Dit tentamen is in elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat. A–Eskwadraat kan niet aansprakelijk worden gesteld voor de gevolgen van eventuele fouten in dit tentamen.

2e deeltentamen Datastructuren 2009/2010

Je heeft drie uur voor dit deeltentamen.

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. Let ook op dat je pseudocode duidelijk en netjes is.

Schrijf leesbaar en netjes.

Zet je mobiel uit voor aanvang van het tentamen.

Je mag geen rekenmachine gebruiken bij dit tentamen.

Zet je naam en collegekaartnummer op elk vel dat je inlevert.

Veel succes!

1. Grenzen (1 pnt, -0.5 voor elk fout of niet gegeven antwoord, je kan niet minder dan 0 pnt halen) Geef bij deze vraag alléén de grenzen in Θ-notatie. Je hoeft de antwoorden niet toe te lichten.

(a) Wat is de maximum diepte van een binaire boom met n knopen?

(b) Wat is de maximum diepte van een rood-zwart boom met n knopen?

(c) Stel een stack wordt geïmplementeerd met behulp van een enkelvoudig gelinkte lijst, zoals besproken op college en in het boek. Stel er zijn n elementen in de stack opgeslagen. Hoeveel tijd kost één POP-operatie?

2. Persistentie (1 pnt) Wat wordt er bedoeld met een *persistente* datastructuur? Leg kort maar duidelijk uit wat dit begrip *persistentie* betekent voor datastructuren.

3. Lijsten (1.5 pnt) Stel we hebben een enkelvoudiggelinkte lijst. Ieder element van de lijst heeft twee velden: *volgende* en *key*, waarbij *volgende* een pointer is naar het volgende element in de lijst, als dat bestaat, anders is de pointer **nil**.

De lijst is gegeven als object L; dit object bevat een pointer *begin*, die naar het begin van de lijst wijst. begin.L is dus het eerste element van de lijst. Als de lijst leeg is, dan heeft begin.L de waarde **nil**.

Geef pseudocode voor een methode DELETEALL(L,*sleutel*), die uit de lijst alle elementen verwijderd wiens key gelijk zijn aan *sleutel*.

Als de lijst $r \geq 1$ elementen bevat, moet je code in $O(r)$ tijd werken. Je code moet ook correct werken op een lege lijst.

4. Bomen-I (2 pnt) Op de universiteit van Harderwijk wordt het vak *Sorteren en Classificeren* gegeven. Dit vak heeft ook een practicum, waarbij de studenten een rood-zwart-boom moeten uitprogrammeren. De practicumleiding wil automatisch testen of bomen, gemaakt door de code van studen-

1

ten, inderdaad rood-zwart-bomen zijn. In deze opgave wordt een gedeelte van deze test gemaakt.

Eén van de eigenschappen van een rood-zwart-boom is dat rode knopen niet aan elkaar grenzen.

We nemen aan dat bladeren met waarde **nil** zwart zijn.

Geef pseudocode van een methode die, gegeven de wortel $root(T)$ van een binaire zoekboom waarbij elke interne knoop een kleur *rood* of *zwart* heeft, test of er geen rode knopen aan elkaar grenzen. Je methode moet in $O(n)$ tijd werken op een boom met n knopen; moet als antwoord *true* geven als er geen rode knopen aan elkaar grenzen, en *false* als er wel rode knopen zijn die aan elkaar grenzen.

5. Bomen-II (2 pnt) Stel we hebben een binaire zoekboom, waarbij elke knoop drie pointers heeft: *parent* (naar de ouder in de boom; alleen de wortel heeft deze pointer niet); *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. Je mag aannemen dat alle knopen een verschillende keywaarde hebben.

Daarnaast heeft elke knoop een integer waarde *formaat*. Het *formaat* van een knoop x is het aantal interne knopen in de boom in de deelboom met x als wortel, of, met andere woorden: het aantal interne knopen y met x voorouder van y of $x = y$. (Bladeren met waarde **nil** tellen dus niet mee.)

Stel de boom heeft hoogte h. Je mag aannemen dat de boom niet leeg is, en we hebben een pointer root(T) die naar de wortel van de boom wijst.

Schrijf een methode aantalKleinerGelijk (in pseudocode of Java-code), die als invoer krijgt:

- de wortel van een binaire zoekboom
- een key x krijgt
- een integer r.

en test in $O(h)$ tijd of er minstens r elementen in de boom zitten wiens keys kleiner zijn aan x. Gegeven is dat x zelf niet in de boom voorkomt. (Je code hoeft dus niet correct te zijn als x in de boom zit; 't mag natuurlijk wel maar geeft geen bonuspunten.) Je methode moet dus een boolean als antwoord opleveren. Je methode moet $O(h)$ tijd gebruiken.

Bijvoorbeeld: als de waarden 5, 6, 15, 17, 27, 28, 30, 31 zijn opgeslagen, dan geeft de query aantalKleinerGelijk(root,7,3) *false*, en aantalKleinerGelijk(root,27,3) *true*.

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.

public bool aantalKleinerGelijk(node *root*, int x, int r)

2

6. Hash-tabellen (1.5 pnt) Voor een datastructuur die zoek-acties en invoegingen ondersteund wordt de volgende implementatie met behulp van hashtabellen met open addressing gekozen. (We nemen hier aan dat er géén weglatingen/deletions gedaan worden.)

We beginnen met een hashtabel van grootte 32. Iedere keer als we een element willen invoegen wordt getest of de load-factor minstens 0.8 zou worden. Als de load-factor kleiner dan 0.8 blijft wordt het element gewoon toegevoegd aan de tabel. Echter, als de load-factor groter wordt dan 0.8 dan gaan we in plaats van de huidige tabel een grotere tabel gebruiken. De nieuwe tabel is twee keer zo groot als de oude tabel.

De tabellen gebruiken open addressing.

Bijvoorbeeld, als we het 26e ($26 = \lceil 32 * 0.8 \rceil$) element invoegen, dan gaan we een tabel gebruiken van grootte 64. Als die tabel weer te vol wordt gaan we een tabel van grootte 128 gebruiken, enzovoorts.

Als bij een invoeging we naar een grotere tabel gaan, dan worden alle elementen overgezet van de oude tabel naar de nieuwe tabel.

(i) Wat is de verwachte tijd van een *onsuccesvolle zoekactie* als we n elementen hebben opgeslagen in de tabel, onder de aanname van *uniform hashing*? Kies een van de onderstaande antwoorden.

1. $\Theta(1)$
2. $\Theta(\log n)$
3. $\Theta(\sqrt{n})$
4. $\Theta(n)$
5. $\Theta(n \log n)$

(ii) Beargumenteer je antwoord van (i). Je mag hierbij verwijzen naar en gebruik maken van resultaten die in het college/boek behandeld zijn.

7. Bomen: analyse (1 punt) We kijken in deze opgave naar binaire bomen. Laat a_v het aantal interne knopen zijn in de deelboom met v als wortel. (We tellen dus v en alle afstammelingen van v; bladeren met waarde **nil** tellen niet mee.)

Noem een binaire boom 2/3-gebalanceerd als voor elke knoop v geldt: voor elk kind w van v geldt $a_w \leq \frac{2}{3} a_v$.

Toon aan dat een binaire boom die 2/3-gebalanceerd en n knopen heeft diepte $O(\log n)$ heeft. (Hint: bewijs dat een 2/3-gebalanceerde boom met diepte r minstens 1.5^r knopen heeft.)