Demo-Klausur “Algorithmen und Datenstrukturen”

Prof. Dr. Ulrike von Luxburg,

Wichtige Informationen (zur Demo-Klausur):

Diese Demo-Klausur bietet *keine* repräsentative Themenauswahl. Die Aufgaben sind *nicht* gründlich darauf überprüft, ob ihr Lösungsaufwand angemessen ist. Trotzdem bietet sie einen Einblick in verschiedene Typen von Fragen, wie sie genau so oder ähnlich in der echten Klausur vorkommen könnten.

Wichtige Informationen (zur echten Klausur):

- Sie dürfen einen von Ihnen einseitig handschriftlich beschriebenen A4-Zettel (Rückseite leer bis auf Ihren Namen) mit zur Klausur bringen. Dieser wird natürlich nicht bewertet, er muss aber mit der Klausur abgegeben werden!
- Sie müssen sich zur Klausur angemeldet haben. Falls das aus irgendeinem Grund schiefgegangen ist, müssen Sie das von uns mitgebrachte Formular “Klausurteilnahme unter Vorbehalt” ausfüllen und mit abgeben.
- Bitte legen Sie Ihren Personalausweis und Studierendenausweis auf den Tisch.
- Die Gesamtbearbeitungszeit beträgt 120 min.
- Die Klausur umfasst mehr Fragen, als Sie zu lösen brauchen! **Sie brauchen nur 10 der 13 Fragen zu beantworten. Wenn Sie mehr als 10 Fragen bearbeiten, dann werden wir nur die besten 10 davon in der endgültigen Summe berücksichtigen.** Beispiel: Wenn die folgenden Punkte erreicht werden, dann werden die unterstrichenen NICHT in der finalen Summe berücksichtigt: $\underline{1}$, 3, $\underline{0}$, $\underline{0}$, 2, 2, 4, 2, 3, 2, 1, 3, 2.

  Daher sollten Sie sich auf genau 10 Fragen konzentrieren, und die übrigen nur dann angehen, wenn Sie wirklich noch Zeit haben.
- Überprüfen Sie, ob Ihre Klausur alle 8 Seiten umfasst.
- Schreiben Sie auf jedes Blatt Ihren Namen. Benötigen Sie weiteres Papier, so erhalten Sie dieses von uns. Benutzen Sie kein eigenes Papier.
- Schreiben Sie nicht mit Bleistift.
- Handys ganz ausschalten und wegpacken, Taschen/Rucksäcke unter den Tisch.

Viel Erfolg!

1. Single Choice - Querbeet (4 Punkte)

Kreuzen Sie für jede Aussage an, ob diese richtig oder falsch ist.
Für jede korrekte Teilaufgabe erhalten Sie +0.5 Punkte, für jeden Fehler -0.5 Punkte, für jede Enthaltung 0 Punkte.
Insgesamt kann diese Aufgabe nicht weniger als 0 Punkte erzielen.

	Aussage	*richtig*	*falsch*
(a)	Die Queue ist eine FIFO-Datenstruktur.	☐	☐
(b)	BubbleSort hat die worst-case Laufzeit $\mathcal{O}(\sqrt{n})$.	☐	☐
(c)	Heap-Sort hat stets für jede Eingabe der Länge n Laufzeit $\Theta(n \log n)$.	☐	☐
(d)	Jeder Graph hat mindestens eine Kante.	☐	☐
(e)	In jedem ungerichteten Baum ist die Länge eines längsten kürzesten Pfades gleich dem Doppelten seiner Höhe.	☐	☐
(f)	Ein stark zusammenhängender, gerichteter Graph enthält mindestens einen Zyklus.	☐	☐
(g)	Ein Sortieralgorithmus mit worst-case Laufzeit $\mathcal{O}(n \log n)$ kann nicht zugleich stabil sein.	☐	☐
(h)	Heap-Sort ist ein stabiler Suchalgorithmus.	☐	☐

2. Landau-Notation (1 + 3 Punkte)

(a) Zum Vergleich des asymptotischen Wachstums zweier Funktionen gibt es die Menge von Relationen $L := \{\Theta, \mathcal{O}, \omega, o, \Omega\}$. Zum Vergleich zweier reeller Zahlen gibt es die Menge von Relationen $R := \{\leq, =, \geq, <, >\}$. Ordnen sie jede Relation aus L einer Relation aus R zu, so dass ihre Semantik bestmöglich beibehalten wird.

(b) Entscheiden Sie für jedes der folgenden Funktionenpaare f und g, in welcher asymptotischen Beziehung sie zueinander stehen. Kreuzen Sie dabei jedes (!) Kästchen an, das eine wahre Aussage darstellt.

f	g	$f \in o(g)$	$f \in \mathcal{O}(g)$	$f \in \Theta(g)$	$f \in \Omega(g)$	$f \in \omega(g)$
n^3	n^2	☐	☐	☐	☐	☐
n	$n^{1.01}$	☐	☐	☐	☐	☐
$\sqrt{7n} - \log n$	$n^{0.5}$	☐	☐	☐	☐	☐

3. Master-Theorem (2 + 2 Punkte)

(a) Lösen Sie mit Hilfe des Master-Theorems die folgenden beiden Rekursionen für $n \geq 1$ unter der Annahme, dass $T(1) = 1$.

$$T(n) \; = \; 100 \cdot T(\lceil n/10 \rceil) + n \qquad\qquad T(n) \; = \; 3 \cdot T(\lceil n/3 \rceil) + n^{3/2}$$

$$\Rightarrow \; T(n) \; \in \; \mathcal{O}(\underline{\hspace{3cm}}) \qquad\qquad \Rightarrow \; T(n) \; \in \; \mathcal{O}(\underline{\hspace{3cm}})$$

(b) Betrachten Sie erneut die rekursive Berechnung der Fibonacci-Zahlen:

```
function Fib(n)
    if n ≤ 1 then
        return n
    else
        return Fib(n − 1) + Fib(n − 2)
    end if
end function
```

Kann hierbei zur Berechnung der Laufzeit das Master-Theorem angewandt werden? Begründen Sie kurz.

4. Sortierverfahren (2 + 2 Punkte)

(a) Sortieren Sie die Eingabe $\boxed{3}\boxed{7}\boxed{2}\boxed{6}\boxed{4}\boxed{1}\boxed{8}\boxed{5}$ mit einem Sortierverfahren Ihrer Wahl. Geben Sie sinnvolle Zwischenschritte an, die Ihr Vorgehen verdeutlichen (zum Beispiel für jede Rekursionstiefe die aktuelle Sortierung).

(b) Was ist mit der schwammigen Aussage "*Kein rein vergleichsbasiertes Sortierverfahren kann schneller als* $n \log n$ *sein.*" gemeint? Formulieren sie die hier nur angedeutete Aussage präzise.

5. Kürzeste Wege mit Floyd Warshall (1 + 1 + 1 Punkte)

(a) Nennen Sie das Problem, welches der Floyd-Warshall-Algorithmus löst.

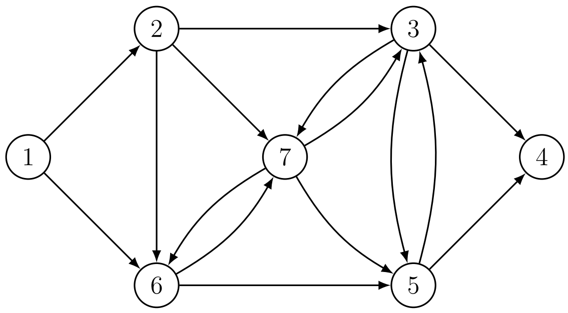
(b) Beschreiben Sie den Floyd-Warshall-Algorithmus in Pseudo-Code.

(c) Was ist seine Laufzeit? Begründung!

6. Breitensuche (2 + 2 Punkte)

(a) Ersetzen Sie im Pseudo-Code der Breitensuche die Queue durch einen Stack. Wie verhält sich der Algorithmus nun?

(b) Führen Sie eine Breitensuche in folgendem Graphen gestartet an 1 aus:



In welcher Reihenfolge werden die Knoten erstmals besucht (= grau gefärbt)? Sie können dabei die Ordnung der Einträge in den Adjanzlisten beliebig wählen.

7. Dynamisches Programmieren (4 Punkte)

Die "k-Schritt Fibonacci-Zahlen" sind für ein festes $k \geq 1$ rekursiv definiert über:

$$F_n := \begin{cases} 0 & n \leq 0 \\ 1 & n = 1 \\ \sum_{i=1}^{k} F_{n-i} & n \geq 2 \end{cases}$$

Schreiben Sie den Pseudo-Code einer Funktion $\textsc{Fib}_k(n)$, die sich diese Struktur mittels dynamischer Programmierung ("bottom-up") zu Nutze macht, um F_n effizient (in Polynomialzeit) zu berechnen.