max planck institut informatik

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Benjamin Doerr, Dr. Reto Spöhel **Wintersemester 2011/12**
Übungsleitung: Alexander Kobel

Übung zu Grundzüge von Algorithmen und Datenstrukturen

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws11/grads/

Übungsblatt 7 Abgabe: Donnerstag, 8. Dezember 2011, 12:00 Uhr

Hinweise zur Abgabe der Aufgaben: Schreiben Sie klar und deutlich in gebührendem Abstand zu weiterem Text Ihren **Namen** und **Matrikelnummer**, ihre **Übungsgruppe als arabische Ziffer** sowie den **Namen Ihres Tutors** auf Ihre Abgabe. **Heften** Sie mehrere Blätter geeignet zusammen. Legen Sie ihre Lösung bis zum Abgabetermin in den **unteren rechten Briefkasten** im Erdgeschoss von Gebäude E1 3 (neben Hörsaal 001). Achtung: Verwechseln Sie den Briefkasten nicht mit dem für die *Stammvorlesung* Algorithms and Data Structures!

Aufgabe 1 (*Schriftliche Übung, 2 Punkte*)

Lesen Sie die beiden kurzen Abschnitte zum Binary-Counter-Beispiel in den Kapiteln 9.1.1. und 9.1.2. im Skript, und beantworten Sie dann folgende Fragen:

a) Im Skript steht, dass für die amortisierten Kosten a_i in Schritt i immer die Ungleichung $a_i \leq 2$ gilt. Geben Sie entweder ein Beispiel eines Schrittes, in dem a_i echt kleiner als zwei ist, oder zeigen Sie, dass immer $a_i = 2$ gilt. [1P.]

b) Wie viele einzelne Bitoperationen sind notwendig, um einen Binärcounter von 0 auf $n = 2011$ hochzuzählen? [1P.]

Aufgabe 2 (*Schriftliche Übung, 6 Punkte*)

Wir betrachten die folgende Implementation von dynamischen Arrays: Zu jedem Zeitpunkt haben wir einen statischen Array A (der z.B. einen Stack oder eine Queue wie in der Vorlesung diskutiert implementiert), in den wir Elemente einzeln einfügen oder entfernen.

Die Größe von A ist immer eine Zweierpotenz. Falls nach einer Einfüge-Operation A komplett voll ist, legen wir sofort einen neuen Array A' der *doppelten* Größe an, kopieren den Inhalt von A geeignet in Zellen von A', geben den für A zugewiesenen Speicher frei und arbeiten mit A' als dem aktuellen Array weiter. Falls nach einer Entferne-Operation der aktuelle Array nur noch zu einem *Viertel* gefüllt ist, legen wir ähnlich einen neuen Array der *halben* Größe an und verfahren ansonsten gleich.

Zu jedem Zeitpunkt sind die für uns relevanten Informationen gegeben durch die Größe m des aktuellen Arrays (m ist immer eine Zweierpotenz) sowie durch die Anzahl n der momentan gespeicherten Elemente, wobei $m/4 < n < m$ gilt. Wir nehmen vereinfachend an, dass das Einfügen und Entfernen eines Elements Aufwand 1 hat, wenn kein neuer Array angelegt wird, und dass der Aufwand in Schritten, in denen ein neuer Array angelegt wird, gegeben ist durch die Anzahl der Elemente, die umkopiert werden müssen (auch dies ist immer eine Zweierpotenz). Zudem nehmen wir an, dass unsere Datenstruktur nie ganz leer ist – konkret, dass zu jedem Zeitpunkt $n \geq 8$ gilt.

In dieser Aufgabe zeigen wir mit der Potential-Methode, dass der amortisierte Aufwand pro Einfüge- und Entferne-Operation konstant ist. Wir definieren die Potentialfunktion

$$\Phi_i = \Phi_i(n, m) := |2n - m|,$$

wobei mit n und m die entsprechenden Werte *nach* dem i-ten Schritt gemeint sind.

a) Begründen Sie informell, warum Φ_i eine „vernünftige“ Potential-Funktion ist, d.h. etwas über mögliche in der Zukunft anfallende Kosten aussagt. [1P.]

b) Bestimmen Sie die amortisierten Kosten

$$a_i = c_i + (\Phi_i - \Phi_{i-1}),$$

die im i-ten Schritt höchstens anfallen, falls dieser

(i) eine Einfüge-Operation ist, bei der sich die Größe des aktuellen Arrays nicht ändert;

(ii) eine Entferne-Operation ist, bei der sich die Größe des aktuellen Arrays nicht ändert;

(iii) eine Einfüge-Operation ist, bei der sich die Größe des aktuellen Arrays verdoppelt;

(iv) eine Entferne-Operation ist, bei der sich die Größe des aktuellen Arrays halbiert.

[je 1P.]

Hinweis: *Lassen Sie sich nicht durch auftretende negative amortisierte Kosten irritieren. Intuitiv bedeuten diese einfach, dass wir in früheren Schritten etwas mehr Rücklagen als nötig gebildet haben, so dass nach Deckung der anfallenden echten Kosten ein Saldo zu unseren Gunsten übrig bleibt.*

c) Zeigen Sie, dass für jede Folge von (in beliebiger Reihenfolge auftretenden) Einfüge- und Entferne-Operationen die totalen echten Kosten beschränkt sind durch $3k + 2n_0$, wobei k die Länge der Folge und n_0 die Anzahl der zu Beginn der Folge gespeicherten Elemente bezeichnet. [1P.]

Aufgabe 3 (*Schriftliche Übung, 2 Punkte*)

Warum ist es keine gute Idee, bei den dynamischen Arrays in Aufgabe 2 bereits dann einen neuen, halb so großen Array anzulegen, wenn nach einem Entferne-Vorgang der aktuelle Array nur noch zur *Hälfte* (statt einem Viertel) voll ist? [1P.] Zeigen Sie, dass es mit dieser Implementations-Variante für jede Konstante C eine unendliche Folge von Einfüge- und Entferne-Operationen gibt, für die die durchschnittlichen Kosten pro Operation C übersteigen. [1P.]

Aufgabe 4 (*Schriftliche Übung, 2 Punkte*)

Nehmen Sie an, dass der dynamische Array aus Aufgabe 2 eine Queue auf die im Skript beschriebene Art implementiert, und führen Sie aus, was es in diesem Fall heißt, „den Inhalt von A geeignet in Zellen von A' zu kopieren“.

Aufgabe 5 (*Schriftliche Übung, 4 Punkte*)

Stellen Sie sich vor, dass Sie in einer Sprache programmieren müssen, die zwar über Stacks, aber nicht über Queues verfügt. Die verfügbaren Stacks seien so implementiert, dass sie *beliebig viele* Elemente aufnehmen können und trotzdem die Standardoperationen Is-Empty, Push und Pop *immer* in konstanter Zeit ausführen.

a) Beschreiben Sie, wie man die Funktionalität einer Queue durch Verwendung von zwei Stacks S_1, S_2 und deren Standardoperationen so emulieren kann, dass die emulierten Queue-Operationen *amortisiert konstante Laufzeit* haben. Geben Sie ein informelles Argument für „amortisiert konstante Laufzeit“. [2P.]

b) Definieren Sie eine geeignete Potentialfunktion und beweisen Sie mit der Potentialmethode, dass Ihre Implementation in der Tat amortisiert konstante Laufzeit pro Operation hat. [2P.]

Knobelaufgabe

Diese Aufgabe ist schwer und zum Knobeln gedacht. Lösungsvorschläge senden Sie bitte direkt an `rspoehel@mpi-inf.mpg.de`*; für die erste korrekte Einsendung gibt's einen Buchpreis! Es ist natürlich Ehrensache, dass Sie nur Lösungen einsenden, auf die Sie selbst gekommen sind.*

n Gefangene werden in Einzelzellen gesperrt. In unregelmäßigen Abständen wird jeweils einer von ihnen zufällig ausgewählt und in den Versammlungsraum des Gefängnisses geführt. Dort befinden sich m Glühbirnen, jede davon einzeln in einem Kästchen, so dass man nur durch Öffnen des jeweiligen Kästchens festellen kann, ob eine bestimmte Glühbirne brennt oder nicht. In diesem Kästchen befindet sich jeweils auch der zugehörige Schalter, mit dem man die Glühbirne ein- und ausschalten kann. Jeder Gefangene darf höchstens x Kästchen öffnen, wenn er in den Versammlungsraum geführt wird. Die darin enthaltenen x Glühbirnen darf er sowohl anschauen als auch ein- oder ausschalten, falls er das möchte.

Der Wärter verspricht den Gefangenen, dass sie alle freigelassen werden, wenn in dem Moment, wo der letzte der Gefangenen zum ersten Mal in den Versammlungsraum geführt wird (das ist also der erste Zeitpunkt, zu dem alle mindestens einmal im Versammlungsraum waren), dieser das feststellt und die Ansage macht „Jetzt waren alle mal im Versammlungsraum." Wenn ein Gefangener diese Ansage macht, obwohl sie nicht stimmt, oder wenn der Gefangene, der als letzter in den Versammlungsraum geführt wird, diese Ansage nicht macht, werden alle Gefangenen hingerichtet.

Bevor die Gefangenen eingesperrt werden, dürfen sie sich auf eine gemeinsame Strategie einigen. Danach können sie nur noch auf die beschriebene Weise mittels der Glühbirnen miteinander kommunizieren. Zu Beginn sind alle Glühbirnen ausgeschaltet. Es sind keine schmutzigen Tricks erlaubt (wie Glühbirnen rausdrehen, kaputtschlagen etc.); dafür haben die Glühbirnen eine unbegrenzte Lebensdauer. ;-)

Die Frage ist natürlich, für welche Werte von n (Anzahl Gefangener), m (Anzahl Glühbirnen) und x (Anzahl Glühbirnen, auf die ein Gefangener jeweils zugreifen darf) die Gefangenen eine Strategie haben, die ihr Überleben garantiert. Es gibt eine triviale solche Strategie für $m = x = n$ (welche?), und eine fast triviale Strategie für $m = x = \lceil \log_2(n+1) \rceil$ (welche?).

Ihre Aufgabe: Geben Sie eine Gewinnstrategie an mit $x \in O(\log \log n)$. Sie dürfen den Wert von m beliebig groß wählen und zudem voraussetzen, dass n eine (große) Zweierpotenz ist.

Tipp: *Die mir bekannte Lösung hat im weitesten Sinne etwas mit den amortisierten Kosten des Binärzählers aus Aufgabe 1 zu tun.*