Karlsruher Institut für Technologie

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

Übungsblatt 03

Algorithmen I - Sommersemester 2022

Abgabe im ILIAS bis 11.05.2022, 14:00 Uhr

Bitte beschrifte Deine Abgabe gut sichtbar mit Deinem Namen und Deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe Deines Tutoriums im ILIAS. Gib Deine Ausarbeitungen in *einer* PDF-Datei ab.

Aufgabe 1 - Schreib-/Lese-/Hitzkopf (9 Punkte)

Um Überhitzung einer Festplatte zu verhindern, entwerfen wir eine Datenstruktur die nicht zu häufig an die gleiche Speicherstelle schreibt. Diese Datenstruktur D besteht aus m Arrays. Das i-te Array, welches wir A_i nennen, hat die feste Größe 2^i. Damit kann D maximal $n = \sum_{i=0}^{m-1} 2^i$ Einträge halten. Der Inhalt eines Arrays A_i kann geschützt oder ungeschützt sein. Geschützte Arrays dürfen nicht direkt überschrieben werden. Um uns merken zu können, welche Arrays geschützt sind, steht ein weiteres Array PROTECTED : $[\text{Bool}, m]$ bereit. Es ist PROTECTED$[i]$ = `true` genau dann, wenn A_i geschützt ist. Wir gehen in dieser Aufgabe stets davon aus, dass es mindestens ein ungeschütztes Array in D gibt.

Um ein neues Element x in diese Datenstruktur einzufügen, nutzen wir den folgenden Algorithmus. Wir iterieren über die Arrays $A_0, \dots, A_{m-1}$ und suchen das erste ungeschützte Array A_i. An diese Stelle setzen wir das Array A_{neu}, wobei A_{neu} alle Elemente aus $\{x\} \cup \bigcup_{j<i} A_j$ enthält. Danach gilt Array A_i als geschützt und alle vorherigen Arrays A_j mit $j < i$ als ungeschützt.

Die Laufzeit des Erstellens von A_{neu} ist linear in der Anzahl der vereinigten Elemente.

1. Sei beispielsweise $m = 4$ und D liegt im folgenden Zustand vor:

$$\begin{aligned} \textsc{Protected} &= \langle \texttt{ true } , \texttt{ false } , \texttt{ false } , \texttt{ true } \rangle \\ A_0 &= \langle 5 \rangle \\ A_1 &= \langle 1, 13 \rangle \\ A_2 &= \langle 5, 7, 9, 10 \rangle \\ A_3 &= \langle 2, 3, 4, 6, 8, 57, 90, 400 \rangle \end{aligned}$$

   Nun wird Element 30 eingefügt. Gib an, welche Veränderungen in D dadurch entstehen. (1 Punkt)

2. Mit welcher Laufzeit müssen wir im schlimmsten Fall für bei einer Einfügeoperation rechnen? Gib eine Abschätzung im O-Kalkül in Abhängigkeit von n an. (1 Punkt)

Bei genauerer Betrachtung fällt auf, dass eine Einfügeoperation oft nicht diese worstcase Laufzeit benötigt. Wir wollen nun die Laufzeit mit Hilfe von amortisierter Analyse abschätzen.

3. Gib die Laufzeit einer Einfügeoperation in Abhängigkeit des Zustands der Datenstruktur im O-Kalkül an.
   Hinweis: Überlege wann eine Einfügeoperation in Abhängigkeit der Werte von Protected besonders günstig / teuer ist. (2 Punkte)

Im Folgenden betrachten wir jeweils den Fall indem zunächst alle Arrays in D ungeschützt sind und wollen nun $k \leq n$ Elemente in D einfügen.

4. Zeige mit der Aggregatmethode, dass die amortisierte Laufzeit einer Einfügeoperation in $O(\log n)$ ist. (2 Punkte)

5. Formuliere eine Potentialfunktion und zeige mit der Potentialmethode, dass die amortisierte Laufzeit einer Einfügeoperation in $O(\log n)$ ist. (3 Punkte)

Aufgabe 2 - Dr. Meta traut niemandem (6 Punkte)

Zunächst die gute Nachricht: Mit deiner Hilfe konnte Dr. Meta die Spur des Eindringlings durch sein Labor bis zu ihrem Beginn zurückverfolgen. Bravo!
Doch nun die schlechte Nachricht: Der Weg des Eindringlings begann an einem der Seitenausgänge. Dort finden sich jedoch keine Anzeichen für ein gewaltsames Eindringen. Dr. Meta wittert Verrat in den eigenen Reihen.
Er will deswegen seine Handlanger noch einmal gründlich durchleuchten. Dazu benötigt er die Daten, die er über seine Handlanger gesammelt hat. Da Dr. Meta zu seinem eigenen Leidwesen sehr unorganisiert ist, hat er die Daten seiner n Handlanger in einem unsortierten Array A gespeichert. Jedem Handlanger ist genau ein Eintrag in A zugeordnet. Dieser beinhaltet insbesondere dessen Identifikationsnummer. Auf die an $A[i]$ gespeicherte Identifikationsnummer kannst du mit $A[i].id$ in $\Theta(1)$ zugreifen.

Einen Handlanger in $O(n)$ suchen zu müssen, dauert Dr. Meta viel zu lange. Er hat seinen Handlangern schließlich nicht grundlos eindeutige Identifikationsnummern (IDs) gegeben.
Nun liegt es wieder an dir! Konstruiere eine Datenstruktur, mit deren Hilfe der Eintrag x in A gefunden werden kann, der zu einer gegebenen ID $x.id$ gehört. Dabei soll die Datenstruktur in $O(\log n)$ den Index von x in A zurückgeben und A dabei nicht verändert werden. Du darfst davon ausgehen, dass A keine Duplikate enthält. Zudem darfst du annehmen, dass du ein Feld der Größe n in $O(n \cdot \log(n))$ sortieren kannst.

Hinweis: Denk daran, dass die Aufgabenstellung "Beschreibe" von dir eine textuelle Beschreibung verlangt. Demzufolge ist Pseudocode keine gültige Lösung.

1. Beschreibe, wie eine solche Datenstruktur aufgebaut ist und wie mit ihrer Hilfe ein Eintrag in A gesucht wird. (2 Punkte)

2. Welche asymptotische Laufzeit benötigt die Konstruktion der Datenstruktur, die du in Teilaufgabe 1 beschrieben hast? (1 Punkt)

3. Welchen Speicherbedarf hat deine Datenstruktur? Gib eine möglichst enge Abschätzung im O-Kalkül an. (1 Punkt)

4. Begründe, warum das Suchen eines Eintrags in A mit Hilfe der neuen Datenstruktur nun in $O(\log n)$ funktioniert. (1 Punkt)

5. Angenommen, du bekommst die Garantie, dass die IDs natürliche Zahlen kleiner als eine Konstante $c_{\max}$ sind. Kannst du die Laufzeit für die Suche nach einem Eintrag in A weiter verringern? Begründe deine Antwort. (1 Punkt)

Aufgabe 3 - Queues (5 Punkte)

In dieser Aufgabe wollen wir uns näher mit der Modellierung einer Queue beschäftigen (siehe Folie 10 der vierten Vorlesung). Diese soll die folgenden Operationen unterstützen:

- pushBack(elem) fügt elem in die Queue ein
- popFront() gibt ältestes Element in Queue zurück und löscht es aus der Queue

1. Wie kann eine solche Queue mit Hilfe einer einfach verketteten Liste modelliert werden? Gib an, welche zusätzlichen Daten benötigt werden und wie pushBack und popFront funktionieren. Beschreibe beide Operationen kurz. Bestimme und begründe ihre asymptotische Laufzeit. (2 Punkte)

2. Wie kann eine solche Queue mit Hilfe *eines* dynamischen Arrays modelliert werden? Gib an, welche zusätzlichen Daten benötigt werden und wie pushBack und popFront funktionieren. Beschreibe beide Operationen kurz. Bestimme und begründe ihre asymptotische Laufzeit.
Hinweis: Beide Operationen sind amortisiert in $\Theta(1)$ durchführbar. (3 Punkte)