Deeltentamen Datastructuren

3 juli 2003, 14.00 - 16.00 uur

- Je mag geen boek, geen eigen aantekeningen en geen uitwerkingen van werkcollege opgaven raadplegen, en ook geen zakrekenmachine gebruiken.
- Er zijn **4** opgaven. Bij elke opgave staat het aantal punten dat je er mee kunt verdienen. Bedenk dat deze punten niet noodzakelijkerwijs de moeilijkheidsgraad van de opgave aangeven. De sommatie van de punten is 11.
- Je mag sommen maken in een volgorde die jou het beste uitkomt.
- Schrijf niet met rood en niet met potlood.
- Schrijf op ieder vel dat je inlevert je **naam**, en op het eerste vel je **collegekaartnummer**, het **totaal aantal vellen** dat je inlevert, en (zo je wenst) een **e-mailadres** waar je bericht van een officieus tentamencijfer wilt ontvangen.
- Bij inleveren dien je je **collegekaart** te laten zien.
- Algoritmen dienen geformuleerd te worden op een hoog niveau, eventueel in pseudocode, natuurlijk met de benodigde precisie. Beschrijving van algoritmen in JAVA is toegestaan, maar wordt door de docent niet aangemoedigd. Het schrijven van JAVA-code vereist veel precisie, en dus tentamentijd.
- Logaritmen hebben grondtal 2.

1 Ja/Nee vragen (1 punt)

Zeg van elk van onderstaande beweringen of hij waar (W) of onwaar (N) is.
SCORE: k goede antwoorden levert een score van $0.25 \max(0, k-2)$ punt op.

1. Bewering: Bij het doorlopen van een binaire boom in preorder kom je elke interne knoop eerder tegen dan elke externe knoop.

2. Bewering: Voor de functie $f(n) = 1000(\log n)^{1000}$ geldt $f(n)$ is $O(\frac{n^{1/1000}}{1000 \log n})$.

3. Bewering: Voor elk positief geheel getal h is er een AVL-boom van hoogte h die minder dan h^5 interne knopen heeft.

4. Bewering: Voor elke boom T en elke interne knoop u in T geldt: $height(u) = height(T) - depth(u)$

5. Zij gegeven twee functies f en g, met elk als parameter een niet-negatieve integer, en als uitkomst ook niet-negatieve integer.
   Bewering: Als $f(n)$ is $O((\log n)^2)$ en $g(n)$ is $O(n^3)$ dan is $f(g(n))$ van de orde $O(n)$.

6. Zij H een heap van n keys, op de standaard manier opgeslagen in een array.
   Bewering: Het verwijderen van het element met de hoogste prioriteit (i.e., met de kleinste key) uit H en het herstellen van H tot een heap werkt in $O((\log n)^2)$ tijd.

1

2 Zoekbomen en traversals

In deze opgave kijken we naar ordered dictionaries met als elementen integers. We willen op zo'n dictionary D een extra operatie toestaan:

$D.partialsums(y_1, y_2)$ geeft als resultaat: $\sum\{x \in D : y_1 \leq x \leq y_2\}$

Als concrete datastructuur nemen we een binaire boom met per interne knoop v:

– een veld *key* van het type integer, en
– een veld *som* van het type integer, en
– een pointer *left* naar het linker kind van v, en
– een pointer *right* naar het rechter kind van v.

Definitie
We zeggen dat een boom T met bovengenoemde velden per interne knoop, een **ps-boom** is als:

1. T heeft zodanige *key*'s dat het een zoekboom is (i.e., de *key*-velden voldoen aan de zoekboom eigenschap), èn

2. het *som*-veld in elke interne knoop v voldoet aan de *sommatie-eigenschap*:

$$v.som = \sum\{w.key \mid w \text{ is een interne afstammeling van } v\}$$

einde definitie
Bedenk dat elke knoop een afstammeling van zichzelf is.

(**a**) **(1.5 punt)** Zij T een propere binaire boom met de velden *key*, *som*, *left*, *right* en met in totaal n interne knopen. Geef een algoritme in pseudo-code dat test of T een ps-boom is, en zo niet print dan de keys van precies die knopen v waarvoor geldt dat $T(v)$ geen ps-boom is terwijl de twee deelbomen van v wel ps-bomen zijn. Je algoritme moet werken in tijd $O(n)$.

Beargumenteer dat je algoritme correct is, en voldoet aan de tijdgrens.

We gaan ordered dictionaries D met integer elementen opslaan in ps-bomen T. De dictionary zal nooit dubbele elementen met dezelfde waarde bevatten. Elk element $d \in D$ wordt opgeslagen in het *key*-veld van precies één interne knoop, en elke interne knoop zal een element van D bevatten. Operaties op D worden dan natuurlijk feitelijk uitgevoerd op de ps-boom T. Neem aan dat T hoogte h heeft.

(**b**) **(1.5 punt)** Geef een algoritme dat in $O(h)$ tijd een gegeven element k toevoegt aan T. Het is niet uitgesloten dat de door T gerepresenteerde dictionary k al bevat. Na afloop van je algoritme moet T nog steeds een ps-boom zijn.

Beargumenteer dat je algoritme correct is, en voldoet aan de tijdgrens.

(**c**) **(1.5 punt)** Geef een $O(h)$ tijd algoritme dat de operatie $partialsums(y_1, y_2)$ voor gegeven y_1 en y_2 uitvoert op T.

Beargumenteer dat je algoritme correct is, en voldoet aan de tijdgrens.

(**d**) **(1.5 punt) voor de liefhebber**
Geef een $O(h)$ tijd algoritme dat voor gegeven key x en gegeven integer K de grootste key $y \in D$ bepaalt zodanig dat $D.partialsums(x, y) \leq K$. (Natuurlijk met correctheidsuitleg en bewijs van behaalde tijdgrens.)

3 Sorteren

(**a**) **(0.5 punt)** Geef de definitie wanneer een sorteer algoritme een comparisonsort is.

(**b**) **(2 punten)** Toon aan dat elk comparisonsort algoritme toegepast op n items in het slechtste geval $\Omega(n \log n)$ tijd vergt.
OPMERKING. Je mag hierover niet meer dan één kantje A4 in **klein** handschrift over vol schrijven. Wees dus compact in je formuleringen en bedenk wat je wel en niet opschrijft.

4 Heaps

(**a**) (0.5 punt) Geef de definitie van een heap.

(**b**) (0.5.punt) Beschrijf in het kort de standaard array-implementatie van een heap.

(**c**) (1.0 punt) Gegeven een heap H van n items, en gegeven een verwijzing p naar een interne knoop in H. We willen een extra operatie toestaan $removeItem(p)$ die het item in de knoop aangewezen door p verwijdert uit de heap, en er voor zorgt dat H na afloop nog steeds een heap is.
Geef een efficient algoritme voor de operatie $removeItem$. Toon aan dat je algoritme in een array-implementatie voor de heap, werkt in $O(\log n)$ tijd.

========== Einde deeltentamen juli 2003 ==========