Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn

Informatik II: Algorithmen & Datenstrukturen

Donnerstag, 28. August, 2014, 14:00 – 17:00

Name: .....

Matrikelnummer: .....

Unterschrift: .....

Blättern Sie nicht um bevor Sie dazu aufgefordert werden!

- Schreiben Sie **auf alle Blätter** (inklusive Deckblatt und etwaiger zusätzlicher Blätter) Ihren **Vornamen, Nachnamen** und Ihre **Matrikelnummer**.
- **Unterschreiben Sie das Deckblatt!** Ihre Unterschrift bestätigt, dass Sie alle Fragen ohne nicht erlaubte Hilfsmittel beantwortet haben.
- Schreiben Sie **lesbar** und nur mit **dokumentenechten** Stiften. Schreiben Sie nicht in **rot** oder **grün** und benutzen Sie keinen Bleistift!
- Alle schriftlichen Hilfsmittel (Bücher, Vorlesungsunterlagen, handschriftliche Notizen, etc.) sind erlaubt. **Elektronische Hilfsmittel sind nicht erlaubt.**
- Die Klausur besteht aus 7 Aufgaben und 120 Punkten. Zum Bestehen reichen 50 Punkte.
- Benutzen Sie für jede Aufgabe eine eigene Seite.
- Es wird nur eine Lösung pro Aufgabe gewertet. Vergewissern Sie sich, dass Sie zusätzliche Lösungen selbst entwerten. Falls mehrere Lösungen zu einer Aufgabe existieren, so wird die schlechtere Lösung gewertet.
- Die folgenden Regeln gelten überall, außer sie werden explizit außer Kraft gesetzt. Bei Laufzeitfragen ist wie üblich nur die asymptotische Laufzeit notwendig. Wenn Sie einen Algorithmus angeben sollen, so können Sie Pseudocode angeben, eine Beschreibung der Funktionsweise Ihres Algorithmus ist allerdings ausreichend. Algorithmen sind immer effizient zu konstruieren, d.h., mindestens polynomiell und i.d.r. schneller als eine naive Lösungsmethode. Algorithmen aus der Vorlesung können grundsätzlich als Blackbox verwendet werden.
- **Erklären Sie Ihre Lösungen, außer es wird explizit darauf hingewiesen, dass dies nicht nötig ist! Nur das Endresultat aufzuschreiben ist nicht ausreichend.**

Frage	1	2	3	4	5	6	7	**Total**
Punkte								
Maximum	26	8	13	22	17	16	18	120

Aufgabe 1: Kurze Fragen (26 Punkte)

Beantworten Sie die folgenden Fragen kurz.

(a) **(2 Punkte)** Was ist die Best-Case Laufzeit von Selection Sort (in $\mathcal{O}$-Notation, ohne Begründung).

(b) **(2 Punkte)** Welches der in der Vorlesung besprochenen Sortierverfahren verhält sich am Besten, wenn die Eingabe ein bereits sortiertes Array ist? (ohne Begründung)

(c) **(3 Punkte)** Eine Anwendung verwendet eine Hashtabelle so, dass zur einmaligen Initialisierung alle (Schlüssel,Wert)-Paare eingefügt werden und danach nur noch (sehr viele) *find*-Operationen ausgeführt werden. Welche Hashtabellen-Implementierung aus der Vorlesung eignet sich besonders für diese Anwendung und wieso?

(d) **(3 Punkte)** Bestimmen Sie die Editierdistanz von `Tier` und `Tor` und geben Sie die notwendigen Operationen an. Begründen Sie auch, wieso es nicht besser geht?

(e) **(4 Punkte)** Geben Sie einen effizienten Algorithmus an, um einen Spannbaum mit maximalem Gesamtgewicht zu berechnen (kurze Beschreibung genügt).

(f) **(4 Punkte)** Ihnen ist von einem großen Netzwerk eine Liste der Knoten gegeben, zusammen mit den Knotengraden. Ihre Aufgabe ist es, die Knoten des Netzwerks nach ihren Knotengraden zu sortieren. Welches Sortierverfahren aus der Vorlesung eignet sich am besten dazu (mit Begründung)?

(g) **(4 Punkte)** Beschreiben Sie, wie eine FIFO-Warteschlange *Queue* mit Hilfe einer Prioritätswarteschlange (Heap) implementiert werden kann.

(h) **(4 Punkte)** Gegeben sei ein Baum T mit n Elementen, welcher für die gleichen Schlüssel sowohl ein binärer Suchbaum, als auch ein Min-Heap ist (alle Schlüssel seien verschieden). Welche Höhe kann T haben?

Hinweis: Überlegen Sie sich zuerst an einem kleinen Beispiel, was passiert.

Aufgabe 2: Landau-Notation (8 Punkte)

Beweisen oder widerlegen Sie die folgenden Behauptungen durch Benutzen der exakten Definitionen von $\mathcal{O}(\cdot)$ und $\Omega(\cdot)$.

(a) **(4 Punkte)** $\sqrt{n+7} \in \mathcal{O}(\sqrt{n})$.

(b) **(4 Punkte)** $\log(n^2) \in \Omega\big((\log n)^2\big)$.

Aufgabe 3: Mystische Multiplikationen (13 Punkte)

Gegeben sei ein Integer Array A der Länge n und folgender Code:

```
boolean myst(int[] A) {
   int n = A.length;
   for (int i = 0; i < n-1; i++) {
      for (int j = i+1; j < n; j++) {
         for (int k = 0; k < n; k++) {
            if (A[i] * A[j] == A[k]) {
               return true;
            }
         }
      }
   }
   return false;
}
```

(a) **(4 Punkte)** Was berechnet die Funktion *myst* (bzw., in welchen Fällen gibt *myst* true zurück)?

(b) **(3 Punkte)** Welche (asymptotische) Laufzeit hat *myst(int[] A)* im Worst Case und im Best Case (in Abhängigkeit von n, ohne Begründung)?

(c) **(6 Punkte)** Geben Sie, z.B. unter Zuhilfenahme einer geeigneten Datenstruktur, einen Algorithmus mit gleicher Ausgabe wie *myst* an, dessen *asymptotische* Laufzeit im Worst Case jedoch strikt besser als diejenige von *myst* ist. Was ist die Laufzeit Ihres Algorithmus?

Aufgabe 4: Binäre Suchbäume (22 Punkte)

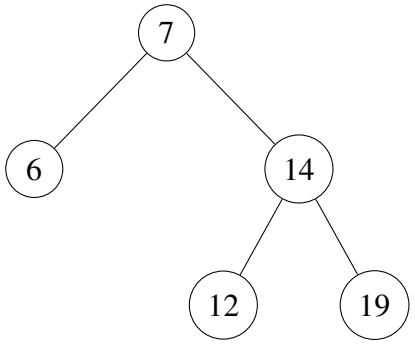
(a) **(6 Punkte)** Führen Sie nacheinander die nachfolgenden 8 Einfüge- und Lösch-Operationen auf den unten angegebenen binären Suchbaum aus und zeichnen Sie den resultierenden Baum nach **jeder Gruppe** von Operationen.

1) *insert(23)*, *delete(19)*

2) *insert(13)*, *insert(10)*, *insert(4)*, *insert(18)*

3) *delete(6)*

4) *delete(14)*



(b) **(6 Punkte)** Können die folgenden Folgen in einer *Suchabfrage* in einem binären Suchbaum auftreten? Zeichnen Sie einen Beispielbaum auf, wenn die Antwort "ja" lautet, andernfalls begründen Sie Ihre Antwort.

- Folge 1: $44, 12, 15, 37, 19, 29, 28$
- Folge 2: $53, 64, 83, 75, 69, 60, 66$

(c) **(10 Punkte)** Man nennt einen binären Suchbaum T einen *AVL-Baum*, wenn für *jeden* Knoten u gilt, dass die Höhen der beiden in u.`left` und u.`right` gewurzelten Teilbäume sich um maximal 1 unterscheiden. Beschreiben Sie eine Methode *isAVL* als Pseudocode, welche einen binären Suchbaum T auf diese Eigenschaft testet. Was ist die Laufzeit Ihres Algorithmus, und warum?

Hinweis: Schreiben Sie zuerst eine rekursive Methode avlHeight(v)*, welche, angewandt auf einen Knoten v, die Höhe des in v gewurzelten Teilbaums wiedergibt* falls *der Teilbaum selbst ein AVL-Baum ist, und -1 sonst.*

Aufgabe 5: Divide & Conquer, Sortieren (17 Punkte)

(a) **(5 Punkte)** Gegeben sei ein Integer-Array A der Länge n. Finden Sie einen Divide & Conquer Algorithmus, der das Maximum der Elemente in A findet, ohne das Array zu sortieren. Geben Sie die Rekursionsgleichung für die Laufzeit an – diese muss *nicht* gelöst werden.

(b) **(12 Punkte)** Gegeben sei ein aufsteigend sortiertes Integer-Array B mit n Elementen. Nun werden k beliebige Elemente ($k \ll n$) individuell verringert (man weiß nicht, welche k Elemente verändert wurden), so dass das resultierende Array im Allgemeinen nicht mehr sortiert ist.

Geben Sie einen Algorithmus an, der das Array in Zeit $\mathcal{O}(n + k \log k)$ erneut in einen sortierten Zustand bringt.

Hinweis: Finden Sie zuerst alle Elemente, welche dadurch, dass sie heruntergesetzt wurden, die Sortierreihenfolge zerstören (wenn man diese Elemente entfernt, sind die restlichen Elemente richtig sortiert). Beachten Sie, dass das höchstens k *Elemente sind, aber nicht exakt* k *Elemente sein müssen.*

Aufgabe 6: Graphen & Graphtraversierung (16 Punkte)

Gegeben sei ein *gerichteter* Graph $G = (V, E)$. Herausgefunden werden soll, ob G einen Kreis enthält. Ein Kreis ist eine Knotenfolge $u_1, u_2, \ldots, u_k$, so dass $(u_i, u_{i+1}) \in E$ für alle $1 \leq i < k$, sowie $(u_k, u_1) \in E$. Es gelte $k \geq 2$, d.h., es kann Kreise der Länge 2 geben.

(a) **(4 Punkte)** Erläutern Sie einen Algorithmus, welcher möglichst effizient herausfindet, ob ein gegebener gerichteter Graph G einen Kreis enthält.

(b) **(2 Punkte)** Welche Datenstruktur würden Sie verwenden, um den Graph zu repräsentieren und warum: Adjazenzlisten oder Ajazenzmatrix?

(c) **(10 Punkte)** Geben Sie Pseudocode für Ihren Algorithmus an.

Aufgabe 7: Kürzeste Wege zur Arbeit (über den Berg) (18 Punkte)

Geben Sie jeweils einen Algorithmus an, um die folgenden Probleme zu lösen (dabei dürfen in der Vorlesung behandelte Algorithmen als Blackbox verwendet werden):

Gegeben sei ein Straßennetz als ein ungerichteter, gewichteter Graph $G = (V, E, w)$, $n = |V|$, $m = |E|$.

(a) **(2 Punkte)** Sie wohnen in Punkt $a \in V$ und arbeiten in Punkt $b \in V$. Wie berechnen Sie den kürzesten Weg zur Arbeit?

Geben Sie außerdem die asymptotische Laufzeit Ihres Algorithmus (ohne Begründung) an.

(b) **(4 Punkte)** Sie möchten umziehen in eine neue Wohnung $a' \in W$, wobei $W \subseteq V$ eine Menge von potentiellen Wohnungen ist. Wie finden Sie a', wenn diese von allen Wohnungen in W den kürzesten Weg zur Arbeit $b \in V$ haben soll?

Geben Sie außerdem die asymptotische Laufzeit Ihres Algorithmus an (ohne Begründung).

(c) **(12 Punkte)** Sie haben sich wegen der schönen Lage für eine andere Wohnung $c \in V$ mit Fahrradnähe zu $b \in V$ entschieden. Jeder Knoten v im Graphen hat eine Höhe $h_v \in \mathbb{R}$ (der Einfachheit halber seien alle Knotenhöhen voneinander verschieden) und je nachdem, in welcher Richtung man über eine Kante $e = \{u, v\}$ geht, ist diese eine Bergaufkante oder eine Bergabkante. Aus unbekanntem Grund wollen Sie von Ihrer Wohnung c aus zunächst ausschließlich bergauf fahren – bis zu einem Punkt $x \in V$ – und von dort aus ausschließlich bergab bis zum Punkt b.

Beschreiben Sie einen (effizienten) Algorithmus, welcher herausfindet, ob so ein Weg existiert und welcher bei Existenz den kürzesten aller solchen Wege, inklusive des höchsten Punkts x, zurückgibt. Welche Laufzeit hat Ihr Algorithmus (ohne Begründung)?

Hinweis: Verfahren aus der Vorlesung können als Blackbox verwendet werden. Beschreiben Sie Ihren Algorithmus nur, zu hohe Genauigkeit (z.B. via Pseudocode) kostet Sie zu viel Zeit.