ALGORITHMS & DATA STRUCTURES

LES 5 : TOEPASSINGEN BINAIRE BOMEN

THEORIE

Actie	**Hoofdstuk**	**2010**	**2012**
Bestudeer	Binary search trees	19.1 – 19.4	18.1 – 18.4
Bestudeer	A priority queue: the binary heap	21.1 – 21.3	20.1 – 20.3

TIP

Controleer je antwoord op de vragen zo mogelijk met behulp van de volgende applets:

- http://people.ksp.sk/~kuko/bak/index.html
- http://webdiis.unizar.es/asignaturas/EDA/AVLTree/avltree.html

THEORIE OPGAVEN

OPGAVE 1 (19.4)

Show the result of inserting 3, 1, 4, 6, 9, 2, 5 and 7 in an initially empty binary search tree. Then show the result of deleting the root.

OPGAVE 2

Voeg aan een lege AVL-boom de volgende elementen toe:

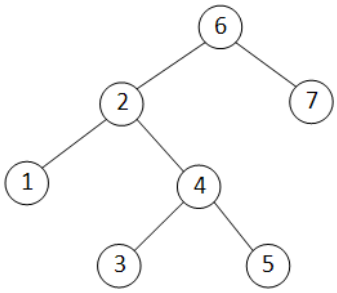
2, 5, 4, 1, 9, 6, 7, 13, 8

OPGAVE 3 (21.1)

Show the result of inserting 14, 6, 5, 8, 1, 3, 12, 9, 7, 13 and 2, one at the time, in an initially empty heap. Then show the result of using the linear-time `BuildHeap` algorithm instead.

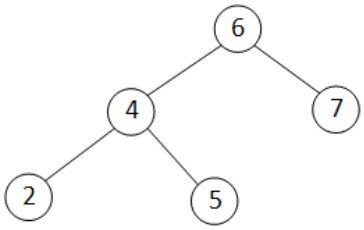
OPGAVE 4

Is de onderstaande boom een AVL-boom? Licht je antwoord toe.



OPGAVE 5

De onderstaande figuur laat een AVL-boom zien. Teken de boom nadat een node met waarde 1 is toegevoegd (zonder dat rotation is uitgevoerd). Is de boom nog steeds een AVL-boom? Als dat niet het geval is pas dan rotatie toe om de boom te corrigeren.



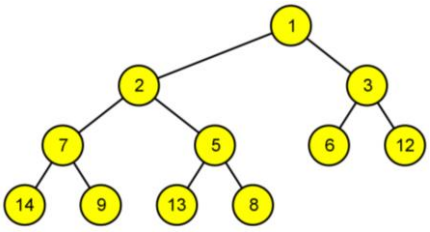
OPGAVE 6

Voeg de volgende items één voor één toe aan een AVL-boom.

10, 1, 15, 5, 2, 3, 20, 6

OPGAVE 7

Gegeven de volgende priority queue:



Toon het resultaat van een `deleteMin`-operatie. Doe dit in totaal 4 keer achter elkaar.

PRAKTIJK

OPDRACHT 1 (KAN NODIG ZIJN VOOR PRACTICUM!)

a) Implementeer de klasse `BinarySearchTree`. Deze klasse is een implementatie van `IBinarySearchTree`. Uiteraard is `BinarySearchTree` een specificatie van de `BinaryTree` die je vorige week hebt gemaakt. Zorg dat je de verschillende exceptions afvangt (zie de interface).
b) Voeg aan de klasse een `ToString()` method toe. Deze method retourneert "" in het geval van een lege boom. In alle andere gevallen dienen alle elementen, gescheiden door een spatie, in inorder volgorde te worden opgeleverd (bijvoorbeeld "2 4 5 6 7" voor de boom uit opgave 5).
c) Test je datastructuur met de bijgevoegde unittests.

OPDRACHT 2 (KAN NODIG ZIJN VOOR PRACTICUM!)

a) Implementeer de klasse `PriorityQueue`. Deze klasse is een implementatie van `IPriorityQueue`.
b) Voeg aan de klasse een `ToString()` method toe. Deze method retourneert "" voor een lege queue. In de andere gevallen wordt een lijst van elementen opgeleverd, gescheiden door een spatie. De volgorde is hetzelfde als de volgorde van de interne array. De queue uit opgave 7 zal dus het volgende resultaat opleveren:

```
1 2 3 7 5 6 12 14 9 13 8
```

Tip: Maak met het oog op de volgende opdracht in ieder geval een aparte method `PercolateDown(int node)`
c) Test je datastructuur met de bijgevoegde unittests.

OPDRACHT 3

In paragraaf 21.3 wordt een method BuildHeap beschreven, die in lineaire tijd een ongeordende lijst kan structureren tot een Binary Heap. Voeg aan de Binary Heap die je in de vorige opdracht hebt gemaakt een method `AddFreely(T x)` toe, die een nieuwe integer toevoegt aan het eind van de array ongeacht de grootte van de integer (de spelregels voor de Binary Heap worden dus niet in acht genomen). Implementeer vervolgens de method `BuildHeap` en test je code.

VOORBEREIDING VOLGENDE WEEK (GRAFEN)

Actie	**Hoofdstuk**	**2010**	**2012**
Bestudeer	Graphs and paths	14.1	13.1