



*Dit tentamen bestaat uit 7 opgaven.*

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

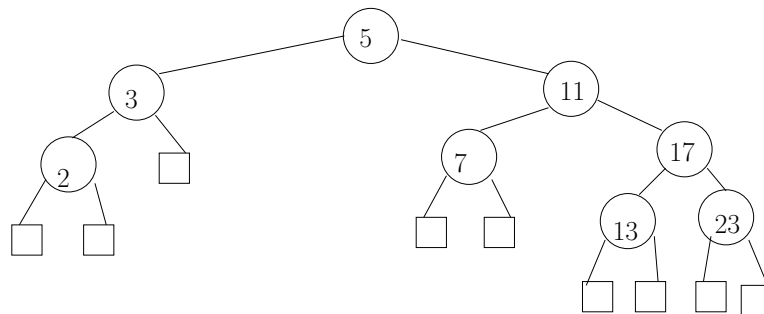
*Veel succes!*

---

**Opgave 1.** (5+5 punten)

Deze opgave gaat over AVL-bomen.

- (a) Geef een klein voorbeeld van het verwijderen van een knoop uit een AVL-boom waardoor de boom uit balans raakt. Herbalanceer de boom.
- (b) Gegeven is de volgende AVL-boom:



Geef met plaatjes stap voor stap aan hoe een knoop met key 14 wordt toegevoegd.

**Opgave 2.** (6+4 punten)

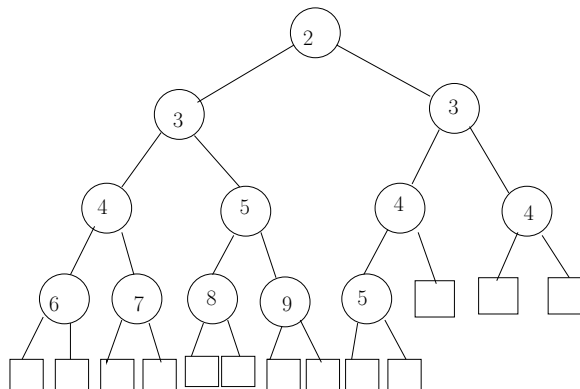
Deze opgave gaat over stacks en queues.

- (a) Gegeven zijn twee queues met operaties `enqueue` en `dequeue` in  $\mathcal{O}(1)$ . Implementeer hiermee de operaties `push` en `pop` van het abstracte datatype voor stacks.
- (b) Geef de worst-case tijdscomplexiteit van je implementaties uit (a), met een korte toelichting.

**Opgave 3.** (5+4+6 punten)

Deze opgave gaat over heaps.

- (a) Verwijder in de volgende max-heap de wortel (geef alle stappen):



- (b) Geef de vector-representatie van de oorspronkelijke max-heap uit (a).  
(c) Beschrijf het principe van heap-sort (hoeft niet in pseudo-code).

**Opgave 4.** (5+4+6 punten)

Deze opgave gaat over binaire bomen met labels op de interne en externe knopen.

- (a) Gegeven is dat de preorder traversal van een binaire boom  $B$  de rij 7, 6, 5, 4, 3, 2, 1 oplevert. Geef drie verschillende mogelijkheden voor  $B$ .  
(b) Er is nu bovendien gegeven dat de inorder traversal van  $B$  de rij 5, 6, 4, 7, 2, 3, 1 oplevert. Wat is  $B$ ? Zijn er meer mogelijkheden voor  $B$ ? Waarom (niet)?  
(c) Geef pseudo-code voor een level-order traversal van een binaire boom.

**Opgave 5.** (5+5+5 punten)

Een gemengde opgave.

- (a) Is een algoritme in  $\mathcal{O}(n)$  *altijd* sneller dan een algoritme in  $\mathcal{O}(n^2)$ ? Licht je antwoord kort toe.

(b) De recurrente betrekking die bij merge-sort hoort is:

$$T(n) = \begin{cases} 1 & \text{als } n = 1 \\ 2T(\frac{n}{2}) + n & \text{als } n > 1 \end{cases}$$

Los deze recurrente betrekking op, en geef aan wat dus de tijdscomplexiteit van merge-sort is in termen van grote- $\mathcal{O}$ .

(c) We bekijken een stack met behalve de gebruikelijke operaties **push** en **pop** ook de operatie **multipop**:

**Algorithm** multipop( $k$ ):

```
while not isEmpty() and  $k \geq 0$  do
  pop()
   $k := k - 1$ 
```

Leg uit waarom in een rijtje van  $n$  operaties de amortized complexiteit per operatie in  $\mathcal{O}(1)$  is.

**Opgave 6.** (5+5+5 punten)

Deze opgave gaat over gulzige (greedy) algoritmen en dynamisch programmeren.

(a) Pas het algoritme voor knapsack01 toe op de volgende verzameling  $S$ , met items  $s_i$  met benefit  $b_i$  en gewicht  $w_i$ :

	$b$	$w$
$s_1$	2	1
$s_2$	5	3
$s_3$	4	2

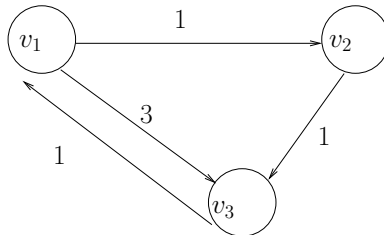
met maximaal totaalgewicht  $W = 4$ .

Geef je antwoord in de vorm van een tabel:

$k \setminus w$	...
$\vdots$	

en licht het totstandkomen van de waarde van het hokje met  $k = 3$  en  $w = 3$  kort toe.

- (b) Pas het dynamic programming algoritme voor het vinden van alle kortste paden toe op de volgende graaf:



- (c) Gegeven: een verzameling taken, elk met begin- en eindtijd. Geef een greedy (scheduling) algoritme dat een zo groot mogelijke deelverzameling taken geeft die op één machine uitgevoerd kunnen worden (sequentieel).

**Opgave 7.** (5+5 punten)

Deze opgave gaat over het Boyer–Moore pattern-matching algoritme, waarbij een functie *last* gebruikt wordt om het patroon handig op te schuiven.

- (a) Gegeven zijn de volgende tekst  $T$  en patroon  $P$ . Pas het Boyer–Moore pattern-matching algoritme toe; geef en nummer alle stappen.

$$\begin{aligned} T &= abaddbacabadabac \\ P &= abac \end{aligned}$$

- (b) Wat is de worst-case tijdscomplexiteit van het brute-force pattern-matching algoritme?

Geef een voorbeeld dat illustreert dat deze afchatting scherp is.