Karlsruher Institut für Technologie

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

Übungsblatt 03

Algorithmen I - Sommersemester 2024

Abgabe im ILIAS bis 10.05.2024, 18:00 Uhr

Die Abgabe erfolgt alleine oder zu zweit als eine PDF-Datei über das Übungsmodul in der Gruppe eures Tutoriums im ILIAS. Beschriftet die Abgabe deutlich mit Matrikelnummer(n) und Name(n). Bei Abgaben zu zweit reicht es, wenn eine Person die Abgabe im ILIAS hochlädt.

- Achtet bei handschriftlichen Abgaben auf Lesbarkeit.
- Achtet darauf, ob Algorithmen in Worten oder Pseudocode beschrieben werden sollen.
- Es werden immer asymptotisch möglichst effiziente Algorithmen erwartet, wenn nicht anders angegeben.
- Wenn Korrektheits- oder Laufzeitanalysen gefordert sind, behandelt diese separat von der Algorithmenbeschreibung.

Gesamtpunkte: 20

Aufgabe 1 - The odds are stacked against you (7 Punkte)

In dieser Aufgabe soll die Datenstruktur **Stack** mithilfe von Listen und Arrays umgesetzt werden.
Ein Stack unterstützt hierbei die folgenden Operationen:

- `push`(x): Legt das Element x oben auf den Stack.
- `pop`(x): Entfernt das oberste Element vom Stack und gibt es zurück.

Viele Stacks unterstützen außerdem die Operation `peek`. Da Dr. Meta diese allerdings sehr langweilig findet, möchte er, dass dein Stack stattdessen die folgende (laut ihm spannendere) Operation umsetzt.

- `spy`(): Gibt das aufgerundet mittlere Element des Stacks zurück ohne es zu entfernen.

In dieser Aufgabe sollst du auf Zeiger- bzw. Arrayeintragebene argumentieren und nicht die angebotenen Operationen der Datenstrukturen aufrufen.

a) Beschreibe, wie ein solcher Stack mit Hilfe einer **doppelt** verketteten Liste (Implementierung von Folie 3, ohne Dummy-Knoten) effizient modelliert werden kann. Gib an, welche zusätzlichen Daten (wie z. B. Zeiger oder Variablen) benötigt werden und wie `push`(x), `pop`(x) und `spy`() funktionieren. Gib an, welche Laufzeit deine Operationen benötigen und begründe warum. (3 Punkte)

b) Begründe, warum bei Verwendung, von **einfach** verketteten Listen mindestens eine der drei Operationen mehr als konstante Laufzeit benötigt. (1 Punkt)

c) Beschreibe, wie ein solcher Stack mit Hilfe eines dynamischen Arrays effizient modelliert werden kann. Gib an, welche zusätzlichen Daten (wie z. B. Zeiger oder Variablen) benötigt werden und wie `push`(x), `pop`(x) und `spy()` funktionieren. Gib an, welche amortisierte Laufzeit deine Operationen benötigen und begründe warum.

Hinweis: Du kannst hier die amortisierten Laufzeiten aus der Vorlesung verwenden und musst selbst keine amortisierte Analyse durchführen. (3 Punkte)

Aufgabe 2 - Ich versteh nur Bahnhof (6 Punkte)

Wir betrachten die folgenden zwei Algorithmen REVERSE und BAHNSTREIK. Beachte, dass wir dabei häufig über Mengen iterieren, bei denen eine implizite Ordnung angenommen wird.

REVERSE($A : [\mathbb{N}_0, n]$, start : $\mathbb{N}_0$, end : $\mathbb{N}_0$) kehrt die Reihenfolge der Elemente in A im Intervall $[\text{start}, \text{end}]$ um. Die Grenzen werden hierbei mitgetauscht. Dafür geht der Algorithmus von außen nach innen über das Array und vertauscht wiederholt die äußeren beiden Elemente.

```
1:  BAHNSTREIK(A : [ℕ_0, n])
2:      for Verhandlungsrunde ∈ {n, ..., 2} do   \\Absteigend
3:          Vermittlungsvorschlag := 0
4:          for Forderung ∈ {0, ..., Verhandlungsrunde − 1} do
5:              if A[Forderung] > A[Vermittlungsvorschlag] then
6:                  Vermittlungsvorschlag = Forderung
7:              end
8:          end
9:          REVERSE(A, 0, Vermittlungsvorschlag)
10:         REVERSE(A, 0, Verhandlungsrunde − 1)
11:     end
```

a) Gegeben sei das Array: $A = [7, 4, 33, 1, 0, 5, 12, 13, 2, 11]$
Führe jeden der zwei Algorithmen auf A aus und gib den resultierenden Inhalt von A an. Verwende für REVERSE die Parameter start $= 2$, end $= 7$. (2 Punkte)

b) Gebe für REVERSE Pseudcode an und beschreibe BAHNSTREIK in Worten. Bestimme und begründe außerdem deren asymptotische Laufzeiten in Abhängigkeit aller Eingabeparameter. (4 Punkte)

Aufgabe 3 - Knapp daneben ist ~~auch vorbei~~ nicht sofort vorbei (7 Punkte)

Nachdem sich die Klon-Maschine, wie von dir auf dem letzten Blatt bewiesen, als höchst erfolgreich herausgestellt hat, möchte sich Dr. Meta einen Überblick über seine Biber verschaffen. Dafür hat er die Biber eindeutig durchnummeriert. Für die nächste Stufe seines Geheimplans möchte er bestimmte Biber in der Menge der Biber finden können. Da der Biber, der die Nummerierung durchgeführt hat, einige Nummern übersprungen hat, lässt Dr. Meta alle Biber sich in einer Reihe nach ihrer Nummer sortiert aufstellen. Hier möchte er mittels einem von ihm eigens entwickelten und hoch innovativem Algorithmus namens binärer Suche die gesuchten Biber ausfindig machen. Da er in weiser Voraussicht den Boden mit Indizes beschriftet hat, kann er bei gegebenem Index in konstanter Zeit auf den Biber an dieser Position zugreifen.

Nach einigen Testläufen stellt Dr. Meta allerdings fest, dass die Biber einen Fehler bei ihrer Aufstellung gemacht haben: Jeder Biber steht höchstens ein Feld entfernt von der Position, die er in einer korrekt sortierten Folge hätte, allerdings nicht zwingend an dieser. Somit sind die Biber nur *beinahe sortiert*.

a) Zeige, dass die aus der Vorlesung bekannte binäre Suche auf beinahe sortierten Aufstellungen falsche Rückgaben liefern kann. Führe dafür die klassische binäre Suche (Folie 14) auf der folgenden, als Array repräsentierten, Aufstellung von Bibern aus. Suche dabei die Zahl 7. Gebe hierbei die Werte von `beg`, `end` und `mid` in jedem Aufruf an. Gebe außerdem den Rückgabewert an. (1.5 Punkte)

0	1	2	3	4	5	6	7	8
1	7	2	9	12	10	21	20	42

Das Array mit Indizes.

Da die bisher bekannte binäre Suche in diesem Szenario nicht immer korrekt ist, schlägst du Dr. Meta vor, den Algorithmus leicht anzupassen und das Problem so zu lösen. Dr. Meta ist zwar nicht völlig von deiner Idee überzeugt, beauftragt dich aber mit der Ausarbeitung des modifizierten Algorithmus.

b) Du überlegst dir zuerst, an welchen Indizes ein Biber in einer beinahe sortierten Aufstellung stehen kann. Betrachte hierfür einen Biber, der in der sortierten Aufstellung an Stelle i stehen würde. Wo kann dieser in einer beinahe sortierten Aufstellung stehen? (1 Punkt)

c) Modifiziere den Algorithmus der binären Suche so, dass er die Position des Bibers mit der gegebenen Nummer in einer beinahe sortierten Aufstellung, in welcher der Biber vorkommt, in logarithmischer Zeit bestimmt. Beschreibe deinen Algorithmus in Worten und begründe sowohl seine Laufzeit als auch seine Korrektheit. (3.5 Punkte).

Hinweis: Stelle sicher, dass die binäre Suche das eigentliche mittlere Element auch wirklich betrachtet.

d) Dr. Meta gefällt zwar deine Lösung sehr, um aber spätere Verwirrung zu vermeiden möchte er das zugrunde liegende Problem beheben. Erkläre kurz, wie du in linearer Zeit eine beinahe sortierte Aufstellung sortieren kannst. (1 Punkt)

Aufgabe 4 - Zum Knobeln (0+2 Punkte)

Gegeben seien die Operationen `pushFront`(x : Element), `popFront`(), `pushBack`(x : Element), `popBack`(), `insertAfter`(k : Knoten, x : Element), `remove`(k : Knoten), `concatenate`(l: Liste, m : Liste)[1] und `splice`(a : Knoten, b : Knoten, c : Knoten) auf doppelt verketteten Listen mit Dummy-Headern.

a*) Mit welcher dieser Operationen kannst du alle anderen in konstanter Zeit umsetzen? (0.5 Punkte)

b*) Beschreibe, wie die anderen Operationen mit deiner in a*) gewähltern Operation umgesetzt werden können. (1.5 Punkte)

[1] `concatenate`(l: Liste, m : Liste) hängt alle Elemente der Liste m hinter das letzte Element der Liste l. Die Reihenfolge der Elemente in jeder Liste bleibt erhalten.