2e deeltentamen Datastructuren 2011/2012

Je hebt drie uur voor dit deeltentamen.

Lees de opgaven goed!

Bij verschillende opgaven wordt om een algoritme gevraagd. Probeer hierbij duidelijke antwoorden te geven. Pseudocode met een toelichting in tekst is daarbij een goede optie.

Schrijf op elk blad je naam, en op het eerste vel je collegekaartnummer. Schrijf ook op het eerste blad hoeveel bladen je inlevert. Schrijf leesbaar en netjes, en geef duidelijke antwoorden.

Zet je mobiel uit voor aanvang van het tentamen.

Je mag geen rekenmachine gebruiken bij dit tentamen.

Veel succes!

Opgave 1. Stacks en queues. (0.5+0.5 pnt)
(a) Stel op een lege stack S worden achtereenvolgens de volgende zes operaties uitgevoerd:
Push(S,23); Push(S,100); Pop(S); Push(S,17); Push(S,52); Pop(S).
Wat is de uitvoer van de laatste operatie?

(b) Stel op een lege queue Q worden achtereenvolgens de volgende zes operaties uitgevoerd:
Enqueue(Q,31); Enqueue(Q,42); Dequeue(Q); Enqueue(Q,53);
Enqueue(Q,64); Dequeue(Q).
Wat is de uitvoer van de laatste operatie?

Opgave 2. Open addressing hashing met deletions. (1 punt) Beschouw hashing met open addressing. Stel we hebben een tabel met grootte 100. Keys zijn integers, en als hashfunctie gebruiken we

$$h(k, i) = (k + i) \bmod 100$$

waarbij k de keywaarde en i het aantal eerdere pogingen voor key k is. (N.B. We volgen de in het boek en college gebruikte notatie.)

Zoals behandeld kan er iets misgaan bij hashing met open addressing als we bij een deletion het element gewoon weglaten en dus door een lege positie vervangen.

(a) **Wat kan er misgaan? (Leg dit uit.)**

(b) **Geef hiervan een voorbeeld, voor de tabel en hashfunctie die hierboven gegeven zijn.**

1

Opgave 3 (1.25 pnt) Stel we hebben een hashtabel met chaining. De tabel heeft m posities, en er zijn n keys opgeslagen, waarbij we de simple uniform hashing aanname maken.

(a) **Wat is de kans (onder de aanname van simple uniform hashing) dat in de tabel positie 1 geen enkele key heeft?** (Met andere woorden: dat positie 1 leeg is?) Geef deze kans als functie van m en n, zonder O, Θ of Ω-notatie te gebruiken.

(b) **Leg je antwoord hierboven kort uit.**

Opgave 4. (1.25 punt) Ik wil onderzoek gaan doen naar het gedrag van rood-zwart-bomen. Daarom wil ik onder andere van knopen in de rood-zwart-boom weten hoeveel rode afstammelingen ze hebben.

We hebben een rood-zwart-boom. Iedere knoop heeft, behalve de gebruikelijke informatie (*key*, verwijzing naar linkerkind, verwijzing naar rechterkind, verwijzing naar ouder en kleur (rood of zwart)), ook een extra integer variabele, die *roodformaat* heet.

Ik wil graag dat het volgende geldt: Het *roodformaat* van een knoop is het aantal rode knopen in de deelboom met die knoop als wortel. Het roodformaat van een knoop x is dus gelijk aan het aantal knopen y dat: (a) rood is en (b) y is een afstammeling van x of $y = x$.

Stel we beginnen met een lege rood-zwart-boom, en we doen insertions (invoegingen) op de boom, en een aantal queries.

(a) **Is het mogelijk om insertions zodanig te doen** zodat:

- de boom blijft voldoen aan de eigenschappen van een rood-zwart-boom;
- een insertion $O(\log n)$ tijd kost (n is het aantal keys opgeslagen in de boom);
- na afloop van de operatie elke knoop in de boom de juiste waarde voor *roodformaat* heeft.

(Schrijf op: JA of NEE.)

(b) **Beargumenteer je antwoord**. Je mag gebruik maken van resultaten die in het college behandeld zijn.

2

Opgave 5 (1.25 + 0.5 pnt) Stel we hebben een AVLboom, waarbij elke knoop drie verwijzingen naar andere knopen heeft: *parent* (naar de ouder in de boom; voor de wortel is deze verwijzing NIL); *left* (naar het linkerkind, kan NIL zijn), en *right* (naar het rechterkind, kan NIL zijn). Iedere knoop heeft een integer *key*; deze voldoen aan de eigenschap van zoekbomen. Alle knopen hebben een verschillende *key*-waarde.

Je mag aannemen dat de boom niet leeg is, en we hebben een verwijzing $root(T)$ die naar de wortel van de boom wijst.

(a) **Geef een efficiente methode die de een-na-grootste keywaarde die opgeslagen is in de boom als antwoord geeft.** Bijvoorbeeld: als de keys 3, 5, 7, 9 en 11 zijn opgeslagen, dan moet je methode de waarde 9 als return-waarde opleveren.

Als minder dan twee waarden in de boom zijn opgeslagen, moet je methode een foutmelding geven.

Toelichting in woorden van je methode wordt op prijs gesteld, maar is niet vereisd.

(b) **Hoeveel tijd kost je methode? Leg uit.** Druk de tijd uit als functie van het aantal in de boom opgeslagen keys n, waarbij je of de O-notatie of of de Θ-notatie mag gebruiken.

Opgave 6. (1.25+0.5 pnt) Stel we hebben een rood-zwart boom, waarbij elke knoop drie verwijzingen heeft: *parent* (naar de ouder in de boom; voor de wortel is deze verwijzing NIL); *left* (naar het linkerkind, kan NIL zijn), en *right* (naar het rechterkind, kan NIL zijn). Iedere knoop heeft een integer *key*; deze voldoen aan de eigenschap van zoekbomen. Het is mogelijk dat sommige knopen dezelfde *key*-waarde hebben.

Daarnaast heeft elke knoop een integer waarde *formaat*. Het *formaat* van een knoop x is het aantal knopen in de boom in de deelboom met x als wortel, of, met andere woorden: het aantal knopen y met x als voorouder van y of $x = y$.

Knopen hebben ook een kleur *rood* of *zwart*; en deze kleuren voldoen aan alle eisen van een rood-zwart boom.

Je mag aannemen dat de boom niet leeg is, en we hebben een verwijzing $root(T)$ die naar de wortel van de boom wijst.

(a) **Geef een methode, die als invoer een key k krijgt en die in $O(\log n)$ tijd uitrekent hoeveel knopen een key hebben die gelijk is aan k.** Je methode moet dus een integer als antwoord opleveren.

Je mag in je code niet verwijzen naar subroutines die in 't college behandeld zijn: als je die wilt gebruiken moet je ook daarvoor pseudocode geven. Correctheid van de methode hoeft niet beargumenteerd te worden. Bijvoorbeeld: als de waarden 5, 5, 5, 7, 7, 8, 10, 11 zijn opgeslagen, en $k = 7$ dan is het antwoord 2; als $k = 9$ dan is het antwoord 0.

Toelichting in woorden van je methode wordt op prijs gesteld, maar is niet vereisd.

(b) **Leg uit waarom je methode $O(\log n)$ tijd gebruikt.** Hierbij mag je verwijzen naar resultaten die in het college of boek behandeld zijn.

7. Geheugengebruik van Skiplists. (1+1 punt) Beschouw de skiplist, zoals behandeld in het college. We nemen aan dat de kans dat een element in een hogere laag gezet wordt elke keer 1/2 is. Stel dat we de skiplist gebruiken om n keys op te slaan.

(a) Wat is het **verwachte geheugengebruik**? Kies één van de antwoorden hieronder en schrijf die op.

1. $\Theta(1)$
2. $\Theta(\log n)$
3. $\Theta(n)$
4. $\Theta(n \log n)$
5. $\Theta(n^2)$
6. Oneindig

(b) **Beargumenteer het antwoord dat je net gaf.**

(c) Wat is het **geheugengebruik in het slechtste geval** ("worst case")? Kies één van de antwoorden hieronder en schrijf die op.

1. $\Theta(1)$
2. $\Theta(\log n)$
3. $\Theta(n)$
4. $\Theta(n \log n)$
5. $\Theta(n^2)$
6. Oneindig

(d) **Beargumenteer het antwoord dat je net gaf.**