Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast
Axel Lehmann

Algorithmen und Datenstrukturen
(Informatik II) SS 2013

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

UNI FREIBURG

Klausur

Mittwoch 28. August 2013, 14:10 - 16:10 Uhr, Gebäude 082, Kinohörsaal

Es gibt *sechs* Aufgaben. Für jede Aufgabe 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 Stunden Zeit. Wenn Sie fünf Aufgaben bearbeiten, haben Sie im Durchschnitt 24 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 noch am selben Tag korrigieren. Klausureinsicht ist dann gleich am Donnerstag, 29. August 2013, um 14:00 Uhr, in Gebäude 51, 2. OG, Raum 02-028 (Büro Prof. Bast).

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 (Diverses, 20 Punkte)

1.1 (5 Punkte) Zeigen Sie, dass $\log n = O((\log n)^2)$, aber nicht $\log n = \Theta((\log n)^2)$. Argumentieren Sie dabei über die Definitionen mittels C und n_0.

1.2 (5 Punkte) Beschreiben Sie (möglichst prägnant), welche Eigenschaft von dem Eingabefeld A die folgende Methode berechnet und zurückgibt. Mit Begründung!

```
int f(int[] A) {
  int count = A.length;
  for (int i = 0; i < A.length; i++) {
    for (int j = 0; j < i; j++) {
      if (A[j] == A[i]) {
        count--;
        break;
      }
    }
  }
  return count;
}
```

1.3 (10 Punkte) Geben Sie die Laufzeit dieser Methode als $\Theta(\ldots)$ an, einmal im schlechtesten Fall und einmal im besten Fall. Jeweils mit Begründung!

Aufgabe 2 (Hashing, 20 Punkte)

2.1 (5 Punkte) Sei $h(x) = x^2 \bmod 5$. Zeichnen Sie den Zustand der zugehörigen Hashtabelle der Größe 5 nach dem Einfügen der Zahlen 1, ..., 10.

2.2 (5 Punkte) Beweisen Sie, dass sich die Werte dieser Hashfunktion mit Periode 5 wiederholen, das heißt $h(x + 5) = h(x)$, für alle $x \in \mathbb{N}$.

2.3 (5 Punkte) Berechnen Sie für jedes $i \in \{0, 1, 2, 3, 4\}$ die Wahrscheinlichkeit, dass $\Pr(h(x) = i)$ wenn $x \in \mathbb{N}$ zufällig gewählt wird.

2.4 (5 Punkte) Seien H_1 und H_2 zwei c-universelle Klassen von Hashfunktionen (für dasselbe c). Seien außerdem H_1 und H_2 disjunkt, das heißt sie haben keine Hashfunktion gemeinsam. Beweisen Sie, dass dann auch die Vereinigungsmenge $H_1 \cup H_2$ c-universell ist.

Aufgabe 3 (Dynamische Felder + IO-Effizienz, 20 Punkte)

3.1 (10 Punkte) Nehmen wir an, ein dynamisches Feld ist so implementiert, dass wenn vor dem Einfügen eines neuen Elementes das interne Feld voll ist (Kapazität = Anzahl Elemente), die Kapazität genau verdoppelt wird (gegenüber der Kapazität vor dem Einfügen des neuen Elementes). Zu Anfang sei das Feld leer (ohne Elemente), mit Kapazität 1. Wie groß ist die Kapazität dann direkt nach der i-ten Reallokation, für alle $i \in \mathbb{N}$? Beweisen Sie Ihre Formel über vollständige Induktion.

3.2 (10 Punkte) Wie groß, ist die Gesamtanzahl Blockoperationen, wenn man auf diese Weise n Elemente einfügt (und das Feld zu Beginn leer ist). Geben Sie die Anzahl als $\Theta(\ldots)$ an, in Abhängigkeit von n und der Blockgröße B.

Aufgabe 4 ((a, b)-Bäume, 20 Punkte)

4.1 (5 Punkte) Zeichnen Sie den Zustand eines $(2, 3)$-Baumes nach Einfügen der Elemente $1, 2, 3, 4, 5, 6$ (in dieser Reihenfolge) in einen zu Beginn leeren Baum.

4.2 (5 Punkte) Nehmen wir an, wir fügen jetzt weiter Elemente $7, 8, 9, 10, 11, \ldots$ ein. Wann bekommt der Baum dann zum ersten Mal Tiefe 3? Mit Begründung! Mit Tiefe ist dabei die Anzahl Kanten auf jedem Weg von der Wurzel zu einem Blatt gemeint.

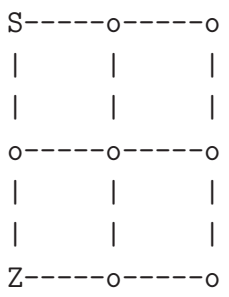
4.3 (5 Punkte) Und wann bekommt der Baum dann zum ersten Mal Tiefe 4? Mit Begründung.

4.4 (5 Punkte) Wie hoch sind, im schlechtesten Fall und als $\Theta(\ldots)$, die amortisierten Kosten für eine beliebige Folge von n Operationen (jeweils Einfügen oder Entfernen) auf einem zu Beginn leeren $(2, 3)$-Baum. Mit Begründung!

Aufgabe 5 (Graphen, 20 Punkte)

5.1 (10 Punkte) Führen Sie Dijkstras Algorithmus auf dem folgenden ungerichteten 3×3 Gittergraphen aus. Alle waagerechten (links-rechts) Kanten haben dabei Gewicht 1, alle senkrechten (oben-unten) Kanten haben Gewicht 2.

Wählen Sie als Startknoten oben links (S), und als Zielknoten unten links (Z). Führen Sie den Algorithmus so lange aus, bis er mit Sicherheit die Distanz zum Zielknoten kennt. Markieren Sie dabei *im Graphen* für jeden Knoten: (i) in welcher Runde er gelöst wurde, und (ii) in welcher Runde er welchen neuen Distanzwert erhalten hat. Achten Sie dabei bitte auf Übersichtlichkeit und benutzen Sie für (i) und (ii) verschiedene Farben.



5.2 (10 Punkte) Schreiben Sie (in Java oder in C++) eine Methode *computeDiameter(Graph G)*, die den Durchmesser eines gegebenen ungerichteten Graphen G berechnet.

Sie können dabei annehmen, dass Sie eine Methode *int[] computeDijkstra(Graph G, int s)* zur Verfügung haben, die für einen gegebenen ungerichteten Graphen G und einen gegebenen Knoten s die Distanzen zwischen s und allen anderen Knoten im Graphen berechnet.

Aufgabe 6 (String Matching, 20 Punkte)

6.1 (10 Punkte) Führen Sie für den Text `BANANANENA` und das Muster `NANA` den Algorithmus von Knuth-Morris-Pratt aus, d.h. verarbeiten Sie das Muster geeignet vor und suchen Sie dann mit Hilfe der vorberechneten Information nach den Vorkommen des Musters im Text. Benutzen Sie das folgende Raster für eine übersichtliche Darstellung.

6.2 (5 Punkte) Wie viele Vergleiche benötigt der triviale Algorithmus für diese Eingabe. Mit Begründung!

6.3 (5 Punkte) Geben Sie für alle $n, m \in \mathbb{N}$ mit $m \leq n$ einen Text der Länge n und ein Muster der Länge m an, so dass der triviale Algorithmus maximal viele Vergleiche benötigt. Mit Begründung!