Name:	Matrikelnummer:

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

Algorithmen und Datenstrukturen
(Informatik II) SS 2017

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

UNI FREIBURG

Klausur

Dienstag 29. August 2017, 14:00 - 16:30 Uhr, HS 00-026 + HS 00-036 + SR 00-010/14

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 selben Tag noch korrigieren. Klausureinsicht ist dann gleich am darauffolgenden Tag (Mittwoch, den 30. August 2017) um 15:00 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.

Wichtig: alle Programmieraufgaben dieser Klausur lassen sich mit 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. Einzige Ausnahme sind if *und* else *Anweisungen in Python: hier darf die Anweisung nach dem Doppelpunkt ausnahmsweise in derselben Zeile stehen.*

Was Sie wie abgeben sollen: Schreiben Sie bitte oben in die blaue Box 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!

A1	A2	A3	A4	A5	A6	Punkte	Note

Aufgabe 1 (O-Notation und Laufzeitbestimmung, 20 Punkte)

1.1 (5 Punkte) Zeigen Sie, dass $\sqrt{n} = O(n)$, aber nicht $\sqrt{n} = \Theta(n)$. Benutzen Sie dabei die Definitionen von O usw. mittels C und n_0. Sie dürfen also *nicht* über den Grenzwert argumentieren.

1.2 (5 Punkte) Vereinfachen Sie, mit Hilfe der gewöhnlichen Rechenregeln für den Logarithmus und für das Potenzieren, den Ausdruck $2^{\log_4 n}$ so weit wie möglich.

1.3 (10 Punkte) Nehmen wir an, ein Algorithmus löst ein Problem der Größe n, indem er es rekursiv in höchstens $A \cdot \sqrt{n}$ Zeit auf *zwei* Probleme der gleichen Art der Größe jeweils $n/4$ zurückführt. Stellen Sie eine rekursive Formel für die Laufzeit $T(n)$ auf und lösen Sie die Rekursion durch wiederholtes Einsetzen in sich selber auf. Am Ende sollte eine möglichst einfach nichtrekursive Formel für $T(n)$ in der Form $\Theta(...)$ in Abhängigkeit von n stehen. Sie können der Einfachheit halber annehmen, dass n eine Viererpotenz ist.

Aufgabe 2 (Programm, Laufzeitbestimmung, Hashing, 20 Punkte)

2.1 (5 Punkte) Schreiben Sie eine Funktion *num_unique*, die für eine gegebene Folge von n ganzen Zahlen (vom Typ *int*) die Anzahl der verschiedenen Zahlen ausgibt. Sie dürfen dabei die üblichen Datenstrukturen der Programmiersprache verwenden. Beispiel 1: Eingabe $5, 7, 5, 1, 1, 5$ $\rightarrow$ Ausgabe 3. Beispiel 2: Eingabe $1, 2, 3, 4, 2 \rightarrow$ Ausgabe 4.

2.2 (5 Punkte) Bestimmen Sie die Laufzeit von Ihrem Programm als $\Theta(...)$ in Abhängigkeit von n. Mit Begründung! Stellen Sie dabei klar, welche Annahmen Sie über die eingebauten Datenstrukturen der verwendeten Programmiersprache machen.

2.3 (5 Punkte) Nehmen Sie an, für eine Eingabe der Länge n wird jede Zahl der Eingabe zufällig aus der Menge $\{0, 1\}$ gewählt, unabhängig von den anderen Zahlen. Wie groß ist dann der Erwartungswert der Ausgabe in Abhängigkeit von n? Wie groß ist der Erwartungswert für $n = 3$? Mit Begründung!

2.4 (5 Punkte) Sei $U = \{0, ..., |U| - 1\}$ das Schlüsseluniversum und $m = |U|$ (das heißt, die Hashtabelle ist genauso groß wie das Universum). Sei H eine Klasse von Hashfunktionen, die aus genau einer Funktion besteht und zwar der Funktion h mit $h(x) = x$. Ist diese Klasse universell? Mit Begründung!

Aufgabe 3 (Programm, Laufzeitbestimmung, Potenzialfunktion, 20 Punkte)

3.1 (5 Punkte) Schreiben Sie eine Funktion *binary_inc*, die einen Binärzähler der Länge n, gegeben als Feld der Größe n mit Einträgen jeweils 0 oder 1, um eins hochzählt. Wenn der Zähler mit dem höchstmöglichen Wert (alle Einträge 1) übergeben wird, soll der Zähler auf den niedrigstmöglichen Wert (alle Einträge 0) zurückgesetzt werden.

Beispiel 1: Feld vor dem Aufruf: $[0, 0, 1, 1] \rightarrow$ Feld nach dem Aufruf: $[0, 1, 0, 0]$
Beispiel 2: Feld vor dem Aufruf: $[1, 1, 1, 0] \rightarrow$ Feld nach dem Aufruf: $[1, 1, 1, 1]$
Beispiel 3: Feld vor dem Aufruf: $[1, 1, 1, 1] \rightarrow$ Feld nach dem Aufruf: $[0, 0, 0, 0]$

3.2 (5 Punkte) Bestimmen Sie die Laufzeit eines Aufrufes Ihrer Funktion *binary_inc* als $\Theta(...)$ in Abhängigkeit von m, wobei m eine geeignete Größe ist, die von der Eingabe abhängt.

3.3 (5 Punkte) Bestimmen Sie die Gesamtlaufzeit von 2^n aufeinanderfolgenden Aufrufen Ihrer Funktion *binary_inc*, wenn in dem Feld beim ersten Aufruf alle Einträge 0 sind. Wählen Sie dazu eine geeignete Potenzialfunktion Φ und zeigen Sie, dass die Laufzeit T_i für die i-te Operation $\leq A + B \cdot (\Phi_i - \Phi_{i-1})$ ist, für irgendwelche Konstanten $A, B > 0$. Sie können das Mastertheorem aus der Vorlesung ohne Beweis verwenden.

3.4 (5 Punkte) Wie verändert sich die Gesamtlaufzeit für 2^n aufeinanderfolgende Aufrufe, wenn man mit einem anderen Zählerstand als in Aufgabe 3.3 beginnt? Mit Begründung!

Aufgabe 4 (Bäume, 20 Punkte)

4.1 (5 Punkte) Malen Sie einen vollständigen binären Baum der Tiefe 3 (= mit 7 Knoten), der die Heapeigenschaft erfüllt, aber *kein* binärer Suchbaum ist. Mit Begründung! Malen Sie für jedes Element nur den Schlüssel (Key). Die Werte (Values) spielen für die Aufgabe keine Rolle. Der kleinste Schlüssel soll oben im Baum stehen und die Schlüssel sollen alle verschieden sein.

Für die folgenden drei Teilaufgaben betrachten wir nun vollständige *ternäre* Bäume. Das sind Bäume, bei denen jeder innere Knoten genau *drei* Kinder hat und bei denen (wie beim vollständigen binären Baum auch) die Blätter alle auf derselben Tiefe liegen.

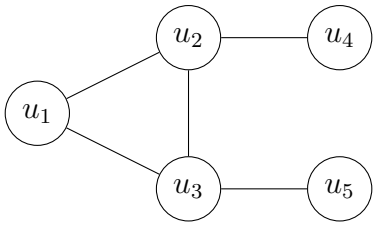
4.2 (5 Punkte) Malen Sie einen vollständigen ternären Baum der Tiefe 3 (= mit 13 Knoten). Die Darstellung sollte analog zu der eines vollständigen binären Baumes sein: alle Knoten einer bestimmten Tiefe sollen auf derselben (horizontalen) Ebene stehen, mit der Wurzel auf der obersten Ebene. Nummerieren Sie die Knoten mit den Zahlen $1, ..., 13$ von oben nach unten und von links nach rechts durch. (Schlüssel und Werte spielen für diese Aufgabe keine Rolle.)

4.3 (5 Punkte) Bestimmen Sie die Anzahl Knoten eines vollständigen ternären Baumes der Tiefe d, in Abhängigkeit von d. Mit Begründung! Die Formel sollte so einfach wie möglich sein. Hinweis: Sie dürfen ohne Beweis die Formel für die Summe einer geometrischen Reihe benutzen: $\sum_{i=0}^{n} q^i = (q^{n+1} - 1)/(q - 1)$.

4.4 (5 Punkte) Nehmen wir an, wir speichern die Elemente von einem ternären Baum in einem Feld, beginnend ab Indexposition 1 (nicht 0). Sei i der Index von einem Knoten, der nicht die Wurzel und kein Blatt ist. Wie sind dann (in Abhängigkeit von i) die Indizes der drei Kinder und wie der Index des Elternknotens? Es reichen die Formeln, ohne weitere Begründung.

Aufgabe 5 (Graphen, 20 Punkte)

In dieser Aufgabe geht es um zusammenhängende ungerichtete Graphen, abgekürzt ZUG. Hier ein Beispiel-ZUG mit fünf Knoten und fünf Kanten:



5.1 (5 Punkte) Bestimmen Sie die Adjazenzlisten für den Beispielgraphen oben (als Feld von Feldern). Gehen Sie davon aus, dass jede Kante in beide Richtungen eingetragen ist. Bestimmen Sie außerdem für jedes Paar von Knoten $\{u_i, u_j\}$ mit $i < j$ (das sind 10 Paare) die Länge des kürzesten Weges (hier gemessen in der Anzahl der Kanten) zwischen u_i und u_j. Es reicht, wenn Sie für jedes Paar die Länge hinschreiben, ohne weitere Begründung.

5.2 (10 Punkte) Schreiben Sie eine Funktion *dist*, die, gegeben die Adjazenzlisten eines ZUGs wie oben beschrieben und zwei Knoten u und v, die Länge (= Anzahl Kanten) des kürzesten Weges zwischen u und v zurückgibt. Hinweis: Sie brauchen hier nicht Dikjstra's Algorithmus zu implementieren, es geht viel einfacher mit einer Variante der Breitensuche.

5.3 (5 Punkte) Gegeben ein ZUG, seien u und v zwei Knoten mit einem kürzesten Pfad maximaler Länge (das heißt, kein anderer kürzester Pfad in dem Graphen ist länger). Wie ist dann die Laufzeit Ihrer Funktion *dist* aus Aufgabe 5.2 für diese u und v, als $\Theta(...)$ in Abhängigkeit von der Anzahl n der Knoten und der Anzahl m der Kanten des Graphen. Mit Begründung!

Aufgabe 6 (String-Suche, 20 Punkte)

6.1 (10 Punkte) Schreiben Sie eine Funktion *num_hits*, die die Anzahl der Vorkommen eines gegebenen Musters P in einem gegebenen Text T findet. Sie sollen dabei den Karp-Rabin Algorithmus verwenden, und zwar mit folgenden vereinfachenden Annahmen:

1. Die möglichen "Buchstaben" des Textes und des Musters sind die Ziffern 0..9
2. Der Text und das Muster sind jeweils als Feld von Ziffern (vom Typ *int*) gegeben
3. Das Muster hat genau Länge 3; insbesondere können Sie annehmen, dass das Muster und das jeweils aktuelle Textfenster in einen *int* passen, so dass Sie keine Hashfunktion benötigen

Beispiel: $T = [5, 3, 2, 8, 2, 8, 2, 8, 8]$, $P = [2, 8, 2] \rightarrow$ Ausgabe 2 (zwei Vorkommen)

6.2 (5 Punkte) Bestimmen Sie die Anzahl der Blockoperation Ihrer Funktion *num_hits* aus Aufgabe 6.1 als $\Theta(...)$ in Abhängigkeit von der Länge n des Textes und der Blockgröße B. Mit Begründung! Sie können dabei annehmen, dass für die Größe M des schnellen Speichers gilt, dass $M \gg B$.

6.3 (5 Punkte) Angenommen, jede Ziffer in dem Text T (der Länge n) und in dem Muster P (der Länge 3) ist zufällig gewählt, unabhängig von den jeweils anderen Ziffern. Wie groß ist dann die Anzahl der Vorkommen von P in T im Erwartungsfall, in Abhängigkeit von der Länge n des Textes? Insbesondere: wie groß ist der Erwartungswert für $n = 3$ und für $n = 1002$? Mit Begründung! Hinweis: führen Sie Indikatorvariablen I_i ein, so dass $I_i = 1$ genau dann, wenn das Muster an Stelle i matched.