Probeklausur Algorithmen

Prof. Kaufmann/Kuckuk/Förster, Wintersemester 2020/21

- Die Bearbeitungszeit beträgt **90 Minuten**, elektronische Hilfmittel sind verboten, nur ein handbeschriebenes DIN-A4-Blatt mit Notizen ist erlaubt.
- Von den 6 Aufgaben mit jeweils 6 möglichen Punkten werden 5 bewertet. **Markieren Sie die Aufgabe, die nicht bewertet werden soll.**

Viel Erfolg!

Name: .....

Matr.-Nr.: .....

Studiengang: .....

Angestr. Abschluss: .....

Sitzplatznummer: .....

	Punkte	max
1		6
2		6
3		6
4		6
5		6
6		6
Σ		30

Aufgabe 1: **Rekursionsgleichungen** *6 Punkte*

Gegeben ist die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 4 \cdot T\left(\frac{n}{2}\right) + n^2 & \text{falls } n > 1 \\ 1 & \text{sonst} \end{cases}.$$

Dabei kann angenommen werden, dass n eine Zweierpotenz ist, also $n = 2^k$ für eine natürliche Zahl k.

a) Wenden Sie das Mastertheorem auf $T(n)$ an!

b) Finden Sie für $T(n)$ eine geschlossene Form und beweisen Sie die Korrektheit mittels Induktion.

Tipp: Überprüfen Sie auch mittels a), ob Ihre geschlossene Form stimmen kann! Erinnern Sie sich auch daran, dass $(2^a)^b = 2^{a \cdot b} = (2^b)^a$ und dass $2^a 2^b = 2^{a+b}$.

Aufgabe 2: **Hashing** *6 Punkte*

Wir betrachten Hashing mit Verkettung.

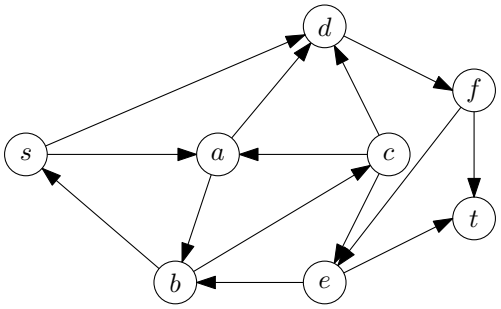
a) Fügen Sie die Elemente 3,6,2,11,20,9,4,8 nacheinander in eine zu Beginn leere Hashtabelle der Größe 9 ein. Verwenden Sie dabei die Hashfunktion $h(x) = (2x + 3) \bmod 9$ und zeigen Sie alle Schritte.

b) Betrachten Sie nun die Hashfunktion $h_{a,b}(x) = (ax + b) \bmod 11$. Gibt es Werte für a and b, so dass alle Elemente aus a) auf verschiedene Hashwerte abgebildet werden, wenn sie in eine zu Beginn leere Hashtabelle der Größe 13 eingefügt werden? Begründen Sie Ihre Antwort!

c) Beim Hashing mit Verkettung kann es passieren, dass zwar noch einige Hashzellen frei sind, aber einige der Liste bereits sehr lang sind. In diesen ist dann das Abrufen von Werten sehr zeitaufwendig. Geben Sie zwei effiziente Strategien an, durch die verhindert werden kann, dass der Abruf von Werten durch das Entstehen solch langer Listen so lange dauert. Begründen Sie kurz, warum diese funktionieren und effizient sind.

Aufgabe 3: **Graphdurchmusterung** *6 Punkte*

Betrachten Sie den folgenden gerichteten Graph:



a) Berechnen Sie mit BFS den kürzesten Weg von s zu allen anderen Knoten. Verwenden Sie dafür die untenstehende Tabelle und geben Sie die Einträge in der folgenden Form an:

(Knoten, Länge des kürzesten Wegs zum Knoten, Vorgängerknoten)

b) Sei $G = (V, E)$ ein (beliebiger) gerichteter Graph und p ein kürzester Pfad in G von $s \in V$ nach $t \in V$. Zeigen oder widerlegen Sie: Wird jede Kante (u, v) von G durch einen gerichteten Pfad von u nach v der Länge 7 ersetzt, so gibt es im resultierenden Graphen einen kürzesten Pfad von s nach t, der durch alle Knoten geht, durch die p geht.

Schritt	S (fertig)	S' (Warteschlange)
1	(s,0,-)	
2		
3		
4		
5		
6		
7		
8		

Aufgabe 4: **Sortieralgorithmen** *6 Punkte*

In der Vorlesung haben Sie den Sortieralgorithmus Quicksort kennengelernt. Dieser hat eine erwartete Laufzeit von $\mathcal{O}(n \log n)$. Die Laufzeit ist im Worst-Case jedoch schlechter, da man nicht zwingend ein gutes Pivot-Element wählt.

Um dies zu verdeutlichen, nehmen wir in dieser Aufgabe an, dass stets der Median als Pivot-Element in konstanter Zeit gewählt werden kann. Dadurch wird das zu sortierende Array A beim Vergleich mit dem Pivot stets in zwei gleich große Teilarrays A_1 und A_2, die dann rekursiv sortiert werden, zerlegt.

a) Beweisen Sie, dass Quicksort unter der Annahme, dass der Median in konstanter Zeit bestimmt werden kann, ein Array der Größe n in Zeit $\mathcal{O}(n \log n)$ sortiert.

Wenn man das Pivot stets optimal bestimmt, lässt sich Quicksort auch sehr gut parallelisieren. Grafikkarten haben heute sehr viele Cores. Vereinfacht nehmen wir an, dass es so viele Cores gibt, dass alle Teilprobleme derselben Größe parallel berechnet werden können, d.h. alle diese Berechnungen laufen in derselben Zeit ab. Dies ermöglicht eine starke Verbesserung der Gesamtlaufzeit.

b) Beweisen Sie, dass parallelisiertes Quicksort unter der Annahme, dass der Median in konstanter Zeit bestimmt werden kann, ein Array der Größe n in Zeit $\mathcal{O}(n)$ sortiert.

Hinweis: Sie dürfen jeweils annehmen, dass die Größe n des Arrays A eine Zweierpotenz ist.

Aufgabe 5: **Dynamische Programmierung** *6 Punkte*

Seien $u, v, w \in \Sigma^*$ drei Wörter über einem Alphabet Σ. Wort w heißt *Mischung* der Wörter u und v, falls es die Zerlegungen $u = u_1 \dots u_k$ und $v = v_1 \dots v_l$ gibt, so dass $u_i, v_j \in \Sigma^*$ für $i = 1, 2, \dots, k$ und $j = 1, 2, \dots, l$ und $w = u_1 v_1 u_2 v_2 \dots$

Betrachten wir als Beispiel die Wörter $u = \mathit{INFORMATIK}$ und $v = \mathit{algorithmen}$. Dann ist das Wort $w = \mathit{INaFOlgoRMrithAmTIenK}$ eine Mischung der Wörter u und v.

a) Geben Sie einen Algorithmus an, der für drei gegebene Wörter u, v, w entscheidet, ob w eine Mischung von u und v ist. Verwenden Sie dabei **Dynamische Programmierung**.

b) Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus.

Aufgabe 6: **(3,6)-Bäume** *6 Punkte*

Betrachten Sie den untenstehenden (3,6)-Baum.

a) Fügen Sie die fehlenden Labels ein.

b) Fügen Sie die Schlüssel 41 und 14 in dieser Reihenfolge in den (3,6)-Baum ein und zeichnen Sie die resultierenden Bäume. Erläutern Sie Ihr Vorgehen kurz.

c) Sei T der Baum nach dem Einfügen von 41 und 13. Löschen Sie den Schlüssel 4 aus T und zeichnen Sie den resultierenden Baum. Erläutern Sie Ihr Vorgehen kurz.

d) Skizzieren Sie einen Linearzeit-Algorithmus, der einen gegebenen (2,4)-Baum in einen (3,6)-Baum umwandelt. Es genügt, wenn Sie die prinzipielle Idee dieses Algorithmus erklären!