UNIVERSITÄT SALZBURG
FB Computerwissenschaften

VO Algorithmen und Datenstrukturen
Robert Elsässer

Probeklausur (12. Juni 2013)

Name: .... Matr.Nr.: ....

Beantworten Sie die folgenden Fragen sauber und leserlich auf den ausgegebenen Klausurbögen. Von Ihren Unterlagen dürfen Sie ein mitgebrachtes doppelseitig handbeschriebenes Blatt verwenden. Darüberhinausgehende Verwendung von Unterlagen oder Geräten wie Smartphones ist nicht gestattet.

Teil 1: Multiple Choice *10 P*

Bei den folgenden Fragen ist jeweils genau eine Antwort zu wählen. Falsche Antworten führen zu Punkteabzügen. Die Variablen V, E, n, k haben die in der Vorlesung definierte Bedeutung.

Frage 1.a: Gegeben sind die folgenden Funktionen: ☐ ja ☐ nein

$f(n) = n^2$ $g(n) = n \log n$ $h(n) = \sqrt{n}$

Gilt $f \in O(g) \vee h \in \Omega(g) \vee f \in \Theta(h)$?

Frage 1.b: Quicksort ist nie asymptotisch langsamer als Mergesort. ☐ ja ☐ nein

Frage 1.c: Jeder Sortieralgorithmus hat eine worst-case Laufzeit von $\Omega(n \log n)$. ☐ ja ☐ nein

Frage 1.d: Matrix-Multiplikationen benötigen immer Zeit $\Omega(n^3)$ ☐ ja ☐ nein

Frage 1.e: Mit Hilfe einer Tiefensuche kann für einen ungerichteten Graphen festgestellt werden, ob der Graph zusammenhängend ist. ☐ ja ☐ nein

Frage 1.f: Counting-Sort hat eine Laufzeit in

☐ $\Theta(n)$ ☐ $\Theta(n+k)$ ☐ $\Theta(n \cdot k)$ ☐ $\Theta(n \log n)$ ☐ $\Theta(n \log n + k)$

Frage 1.g: Build-Max-Heap hat eine Laufzeit von

☐ $\Theta(n \log n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(n)$ ☐ $\Theta(\log n)$ ☐ $\Theta(n^{\log n})$

Frage 1.h: Ein AVL-Baum lässt sich nach dem Einfügen eines neuen Elements balanzieren in worst case Zeit von

☐ $\Theta(1)$ ☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(\sqrt{n})$

Frage 1.i: Der Algorithmus von Strassen hat eine Laufzeit von

☐ $O(n^{2.7})$ ☐ $O(n^{\log_2 7})$ ☐ $O(n^3)$ ☐ $O(n \log n)$ ☐ $O(n^3 + m \log^2 m)$

Frage 1.j: Eine Breitensuche in einem Graphen benötigt Zeit

☐ $O(|V|)$ ☐ $O(|E|)$ ☐ $O(|V| \log(|V|))$ ☐ $O(|V| + |E|)$ ☐ $O(|V| \cdot |E|)$

Teil 2: Sortieren *5 P*

Frage 2.a: Verwenden Sie Quicksort wie in der Vorlesung vorgestellt, um folgendes Array zu sortieren:

9	3	8	2	1	4

Frage 2.b: Angenommen, Sie bekommen bei Quicksort die Garantie, dass die Wahl Ihres Pivot-Elementes immer zu einer Partitionierung in zwei Teile führt, sodass beide Teile kleiner als ein konstanter Faktor α der ursprünglichen Größe sind. Beweisen Sie, dass für ein konstantes α mit $0.5 \leq \alpha < 1$ dieses Verfahren immer eine Laufzeitkomplexität in $O(n \log n)$ hat.

Teil 3: Datenstrukturen *5 P*

Frage 3.a: Geben Sie eine Datenstruktur an, die folgende Eigenschaften mit sich bringt:

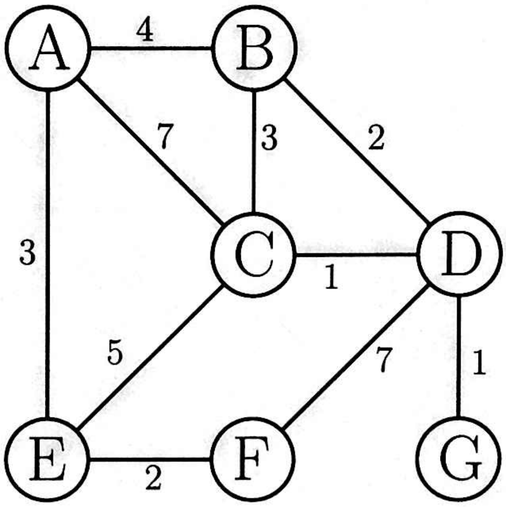
- Der Zugriff soll in erwartet konstanter Zeit $O(1)$ möglich sein.
- Einfügen und Löschen soll erwartet in konstanter Zeit $O(1)$ möglich sein.
- Alle Operationen sollen im worst case $O(\log n)$ Zeit benötigen.

Gehen Sie davon aus, dass die einzufügenden Daten einer Gleichverteilung unterliegen.

Frage 3.b: Begründen Sie die Korrektheit Ihrer Lösung!

Teil 4: Graphentheoretische Algorithmen *5 P*

Gegeben ist folgender Graph:



Frage 4.a: Verwenden Sie Prims Algorithmus, um für den Graphen einen minimalen Spannbaum zu ermitteln.

Frage 4.b: Skizzieren Sie den Korrektheitsbeweis für Prims Algorithmus.

Teil 5: O-Kalkül *5 P*

Für Funktionen $f, g : \mathbb{N} \to \mathbb{R}^+$ lässt sich ähnlich der Klasse $O(f)$ auch $o(f)$ (*Klein-O*) definieren. Informell bedeutet $f \in o(g)$, dass für beliebige positive konstante k die Ungleichung $f(n) \leq k \cdot g(n)$ ab einem gewissen n gilt.
Formal können wir die Menge von Funktionen $o(f)$ wie folgt definieren:

$$o(f) = \{g : \mathbb{N} \to \mathbb{R}^+ | \forall k > 0 \ \exists n_0 \in \mathbb{N} \ \forall n > n_0 \quad g(n) \leq k \cdot f(n)\} \ .$$

Es gilt:

$$\lim_{n \to \infty} \frac{g(x)}{f(x)} = 0 \Leftrightarrow g \in o(f)$$

Frage 5.a: Verwenden Sie dies, um zu Zeigen, dass für beliebige konstante $k \in \mathbb{N}$ gilt:

$$\log^k(n) \in o(n)$$