



*Dit tentamen bestaat uit 8 opgaven.*

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

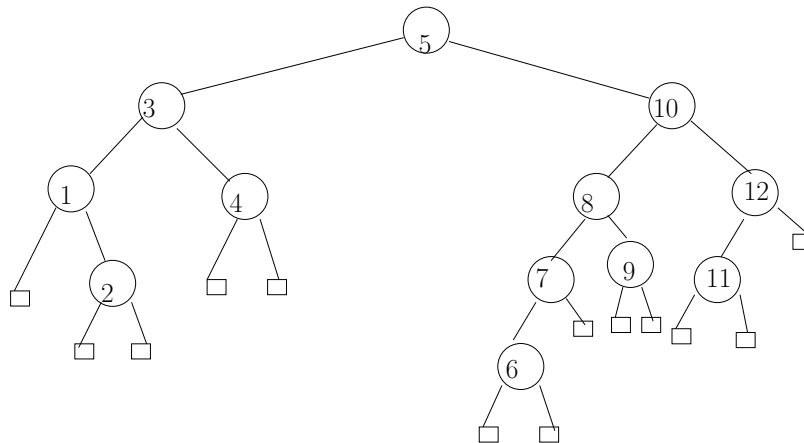
*Veel succes!*

---

**Opgave 1.** (2+2+6 punten)

Deze opgave gaat over binaire zoekbomen en AVL-bomen. De interne knopen zijn gelabeld met keys; de externe knopen zijn leeg.

- (a) Gegeven een binaire zoekboom met  $n$  interne knopen. Wat is de worst-case hoogte in termen van grote- $O$ ?
- (b) Gegeven een AVL-boom met  $n$  interne knopen. Wat is de worst-case hoogte in termen van grote- $O$ ?
- (c) Gegeven is de volgende AVL-boom:

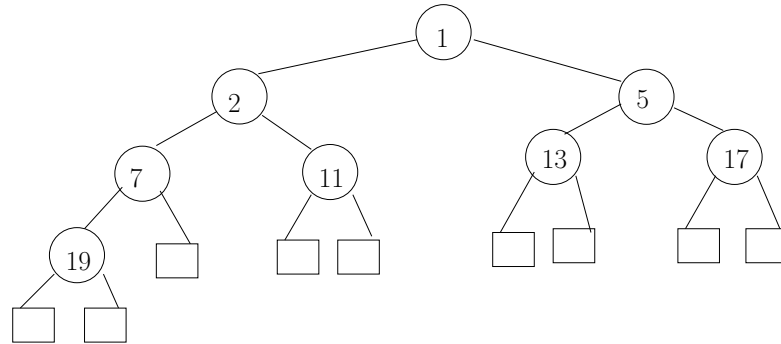


Verwijder de knoop met key 4 en herbalanceer zodat het weer een AVL-boom wordt; geef alle stappen.

**Opgave 2.** (5+5 punten)

Deze opgave gaat over heaps. De interne knopen zijn gelabeld met keys; de externe knopen zijn leeg.

- (a) Gegeven is de volgende min-heap:



Verwijder de wortel en herstel de min-heap-eigenschappen door bubbelen. Geef het eindresultaat, en geef het totaal aantal bubbel-stappen.

- (b) Heap-sort kan gezien worden als priority-queue-sort waarbij de priority queue geïmplementeerd is met een (min- of max-)heap. Geef en verklaar de worst-case tijdscomplexiteit van heap-sort in termen van grote- $O$ .

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

Deze opgave gaat over stacks, queues, en amortized complexiteit.

- (a) Gegeven zijn twee stacks met operaties `push` en `pop` in  $\mathcal{O}(1)$ . Implementeer hiermee (efficiënt) de operaties `enqueue` en `dequeue` van het abstract datatype voor queues.
- (b) Geef de amortized (uitgesmeerde) complexiteit van de operaties `enqueue` en `dequeue` uit je antwoord op de vorige opgave.

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

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

- (a) Geef pseudo-code voor een recursief algoritme voor inorder-traversal van een (complete) binaire boom.
- (b) Geef pseudo-code voor een niet-recursief algoritme voor inorder-traversal van een (complete) binaire boom; gebruik bijvoorbeeld een stack.

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

Een gemengde opgave.

- (a) Geef een voorbeeld van een (bekend) algoritme met worst-case tijdscomplexiteit in  $\mathcal{O}(n^2)$ , dat vaak veel beter presteert dan in de worst-case. Geef om dat te illustreren ook een voorbeeld van een input voor het algoritme waarvoor de tijdscomplexiteit in  $\mathcal{O}(n \log(n))$  is.
- (b) Gegeven is een array  $A$  ter lengte  $n$  van gesorteerde integers (die kunnen negatief zijn). Geef een algoritme (mag maar hoeft niet in pseudo-code) in  $\mathcal{O}(\log n)$  dat een index vindt met  $A[i] = i$  als zo'n  $i$  bestaat.

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

Deze opgave gaat over greedy (gulzige) algoritmen. Gegeven is een verzameling taken, elk met begintijd en eindtijd. Twee taken zijn *compatibel* als ze sequentieel uitgevoerd kunnen worden op één machine. Het doel is om een zo groot mogelijke deelverzameling van onderling compatibele taken te vinden.

- (a) Geef een voorbeeld waaruit blijkt dat herhaaldelijk kiezen voor een compatibele taak met de kleinste starttijd niet leidt tot een optimale oplossing.
- (b) Geef een voorbeeld waaruit blijkt dat herhaaldelijk kiezen voor een compatibele taak met een kortste duur niet leidt tot een optimale oplossing.
- (c) Welke (greedy) keuze geeft wel een algoritme om een grootste deelverzameling van compatibele taken te vinden? Licht kort toe.

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

Deze opgave gaat over dynamisch programmeren.

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

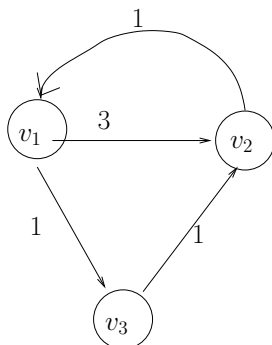
	$b$	$w$
$s_1$	3	1
$s_2$	6	3
$s_3$	5	2

met maximaal totaalgewicht  $W = 5$ . Geef je antwoord in een tabel:

$k \setminus w$	...
:	

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

- (b) Pas het algoritme voor het vinden van alle kortste paden ('all-pairs shortest paths', voor een 'afstandstabel') toe op de volgende graaf:



- (c) Geef een algoritme in  $\mathcal{O}(n)$  voor het berekenen van het  $n$ -de Fibonacci getal. De Fibonacci-getallen zijn als volgt gedefinieerd:

$$\begin{aligned}
 F(1) &= 1 \\
 F(2) &= 1 \\
 F(n) &= F(n-1) + F(n-2) \quad (\text{voor } n > 2)
 \end{aligned}$$

**Opgave 8.** (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.

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

- (b) Geef een voorbeeld van een tekst  $T$  en een patroon  $P$  waarvoor het Boyer–Moore pattern-matching algoritme niet significant efficiënter werkt dan brute-force pattern-matching.