Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Abteilung Algorithmik

Prof. Dr. Sándor Fekete
Arne Schmidt

Sommersemester 2020

Klausur
Algorithmen und Datenstrukturen II
14.08.2020

Name: .....

Vorname: .....

Matr.-Nr.: .....

Studiengang: .....

Klausurraum: .....

Platznummer: .....

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

☐ Bachelor ☐ Master ☐ Andere

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**	**Σ**
Punkte	**15**	**20**	**10**	**25**	**10**	**20**	**100**
Erreicht							

Aufgabe 1: Hashing **(7+1+1+2+4 Punkte)**

a) Betrachte ein anfangs leeres Array A der Größe 13 mit Speicherzellen $A[0], \ldots, A[12]$. In diesem Array führen wir Hashing mit offener Adressierung mit der folgenden Hashfunktion durch:

$$t(x,i) = 3x + 4 + i \cdot h(x) \mod 13 \quad \text{mit} \quad h(x) = (5x + 1 \mod 12) + 1$$

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:

$$3, \quad 16, \quad 8, \quad 2$$

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	11	12
$A[j]$													

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

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

d) Sei $h(x)$ eine Hashfunktion und $\text{Prob}[h(x) = j] = \frac{1}{m}$, wobei m die Größe der Hashtabelle bezeichnet. Wie lautet die Formel zur Bestimmung der Wahrscheinlichkeit, dass k Schlüssel kollisionsfrei in eine Hashtabelle eingefügt werden können?

e) Begründe kurz, warum beim Hashing mit offener Adressierung Schlüssel nicht einfach gelöscht werden dürfen. Beschreibe außerdem kurz, wie das Problem umgangen werden kann.

Aufgabe 2: Knapsack I - Greedy-Algorithmen **(6+3+5+6 Punkte)**

a) Betrachte folgende Instanz für MAXIMUM KNAPSACK:

i	1	2	3	4	5	
z_i	20	10	3	13	4	mit $Z = 30$
p_i	20	11	5	9	5	

Wende den Greedy-Algorithmus für MAXIMUM KNAPSACK auf diese Instanz an. Gib in jeder Iteration die folgenden Werte an: den aktuellen Gegenstand, ob er gepackt wird, das neue Gesamtgewicht und den neuen Gesamtwert.

Iteration	Aktueller Gegenstand	Gepackt?	Gesamtgewicht	Gesamtwert
1				
2				
3				
4				
5				

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

c) Zeige, dass der Greedy-Algorithmus für MAXIMUM KNAPSACK keine Approximation liefert.

d) Wie kann FRACTIONAL KNAPSACK optimal in $O(n \log n)$ Zeit gelöst werden? Begründe außerdem kurz die Korrektheit und die Laufzeit.

Aufgabe 3: Hörsaal-Belegung **(3+3+4 Punkte)**

Betrachte das Problem HÖRSAAL-BELEGUNG:

Gegeben: Menge von Intervallen $\mathcal{I} := \{I_1 = [s_1, e_1), \ldots, I_n = [s_n, e_n)\}$

Gesucht: Eine größtmögliche Teilmenge $\mathcal{I}' \subseteq \mathcal{I}$ von disjunkten Intervallen.

a) Beschreibe kurz, wie das Problem optimal in $O(n \log n)$ Zeit gelöst werden kann.

b) Zeige, dass man nicht unbedingt eine optimale Lösung bekommen, wenn 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 4: Knapsack II - Exakte Algorithmen (6+2+4+7+6 Punkte)

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

Objekt i	1	2	3	4	5
Gewicht z_i	4	5	3	4	4

mit $Z = 18$

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 \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0																			
1																			
2																			
3																			
4																			
5																			

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

c) Wie lautet die Rekursionsgleichung, um Subset Sum zu lösen, wenn Objekte beliebig oft verwendet werden dürfen? Sei dazu $\mathcal{S}'(x, i) = 1$, wenn x mit den ersten i Objekten erzeugt werden kann und 0 andernfalls.

d) 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	3	12	6
p_i	4	12	5

und $Z = 20$

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

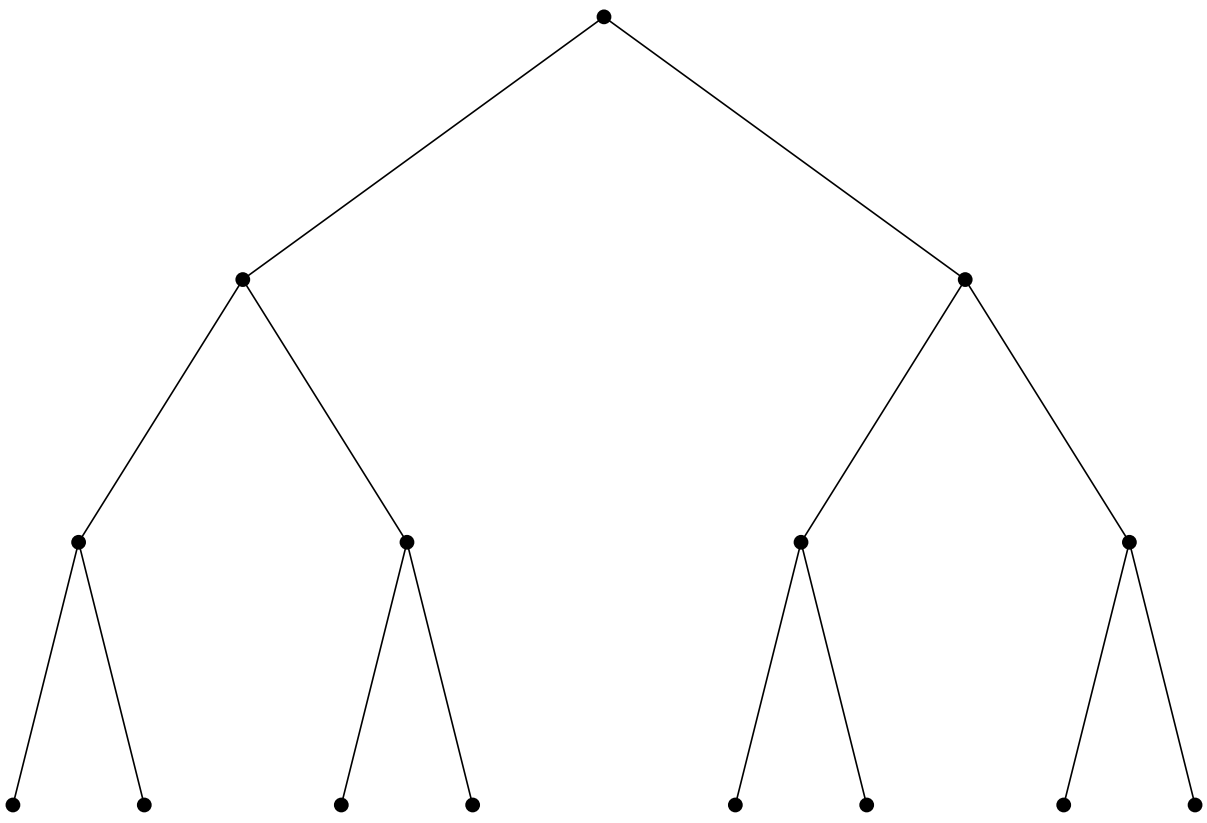

Abbildung 1: Ein Entscheidungsbaum.

e) Betrachte das Problem Maximum Knapsack. Kreuze an, welche Aussagen korrekt sind.
(Hinweis: In jeder Teilaufgabe ist **genau eine** Aussage korrekt.)

(i) Die Laufzeit $O(nZ)$ des dynamischen Programms heißt...

- ...quasipolynomiell. ☐
- ...pseudopolynomiell. ☐
- ...polynomiell. ☐

(ii) Zur Berechnung der unteren Schranke im Branch-and-Bound-Algorithmus benutzen wir...

- ...Greedy_0. ☐
- ...Dynamic Programming. ☐
- ...Greedy für Fractional Knapsack. ☐

(iii) Sei v ein Knoten im Enumerationsbaum und UB eine dazu passende obere Schranke. UB gilt auf jeden Fall...

- ...im gesamten Enumerationsbaum. ☐
- ...oberhalb von v. ☐
- ...unterhalb in v. ☐

Aufgabe 5: Dynamische Programmierung **(3+4+3 Punkte)**

Gegeben sei ein Holzbrett H der Länge L und eine Preisliste $p_1, \ldots, p_n$, wobei p_i den Preis für ein Holzbrett der Länge i für $i = 1, \ldots, n$ angibt. Wir wollen H in kleinere Stücke teilen, um maximalen Profit zu bekommen, d.h. wir suchen Längen $\ell_1, \ldots, \ell_m$ mit $\sum_{j=1}^{m} \ell_j = L$ und $\sum_{j=1}^{m} p_{\ell_j}$ maximal.

a) Betrachte folgende Preisliste:

i	1	2	3	4	5
p_i	2	6	9	13	16

Was ist der maximale Profit für Holzbretter der Längen $0, \ldots, 9$? Fülle dazu die folgende Tabelle aus:

Länge	0	1	2	3	4	5	6	7	8	9
Profit										

b) Sei $\mathcal{P}(L)$ der maximale Profit ein Holzbrett der Länge L in Teilbretter der Längen $1, \ldots, n$ zu teilen.

(i) Stelle eine Rekursionsgleichung auf, um $\mathcal{P}(L)$ zu bestimmen.

(ii) Begründe kurz, dass deine Rekursionsgleichung korrekt ist.

Aufgabe 6: Komplexität **(5+5+2+1+2+5 Punkte)**

a) Wie ist die Problemklasse P definiert? Gib außerdem ein Problem in P an und begründe die Zugehörigkeit.

b) Wie ist die Problemklasse NP definiert? Gib außerdem ein Problem in NP an und begründe die Zugehörigkeit.

c) Wie zeigt man, dass ein Problem NP-schwer ist?

d) Wie zeigt man, dass ein Problem NP-vollständig ist?

e) Angenommen es gilt P $\neq$ NP. Gibt es Probleme in NP, die nicht NP-schwer sind? Begründe deine Antwort.

f) Reduziere die folgende 3SAT-Formel auf eine Instanz von Subset Sum mit der in der Vorlesung vorgestellten Reduktion.

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4)$$

Viel Erfolg ☺