Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast
Claudius Korzen

Algorithmen und Datenstrukturen
(Informatik II) SS 2015

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Klausur

Freitag 28. August 2015, 14:00 - 16:30 Uhr, HS 026 + HS 036 + SR 00/10+14

Es gibt *sechs* Aufgaben. Für jede davon gibt es maximal 20 Punkte. Wir zählen nur die *fünf besten* Aufgaben. Sie können also maximal 100 Punkte erreichen. Zum Bestehen reichen 50 Punkte.

Sie haben insgesamt 2 1/2 Stunden Zeit. Wenn Sie fünf Aufgaben bearbeiten, haben Sie im Durchschnitt 30 Minuten pro Aufgabe Zeit.

Sie dürfen eine beliebige Menge an Papier, Büchern, etc. verwenden. Sie dürfen keinerlei elektronische Geräte wie Notebook, Mobiltelefon, etc. verwenden, insbesondere keine Geräte, mit denen Sie mit Dritten kommunizieren oder sich mit dem Internet verbinden können.

Wir werden die Klausur am Montag, den 31. August 2015 korrigieren. Klausureinsicht ist dann gleich am darauffolgenden Tag (Dienstag, den 1. September 2015) um 14:00 Uhr, in Gebäude 51, 2. OG, Raum 02-028 (Büro Prof. Bast).

Bemerkung zu den Programmieraufgaben: Wann immer das Schreiben von Code gefragt ist, können Sie frei zwischen Python, Java und C++ wählen. Für kleinere, rein syntaktische Fehler, wie z.B. ein fehlendes Semikolon, gibt es keinen Punktabzug. Ansonsten sollten Sie aber schon vernünftigen Code schreiben.

Wichtig: alle Programmieraufgaben dieser Klausur lassen sich mit sehr wenig Code lösen. Keines Ihrer Programme sollte länger als 10 Zeilen sein, sonst gibt es Punktabzug oder gar keine Punkte. Dabei sollte in jeder Zeile nur eine Anweisung stehen, wie in der jeweiligen Sprache üblich.

Was Sie wie abgeben sollen: Schreiben Sie bitte oben rechts auf das Deckblatt der Klausur Ihren Namen und Ihre Matrikelnummer. Schreiben Sie Ihre Lösungen bitte auch auf die Klausur: benutzen Sie dabei für jede Aufgabe zuerst die Vorderseite und dann die Rückseite des entsprechenden Blattes. Wenn Sie zusätzliches Papier benötigen, schreiben Sie ebenfalls auf jedes dieser Blätter Ihren Namen und Ihre Matrikelnummer.

Wir wünschen Ihnen viel Erfolg!

Aufgabe 1 (Programmieren + Laufzeitbestimmung, 20 Punkte)

1.1 (10 Punkte) Schreiben Sie eine Funktion *secondLargest*, das die zweitgrößte Zahl in einem gegebenen Feld von Zahlen zurückgibt. Sie können annehmen, dass das Feld mindestens Größe 2 hat. Über die Reihenfolge der Zahlen in dem Feld ist nichts bekannt. Das Programm sollte in Linearzeit laufen. Für diese Funktion können Sie ausnahmsweise bis zu 15 Zeilen verwenden.

1.2 (5 Punkte) Die Fibonacci-Zahlen sind definiert durch $F_1 = 1$, $F_2 = 1$ und $F_n = F_{n-1} + F_{n-2}$ für $n \geq 3$. Sei T_n die Laufzeit von folgender Funktion zur Berechnung von F_n. Zeigen Sie über vollständige Induktion, dass $T_n \geq F_n$.

```
def fibonacci(n):
    if n <= 2:
        return 1
    else:
        f1 = fibonacci(n - 1)
        f2 = fibonacci(n - 2)
        return f1 + f2
```

1.3 (5 Punkte) Schreiben Sie eine Funktion *fastFibonacci*, die für eine gegebene ganze Zahl $n \geq 2$ die Zahl F_n in Zeit $O(n)$ berechnet. Mit Begründung, dass diese Laufzeit erreicht wird.

Aufgabe 2 (Felder + Hashing, 20 Punkte)

2.1 (5 Punkte) Schreiben Sie eine Funktion *removeFromArray*, die das erste Vorkommen einer gegebenen Zahl x in einem gegebenen (dynamischen) Feld von Zahlen sucht und löscht. Es soll das gegebene Feld modifiziert und kein neues erzeugt werden. Über die Reihenfolge der Zahlen in dem Feld ist nichts bekannt und die Reihenfolge kann aber muss nicht beibehalten werden.

2.2 (5 Punkte) Schreiben Sie eine Funktion *removeFromHashTable*, die eine gegebene Zahl x aus einer gegebenen Hashtabelle T mit gegebener Hashfunktion h löscht, falls vorhanden. Nehmen Sie dabei an, dass die Hashtabelle als Feld von Feldern implementiert ist. Sie können weiterhin annehmen, dass die Elemente nur aus Zahlen bestehen, ohne weitere Daten.

2.3 (5 Punkte) Bestimmen Sie die Laufzeit von Ihrer Funktion aus Aufgabe 2.1 und der Funktion aus Aufgabe 2.2 als $\Theta(...)$, jeweils im besten und im schlechtesten Fall.

2.4 (5 Punkte) Sei H eine Klasse von Hashfunktionen für eine Hashtabelle der Größe m und Schlüssel aus einem Universum U mit $|U| > m$. Zeigen Sie, dass die Klasse nicht 1-universell sein kann, wenn H weniger als m Hashfunktionen enthält.

Aufgabe 3 (Binäre Heaps + Anzahl Blockoperationen, 20 Punkte)

In dieser Aufgabe betrachten wir binäre Heaps, bei denen das kleinste Element oben steht und die in einem Feld gespeichert sind. Wie in der Vorlesung besprochen, sollte dabei Position 0 des Feldes stets leer gelassen werden. Das Feld ist somit immer eins größer als die Anzahl der im Heap gespeicherten Elemente.

3.1 (5 Punkte) Zeichnen Sie den Zustand eines binären Heaps nach Einfügen der Elemente: $5, 4, 3, 2, 1$ (in dieser Reihenfolge). Zeichnen Sie dazu sowohl den dazugehörigen Baum als auch das dazugehörige Feld nach jeder Einfügung.

3.2 (5 Punkte) Schreiben Sie eine Funktion *checkHeapProperty*, die einen als Feld gegebenen binären Heap daraufhin überprüft, ob die Heapeigenschaft überall erfüllt ist.

3.3 (5 Punkte) Bestimmen Sie die Laufzeit Ihrer Funktion aus Aufgabe 3.2 als $\Theta(...)$, mit Begründung.

3.4 (5 Punkte) Bestimmen Sie die Anzahl Blockoperationen Ihrer Funktion aus Aufgabe 3.2 als $\Theta(...)$, mit Begründung. Sie können annehmen, dass das Feld viel größer als die Größe M des schnellen Speichers ist, und dass M viel größer als die Blockgröße B ist.

Aufgabe 4 (Graphen, 20 Punkte)

Alle Graphen in dieser Aufgabe sind gerichtet und gewichtet (d.h. mit Kantenkosten).

4.1 (5 Punkte) Gegeben ein Graph mit drei Knoten u_1, u_2, u_3, drei Kanten (u_1, u_2), (u_2, u_3), (u_3, u_1) und Kosten 2 für alle Kanten. Fertigen Sie drei Zeichnungen an: den Graph, seine Darstellung durch eine Adjazenzmatrix und seine Darstellung durch Adjazenzlisten.

4.2 (5 Punkte) Schreiben Sie eine Funktion *numEdges*, die die Anzahl Kanten in einem durch Adjazenzlisten dargestellten Graphen zählt. Beschreiben Sie zuerst einen geeigneten Datentyp für diese Darstellung des Graphen und schreiben Sie dann die Funktion.

4.3 (5 Punkte) Wie Aufgabe 4.2, aber der Graph soll jetzt durch eine *Adjazenzmatrix* dargestellt sein.

4.4 (5 Punkte) Bestimmen Sie die Laufzeit Ihrer beiden Funktionen aus Aufgabe 4.2 und 4.3, jeweils als $\Theta(...)$ in Abhängigkeit von einer geeigneten mit dem Graph zusammenhängenden Größe.

Aufgabe 5 (Edi-Tier, 20 Punkte)

5.1 (5 Punkte) Bestimmen Sie die Edi-Tier-Distanz zwischen *REGAL* und *LAGER* unter Verwendung des iterativen Schemas aus der Vorlesung, indem Sie die dazugehörige Tabelle ausfüllen.

Für die folgenden drei Teilaufgaben seien x und y zwei Worte gleicher Länge, wobei y gerade das Wort x rückwärts ist und umgekehrt (wie für die beiden Worte aus Aufgabe 5.1 der Fall).

5.2 (5 Punkte) Angenommen, die Länge n von x und y ist *ungerade* und es werden n verschiedene Buchstaben verwendet (wie für die beiden Worte aus Aufgabe 5.1 der Fall). Argumentieren Sie, dass dann immer gilt: $\mathrm{ED}(x, y) = n - 1$.

5.3 (5 Punkte) Angenommen, die Länge n von x und y ist *gerade* und es werden n verschiedene Buchstaben verwendet. Wie ist dann der Wert von $\mathrm{ED}(x, y)$? Ebenfalls mit Begründung.

5.4 (5 Punkte) Angenommen jetzt, es können beliebige Buchstaben verwendet werden, aber mindestens zwei verschiedene. Wie ist dann für eine gegebene Länge n, egal ob gerade oder ungerade, $\mathrm{ED}(x, y)$ im niedrigsten Fall? Wieder mit Begründung.

Aufgabe 6 (O-Notation, Logarith-Mus, Wahrscheinlichkeit, 20 Punkte)

6.1 (5 Punkte) Zeigen Sie, dass $2n + 17 = O(n^2)$. Verwenden Sie dabei die Definition von O mittels C und n_0.

6.2 (5 Punkte) Berechnen Sie den Wert von $\log_2(2^{64} \cdot 8^{12})$.

6.3 (5 Punkte) Quicksort teilt in jedem Rekursionsschritt das gegebene Feld mit Hilfe eines Pivots in zwei Teile. Wie groß ist die Wahrscheinlichkeit, dass jeder Teil mindestens $n/4$ groß ist, wenn der Pivot zufällig gewählt wird und n die Größe des Feldes ist. Mit Begründung. Sie können annehmen, dass die Zahlen in dem Feld alle verschieden sind, n durch 4 teilbar ist und der Pivot zu keinem der beiden Teile gehört.

6.4 (5 Punkte) Wie groß ist die Wahrscheinlichkeit aus Aufgabe 6.3, wenn man drei zufällige Pivots wählt und den nimmt, der die beste (ausgeglichenste) Aufteilung macht. Sie können dabei annehmen, dass derselbe Pivot (wenn auch unwahrscheinlich) mehrfach gewählt werden kann. Wieder mit Begründung.