Universiteit Twente
Afdeling Informatica, Faculteit EWI
vrijdag 8 april 8.45–12.15

Semester 2010-2011
Tentamen

Uitwerkingen

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

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.

   Uitwerking:

   Merk op dat `func(n div 8)` twee keer wordt aangeroepen (niet erg handig, maar het staat er nu eenmaal). Verder worden er 6 rekenkundige bewerkingen gebruikt (een vermenigvuldiging, twee optellingen, twee divs, en een kwadraat), dus de recurrente betrekking wordt $T(n) = 2 \cdot T(\lfloor n/8 \rfloor) + 6$.

2. Wat is de complexiteitsklasse van dit algoritme?

Uitwerking:

Neem aan dat n een macht van 8 is (maakt voor het bepalen van de complexiteitsklasse niets uit), dan hebben we $T(n) = 2 \cdot T(n/8) + 6$. We passen nu het Master Theorema toe, met $b = 2$ en $c = 8$, dus $E = log2/log8 = 1/3$. Er geldt dat $6 \in O(n^{1/3-\epsilon})$ voor een ϵ, dus we hebben geval 1, dus $T(n) \in \Theta(n^{1/3})$.

Opgave 2

20 pt

Gegeven een binaire zoekboom waarbij alle keys (positieve getallen) uniek zijn. Geef een algoritme die, met als invoer een node uit de boom met key k, oplevert de node met de grootste waarde kleiner dan k (en *nil* als er niet zo'n node is).

Uitwerking:

Stel x is de node die k bevat. Er zijn 2 mogelijkheden:

- x heeft linkerkinderen. Ga dan 1 keer naar linksbeneden, en blijf dan vervolgens naar rechtsbeneden gaan zolang als dat kan. Je hebt dan de grootste waarde kleiner dan k.
- x heeft geen linkerkinderen. Ga eerst net zolang rechtsomhoog tot dat niet meer kan, en ga dan linksomhoog. Kun je niet linksomhoog, dan was k het kleinste element en lever je *null* op; anders bereik je zo de grootste waarde kleiner dan k.

Het algoritme:

```
node bstPred(node x)
{ if x.left != null
     { x = x.left;           //ga naar linksbeneden
       while x.right != null //ga zover mogelijk naar rechts-
          x = x.right;       //beneden
       return x;
     }
  else node y = x.parent;               //ga naar boven
       while (y != null && y.left == x) //tot je niet verder kan,
          { x =y;                       //of je uiteindelijk een
            y = y.parent;               //een keer naar links-
          }                             //boven bent gegaan
       return y;
}
```

Opgave 3

15 pt

Een 4-clique is een deelgraaf met 4 vertices die compleet is, dus edges tussen elke twee vertices. Gegeven een ongerichte graaf in adjacency matrix representatie met n vertices. Geef een algoritme dat in polynomiale tijd bepaalt of deze graaf een 4-clique heeft. Hint: wat is het aantal deelverzamelingen van $1..n$ met 4 elementen?

Uitwerking:

Stel er zijn n vertices; het aantal deelverzamelingen met 4 vertices is $n(n-1)(n-2)(n-3)/24$. Per deelverzameling moeten we de aanwezigheid van 6 edges checken. Dus als we simpelweg alle deelverzamelingen afgaan, hebben we een algoritme met complexiteit $\Theta(n^4)$, dus polynomiaal. Het algoritme:

```
bool 4clique(bool[][]A, int n)
{ for (int i=1; i<=n; i++)
    for  (int j=i+1; j<=n; j++)
      for  (int k=j+1; k<=n; k++)
        for  (int l=k+1; l<=n; l++)
            if (A[i,j]&&A[j,k]&&A[k,l]&&
                A[i,k]&&A[i,l]&&A[j,l]) return true;
  return false;
}
```

Opgave 4

25 pt

Gegeven een array E met lengte n, dat n verschillende integers bevat.

1. Geef een algoritme dat de lengte van de langste stijgende subreeks van E vindt. 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.

   Uitwerking:

   Analyse: de lengte van de langste subreeks die met $E[i]$ begint, is 1 plus de lengte van de langste subreeks na i waar je $E[i]$ voor kan plakken. Dus:
   $A[i] = 1 + \max_{i<j\leqslant n}\{A[j] \mid E[i] < E[j]\}$
   en de gevraagde lengte: $lengte = \max_{1\leqslant i\leqslant n}\{A[i]\}$
   Het algoritme:

```
int lls(int[] E, int n)
{
  int[1..n] A;
  int maxl;

  for (i=n; i>=1; i--)
   { maxl=0;
     for (j=i+1; j<=n; j++)
       if E[i]<E[j] then maxl = max(maxl, A[j])

     A[i] = maxl +1
   }

  maxl=0;
  for (i=1; i<=n; i++)
   maxl = max(maxl, A[i])
  return maxl
}
```

2. Zorg dat de algoritme ook bepaalt wat de langste stijgende subreeks is.

Uitwerking:

We nemen aan dat er globaal een array $int[0, n]loc$ gedefinieerd is. Uiteindelijk zal $loc[0]$ de index bevatten van het eerste element in de langste subreeks, $loc[loc[0]]$ de index van het tweede element, enzovoorts; als $loc[k] == 0$ is k de index van het laatste element van de subreeks.

```
int lls(int[] E, int n)
{
  int[1..n] A;
  int maxl;

  for (i=n; i>=1; i--)
   { maxl=0;
     loc[i]=0;
     for (j=i+1; j<=n; j++)
       if E[i]<E[j] and A[j]>maxl then {maxl=A[j]; loc[i]=j}

     A[i] = maxl+1
   }

  maxl=0;
  for (i=1; i<=n; i++)
   if A[i]>maxl then {maxl=A[i]; loc[0]=i}
  return maxl
}
```

Opgave 5

15 pt

Beargumenteer voor de volgende uitspraken of ze waar of niet waar zijn:

1. Het handelsreizigerprobleem is reduceerbaar tot het satisfiability probleem (SAT).

   Uitwerking:

   Het handelsreizigersprobleem zit in NP, SAT zit in NPC, dus de bewering is waar.

2. Stel P $\neq$ NP, en we hebben een probleem $X \notin$ P, en alle problemen in NP zijn reduceerbaar tot X. Dan geldt $X \in$ NP.

   Uitwerking:

   De bewering is niet waar: X is NP-hard, maar we weten niet of X in NP zit.

3. Iedere ongerichte graaf G is in polynomiale tijd te transformeren naar een graaf G' zodat tussen G en G' de volgende relatie bestaat: Als G' met $n+1$ kleuren te kleueren is, dan is G met n kleuren te kleuren (zonder buren met dezelfde kleur).

   Uitwerking:
   Is waar. Construeer G' uit $G = \langle V, E \rangle$ door aan G een knoop $w \notin V$ toe te

voegen, en door E uit te breiden met kanten die iedere knoop $v \in V$ met de toegevoegde knoop w verbinden (d.w.z. met de verzameling $\{vw|v \in V\}$. Een kleuring van deze G' met $n+1$ kleuren bevat een kleuring van G met n kleuren. Immers: alle knopen van de oorspronkelijke G hebben een andere kleur dan de kleur van de toegevoegde w.

4. Het n-kleur probleem voor grafen is polynomiaal reduceerbaar tot het $n+1$-kleur probleem.

   Uitwerking:
   Is waar, volgt uit de vorige bewering.