Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Zusatzaufgaben 01

Algorithmen I – Sommersemester 2023

Gesamtpunkte: 26

Allgemeiner Hinweis:
Für $0 \leq c < 1$ gilt sowohl $0 < \sum_{i=0}^{\infty} c^i < \infty$ als auch $0 < \sum_{i=0}^{\infty} i \cdot c^i < \infty$. Diese Abschätzung darfst du in allen Aufgaben verwenden.

Aufgabe 1 - Verbindungen knüpfen (5 Punkte)

Im folgenden Graphen sollen Kanten eingefügt werden. Diese kannst du entweder einzeichnen oder als Tupel angeben. Deine Kantenmenge soll die folgenden Eigenschaften erfüllen:

1. Zu jedem Knoten auf der linken Seite soll es genau dann eine gerichtete Kante zu einem Knoten auf der rechten Seite geben, wenn die zugehörige Funktion in der jeweiligen Menge enthalten ist.
2. Zu jedem Knoten v auf der rechten Seite soll es genau dann eine gerichtete Kante zu einem anderen Knoten u auf der rechten Seite geben, wenn die Menge zu v eine Untermenge der Menge zu u ist.

A	$\log \log n$	$O(\sqrt{n})$	1
B	$\log_2(4^{2n})$	$\Theta(\log_5(n))$	2
C	$2^{\log \log(n^3)}$	$\Theta(n \log(n))$	3
D	$7^{\log_7(n) \cdot \log_7(n)}$	$\omega(n)$	4
E	$\frac{6n}{\log_n(2)}$	$o(n^2)$	5
F	$\frac{n^3}{2^{6 \log(n)}}$	$\Omega(n)$	6

Aufgabe 2 - Fleißige Heinzelmännchen (3 Punkte)

Gegeben sei der folgende Sortieralgorithmus:

```
gnomeSort(A: [ℕ; n])
    pos: ℕ = 0
    while pos < n do
        if pos = 0 ∨ A[pos] ≥ A[pos − 1] then
            pos := pos + 1
        else
            swap(A[pos], A[pos − 1])
            pos := pos − 1
        end
    end
```

1. Führe gnomeSort für die Eingabe $A = \langle 4, 2, 3, 7, 6 \rangle$. Gib dabei den Inhalt von A nach jeder swap-Operation an. (1 Punkt)

2. Gib die asymptotische Laufzeit von GNOMESORT an und begründe deine Antwort. (2 Punkte)

Aufgabe 3 - Dr. Meta räumt auf (5 Punkte)

Da Dr. Meta ein sehr paranoider Superschurke ist, hält er jeden Biberfeind und jeden Biberfreund sehr genau im Auge. Um dies zu bewerkstelligen hat er eigens eine Datenstruktur D angelegt, in der er alle ihm jemals bekannten Biber hält. Die Ordnung in dieser Datenstruktur ist dadurch bestimmt, wie wohlgesonnen ein Biber seinem großen Meisterplan ist. Dem wohlgesonnensten Biber ist hierbei der Wert high zugewiesen und dem rebellischsten Biber der Wert low.
Zusätzlich zu dieser Ordnung werden auch die bekannten Freunde eines jeden Bibers in D gespeichert. Hierfür bietet sein Überwachungsapparat die Funktion SPIONIERE an, welche eine noch nicht bekannte Freundschaft zwischen zwei Bibern in $\Theta(1)$ in D einträgt.
Da nicht jeder Biber dem Plan zuträglich ist kommt es manchmal vor, das Bibern gekündigt wird. Die entsprechende Funktion EXPIRE beendet das Arbeitsverhältnis eines Bibers und entfernt ihn aus D in $\Theta(1)$. Zusätzlich wird, wenn Biber b entlassen wird, jeder mit b befreundete Biber als verdächtig markiert und die Freundschaftsbeziehung aus D entfernt. Dies benötigt Zeit in $\Theta(\texttt{friends}(b))$, wobei $\texttt{friends}(b)$ die Anzahl der Freunde von b bezeichnet.
Das Einfügen eines Bibers in D lässt sich durch INSERTMIDDLE in $\mathcal{O}(1)$ bewerkstelligen.

1. Zeige mittels einer Methode deiner Wahl, dass EXPIRE eine amortisiert konstante Laufzeit hat. Du darfst dabei annehmen, dass D zu Beginn leer ist. (3 Punkte)

Zusätzlich zum Ausspionieren kann Dr. Metas Überwachungsapparat auch Biber mittels QUESTION befragen. Wird ein Biber befragt, so wird er in D in $\Theta(1)$ als verdächtig markiert. Wird aber ein schon als verdächtig eingetragener Biber befragt, so wird diesem mit EXPIRE gekündigt.

* Zeige, dass EXPIRE eine amortisierte Laufzeit in $\Theta(1)$ hat. (2 Punkte)

Aufgabe 4 - Das Genie beherrscht das Chaos (7 Punkte)

Dr. Meta ist ein sehr kreativer Kopf. Ständig fallen ihm neue Möglichkeiten ein, die Weltherrschaft an sich zu reißen (oder Biber zu koordinieren - ein nie endendes Unterfangen). Um seine ganzen Ideen zu sammeln, hat er sich eine nagelneue Datenstruktur angelegt; doch nicht nur irgendeine Datenstruktur! Auf seine neuesten Einfälle möchte Dr. Meta besonders schnell zugreifen können, denn er hält eine Idee, wenn sie ihm neu einfällt, immer für besonders gut. Als großes Genie hat er natürlich keine schlechten Ideen, doch er weiß auch, dass er schnell den Überblick verlieren kann. Schweren Herzens beschließt er sich deswegen, stets höchstens wurzelig viele Ideen schnell zugreifbar zu machen. Wir bezeichnen die Anzahl aller Ideen mit $n \in \mathbb{N}$ und die Anzahl der neuesten Ideen mit $n_{\text{new}} \in \mathbb{N}$, wobei $n_{\text{new}} \leq \sqrt{n}$ gilt.
Insgesamt kann Dr. Meta folgende Operationen durchführen:

- INSERT(`idea`)
  Fügt die neue Idee `idea` ein. Gilt $n_{\text{new}} < \sqrt{n}$ Einträge, so wird `idea` in die Menge der neuesten Ideen eingefügt, was $\Theta(1)$ Zeit benötigt. Ansonsten muss Dr. Meta aufräumen. Dazu fügt er die neuesten Ideen in die Menge aller anderen Ideen ein, um Platz zu schaffen. Außerdem löscht er alle Ideen, die veraltet sind. Das Aufräumen benötigt Zeit in $\Theta(n + n_{\text{new}})$.
  Beachte: Nach dem Aufräumen gilt $n_{\text{new}} = 0$.

- HAS(`idea`)
  Sucht nach einer Idee. Dazu werden zunächst in $\Theta(n_{\text{new}})$ alle neusten Ideen überprüft. Wurde die Idee dort nicht gefunden, wird wird die Menge der übrigen Ideen in $\Theta(\log n)$ Zeit nach ihr durchsucht.

- DELETE(`idea`)
  Löscht die Idee `idea`. Nunja... fast. Dr. Meta ist faul und markiert erst einmal Ideen nur als veraltet. Wir bezeichnen mit n_{old} die Anzahl der als veraltet markierten Ideen. Ist $n_{\text{old}} < \sqrt{n}$, so wird `idea` wie bei HAS gesucht. Ist `idea` eine der neuesten Ideen, wird sie sofort gelöscht, ansonsten als veraltet markiert; beides benötigt Zeit in $\Theta(1)$.
  Ist $n_{\text{old}} \geq \sqrt{n}$, so ist es wieder dringend Zeit zum Aufräumen. Das passiert wie bei INSERT.

Beachte: Nach dem Umsortieren gilt $n_{\text{old}} = 0$.

1. Wir betrachten nun den Zustand der Datenstruktur direkt nach einer Aufräumaktion, d.h. es gilt $n_{\text{new}} = 0$. Wie oft hintereinander kann Dr. Meta nun neue Ideen einfügen, bis er wieder aufräumen muss? Wie oft kann er andererseits Ideen löschen, bis er wieder aufräumen muss? (2 Punkte)

2. Dr. Meta fragt sich, wie unaufgeräumt seine Datenstruktur eigentlich werden kann. Beschreibe den Zustand der Datenstruktur, von dem aus möglichst viele Ideen gelöscht werden können, ehe aufgeräumt werden muss. Wie viele Ideen können in Abhängigkeit von n, n_{new} gelöscht werden? (1 Punkt)

3. Zeige mit einer Methode deiner Wahl, dass alle Operationen eine amortisierte Laufzeit in $O(\sqrt{n})$ haben. (4 Punkte)

Aufgabe 5 - Wieder und wieder und wieder (6 Punkte)

Bestimme die Laufzeit der folgenden rekurrsiven Algorithmen.

Dazu kannst du die vorgestellte Technik verwenden, in dem du die folgenden Fragen für den Rekursionsbaum beantwortest:

- Wie viele Lagen gibt es?
- Wie viele Knoten sind auf Lage i?
- Wie groß ist das n auf Lage i?
- Wie viel Zeit kostet ein Knoten in Lage i?

um somit die Gesamtkosten für Lage i und hieraus die Gesamtkosten abzuleiten.

1.
$$T_1(n) = \begin{cases} 1 & | \; n = 1 \\ 243 \cdot T_1(\frac{n}{9}) + n^2\sqrt{n} & | \; \text{sonst} \end{cases}$$

2.
$$T_2(n) = \begin{cases} 1 & | \; n = 1 \\ 2 \cdot T_2(\frac{n}{3}) + n\log(n) & | \; \text{sonst} \end{cases}$$

3.

```
1:  ZERKLEINERN(L : List⟨ℕ⟩): ℕ
2:      n : ℕ = L.LENGTH()
3:      if n == 1 then
4:          return 1
5:      end
6:      L_1 : LIST⟨ℕ⟩
7:      L_2 : LIST⟨ℕ⟩
8:      L_3 : LIST⟨ℕ⟩
9:      L_4 : LIST⟨ℕ⟩
10:     i = 0
11:     for x ∈ L do
12:         if 0 ≤ i ≤ ⌊n/2⌋ then
13:             L_1.PUSHBACK(x)
14:         end
15:         if ⌊n/2⌋ ≤ i ≤ n then
16:             L_2.PUSHBACK(x)
17:         end
18:         if ⌊n/4⌋ ≤ i ≤ ⌊3n/4⌋ then
19:             L_3.PUSHBACK(x)
20:         end
21:         if (0 ≤ i ≤ ⌊n/4⌋) ∨ (⌊3n/4⌋ ≤ i ≤ n) then
22:             L_4.PUSHBACK(x)
23:         end
24:         i++
25:     end
26:     a =ZERKLEINERN(L_1)
27:     b =ZERKLEINERN(L_2)
28:     c =ZERKLEINERN(L_3)
29:     d =ZERKLEINERN(L_4)
30:     return a + b + c + d
```