Algorithmen und Datenstrukturen 2

WiSe 2020 · Prof. Dr. U. Meyer
30. März 2021
Klausurnummer: **0001**

GOETHE UNIVERSITÄT
FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

Studiengang: _____

Bitte Hinweise beachten:

- Schreiben Sie Ihren **Namen *nur* auf dieses Titelblatt**. Sollten Sie Zusatzpapier bekommen, vermerken Sie darauf unbedingt die Klausurnummer **0001**.
- Die Klausur besteht aus den Aufgaben **1 − 10**.
- Merken oder notieren Sie sich Ihre Klausurnummer **0001**, da nur unter dieser Nummer die Ergebnisse veröffentlicht werden.
- Es dürfen nur **dokumentenechte Stifte** in den Farben blau und schwarz verwendet werden. Insb. ist die Nutzung von Tintenlöschern, Tipp-Ex u. Ä. untersagt.
  Zugelassene Hilfsmittel:
  1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
- Das Mitbringen nicht zugelassener Hilfsmittel stellt einen Täuschungsversuch dar und führt zum Nichtbestehen der Klausur. **Schalten Sie bitte deshalb alle elektronischen Geräte, insbesondere Handys und Smartwatches, vor Beginn der Klausur aus und packen Sie diese weg.**
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung. Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies verlangt.
- Die Klausur ist mit Sicherheit bestanden, wenn (ohne Bonifikation aus den Übungspunkten) mindestens **50%** der Höchstpunktzahl erreicht wird.
- Die Klausur dauert **180 Minuten**.

1 0001202 200013

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Diese Seite ist nur für den internen Gebrauch bestimmt.
Bitte nicht beschreiben.

Aufgabe	1	2	3	4	5	6	7	8	9	10
Erreichbar	6	6	7	8	12	14	12	13	11	11
Erreicht										

Klausur	Bonfikation

1 0 0 2 2 0 2 2 0 0 0 1 0

Aufgabe 1: Sortieren spezieller Eingaben — $2 + 2 + 2 = 6$ Punkt(e)

Betrachten Sie die folgenden Sequenzen der Länge n bestehend aus natürlichen Zahlen. Geben Sie **in Abhängigkeit der Länge n und dem Parameter x**, mit $1 < x < n$, asymptotisch exakt an, wie lange die entsprechenden Algorithmen zum Sortieren der Sequenz benötigen.

Anmerkung: Bubblesort terminiert nach einem Durchlauf ohne Vertauschung.

a)

$$x, 1, \ldots, x-1, \quad x+1, \ldots, n$$

d.h. die Folge ist sortiert bis auf das Element x, das sich am Anfang befindet.

Bubblesort: $\Theta\Big($ _____ $\Big)$

Insertionsort: $\Theta\Big($ _____ $\Big)$

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b)

$$1, \ldots, x-1, \quad x+1, \ldots, n, x$$

d.h. die Folge ist sortiert bis auf das Element x, das sich am Ende befindet.

Bubblesort: $\Theta\Big($ _____ $\Big)$

Insertionsort: $\Theta\Big($ _____ $\Big)$

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

c)

$$x, \ldots, 1, \ 2 \cdot x, \ldots, 1 + x, \ \ldots \ , n, \ldots, n + 1 - x$$

d.h. absteigend sortierte Teilfolgen der Länge x, für ein x, sodass $n = x \cdot k$ für ein $k \in \mathbb{N}$ gilt.

Bubblesort: $\Theta\Big($ _____ $\Big)$

Insertionsort: $\Theta\Big($ _____ $\Big)$

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Aufgabe 2: Sortieren kleiner Werte — 6 Punkt(e)

Gegeben seien n paarweise verschiedene, gekürzte Brüche der Form $\frac{a}{b}$ mit $a \in \{1, 2, 3, 4, 5\}$ und $b \in \mathbb{N}_{>0}$.

Die Brüche sind als Paare der Form (a, b) gegeben.

Geben Sie eine Transformation sowie, falls nötig, eine möglichst großzügige Einschränkung für b an, sodass eine solche Sequenz mit Radixsort inkl. der Transformation in Zeit $\mathcal{O}(n)$ sortiert werden kann.

Hinweis: Nehmen Sie an, dass arithmetische Operationen ($+$, $-$, $\cdot$, $/$) in konstanter Zeit ausgeführt werden können.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

Aufgabe 3: Sortieralgorithmen verbessern — $2 + 3 + 2 = 7$ Punkt(e)

Die Laufzeit von Quicksort hängt maßgeblich von der Wahl des Pivotelements ab. Wir haben in der Vorlesung gesehen, dass sie im Worst-Case bei $\Theta(n^2)$ liegt. Selbst bei einer zufälligen Wahl des Pivotelements bleibt die Laufzeit im Worst-Case quadratisch.

Wir wollen nun wie folgt versuchen, deterministisch ein gutes Pivotelement zu finden:

`getPivot`(Array A)

1. Teile A in disjunkte Blöcke zu je $\sqrt{n}$ Elementen auf (der letzte Block muss nicht voll sein) und sortiere jeden Block mit Bubblesort.
2. Wähle aus jedem Block das mittlere Element aus und bilde mit diesen ein Array A'.
3. Sortiere A' mit Bubblesort.
4. Gib das mittlere Element von A' als Pivot zurück.

a) Welche Worst-Case-Laufzeit hat `getPivot`, wenn das Array A die Länge n hat?

$\Theta($ _____ $)$

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b) Geben Sie eine möglichst gute, nicht-asymptotische untere Schranke für die Anzahl der Elemente aus A an, die höchstens so groß wie das durch `getPivot` gewählte Element sind. Begründen Sie Ihre Antwort kurz. Gehen Sie davon aus, dass alle Elemente paarweise verschieden sind.

Hinweis: Sie können davon ausgehen, dass $\sqrt{n}$ eine ganze Zahl ist.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Geben Sie eine Rekursionsgleichung für die Worst-Case-Laufzeit von Quicksort an, wenn für die Pivotwahl `getPivot` verwendet wird. Sie müssen die Rekursionsgleichung nicht ausrechnen.

Hinweis: Sie können $f(n)$ für die Worst-Case-Laufzeit von `getPivot` und a für den Anteil der Elemente, die höchstens so groß wie das gewählte Pivotelement sind, verwenden.

$T(n) =$ _____

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Aufgabe 4: Entscheidbarkeit — 8 Punkt(e)

Seien L_1 und L_2 rekursiv aufzählbare Sprachen und sei

$$L_1 \circ L_2 = \{uv \,:\, u \in L_1,\, v \in L_2\}$$

die Sprache der konkatenierten Wörter aus L_1 und L_2.

Zeigen oder widerlegen Sie: $L_1 \circ L_2$ ist rekursiv aufzählbar.

⇑⇑⇑ _____ / 8 Punkt(e) ⇑⇑⇑

Aufgabe 5: Turingmaschine — $1 + 4 + 5 + 2 = 12$ Punkt(e)

Gegeben sei die folgende Turingmaschine $M = (Q, \Sigma, \delta, q_0, \Gamma, F)$ mit

$$
\begin{aligned}
Q &= \{q_0, q_1, q_2\} \\
\Sigma &= \{0, 1\} \\
\Gamma &= \{B\} \cup \Sigma \\
F &= \{q_2\}
\end{aligned}
$$

und der Übergangsfunktion δ definiert als

$$
\begin{aligned}
(q_0, B) &\mapsto (q_2, B, rechts) \\
(q_0, 1) &\mapsto (q_1, 1, rechts) \\
(q_1, 1) &\mapsto (q_2, 1, rechts) \\
(q_2, 1) &\mapsto (q_1, 1, rechts) \\
(q_2, 0) &\mapsto (q_1, 0, bleib)
\end{aligned}
$$

a) Geben Sie ein Wort $w \in \Sigma^*$ an, das von M akzeptiert wird.

$w =$ _____

⇑⇑⇑ _____ / 1 Punkt(e) ⇑⇑⇑

b) Welche Sprache $L \subseteq \Sigma^*$ wird von M akzeptiert?

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

c) Hält M auf allen Eingaben? Begründen Sie Ihre Antwort und geben Sie ggf. eine Eingabe an, auf der M nicht hält.

⇑⇑⇑ _____ / 5 Punkt(e) ⇑⇑⇑

d) Gibt es Eingaben, auf denen M in einem nicht-akzeptierenden Zustand hält? Geben Sie ein Beispiel an oder begründen Sie, warum es keine solche Eingabe gibt.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Aufgabe 6: Scheduling — $3 + 4 + 7 = 14$ Punkt(e)

Gegeben seien drei identische Maschinen M_1, M_2 und M_3 und sechs Aufgaben $1, \ldots, 6$ mit den Laufzeiten $t_1, \ldots, t_6$:

i	1	2	3	4	5	6
t_i	2	13	7	5	10	3

Alle Aufgaben sollen auf die Maschinen verteilt werden mit dem Ziel, den Makespan (frühester Zeitpunkt, zu dem alle Aufgaben fertiggestellt sind) zu minimieren.

Für die folgenden Teilaufgaben gilt: Falls beide Maschinen die gleiche Last haben, dürfen Sie die aktuelle Aufgabe einer beliebigen Maschine zuteilen.

a) Verteilen Sie die Aufgaben an die Maschinen unter Verwendung der Greedy-Offline-Heuristik für das Lastverteilungsproblem aus der Vorlesung. Geben Sie dabei für jede Aufgabe an, welcher Maschine sie zugeteilt wird. Geben Sie den resultierenden Makespan an.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Berechnet die Greedy-Offline-Heuristik auf dieser Instanz eine optimale Lösung? Beweisen Sie Ihre Antwort.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

c) Nun stehen nur noch zwei Maschinen M_1 und M_2 zur Verfügung und es sollen folgende Aufgaben verteilt werden:

i	1	2	3	4	5
t_i	2	13	7	5	10

Geben Sie eine Reihenfolge der Aufgaben an, sodass mit der Greedy-Online-Heuristik für das Lastverteilungsproblem aus der Vorlesung der Makespan 25 oder schlechter ist. Geben Sie dabei für jede Aufgabe an, welcher Maschine sie zugeteilt wird.

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Aufgabe 7: NP-Vollständigkeit und Approximation — $3 + 3 + 3 + 3 = 12$ Punkt(e)

Entscheiden Sie, ob die folgenden Aussagen stimmen, und begründen Sie kurz Ihre Antwort. Es ist kein Beweis gefordert.

a) Sei $\mathcal{B}$ die Klasse der ungerichteten Bäume mit mindestens drei Knoten. Es ist ein $\mathcal{NP}$-vollständiges Problem zu entscheiden, ob ein gegebener Baum $B \in \mathcal{B}$ einen Hamiltonischen Kreis enthält.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Für das Rucksackproblem gibt es einen $\frac{2}{3}$-approximativen Algorithmus mit polynomieller Laufzeit in n.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Angenommen $\mathcal{P} \neq \mathcal{NP}$. Dann gibt es eine polynomielle Reduktion von KNF-SAT auf die leere Sprache.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

d) Falls es einen Algorithmus mit polynomieller Laufzeit für VertexCover gibt, dann ist $\mathcal{P} = \mathcal{NP}$.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Aufgabe 8: Gewichtetes VC und SC — $3 + 3 + 7 = 13$ Punkt(e)

Im gewichtetem Vertex-Cover-Problem GVC ist ein ungerichteter Graph $G = (V, E)$ mit $V = \{1, \ldots, n\}$ und Knotengewichten w_v für jeden Knoten $v \in V$ gegeben. Desweiteren ist eine Gewichtsschranke W gegeben.

Gesucht ist eine Knotenüberdeckung $C \subseteq V$, sodass

$$\sum_{i \in C} w_i \leq W.$$

Im gewichtetem Set-Cover-Problem GSC sind m endliche Mengen $A_1, \ldots, A_m$, wobei Menge A_i das Gewicht s_i hat, und eine Gewichtsschanke S gegeben. Gesucht ist eine Indexmenge $I \subseteq \{1, \ldots, m\}$, sodass

$$\bigcup_{i \in I} A_i = \bigcup_{j \in \{1,\ldots,m\}} A_j \qquad \text{und} \qquad \sum_{i \in I} s_i \leq S.$$

a) Zeigen Sie, dass GVC $\in \mathcal{NP}$ ist.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Zeigen Sie, dass GSC $\in \mathcal{NP}$ ist.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Zeigen Sie, dass es eine polynomielle Reduktion GVC $\leq_p$ GSC gibt. Zeigen Sie die Korrektheit dieser Reduktion.

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Aufgabe 9: Approximativer Durchmesser — $3 + 4 + 4 = 11$ Punkt(e)

Gegeben sei ein ungerichteter, ungewichteter, zusammenhängender Graph $G = (V, E)$.

Bei einem ungewichteten Graphen messen wir die Länge eines Weges durch die Anzahl der Kanten entlang des Weges. Für je zwei Knoten $u, v \in V$ sei $d(u, v)$ die Länge des kürzesten Weges von u nach v. Der Durchmesser D eines Graphen ist die Länge des längsten kürzesten Weges im Graphen:

$$D = \max_{u,v \in V} d(u, v).$$

Gegeben sei folgender Algorithmus, der einen approximativen Durchmesser D_a berechnet: Wir führen Breitensuche mit Startknoten 1 aus und bestimmen die Tiefe t des resultierenden Baumes. Die Ausgabe ist $D_a = 2 \cdot t$.

a) Zeigen Sie, dass der Algorithmus nicht immer die optimale Lösung berechnet.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Zeigen Sie, dass $D_a \leq 2 \cdot D$ ist.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

c) Zeigen Sie, dass $D_a \geq D$ ist.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Aufgabe 10: Clique

$8 + 3 = 11$ Punkt(e)

Gegeben sei ein ungerichteter, zusammenhängender Graph $G = (V, E)$ mit $|V| \geq 2$.

Eine Clique C in G ist eine Teilmenge von V, innerhalb derer alle Knotenpaare durch eine Kante miteinander verbunden sind. Das Ziel ist, die Anzahl der Knoten in C zu maximieren.

a) Beschreiben Sie einen effizienten Greedyalgorithmus, der eine nicht-erweiterbare Clique ausgibt. Begründen Sie die Korrektheit Ihres Algorithmus und zeigen Sie, dass ihr Algorithmus effizient ist.

⇑⇑⇑ _____ / 8 Punkt(e) ⇑⇑⇑

b) Zeigen Sie, dass für eine optimale Clique C_{opt} in G und eine nicht-erweiterbare Clique C in G gilt:

$$\frac{|C_{opt}|}{|C|} \leq \frac{|V|}{2}$$

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Hinweis auf Seite 15 platziert wurde.