Voorbeeldexamen Algoritmen en Datastructuren 1

1. **Complexiteit van sorteeralgoritmen (2 pten.)**
   Beschouw het insertion-algoritme voor het sorteren van een rij $(a_1, \ldots, a_n)$:

   1: **for** i **from** 2 **to** n **do**
   2: Bepaal de positie $j \leq i$ waarvoor $a_{j-1} < a_i \leq a_j$
   3: Voeg a_i tussen op positie j, door de elementen $a_j, \ldots, a_{i-1}$ op te schuiven

   (a) Wat is de slechtst mogelijke uitvoeringstijd van dit algoritme? Verklaar.
   (b) Wat is de best mogelijke uitvoeringstijd van dit algoritme? Verklaar.
   (c) Wat is de gemiddelde uitvoeringstijd van dit algoritme? Bewijs.

2. **Gretige algoritmen: algoritme van Kruskal (2 pten.)**

   (a) Geef het algoritme (in pseudocode) van Kruskal voor het bepalen van een minimale-kost opspannende boom in een gewogen graaf.
   (b) Bewijs dat het algoritme van Kruskal correct is.

3. **Binaire zoekbomen (3 pten.)**

   (a) Gegeven een binaire zoekboom die gehele sleutels uit het interval $[0, 1000]$ bevat. We zoeken de sleutel 363. Welk van onderstaande sequenties kan/kunnen **niet** de sequentie van bezochte toppen zijn? Verklaar.
      i. 2, 252, 401, 398, 330, 344, 397, 363.
      ii. 924, 220, 911, 244, 898, 258, 362, 363.
      iii. 925, 202, 911, 240, 912, 245, 363.
      iv. 2, 399, 387, 219, 266, 382, 381, 278, 363.
      v. 935, 278, 347, 621, 299, 392, 358, 363.

   (b) Beschouw een top met twee kinderen in een binaire zoekboom. Toon aan dat zijn opvolger (d.i. de eerstvolgende sleutel uit de binaire zoekboom die groter is dan deze top) geen linkerkind kan hebben, en dat zijn voorganger (d.i. de laatste voorafgaande sleutel uit de binaire zoekboom die kleiner is dan deze top) geen rechterkind kan hebben.

   (c) Professor X meent dat hij een merkwaardige eigenschap van een binaire zoekboom ontdekt heeft. Veronderstel dat de zoektocht naar een zekere sleutel k eindigt in een blad van de boom. Beschouw dan drie verzamelingen:

      - A zijn de sleutels links van het zoekpad;
      - B zijn de sleutels op het zoekpad;
      - C zijn de sleutels rechts van het zoekpad.

      Professor X beweert dat voor elke drie sleutels $a \in A$, $b \in B$ en $c \in C$ geldt dat $a \leq b \leq c$. Geef een tegenvoorbeeld voor deze claim.

4. **Young tableaux als prioriteitswachtlijnen (3 pten.)**
   Een $m \times n$ Young tableau is een $m \times n$ matrix zodanig dat de elementen van elke rij van links naar rechts gesorteerd zijn van klein naar groot, en de elementen van elke kolom van boven naar onder in stijgende volgorde gesorteerd zijn. Sommige elementen van een Young tableau kunnen ∞ zijn, hetgeen we beschouwen als niet-bestaande elementen. Een Young tableau kan dus $r \leq mn$ eindige elementen bevatten.
   Hieronder een voorbeeld van een 4×4 Young tableau dat de elementen $\{9, 16, 3, 2, 4, 8, 5, 14, 12\}$ bevat. Merk op dat dit niet uniek is.

2	4	9	∞
3	8	16	∞
5	14	∞	∞
12	∞	∞	∞

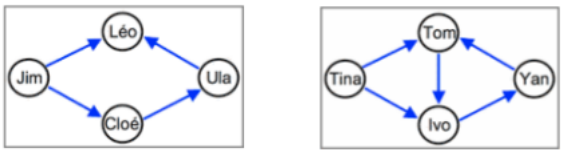
Zij Y een $m \times n$ Young tableau. Het is gemakkelijk in te zien dat Y leeg is als $Y[1,1] = \infty$, en dat Y vol is (m.a.w. mn elementen bevat) als $Y[m,n] < \infty$. Het kleinste element van Y bevindt zich altijd in $Y[1,1]$.

(a) Geef een $\Theta(m+n)$ algoritme voor het verwijderen van het kleinste element uit een $m \times n$ Young tableau.
(b) Geef een $\Theta(m+n)$ algoritme voor het toevoegen van een element aan een niet-vol $m \times n$ Young tableau.
(c) Geef een algoritme dat n^2 getallen sorteert in $\Theta(n^3)$ tijd, door gebruik te maken van een $n \times n$ Young tableau (en zonder gebruik te maken van een andere sorteermethode).

5. **Ontwerp van algoritmen: k-way merge (2 pten.)**
Gegeven zijn k gesorteerde rijen van elk n elementen. Gevraagd is deze rijen samen te voegen tot één gesorteerde rij van kn elementen.

(a) Een eenvoudige strategie werkt als volgt: gebruik makend van het *merge*-algoritme, voeg de eerste twee rijen samen; merge vervolgens de derde rij hierbij, dan de vierde, enz. Wat is de tijdscomplexiteit van dit algoritme in termen van k en n?
(b) Geef een efficiënter algoritme voor dit probleem, dat gebruik maakt van verdeel-en-heers. Wat is de tijdscomplexiteit van dit algoritme in termen van k en n?

6. **Ontwerp van algoritmen: groepswerk (2 pten.)**
Een groep van n personen maakt een groepswerk, waarbij ieder lid van de groep een deeltaak voor zijn rekening neemt. Maar soms moet een persoon wachten totdat een aantal andere personen hun deeltaak afgewerkt hebben, vooraleer hij aan zijn eigen deeltaak kan beginnen. Dergelijke afhankelijkheden kunnen schematisch voorgesteld worden met behulp van pijlen: een pijl van X naar Y betekent dat Y moet wachten totdat X klaar is met zijn deeltaak, vooraleer hij kan beginnen. Enkele voorbeelden van afhankelijkheidsschema's:



Het schema rechts heeft echter een probleem: het werk zal nooit afgeraken. Immers, er is een groepje waarbij iedereen op de volgende moet wachten: Ivo moet wachten totdat Tom klaar is, Yan totdat Ivo klaar is, en Tom totdat Yan klaar is. Ze draaien rond in een cirkeltje.
Geef een algoritme om dergelijk probleem in een gegeven afhankelijkheidsschema op te sporen.

7. **Backtracking algoritmen: Het knapsack-probleem (2 pten.)**
In het knapsack-probleem beschouwt men n objecten met gegeven gewichten $w_0, \ldots, w_{n-1}$ en gegeven winsten $p_0, \ldots, p_{n-1}$. Verder is ook de totale capaciteit M van de *knapsack* gegeven. Er wordt gevraagd een invulling $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ van de *knapsack* te zoeken, zodanig dat de totale winst $P = \sum_{i=0}^{n-1} p_i x_i$ zo groot mogelijk is, waarbij het totale gewicht $\sum_{i=0}^{n-1} w_i x_i \leq M$.
Geef een backtracking-algoritme met snoeifuncties voor dit probleem.