Universiteit Twente
Afdeling Informatica, Faculteit EWI
woensdag 8 april 2015, 9.45 – 11.45

Semester 2014-2015
Tentamen

Algoritmen, Datastructuren en Complexiteit

Opgave 1

20 pt

Beschouw het volgende algoritme (met * vermenigvuldigen, `div` integer division (bv. `7 div 2 = 3`), en `^2` kwadraat):

```
int func(int n)
{ if n == 0 return 1
    else if n < 8 return n
       else return 3*func(n div 8) + 8 + func(n div 8)^2
}
```

1. Geef een recursieve uitdrukking van de tijdscomplexiteit van dit algoritme, uitgedrukt in het aantal rekenkundige operaties.

2. Wat is de complexiteitsklasse van dit algoritme?

Opgave 2

20 pt

1. Geef een zo efficient mogelijk algoritme dat het verschil bepaalt tussen het grootste en het kleinste element van een heap (gegeven als een array).

2. Wat is de complexiteit van dit algoritme (dus niet de complexiteitsklasse), waarbij we tellen het aantal gemaakte vergelijkingen?

Opgave 3

20 pt

Gegeven een strongly connected directed graph in adjacency list representatie. Geef een algoritme dat, gegeven een node v, een spanning subtree met v als root node oplevert, gegeven in adjacency list representatie.

~~Opgave 4~~

20 pt

Gegeven twee karakterstrings A en B. We definiëren de *afstand* van A naar B als het minimale aantal insert, delete en replace operaties dat nodig is om A in B te veranderen. Bijvoorbeeld: de afstand van *dier* naar *dor* is 2, want er zijn twee operaties nodig: verwijder de e, en verander de i in een o. Neem aan dat de strings gegeven zijn als arrays van karakters.

Opgave 4 heeft fout. Is vervangen door opgave 4 op tweede blad.

1. Beargumenteer dat een recursieve uitdrukking voor de afstand $D(i,j)$ van $A[1..i]$ naar $B[1..j]$ gegeven kan worden door
   $D(i,j) = min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} + 1$.
2. Geef een algoritme dat de afstand van A naar B berekent, gebruik makend van dynamisch programmeren.
3. Wat is de tijdscomplexiteit van je algoritme?

Opgave 5

10 pt

Geef van de volgende beweringen aan of ze waar of onwaar zijn, en motiveer je antwoord.

1. Als $T(n)$ wordt gekarakteriseerd door $T(n) = 2 \cdot T(\frac{n}{4}) + 8$ geldt: $T(n) \in \Theta(\sqrt{n})$.
2. Open adressering heeft alleen zin als de loadfactor kleiner of gelijk is dan 1.
3. Laat T een binaire zoekboom zijn met n knopen, waarin de stijgende rij waarden $x_1, \ldots, x_n$ is opgeslagen. Als de knoop N waarin zich x_{i+1} bevindt, dichter bij de wortel van de boom ligt dan de knoop M met x_i, dan is M het rechterkind van zijn ouder.
4. Als het bin packing beslisprobleem in polynomiale tijd kan worden opgelost, dan kan de optimale hoeveelheid bins in polynomiale tijd worden gevonden.
5. Als P=NP, dan bestaat er een polynomiale reductie van elk probleem in P naar het Hamilton cycle probleem.

Universiteit Twente
Afdeling Informatica, Faculteit EWI
woensdag 8 april 2015, 9.45 – 11.45

Semester 2014-2015
Tentamen

Algoritmen, Datastructuren en Complexiteit

Opgave 4

20 pt

Beschouw het volgende spel. Het spel wordt gespeeld op een bord met n bij n vierkante vakjes. Je mag een damsteen op een willekeurig vakje op de onderste rij zetten. De damsteen mag je vervolgens steeds diagonaal linksomhoog of rechtsomhoog schuiven, mits je op het bord blijft. Voor elke zet kun je een bepaald aantal positieve punten krijgen, die in een tabel gegeven zijn: vanuit vakje (i, j) rechtsomhoog levert $p(i, j, R)$ punten op, linksomhoog levert $p(i, j, L)$ punten op. Uiteindelijk mag je op een willekeurig vakje op de bovenste rij eindigen. Neem aan dat het vakje linksonder de coordinaten $(1, 1)$ heeft.

Stel het maximaal aantal punten in vakje (i, j) is $R(i, j)$. Stel dat $1 \leqslant j \leqslant n$ en $0 \leqslant i \leqslant n+1$, met $R(0, j) = 0$ en $R(n+1, j) = 0$, en alle zetten van en naar vakjes met $i = 0$ of $i = n + 1$ leveren 0 punten op. Dan geldt de volgende recurrente betrekking:

- $R(i, j) =$
  $max\{R(i-1, j-1) + p(i-1, j-1, R), R(i+1, j-1) + p(i+1, j-1, L)\}$
  voor $1 \leqslant i \leqslant n$, $1 < j \leqslant n$
- $R(i, j) = 0$ voor $i = 0$ of $i = n + 1$ of $j = 1$

Geef een algoritme om te bepalen hoeveel punten je maximaal kunt winnen. De complexiteit mag niet slechter zijn dan kwadratisch in n.