Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Abteilung Algorithmik

Dr. Linda Kleist
Arne Schmidt

Sommersemester 2019

Klausur
Algorithmen und Datenstrukturen II
09.08.2019

Name: .....

Vorname: .....

Matr.-Nr.: .....

Studiengang: .....

☐ Bachelor ☐ Master ☐ Andere

Klausurcode:
Dieser wird benötigt, um das Ergebnis der Klausur abzurufen.

Hinweise:

- Bitte das Deckblatt in Druckschrift vollständig ausfüllen.
- Die Klausur besteht aus 12 Blättern, bitte auf Vollständigkeit überprüfen.
- Erlaubte Hilfsmittel: keine
- Eigenes Papier ist nicht erlaubt.
- Die Rückseiten der Blätter dürfen beschrieben werden.
- Antworten, die *nicht* gewertet werden sollen, bitte deutlich durchstreichen. Kein Tippex verwenden.
- Mit *Bleistift* oder in *Rot* geschriebene Klausurteile können nicht gewertet werden.
- Werden mehrere Antworten gegeben, werten wir die mit der geringsten Punktzahl.
- Sämtliche Algorithmen, Datenstrukturen, Sätze und Begriffe beziehen sich, sofern nicht explizit anders angegeben, auf die in der Vorlesung vorgestellte Variante.
- Die Bearbeitungszeit für die Klausur beträgt 120 Minuten.

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Σ**
Punkte	**12**	**15**	**10**	**5**	**15**	**8**	**23**	**12**	**100**
Erreicht									

Aufgabe 1: Hashing **(7+1+1+2+1 Punkte)**

a) Betrachte ein anfangs leeres Array A der Größe 11 mit Speicherzellen $A[0], \ldots, A[10]$. In diesem Array führen wir Hashing mit offener Addressierung mit der folgenden Hashfunktion durch:

$$t(x, i) = 2x^2 + 1 + (x^2 + 1)i \mod 11$$

Dabei ist x ein Schlüssel und i die Nummer des Versuchs, x in eine unbesetzte Speicherzelle des Arrays einzufügen, beginnend bei $i = 0$. Berechne zu jedem der folgenden Schlüssel die Position, die er in A bekommt:

$$1, \quad 9, \quad 2, \quad 10$$

Dabei sollen die Schlüssel in der gegebenen Reihenfolge eingefügt werden und der Rechenweg soll klar erkennbar sein. Trage die Elemente in die folgende Tabelle ein.

j	0	1	2	3	4	5	6	7	8	9	10
$A[j]$											

b) Wie ist der Belegungsfaktor β für eine Hashtabelle der Größe m und n eingefügten Elementen definiert?

c) Wie groß ist β in der Hashtabelle aus Aufgabenteil a) nach Einfügen der vier Elemente?

d) Sei $h(x)$ eine Hashfunktion und $\text{Prob}[h(x) = j] = \frac{1}{m}$, wobei m die Größe der Hashtabelle bezeichnet.

(i) Wie lautet die Formel zum Bestimmen der Wahrscheinlichkeit, dass k Elemente kollisionsfrei eingefügt werden können?

(ii) Berechne die Wahrscheinlichkeit, dass drei Elemente kollisionsfrei in eine Hashtabelle der Größe $m = 10$ eingefügt werden können.

Aufgabe 2: Knapsack I - Subset Sum **(4+5+1+5 Punkte)**

a) Seien $z_1, \ldots, z_n$ ganze Zahlen. $\mathcal{S}(x, i)$ nimmt den Wert 1 an, falls x als Summe von einer Auswahl von $z_1, \ldots, z_i$ dargestellt werden kann. Andernfalls ist $\mathcal{S}(x, i) = 0$. Wie lautet die Rekursionsgleichung zur Bestimmung von $\mathcal{S}(x, i)$?

b) Wende das dynamische Programm für Subset Sum auf folgende Instanz an.

Objekt i	1	2	3	4	5
Gewicht z_i	7	5	6	5	8

mit $Z = 15$

Fülle hierzu die folgende Tabelle aus, wobei der Eintrag in Zeile i und Spalte x dem Wert $\mathcal{S}(x, i)$ entspricht. Nullen müssen nicht eingetragen werden.

$i \backslash x$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1																
2																
3																
4																
5																

c) Ist die Subset Sum Instanz aus b) lösbar? Begründe deine Antwort!

d) Sei $\mathcal{A}$ ein Algorithmus, der in polynomieller Zeit Subset Sum entscheidet, das heißt, $\mathcal{A}$ gibt nur *wahr* oder *falsch* aus. Zeige, dass mit Hilfe von $\mathcal{A}$ (als Blackbox!) eine Lösungsmenge S in polynomieller Zeit gefunden werden kann, falls diese existiert.

Aufgabe 3: Knapsack II - Greedy$_k$ **(7+1+2 Punkte)**

a) Wende GREEDY$_k$ auf die folgende, nach $\frac{z_i}{p_i}$ aufsteigend sortierte Instanz für MAXIMUM KNAPSACK an.

i	1	2	3
z_i	4	3	9
p_i	5	3	8

mit $Z = 12$ und $k = 2$.

Gib dazu die folgenden Mengen bzw. Werte tabellarisch an:

- Fixierte Menge: $M := \{i \mid v_i = 1\}$
- Gewicht der fixierten Menge: $\sum_{i \in M} z_i$
- Restkapazität: $Z - \sum_{i \in M} z_i$
- Wert von $\text{GREEDY}_0\left((1 - v_1)z_1, \ldots, (1 - v_n)z_n, Z - \sum_{i \in M} z_i, p_1, \ldots, p_n\right) : G$
- Wert der bisher besten Lösung: G_k
- Bisher beste Lösung: $S := \{i \mid x_i = 1\}$

Achte darauf, dass M mit der kleinsten Menge anfängt und mit einer größten endet. Zusätzlich soll M lexikographisch sortiert sein, das heißt, für zwei gleichgroße Mengen M_1 und M_2 kommt M_1 vor M_2, falls das kleinste Element $x \in M_1 \setminus M_2$ kleiner ist als das kleinste Element $y \in M_2 \setminus M_1$.

M	$\sum_{i \in M} z_i$	$Z - \sum_{i \in M} z_i$	G	G_k	S

b) Umrande in der Tabelle die Rückgabewerte von GREEDY$_k$.

c) Ist die in a) erhaltene Lösung optimal? Begründe deine Antwort!

Aufgabe 4: Knapsack III - Branch-and-Bound **(5 Punkte)**

Wende den Branch-and-Bound-Algorithmus auf folgende, nach $\frac{z_i}{p_i}$ aufsteigend sortierte Instanz für MAXIMUM KNAPSACK an.

i	1	2	3
z_i	7	12	3
p_i	10	14	3

und $Z = 12$

Beachte folgende Punkte:

- Benutze den Entscheidungsbaum aus Abbildung 1.
- Beschrifte die Kanten mit der Auswahl, die getroffen wurde.
- Beschrifte die Knoten mit den aktuell besten Schranken (obere und untere).
- Beschrifte einen Knoten mit *unzulässig*, falls die aktuelle Auswahl unzulässig ist.
- Sollten Kanten nicht benutzt werden, streiche sie durch.
- Halte in einer Tabelle fest, welche Objekte eine neue, beste Lösung liefern.

Menge	Wert

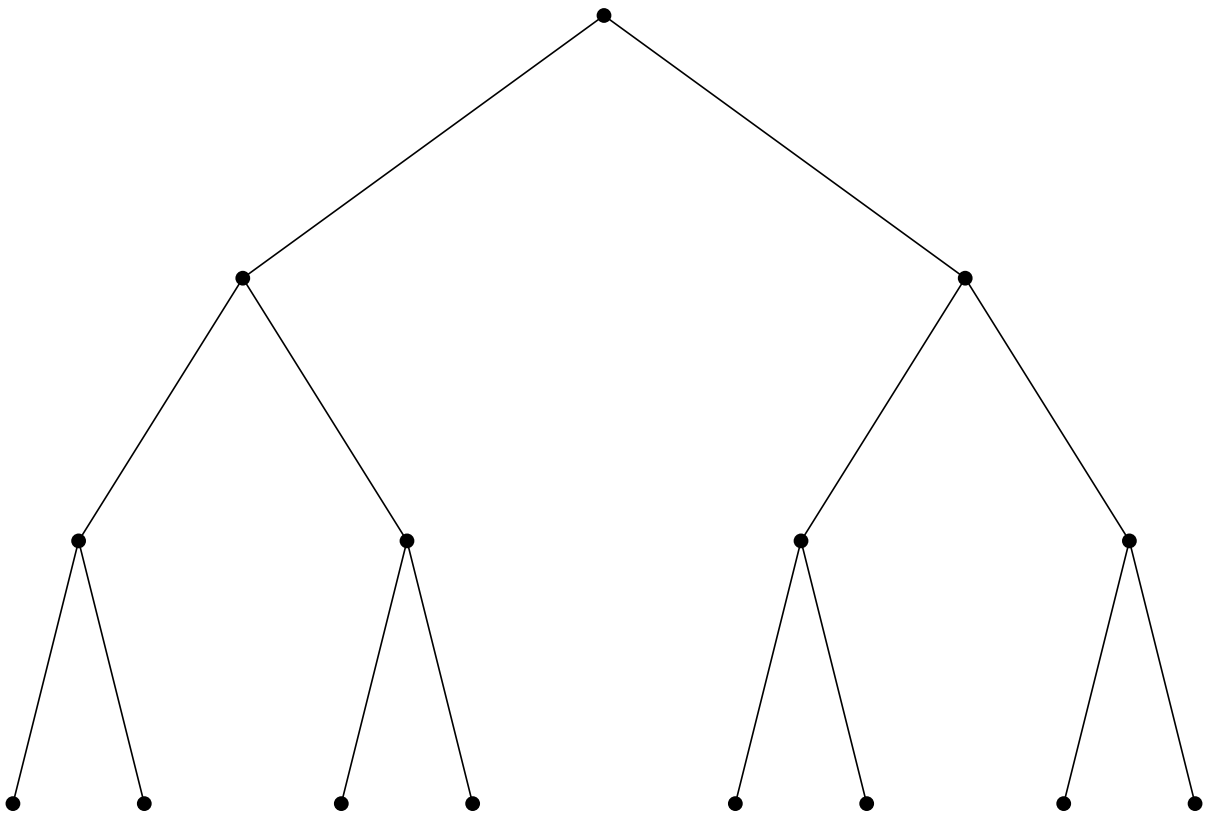
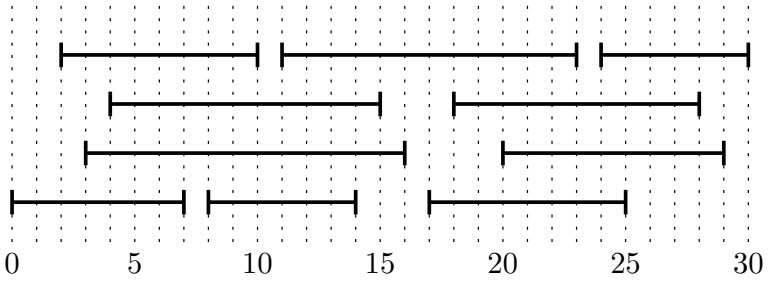


Abbildung 1: Ein Entscheidungsbaum.

Aufgabe 5: Hörsaal-Belegung **(4+5+6 Punkte)**

a) Betrachte die folgende Instanz für das Problem Hörsaal-Belegung



Wende den Greedy-Algorithmus aus der Vorlesung auf diese Instanz an, um eine maximale Anzahl an disjunkten Intervallen zu finden. Nummeriere dazu die Intervalle in der sortierten Reihenfolge durch. Gib dann in der richtigen Reihenfolge an, welches Intervall aufgenommen wird und welche Intervalle dadurch geschnitten werden, indem du die nachfolgende Tabelle ausfüllst.

Aufgenommen	Geschnitten

b) Zeige, dass das Problem Hörsaal-Belegung nicht optimal gelöst werden kann, indem man immer das kürzeste Intervall zuerst aufnimmt.

c) Zeige, dass die Strategie in b) eine $\frac{1}{2}$-Approximation für das Problem Hörsaal-Belegung ist.

Aufgabe 6: Rundreise Problem **(2+3+3 Punkte)**

Gegeben sind n Städte $\mathcal{P} := \{P_1, \ldots, P_n\}$ und eine Gewichtsfunktion $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$, welche die Distanz zwischen den Städten repräsentiert. Gesucht ist eine Rundreise minimaler Länge, die alle Städte (einmal) besucht.

Um das Problem zu lösen, wollen wir dynamische Programmierung nutzen. Sei $S \subseteq \{1, \ldots, n\}$ eine Menge von Indizes mit $1 \in S$ und $\ell \in S$. Wir definieren $D(S, \ell)$ als die Länge eines kürzesten Weges von P_1 zu P_ℓ, der alle Städte P_i mit $i \in S$ besucht.

a) Beschreibe eine Idee, um $D(S, \ell)$ rekursiv zu bestimmen.

b) Stelle eine Rekursionsgleichung auf, die $D(S, \ell)$ für beliebige S und ℓ bestimmt.

c) Wie kann $D(S, \ell)$ genutzt werden, um die kürzeste Rundreise zu bestimmen?

Aufgabe 7: Bin Packing **(8+5+5+5 Punkte)**

Wir betrachten das Bin-Packing-Problem. Gegeben sind Behälter der Größe Z und Objekte $\{O_1, \ldots, O_n\}$, jeweils mit Größe $z_i \leq Z$. Gesucht ist eine Zuordnung der Objekte in möglichst wenige Behälter; wobei jeder Behälter Objekte einer Gesamtgröße von höchstens Z enthalten darf.

NextFit (siehe nachfolgenden Algorithmus) packt der Reihe nach Objekte in den aktuellen Behälter, bis das nächste Objekt nicht mehr passt. Danach wird ein neuer Behälter geöffnet.

```
1:  function NextFit(z_1, ..., z_n, Z)
2:      b := 1                                   ▷ Index des aktuellen Behälters
3:      B_1 := ∅

4:      for i := 1 to n do
5:          if z_i + Σ_{j∈B_b} z_j > Z then
6:              b := b + 1
7:              B_b := ∅                         ▷ öffne neuen Behälter
8:          end if
9:          B_b := B_b ∪ {i}
10:     end for

11:     return b
12: end function
```

a) Wende NextFit auf folgende Instanz an:

i	1	2	3	4	5	6	7	8
z_i	3	4	7	4	1	6	2	10

und $Z = 10$

Gib nach jeder Iteration den Wert von b, sowie die Füllstände der Behälter an.

Iteration	Objekt	b	Füllstand der Behälter				
			B_1	B_2	B_3	B_4	B_5
1	O_1						
2	O_2						
3	O_3						
4	O_4						
5	O_5						
6	O_6						
7	O_7						
8	O_8						

b) Konstruiere für beliebiges $n \in \mathbb{N} \setminus \{0\}$ eine Sequenz von $2n$ Objekten, so dass die optimale Verteilung maximal $\frac{n}{2} + 1$ Behälter, NextFit aber n Behälter benötigt.
(Hinweis: Betrachte abwechselnd kleine und große Objekte.)

c) Zeige: NextFit ist eine 2-Approximation für Bin Packing.

d) Zeige: Falls ein c-Approximationsalgorithmus für Bin Packing mit $c < \frac{3}{2}$ existiert, dann gilt $P = NP$.

Aufgabe 8: Kurzfragen **(2+2+2+2+2+2 Punkte)**

Kreuze an, welche Aussagen korrekt sind. Es gibt nur Punkte für vollständig korrekt angekreuzte Teilaufgaben.

(Hinweis: In jeder Teilaufgabe ist immer mindestens eine Aussage korrekt.)

a) Welche Algorithmen für Maximum Knapsack besitzen polynomielle Laufzeit?

- Greedy$_k$ für $k \in \mathbb{N}$ ☐
- Dynamische Programmierung ☐
- Greedy$_0$ ☐

b) Sei A ein Problem in P. Welche Aussagen sind wahr?

- A lässt sich in polynomieller Zeit auf jedes NP-schwere Problem B reduzieren, d.h. $A \leq_p B$. ☐
- A lässt sich in polynomieller Zeit lösen. ☐
- Eine Lösung für A lässt sich in polynomieller Zeit verifizieren. ☐

c) Der Branch-and-Bound-Algorithmus für Maximum Knapsack...

- ...besitzt eine polynomielle Laufzeit. ☐
- ...nutzt Fractional Knapsack als untere Schranke. ☐
- ...findet eine optimale Lösung. ☐

d) Welche Aussagen zu Fractional Knapsack sind korrekt?

- Das Problem lässt sich in polynomieller Zeit lösen. ☐
- Das Problem ist ein Entscheidungsproblem. ☐
- Die optimale Lösung besitzt mindestens den Wert der Größenschranke Z. ☐

e) Beim Hashing mit offener Addressierung...

- ...enthält die Hashtabelle nur verschiedene Schlüssel. ☐
- ...erfüllt jede Hashfunktion die Permutationsbedingung. ☐
- ...ist der Belegungsfaktor höchstens 1. ☐

f) Welche Aussagen zu Graphenproblemen sind korrekt?

- Für jeden Intervallgraphen lässt sich eine größte unabhängige Menge in polynomieller Zeit bestimmen. ☐
- Es gibt eine 2-Approximation für Min Vertex Cover. ☐
- Für jeden Graphen lässt sich eine größte Clique in polynomieller Zeit finden. ☐

Viel Erfolg ☺