Universiteit Twente
Afdeling Informatica, Faculteit EWI
woensdag 30 januari 2013, 13.45 – 17.15

Semester 2012-2013
Tentamen

Algoritmen, Datastructuren en Complexiteit

Bij dit tentamen mag het boek (zowel Baase-Van Gelder als Cormen et al) worden gebruikt, evenals een uitdraai van de hoorcollegesheets (dit alles zonder eigen aantekeningen).

Bij de opgaven waar om een algoritme wordt gevraagd, geeft u de pseudocode van uw oplossing en een beknopte maar duidelijke uitleg van de werking. Algoritmes zonder duidelijke uitleg worden niet in beschouwing genomen.

Uitspraken die u doet in antwoord op gestelde vragen moeten nauwkeurig worden beargumenteerd.

Er zijn 5 opgaven, waarmee 90 punten behaald kunnen worden. Het tentamenresultaat is (het aantal behaalde punten gedeeld door 10) plus 1.

Vermeld uw naam en de afkorting ADC op ieder los blad. Vermeld ook de werkcollegeleider waar u dit jaar bij was ingedeeld en geef expliciet aan of u beide huiswerkopgaven gemaakt hebt.

Veel succes!

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 < 4 return n
       else return 2*func(n div 4) + 6 + func(n div 4)^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. Stel je verandert in een heap een willekeurig element (stel, met index i). Geef een algoritme *ExtHeapify(E,i)* die er voor zorgt dat de heapeigenschap zonodig hersteld wordt.

2. Geef een algoritme die een element met index i uit een heap verwijdert, en ervoor zorgt dat het resultaat weer een heap is (hint: gebruik *ExtHeapify*.

Opgave 3

20 pt

Gegeven een ongerichte graaf G in adjacency list representatie.

1. Geef een DFS algoritme dat bepaalt of G bipartiet is, dwz. dat de vertices zodanig met twee kleuren gekleurd kunnen worden dat er geen edge is die vertices met dezelfde kleur verbindt.

2. Geef de worst-case complexiteit van dit algoritme.

Opgave 4

20 pt

1. Gegeven een array E met lengte n, dat n verschillende integers bevat. Geef een uitdrukking voor de lengte van de langste stijgende subreeks van E. Zo'n subreeks hoeft niet opeenvolgende te zijn: bijvoorbeeld, de langste stijgende subreeks van 11,17,5,8,6,4,7,12,3 is 5,6,7,12. Hint: laat $A[i]$ de lengte zijn van de langste stijgende subreeks die begint met $E[i]$, geef een recursieve uitdrukking voor $A[i]$, en bepaal de lengte van de langste stijgende subreeks aan de hand van A.

2. 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.

Opgave 5

10 pt

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

1. Beschouw de recurrente betrekking $T(n) = 2 \cdot T(\frac{n}{2}) + 2 \cdot log\ n$, $T(1) = 1$. Volgens het Masters theorema geldt $T(n) \in \Theta(log\ n)$.

2. Stel je gebruikt gesloten hashing. Als de gemiddelde lijstlengte 5 is, en er 5000 elementen zijn, is de loadfactor kleiner dan 1.

3. Als je het travelling salesman probleem polynomiaal zou kunnen reduceren tot een ander probleem, heb je bewezen dat P = NP.

4. Als we zouden kunnen bewijzen dat P ongelijk is aan NP, dan hadden we tevens bewezen dat problemen in NP niet efficient opgelost kunnen worden.