Universiteit Twente
Afdeling Informatica, Faculteit EWI
woensdag 1 februari 2012 13.45–17.15

K2 2011-2012
Tentamen

Algoritmen, Datastructuren en Complexiteit

(192140200 en 192140250)

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

15 pt

1. Gegeven een integer array a ter lengte $n = 2^k$ voor een $k > 0$.

   (a) Geef een algoritme dat het minimum en maximum van a bepaalt door het array één keer te doorlopen. Bepaal het aantal vergelijkingen dat dit algoritme uitvoert.

   (b) Beschouw het volgende algoritme voor het bepalen van het minimum en maximum:

```
(int, int) minmax( int a[1], ..., a[n])
{ if (n==2)
   { if (a[1]<=a[2]) return (a[1],a[2]); else return (a[2],a[1])
   }
  else { (mn1,mx1) = minmax(a[1], ..., a[n/2]);
         (mn2,mx2) = minmax(a[n/2+1], ..., a[n]);
         return(min(mn1,mn2), max(mx1,mx2));
       }
}
```

Geef een recurrente betrekking voor het aantal vergelijkingen van dit algoritme en bepaal een exacte uitdrukking voor dit aantal.

2. Vind de asymptotische orde van de oplossing van de volgende recurrente betrekking:

   $T(n) = 2 \cdot T(\frac{n}{4}) + 8$ voor $n \geqslant 4$, $T(n) = 0$ voor $n < 4$

Opgave 2

20 pt

1. Geef een efficient 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

15 pt

Zij $G = (V, E)$ een gerichte graaf met een centrale knoop c van waaruit alle andere knopen bereikbaar zijn. G heet *equidistant* als voor elke knoop v geldt dat alle paden van c naar v dezelfde lengte hebben. Geef een DFS algoritme dat bepaalt of G equidistant is. Wat is de worst-case complexiteit?

Opgave 4

25 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.

We geven een recursieve uitdrukking voor de afstand van A naar B. Idee: bekijk het aantal operaties als een van de twee leeg is, en begin in de andere gevallen met operaties op het laatste karakter van A en/of B.

Stel $D(i, j)$ is de afstand van $A[1..i]$ naar $B[1..j]$. Merk op dat voor $i = 0$ geldt $D(i, j) = j$ en voor $j = 0$ geldt $D(i, j) = i$. Als $A[i] = B[i]$ geldt $D(i, j) = D(i-1, j-1)$. In de andere gevallen: als je $A[i]$ verwijdert, is het aantal stappen $D(i - 1, j) + 1$. Als je $B[j]$ verwijdert is het aantal stappen $D(i, j - 1) + 1$. Als je $A[i]$ verandert in $B[j]$ is het aantal stappen $D(i - 1, j - 1) + 1$. Je zoekt naar het minste aantal stappen, dus $D(i, j) = min\{D(i - 1, j), D(i, j - 1), D(i - 1, j - 1)\} + 1$.

1. Geef een algoritme dat de afstand van A naar B berekent, gebruik makend van dynamisch programmeren.

2. Wat is de tijdscomplexiteit van je algoritme?

Opgave 5

15 pt

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

1. Beschouw de recurrente betrekking $T(n) = T(\frac{n}{2}) + 2 \cdot log\ n$, $T(1) = 1$. Volgens Masters theorema geldt $T(n) \in \Theta(log\ n)$.
2. Stel je gebruikt voor open adressering een hastabel met 400 locaties. Stel dat de kans, dat een item op een locatie wordt afgebeeld, voor alle items en alle locaties even groot is. De kans dat het vijfde toegevoegde item to een hashcollision leidt is 0,01.
3. Het Euler circuit probleem (heeft een graaf een cykel waarbij elke edge precies 1 keer bezocht wordt, terwijl knopen vaker mogen voorkomen) is polynomiaal reduceerbaar tot het Hamilton cykel probleem.
4. Een NP-volledig probleem is alleen efficient oplosbaar als het ook in P zit.
5. Als we zouden kunnen bewijzen dat P ongelijk is aan NP, dan hadden we tevens bewezen dat de problemen in NP niet efficient opgelost kunnen worden.