



Opgave 1. (*4+5 punten*)

Deze opgave gaat over lineaire datastructuren.

- (a) Een standaard manier om het Abstract Data Type (ADT) voor stacks te implementeren is met een array ter lengte N . Geef in deze setting de implementaties van de operaties **push** en **pop**; gebruik een variable t voor de top van de stack.
- (b) Nu bekijken we een alternatieve manier om stacks te implementeren. We hebben twee queues tot onze beschikking, met operaties als gespecificeerd in het ADT voor queues. Implementeer hiermee de operaties **push** en **pop** van het ADT voor stacks.

Geef de tijdscomplexiteit van **push** en **pop** in termen van grote- $\mathcal{O}$.

Opgave 2. (*4+5+4 punten*)

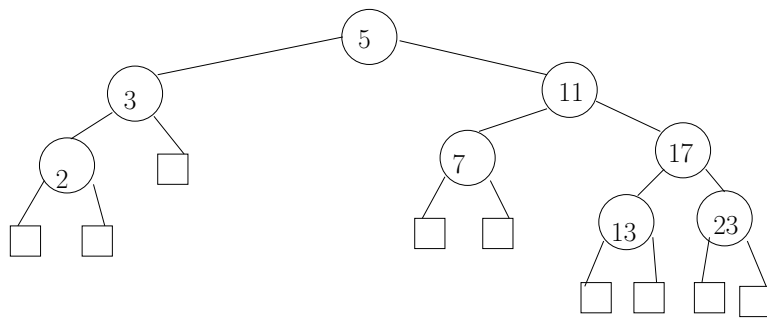
Deze opgave gaat over heaps en heap-sort. In heaps zijn de interne knopen gelabeld met een geheel getal, en de externe knopen zijn leeg.

- (a) Geef drie verschillende max-heaps met de getallen 1, 2, 3, 4, 5, 6, 7.
- (b) Pas heap-sort toe op de max-heap $[-, 4, 3, 2, 1]$.
Geef alle stappen expliciet naar keuze met plaatjes of in de array-representatie.
- (c) Wat is de worst-case tijdscomplexiteit van heap-sort?
Licht je antwoord kort toe.

Opgave 3. (5+4+4)

Deze opgave gaat over binaire zoekbomen en AVL-bomen. Elke knoop heeft 0 of 2 kinderen. Interne knopen zijn gelabeld met positieve gehele getallen, en externe knopen zijn leeg.

- (a) Geef pseudo-code voor een algoritme dat als input neemt een binaire zoekboom T en een knoop v in T , en dat als output geeft de knoop met de kleinste key uit de sub-boom van T die begint in v .
- (b) Gegeven is de volgende AVL-boom:



Verwijder de knoop met key 2. Herbalanceer zo nodig, en geef alle stappen expliciet in plaatjes.

- (c) Wat is de worst-case tijdscomplexiteit voor het verwijderen van een key aan (i) een gesorteerd array (geordend dictionary), (ii) een binaire zoekboom, (iii) een AVL-boom? (Alleen het antwoord volstaat.)

Opgave 4. (4+4+4 punten)

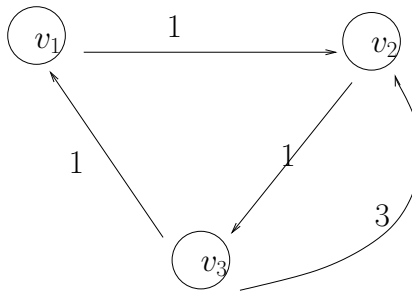
Deze opgave gaat over sorteren.

- (a) Geef een concreet worst-case input-rijtje voor bubble sort ter lengte 5.
Hoe ziet een worst-case input-rijtje voor bubble sort er in het algemeen uit ?
- (b) Beargumenteer dat bubble sort in $\mathcal{O}(n^2)$ is.
- (c) Geef de quick-sort boom voor het sorteren van de input-rij $[3, 6, 1, 7, 5, 8, 2, 4]$.
Neem als pivot steeds het laatste element,

Opgave 5. (5+4 punten)

Deze opgave gaat over bomen en grafen. Alle knopen zijn gelabeld.

- (a) Geef de pseudo-code voor inorder traversal van een binaire boom, zonder recursie, maar gebruikmakend van een stack. (Elke knoop heeft 0 of 2 kinderen.)
- (b) Pas het dynamic programming algoritme voor het verkrijgen van alle kortste paden (All-Pair Shortest Paths in het boek) toe op de volgende graaf:



Opgave 6. (4+4+5 punten)

Gegeven is het dynamic programming algoritme van Kadane voor het berekenen van een grootste som van een sub-array:

Algorithm maxSubArray(A, n):

new B

$B[0] := \max(A[0], 0)$

$m := B[0]$

for $r = 1$ **to** $n - 1$ **do**

$B[r] := \max(0, B[r - 1] + A[r])$

$m := \max(m, B[r])$

return m

- (a) Pas het algoritme toe op het array $[-1, 2, 3, -4]$. Geef je berekeningen voor het verkrijgen van de $B[i]$.
- (b) Wat is de tijdscomplexiteit in termen van grote-Oh van het algoritme van Kadane? Licht je antwoord kort toe.
- (c) Pas het algoritme aan zodat het niet een grootste, maar een kleinste som van een sub-array geeft.

Opgave 7. (*4+5 punten*)

- (a) Is een algoritme in $\mathcal{O}(n)$ *altijd* sneller dan een algoritme in $\mathcal{O}(n^2)$? Licht je antwoord kort toe.
- (b) 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 8. (*4+4+4 punten*)

Deze opgave gaat over pattern matching.

- (a) Gegeven is het alfabet $\{0, 1\}$ en het patroon $P = 001$. Geef een tekst T van lengte tien met de eigenschap dat het zoeken van P in de tekst T met het brute-force algoritme in het slechtste geval (grootst mogelijke aantal stappen) valt. Licht kort toe.
- (b) Gegeven is het alfabet $\{0, 1\}$ en het patroon $P = 100$. Geef een tekst T van lengte tien met de eigenschap dat het zoeken van patroon P in de tekst T met het Knuth–Morris–Pratt algoritme (dat de failure functie gebruikt) evenveel stappen kost als met het brute-force algoritme. Licht kort toe.
- (c) Pas het Boyer–Moore pattern-matching algoritme toe op

$$\begin{array}{lcl} T & = & bbbaaacccabc \\ P & = & abc \end{array}$$

Geef en nummer alle stappen. Je hoeft de functie **last** niet expliciet te geven.

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