Voorbeeldexamen Algoritmen en Datastructuren

1. **Complexiteit van algoritmen (4 pten.)**

   (1.a) Onderstaande tabel geeft de experimentele tijdsmetingen (in milliseconden) voor een programma dat een bepaald algoritme implementeert. Het programma werd uitgevoerd voor verschillende waarden van de probleemgrootte n. Wat is de tijdscomplexiteit (in Θ-notatie) van het algoritme? Verklaar.

n	10	11	12	13	14	15
tijd	16	63	250	1 021	4 087	16 249

   (1.b) Veronderstel dat algoritme A een tijdscomplexiteit $T(n)$ heeft en dat het uitvoeren van een implementatie ervan op een computer X één uur duurt voor een inputgrootte n_0. Veronderstel vervolgens dat we het programma kunnen uitvoeren op een andere computer Y, die 27 keer sneller is dan X. Wat is de inputgrootte n_1 die op Y kan worden verwerkt in één uur, voor een algoritme met uitvoeringstijd $T(n) = n^3$? Verklaar.

   (1.c) Bepaal de tijdscomplexiteit in Θ-notatie voor onderstaand pseudocode-fragment. Verklaar.

```
1: s ← 0
2: for i from 1 to 2n do
3:     for j from 1 to i do
4:         s ← s + 1
```

   (1.d) Gegeven een probleem P waarvoor een algoritme A met tijdscomplexiteit $\Theta(n^3)$ gekend is. Veronderstel dat we een recursief algoritme B voor hetzelfde probleem P kunnen opstellen dat het originele probleem van grootte n opsplitst in twee deelproblemen van grootte $n/2$. Veronderstel bovendien dat het samenvoegen van deze deelresultaten tot het uiteindelijke resultaat een tijdscomplexiteit $\Theta(n^2)$ heeft. Heeft algoritme B een betere, dezelfde of een slechtere tijdscomplexiteit dan algoritme A? Motiveer je antwoord.

2. **Binaire zoekbomen (4 pten.)**

   (2.a) Voor een AVL-boom die reële getallen bevat, ontwerp een algoritme dat het bereik bepaalt (d.i. het verschil tussen de grootste en de kleinste getallen in de boom). Wat is de tijdscomplexiteit van je algoritme?

   (2.b) Waar of niet waar? De grootste en de kleinste sleutel in een AVL-boom bevinden zich steeds in het laatste of het voorlaatste niveau van de boom. Verklaar je antwoord.

   (2.c) Bewijs dat de diepte van een AVL-boom met n toppen $O(\log n)$ is.

3. **Minimale-kost opspannende bomen (4 pten.)**

(3.a) Geef het algoritme (in pseudocode) van Kruskal voor het bepalen van een minimale-kost opspannende boom in een gewogen graaf.

(3.b) Het sorteren van alle bogen in het algoritme van Kruskal vraagt meer werk dan nodig wanneer niet alle bogen moeten onderzocht worden voor mogelijke toevoeging aan de minimale-kost opspannende boom. We bekijken nu de strategie waarbij we, in plaats van alle bogen te sorteren, de bogen in een binaire hoop plaatsen en een boog daaruit verwijderen wanneer nodig. Analyseer de slechtste-geval complexiteit van deze implementatie van het algoritme van Kruskal, in termen van het aantal toppen n, het totale aantal bogen m, en het aantal k van onderzochte bogen.

(3.c) Bewijs dat het algoritme van Kruskal correct is.

4. **Ontwerp van algoritmen (4 pten.)**

(4.a) De lijst $A[1..n]$ bevat alle integers van 0 tot n, behalve één. Door gebruik te maken van een hulparray $B[0..n]$, die bijhoudt welke integers in A optreden, kan de ontbrekende integer gemakkelijk gevonden worden in $O(n)$ tijd.
In dit probleem kunnen we een integer in A echter niet in één enkele bewerking opvragen. De elementen in A zijn als bitstrings voorgesteld, en de enige toegelaten bewerking is 'bepaal bit j in $A[i]$', hetgeen in constante tijd kan gebeuren.
Geef een algoritme dat, gebruik makend van enkel deze bewerking, de ontbrekende integer vindt in $O(n)$ tijd.

(4.b) Gegeven een verzameling van n punten in het vlak. Ontwerp een $\Theta(n \log n)$ algoritme dat de afstand δ tussen een dichtste puntenpaar bepaalt en, wanneer $\delta > 0$, ook alle paren bepaalt die op afstand δ van elkaar liggen.