

# Voorbeeldvragen Algoritmen & Datastructuren I

## Theorievragen

Titularis: Prof. Dr. Wolfgang De Meuter

Assistenten: Nathalie Oostvogels, Simon Van de Water

18 december 2015

### **Vraag 1**

Wat is de worst-case performantie van het KMP algoritme? Toon je antwoord formeel aan. Als houvast vind je in bijlage de code van het algoritme.



## Vraag 2

Op de achterzijde van deze pagina vind je de code van een in de cursus behandeld algoritme.

- a) Benoem het algoritme en leg kort (in ongeveer 5 lijnen) de werking ervan uit.
- b) Geef de worst-case performantie van het algoritme, en verklaar je antwoord.



### Vraag 3

- a) Hoeveel bladeren heeft een heap van  $n$  elementen? Toon je antwoord formeel aan.
- b) Waarom is dit een belangrijke eigenschap?

## Vraag 4

Bewijs dat geen enkel algoritme dat gebaseerd is op vergelijkingen sneller kan sorteren dan  $\Omega(n \log n)$

## Vraag 5

Toon wiskundig aan dat Quicksort een best-case performantie-karakteristiek heeft in  $O(n \log(n))$  en een worst-case performantie-karakteristiek in  $O(n^2)$ . We willen dus 2 afzonderlijke berekeningen zien.

## Vraag 6

Waar of vals? Indien waar, leg uit in een 5-tal zinnen. Indien vals, geef een tegenvoorbeeld of leg kort uit waarom.

- a) KMP zal ieder karakter van de tekst precies 1 keer vergelijken met een karakter uit het patroon.
- b) Binary Search is enkel toepasbaar op gesorteerde vectoren.
- c) Het omvormen van een willekeurige vector tot een heap (ook wel “to heapify” genoemd) neemt  $O(\log(n))$  tijd in beslag.
- d) Een randomized QuickSort zal nooit een  $O(n^2)$  gedrag vertonen.
- e) Iedere BST is automatisch ook een heap.
- f) Een heap is automatisch ook een BST.
- g) Een AVL boom is steeds ook een BST.
- h) Het vermijden van secondary clustering zorgt er automatisch voor dat primary clustering opgelost is.

## Vraag 7

Bewijs dat de procedure `from-scheme-vector` (hieronder afgebeeld) een worst-case performantiekarakteristiek heeft van  $O(n)$ .

```
(define (from-scheme-vector vector <<?)  
  (define size (vector-length vector))  
  (define heap (make vector size <<?))  
  (define (iter index)  
    (sift-down heap index)  
    (if (> index 1)  
        (iter (- index 1))))  
  (iter (div size 2))  
  heap)
```