**

a) (3 Punkte) Zeichnen Sie einen Graphen, auf dem sowohl die Tiefensuche als auch die Breitensuche den unten stehenden Baum erzeugen. Wir nehmen an, dass beide Durchmusterungsverfahren bei A anfangen und bei mehreren Auswahlmöglichkeiten zuerst den lexikographisch kleinsten Knoten besuchen.



b) (2 Punkte) Ist der geforderte Graph aus der vorherigen Unteraufgabe eindeutig? Begründen Sie Ihre Antwort kurz.

c) (3 Punkte) Geben Sie **alle** topologischen Sortierungen des folgenden Graphen an.



d) (3 Punkte) Füllen Sie die Lücken in folgendem Text aus.

Max-Heaps können benutzt werden, um aus einer Menge von Elementen das größte Element in Zeit $O($ [ ] $)$ zu extrahieren. Die Heapeigenschaft fordert für jeden Knoten w des Max-Heaps und jedes Kind u von w, dass [ ].

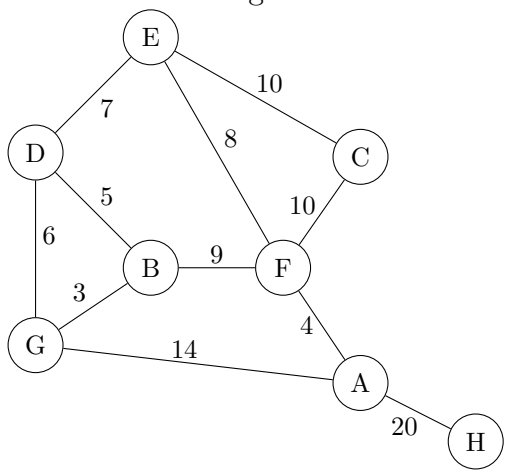
e) (i) (2 Punkte) Nennen Sie eine Datenstruktur aus der Vorlesung, um Graphen $G = (V, E)$ zu repräsentieren, mit der Sie für beliebige Knoten $u, v \in V$ in Zeit $O(1)$ entscheiden können, ob $(u, v) \in E$.

(ii) (4 Punkte) Angenommen Sie bekommen als Eingabe einen schlichten Graphen G, repräsentiert durch diese Datenstruktur. Schreiben Sie einen Pseudocode, der in Zeit $O(n^3)$ entscheidet, ob G ein Dreieck (einen Kreis der Länge drei) enthält.

(iii) (3 Punkte) Können Sie das Problem aus der vorigen Unteraufgabe in der gleichen asymptotischen Laufzeit $O(n^3)$ lösen, wenn Sie anstelle dieser Datenstruktur, den Graphen G als Adjazenzliste gegeben bekommen? Begründen Sie Ihre Antwort kurz.

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

a) (6 Punkte) Berechnen Sie einen minimalen Spannbaum des untenstehenden Graphen. Geben Sie eine mögliche Reihenfolge an, in der der Algorithmus von **Kruskal** die Kanten des Graphen für einen Spannbaum auswählt. Geben Sie die Reihenfolge als Liste von Knotenpaaren an (z.B.: AB, BC, CD, ...).



b) (2 Punkte) Hat der Graph aus Unteraufgabe a) einen eindeutigen minimalen Spannbaum? Begründen Sie Ihre Antwort.

c) (8 Punkte) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

(+2 Punkte für jede richtige, -2 Punkte für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

(Q1) Der Greedy-Algorithmus aus der Vorlesung, der das Interval Scheduling Problem in polynomieller Laufzeit löst, sortiert die Intervalle aufsteigend nach Startzeit.
☐ Wahr ☐ Falsch

(Q2) Sei K ein Kreis in einem gewichteten Graphen G, sodass dieser Kreis zwei Kanten e_1 und e_2 enthält, die das minimale Gewicht aller Kanten **in K** haben. Dann existiert ein minimaler Spannbaum von G der e_1 **und** e_2 enthält.
☐ Wahr ☐ Falsch

(Q3) Sei K ein Kreis in einem gewichteten Graphen G, sodass dieser Kreis zwei Kanten e_1 und e_2 enthält, die das minimale Gewicht aller Kanten **in G** haben. Dann existiert ein minimaler Spannbaum von G der e_1 **und** e_2 enthält.
☐ Wahr ☐ Falsch

(Q4) Sei S eine Teilmenge von Knoten eines zusammenhängen Graphen und e eine Kante maximalen Gewichts, die genau einen Endknoten in S hat. Dann gibt es keinen minimalen Spannbaum, der e enthält.
☐ Wahr ☐ Falsch

d) (4 Punkte) Gegeben ist eine Währung, deren Münzen eine Stückelung von 1,4,10, und 15 haben. Liefert der Greedy-Algorithmus zum Geldwechseln aus der Vorlesung bei dieser Stückelung immer ein optimales Ergebnis? Ein optimales Ergebnis ist ein Ergebnis mit minimaler Anzahl an Münzen.

☐ Ja ☐ Nein

Begründung/Gegenbeispiel:

Aufgabe A4: Sortierverfahren und Divide-and-Conquer (20 Punkte)

a) (10 Punkte) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Bei allen Aufgaben bezeichnet n die Anzahl der Schlüssel in der Eingabe.

(+2 Punkte für jede richtige, -2 Punkte für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

(Q1) Countsort lässt sich auf Arrays mit Einträgen aus den natürlichen Zahlen $\mathbb{N}$ als Eingabe anwenden und hat dann eine Laufzeit in $O(n \cdot \log n)$.

☐ Wahr ☐ Falsch

(Q2) Die Best-Case Laufzeit von Mergesort liegt in $O(n \cdot \log n)$.

☐ Wahr ☐ Falsch

(Q3) Bei einer Ausführung von Mergesort gilt, dass jeder Schlüssel mit $O(\log n)$ anderen verglichen wird.

☐ Wahr ☐ Falsch

(Q4) Die Average-Case Laufzeit von Quicksort liegt in $O(n \cdot \log n)$.

☐ Wahr ☐ Falsch

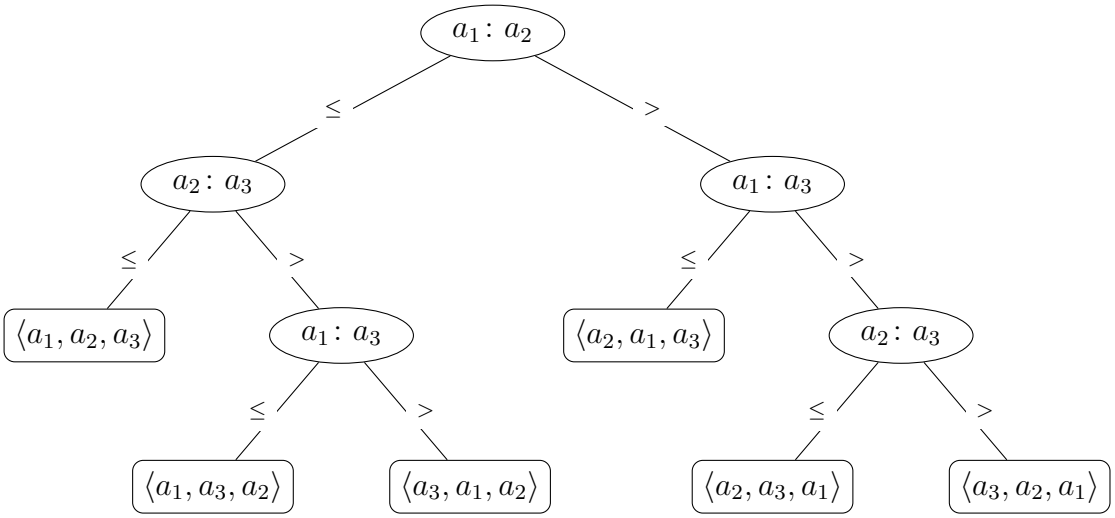
(Q5) Im Best-Case liegt die Höhe des Aufrufbaumes von Mergesort in $\Omega(\log n)$.

☐ Wahr ☐ Falsch

b) (4 Punkte) Führen Sie **Quicksort** auf dem unten gegebenen Array aus, um es aufsteigend zu sortieren. Wählen Sie stets das **letzte** Element als Pivotelement 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 die Zwischenschritte an, wie in der Vorlesung und Übung kennen gelernt. Markieren Sie in jedem Zwischenschritt das ausgewählte Pivotelement indem Sie es einkreisen. Gegeben ist das Array $[12, 8, 6, 3, 5, 9]$.

c) (6 Punkte) In der Vorlesung haben Sie einige vergleichsbasierte Sortieralgorithmen kennengelernt, wie zum Beispiel Quicksort und Mergesort. Ein vergleichsbasierter Algorithmus geht nach dem folgenden Prinzip vor: Er bekommt als Eingabe eine Liste A von paarweise verschiedenen Zahlen und führt eine Serie von Vergleichen von Paaren dieser Zahlen aus. Basierend auf den Ergebnissen dieser Vergleiche, gibt er die Zahlen in A in aufsteigend sortierter Reihenfolge aus.

Untenstehend ein Beispiel dafür, wie ein vergleichsbasierter Algorithmus bei Eingabe der Liste $A = \langle a_1, a_2, a_3 \rangle$ vorgehen könnte. In diesem Beispiel vergleicht er zuerst a_1 mit a_2. Wenn $a_1 \leq a_2$, dann vergleicht er a_2 mit a_3, wenn $a_1 > a_2$, dann vergleicht er a_1 mit a_3 und so weiter.



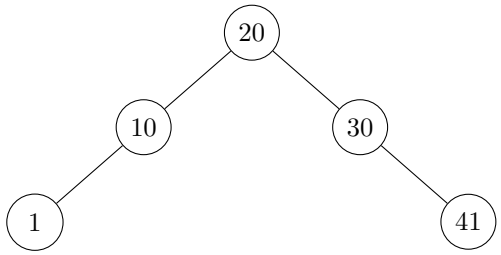
Zeigen Sie, dass jeder vergleichsbasierte Algorithmus im Worst-Case eine Laufzeit von $\Omega(n \cdot \log n)$ hat, wobei n die Anzahl der Elemente in der Eingabeliste A ist. Sie dürfen ohne Beweis annehmen, dass $\log_2(n!) \in \Omega(n \cdot \log n)$.

Aufgabe A5: Suchbäume und Hashing **(20 Punkte)**

a) (9 Punkte) Gegeben ist ein Algorithmus `Foo`, welcher die Wurzel *root* eines natürlichen binären Suchbaumes als Eingabe erhält und die aus der Vorlesung bekannten Funktionen `Minimum`, `Maximum`, `Search` und `Insert` für Suchbäume verwendet.

```
Foo(root):
    a ← Minimum(root)
    b ← Maximum(root)
    c ← ⌊(a + b)/2⌋
    if Search(root, c) = null then
        Insert(root, c)
```

(i) Führen Sie `Foo` mit folgendem Suchbaum als Eingabe aus und geben Sie den Baum nach Ausführung der Funktion an. Sie dürfen dazu die untenstehende Abbildung ergänzen.



(ii) Bestimmen Sie die Worst- und Best-Case Laufzeit von `Foo` in O-Notation abhängig von der Knotenanzahl n des gegebenen Suchbaumes. Wählen Sie eine kleinstmögliche obere Schranke.

Worst-Case Laufzeit: ☐

Best-Case Laufzeit: ☐

(iii) Nehmen Sie jetzt an, die Eingabe für `Foo` ist auf AVL-Bäume beschränkt und die Funktion `Insert`, welche in `Foo` verwendet wird, ist die Einfügeoperation für AVL-Bäume. Bestimmen Sie die Worst-Case und Best-Case Laufzeit von `Foo` in O-Notation unter dieser Einschränkung. Wählen Sie eine kleinstmögliche obere Schranke.

Worst-Case Laufzeit: ☐

Best-Case Laufzeit: ☐

b) (4 Punkte) Bestimmen Sie den binären Wurzelbaum, der durch die folgende Inorder- und Postorder-Durchmusterungsreihenfolge gegeben ist.

Inorder: d, b, c, e, a

Postorder: d, c, a, e, b

c) (7 Punkte) Gegeben ist folgende Hashtabelle T der Länge $m = 7$.

Position	0	1	2	3	4	5	6
Schlüssel			9		2	12	
Flag	frei	frei	besetzt	wieder frei	besetzt	besetzt	frei

Zur Kollisionsbehandlung wird **lineares Sondieren** mit $h(k, i) = (h'(k)+i) \bmod 7$, wobei $h'(k) = k \bmod 7$, verwendet. Für das Sondieren gilt $i = 0, 1, \ldots, m-1$.

(i) Führen Sie eine Suche nach Schlüssel 2 in T aus. Geben Sie alle dabei sondierte Positionen in entsprechender Sondierungsreihenfolge an. Ist die Suche erfolgreich?

Sondierte Positionen: ☐

☐ Suche erfolgreich ☐ Suche nicht erfolgreich

(ii) Führen Sie jetzt eine Suche nach Schlüssel 16 in T aus. Geben Sie erneut alle dabei sondierte Positionen in entsprechender Sondierungsreihenfolge an. Ist die Suche erfolgreich?

Sondierte Positionen: ☐

☐ Suche erfolgreich ☐ Suche nicht erfolgreich

(iii) Fügen Sie den Schlüssel 23 in T ein. An welcher Position wird der Schlüssel eingefügt? Welcher Flag wird nach dem Einfügen an dieser Position gesetzt?

Position: ☐ Flag: ☐

(iv) Angenommen wir löschen den Schlüssel 9 aus T. Welcher Flag wird nach dem Löschen an der betroffenen Position gesetzt?

Flag: ☐