2e deeltentamen Datastructuren 2010/2011

Je hebt 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.

Schrijf op elk blad je naam, en op het eerste vel je collegekaartnummer. 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!

1. Basiskennis (2 pnt) (a) Stel op een lege queue Q worden achtereenvolgens de volgende operaties uitgevoerd: Enqueue(Q,42); Enqueue(Q,47); Enqueue(Q,13); Dequeue(Q). **Wat is de uitvoer van de laatste operatie?**

(b) Stel op een lege stack S worden achtereenvolgens de volgende operaties uitgevoerd: Push(S,42); Push(S,47); Push(S,13); Pop(S). **Wat is de uitvoer van de laatste operatie?**

(c) **Bij welke soorten van hashtabel kan zich het verschijnsel van *primary clustering* (in het college: *primaire klontvorming* genoemd) voordoen:**

- (a) Hashtabellen met chaining
- (b) Hashtabellen met open addressing
- (c) Beide soorten hashtabellen

Hangt dit af van de gebruikte hashfunctie? Licht je antwoord kort toe. (Maximaal 8 regels.)

(Schrijf op: a, b of c; en dan ja of nee. Geef daarna een korte toelichting.)

Opgave 2 (1.25 pnt)

Bij een bordspel hebben spelers steeds zeven vierkante steentjes. Op elk van deze steentjes staat één van de letters van het alfabet. Met deze steentjes worden woorden gemaakt. De spelers kunnen ook letters die al op het bord liggen gebruiken; we nemen hier aan dat ze hooguit drie andere letters gebruiken, en dat woorden dus hooguit tien letters lang zijn.

Er zijn een aantal regels welke woorden toegestaan zijn.

Voor het maken van een computerspelversie van dit bordspel willen we controleren of een bepaald woord toegestaan is. Hiervoor wordt een hashtabel gebruikt. In deze hashtabel worden alle toegestane woorden opgeslagen. Als de speler dan een bepaald woord opgeeft, dan wordt opgezocht of het woord in de tabel voorkomt, op de manier zoals dat gebruikelijk is bij hashtabellen.

Er wordt hashing met chaining gebruikt.

Er worden 100.000 woorden worden opgeslagen. De tabelgrootte is ook 100.000. Als hashfunctie wordt het volgende genomen. Elke letter wordt 'vertaald' naar een getal tussen 1 en 26, naar gelang de positie in het alfabet; met andere woorden, we nemen $a(\text{"A"}) = 1$, $a(\text{"B"}) = 2$, ..., $a(\text{"Z"}) = 26$. Voor een gegeven woord nemen we elk van de letters dit getal, en die getallen tellen we bij elkaar op. Dit geeft de hashwaarde van het woord.

Voorbeeld: Het woord "BY" heeft als hashwaarde 2+25=27. De hashwaarde van "AA" is 1+1=2.

Welke nadelen zie je voor het op deze manier (met deze tabelgrootte en deze hashfunctie) gebruiken van hashing voor deze toepassing? Leg kort uit (maximaal 10 regels.)

~~*Voorbeeld:* Het woord "BY" heeft als hashwaarde $1 \cdot 2 + 2 \cdot 25 = 52$.~~

Is dit een goede manier om een hashtabel te gebruiken voor deze toepassing? Waarom wel / waarom niet? (Maximaal 10 regels.)

Deze vraag vervalt

Opgave 3 (2 pnt)

Stel we hebben een roodzwartboom. We nemen aan dat elke knoop drie pointers heeft: parent (naar de ouder in de boom; bij de wortel wijst deze pointer naar NIL); left (naar het linkerkind, kan NIL zijn), en right (naar het rechterkind, kan NIL zijn). Iedere knoop heeft een int *key*; deze voldoen aan de eigenschap van zoekbomen. Daarnaast heeft elke knoop een kleur *rood* of *zwart*. Het is mogelijk dat keys vaker dan één keer voorkomen. Stel de boom heeft n knopen.

Iemand wil berekenen wat het maximum aantal rode knopen is dat ligt op een pad van de wortel naar een blad. **Geef een methode die dit aantal uitrekent in $O(n)$ tijd.**

Geef je methode in pseudocode **en** licht kort toe wat je methode doet/waarom deze correct werkt.

Je hoeft de tijdgrens niet toe te lichten.

(Hint: het is niet nodig om de eigenschappen van een roodzwartboom te gebruiken; het kan wel.)

Opgave 4 (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

2

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. Alle keywaarden zijn verschillend.

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 voorouder van y of $x = y$.

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.

Geef in pseudocode een methode die als invoer een key x krijgt, en

- **een foutmelding geeft als x niet voorkomt, en**
- **als x wel voorkomt, het aantal keys geeft dat groter is dan x, en**
- **$O(h)$ tijd gebruikt.**

Bijvoorbeeld: als de waarden 20, 30, 50, 70 opgeslagen zijn, en de methode als invoer 30 krijgt, dan heeft het als uitvoer 2, want er zijn twee opgeslagen keys groter dan 30.

Je hoeft de tijdgrens niet toe te lichten.

Opgave 5 (1.5 pnt) In het kader van een experimentele analyse van roodzwartbomen wil iemand voor elke knoop in een roodzwartboom bijhouden wat de zwarthoogte (in het Engels: *black-height*) is.

We hebben een roodzwartboom. Iedere knoop heeft de pointers naar de ouder, het linkerkind en het rechterkind, de variabele die de *key* aangeeft, en de variabele die de kleur (rood of zwart) aangeeft. Daarnaast heeft iedere knoop een variabele zh, die de zwarthoogte van de knoop aangeeft.

Is het mogelijk om invoegingen en weglatingen te doen in $O(\log n)$ tijd zodat de variabelen zh steeds de zwarthoogtes blijven weergeven?

Zo ja: leg duidelijk uit waarom/hoe dit kan.

Zo nee: waarom niet.

Bij je antwoord mag je gebruik maken van resultaten die in het college zijn behandeld. Let er wel op dat je voldoende details geeft, en leg ook uit welke resultaten uit het college je gebruikt en hoe je dit doet.

Opgave 6 (1.25 pnt) In deze opgave kijken we naar de skiplists, zoals besproken in het college. Stel we slaan n keys op in een skiplist. De kans p die gebruikt wordt in het algoritme bij invoegingen wordt op $p = \frac{1}{2}$ gesteld.

Zoals besproken zit elke key in de onderste laag, dus de onderste laag heeft n keys.

(a) **Hoe wordt bij het invoegen bepaald of een key ook op een hogere laag voorkomt?**

(b) **Wat is het verwachte aantal keys in de een-na-onderste laag?** Geef het antwoord *zonder* gebruik te maken van O-notaties.

Leg je antwoord uit.