

voortentamen datastructuren en algoritmen 27 september 2010

veel succes !

Opgave 1.

- (a) Vervang de vraagtekens door een zo eenvoudig mogelijk expressie:

$$5 \cdot n + 7 \in \mathcal{O}(?)$$

$$5 \cdot n + 7 \cdot n^2 \in \mathcal{O}(?)$$

$$5 \cdot n + 7 \cdot \log n \in \mathcal{O}(?)$$

(9 punten)

- (b) Wat is de worst-case tijdscomplexiteit, uitgedrukt in \mathcal{O} , van het volgende algoritme?

Algorithm loopje(n):

$p := 0$

for $i = 1$ **to** n **do**

$p := p + i$

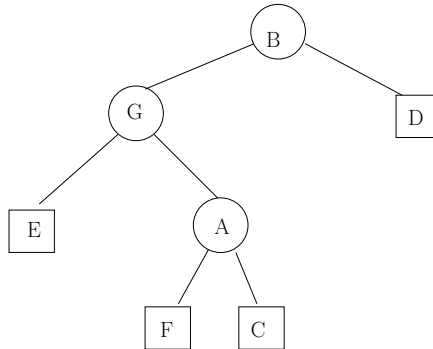
(6 punten)

Opgave 2. We hebben twee stacks tot onze beschikking met alle operaties uit het ADT voor stacks, zoals bijvoorbeeld *push*, *pop*, en *isEmpty*. Implementeer hiermee de operaties *enqueue* en *dequeue* van het ADT voor queues.

(15 punten)

Opgave 3. NB: Deze opgave gaat over algemene binaire bomen; hier zijn zowel de interne als de externe knopen gelabeld. In de heaps (opgave 4) en in de binaire zoekbomen (opgave 7) zijn alleen de interne knopen gelabeld.

- (a) Geef voor de volgende binaire boom de volgorde waarin de knopen bezocht worden bij een postorder traversaal:



(7 punten)

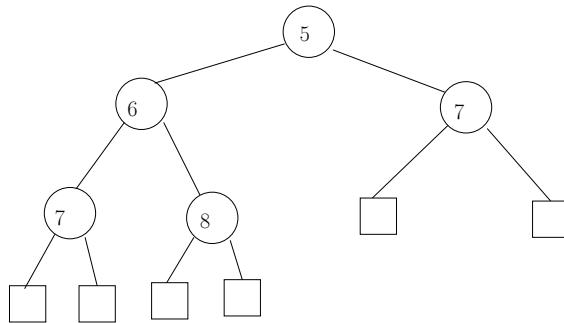
- (b) Geef de pseudo-code voor postorder traversal van een binaire boom. De input is: een boom T en een knoop v in die boom. Je kunt gebruikmaken van de operaties uit het ADT voor binaire bomen, zoals bijvoorbeeld *leftChild* en *rightChild*.

(8 punten)

Opgave 4.

- (a) Voeg aan de volgende heap een knoop met key 6 toe, en vervolgens een knoop met key 1.

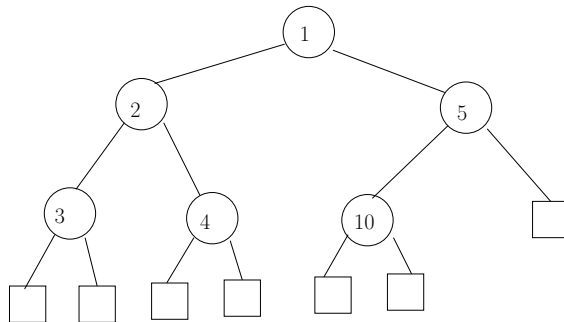
(Je hoeft alleen het eindresultaat van deze twee toevoegingen te geven en niet de tussenstappen.)



(8 punten)

- (b) Verwijder uit de volgende heap de wortel.

(Je hoeft alleen het eindresultaat van deze verwijdering te geven en niet de tussenstappen.)



(7 punten)

- (c) Bewering: een preorder traversal van een heap geeft de elementen in niet-dalende volgorde.

Toon met een voorbeeld aan dat deze bewering onwaar is.

(5 punten)

Opgave 5. Gegeven is de pseudo-code voor het algoritme *PriorityQueueSort* voor het sorteren van een rijtje C gebruikmakend van een priority queue P :

Algorithm *PriorityQueueSort*(C, P):

```
while not  $C$ .isEmpty() do
     $e := C$ .removeFirst
     $P$ .insertItem( $e, e$ )
while not  $P$ .isEmpty() do
     $e := P$ .removeMin()
     $C$ .insertLast( $e$ )
```

We implementeren de priority queue met een ongeordend rijtje. Wat is dan de complexiteit van *PriorityQueueSort* in termen van \mathcal{O} ? Beargumenteer je antwoord.

(9 punten)

Opgave 6. Maak een hash-table ter grootte 7 en gebruik de hash-functie $h(k) = k \bmod 7$. Voeg (in deze volgorde) de volgende getallen toe

1, 2, 8, 9, 11

waarbij collision wordt opgelost

- (a) met chaining (geef alleen het eindresultaat),
(b) met linear probing (geef alleen het eindresultaat).

(10 punten)

Opgave 7. Geef het eindresultaat van de binaire zoekboom die je krijgt door (in deze volgorde) de volgende getallen aan een initieel lege boom toe te voegen:

6, 7, 4, 9, 8, 5, 2, 3, 1

(6 punten)

Het cijfer is (het totaal aantal punten plus 10) gedeeld door 10.