Universiteit Twente
Afdeling Informatica, Faculteit EWI
23 juni 2006, 9:00 – 12:30

Semester 2005-2006/1
Extra tentamen

Algoritmen, Datastructuren en Complexiteit

(214020 ADC en 214025 ADC+)

Bij dit tentamen mag het boek van Baase en Van Gelder worden gebruikt, aangevuld met kopieën van figuren die in het boek verkeerd zijn afgedrukt. Ook een uitdraai van de slides is toegestaan, en het formuleblad dat op het werkcollege aan deelnemende HBO-studenten in een overbruggingsprogramma/-thema is uitgereikt. Dit alles zonder eigen aantekeningen. Ook het gebruik van een rekenmachine is toegestaan. Verder niets.

Bij de opgaven waar om een algoritme wordt gevraagd dient u zowel een beknopte en duidelijke beschrijving te geven van het principe van het algoritme alsmede de beschrijving in pseudocode.

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.

Veel succes!

Opgave 1

15 pt

De methode `tailsort` hieronder is bedoeld om een rij $a[1..n]$ waarvan het eerste deel $a[1..k-1]$, $1 \leqslant k \leqslant n$ opklimmend gesorteerd is, volledig te sorteren. In opeenvolgende recursieve aanroepen worden achtereenvolgens de elementen $a[k], a[k+1], \cdots, a[n]$ "naar hun plaats gebracht".

```
void tailsort (int [ ] a; int k,n):
 if (k<=n)
 {
   int j=k;
   bool increasing=false;
   while (j>1 && !increasing)
     if (a[j]<a[j-1])
      {
       swap(a,j,j-1); //verwisselt a[j-1] en a[j]
       j--
      }
     else
```

```
    increasing=true;
 tailsort(a,k+1,n)
}
```

1. Laat $T(k,n)$ het aantal vergelijkingen tussen elementen van de rij $a[1..n]$ zijn dat in het slechtste geval nodig is voor het uitvoeren van `tailsort(a,k,n)`.

   (a) Wat is de recurrente betrekking voor $T(k,n)$?

   (b) en wat is het asymptotisch gedrag van $T(k,n)$?

   (c) Hoe ziet een rij a eruit waarop `tailsort(a,k,n)` het slechts mogelijk gedrag vertoont, als het gaat om het noodzakelijk aantal vergelijkingen?

2. Geef van de volgende uitspraken aan of ze waar zijn of niet. Het gaat steeds om een uitspraak over de average-case complexiteit van twee verschillende methoden. De veronderstelling is steeds dat het beginstuk $a[1..k]$ van $a[1..n]$ opklimmend gesorteerd is. U moet uw antwoord beargumenteren, uit het argument moet blijken welke complexiteit u voor elk van de gnoemde methoden heeft vastgesteld.

   (a) Als $n-k = \lfloor \log n \rfloor$ dan is het beter om $a[1..n]$ volledig te sorteren met `tailsort(a,k,n)` dan met `quicksort(a,1,n)`

   (b) Als $n-k = \lfloor \sqrt{n} \rfloor$ dan is het beter om $a[1..n]$ volledig te sorteren door eerst $a[k..n]$ te sorteren met *insertion sort* en daarna met een *merge* van $a[1..k-1]$ en $a[k..n]$ tot een volledig gesorteerde rij te komen, dan om `tailsort(a,k,n)` te gebruiken.

Opgave 2

15 pt

Beschouw de volgende 6 functies.

$$
\begin{aligned}
f_1(n) &= n^{2.5} \\
f_2(n) &= \sqrt{2n} \\
f_3(n) &= n+10 \\
f_4(n) &= 2^{\log n} \\
f_5(n) &= \tfrac{n}{\log n} \\
f_6(n) &= n^2 \log n
\end{aligned}
$$

1. Rangschik de functies in deze rij zo dat voor twee opeenvolgende elementen f_i en f_j in uw ordening steeds geldt $f_i \in O(f_j)$ en bewijs de correctheid van uw ordening.

2. Zijn er opeenvolgende elementen f_i en f_j in uw ordening waarvoor geldt dat $f_i \in \Theta(f_j)$?

Opgave 3

20 pt

Het algoritme van Prim voor het bepalen van de minimum spanning tree in een ongerichte gewogen graph G verloopt schematisch als volgt:

```
primMST(graph G, int n):
{
  initialise all vertices as unseen;
  introduce a priority queue Q;
  select an arbitrary vertex s and classify it as tree;
  reclassify all vertices v adjacent to s as fringe and move them to Q,
    their priority (cost) is determined by the weight of edge sv;
  while (there are fringe vertices in Q)
   {
     select the minimum element from Q, that is a the node v in Q with the
       lowest cost, the cheapest edge to a tree vertex t;
     reclassify v as tree and remove it from Q;
     add edge tv to the tree;
     visit all vertices w adjacent to v,
       reclassify the unseen ones as fringe, and add them to Q,
       their priority (cost) is determined by the weight of edge vw, and
       update the cost of the ones that had already been classified as fringe
       by taking the weight of edge vw into account;
   }
}
```

In de uitwerking van deze methode zal zeker een array `classifier [1..n] status` gebruikt worden waarin van de knopen van de graph wordt bijgehouden hoe hun afhandeling vordert. Daarbij gebruiken we drie `classifier`-objecten, namelijk `unseen` (d.w.z. we zijn de knoop nog niet tegengekomen), `fringe`, (d.w.z. de knoop wacht op afhandeling in `Q`), en `tree` (d.w.z. de knoop is afgehandeld en verbonden met de minimum spanning tree).

Daarnaast is er natuurlijk een implementatie van de queue `Q`.

1. Veronderstel dat u de minimum spanning tree moet bepalen voor een graph G die behoort tot een klasse met de eigenschap dat $m \in O(n \log n)$, waarbij n het aantal knopen is en m het aantal kanten. Welke implementatie van `Q` zou u kiezen, en wat kunt u zeggen over de tijdscomplexiteit van de methode op deze klasse graphen, met de door u gekozen implementatie van `Q`?

We onderscheiden in een ongerichte graph G twee soorten knopen. De ene soort noemen we "doorgangsknopen", dat zijn knopen waar 1 of 2 kanten uitkomen. De andere soort noemen we"splitsingsknopen", dat zijn knopen waar 3 of meer kanten uitkomen.

2. Wat is het effect op het aantal elementen in de queue `Q` van het afhandelen van een doorgangsknoop in een doorloop van de herhaling?

3. Veronderstel dat G behoort tot een klasse graphen met de volgende eigenschappen:

   (a) In een knoop komen nooit meer dan c kanten uit, waarbij c een constante is, onafhankelijk van het aantal knopen in de graph.

   (b) Het aantal splitsingsknopen in G is niet meer dan $\log n$, waarbij n het aantal knopen in de graph is.

   Welke implementatie zou u bij deze soort graphen kiezen voor `Q` en wat kunt u zeggen over de asymptotische tijdscomplexiteit van de methode op deze klasse graphen, met de door u gekozen implementatie van `Q`?

4. Kunt u in aanvulling op uw antwoord op 3.3 aangeven wat de grootte-orde is van de maximale omvang is van de door u geïmplementeerd queue `Q`?

Opgave 4

20 pt

De eigenaar van een feestzaal heeft een groot aantal verzoeken liggen van mensen die zijn zaal willen gebruiken. Hij moet uitzoeken welke aanvragen hij kan honoreren en wie hij moet afwijzen. Hij maakt daarom van alle aanvragen een schatting van de extra (bar-)inkomsten die het evenement hem zal opleveren, en gaat op zoek naar die verzameling aanvragen waarvoor geldt:

ze overlappen elkaar niet in de tijd en

de som van de geschatte inkomsten is maximaal.

Aanvragen die niet in die verzameling vallen wil hij afwijzen, met de partijen waarvan de aanvraag in de verzameling zit, sluit hij een contract.

Hij pakt dit probleem systematisch aan, en gaat op zoek naar een efficiënt algoritme dat een algemene oplossing biedt. De formulering van het abstracte probleem luidt als volgt.

Gegeven zijn drie even lange rijen positieve getallen $s_1, \ldots, s_n$, $f_1, \ldots, f_n$ en $v_1, \ldots, v_n$ (in deze rijen zijn de relevante gegevens van de n aanvragen vastgelegd) met de volgende eigenschappen:

- $s_i < f_i$, voor $i = 1, \ldots, n$.

  (s en f staan voor start en finish, s_i en f_i zijn begin- en eindtijd van de i-de aanvraag.)

- $v_i > 0$, voor $i = 1, \ldots, n$.

  (v staat voor value, het gaat om de verwachte inkomsten van de i-de aanvraag.)

Gevraagd wordt een rij $i_1, \ldots, i_k$ te kiezen, met $1 \leqslant i_j \leqslant n$, (d.w.z. gevraagd wordt k aanvragen uit de gehele rij te kiezen) waarvoor geldt

- $s_{i_{j+1}} > f_{i_j}$, voor $1 \leqslant j < k$

  (D.w.z.twee opeenvolgende aanvragen uit de gekozen rij overlappen elkaar niet.)

- $\sum_{j=1}^{k} v_{i_j} \geqslant \sum_{j=1}^{k'} v_{i'_j}$ voor iedere rij indices $i'_1, \ldots, i'_{k'}$ met $1 \leqslant i'_j \leqslant n$ en $s_{i'_{j+1}} > f_{i'_j}$.

  (D.w.z. het totaal van de opbrengsten van de gekozen rij aanvragen is even groot of zelfs groter dan de opbrengsten van iedere andere rij aanvragen die gekozen had kunnen worden.)

1. Laat de volgende 6 aanvragen gegeven zijn:

i	s_i	f_i	v_i
1	2	11	7
2	0	4	2
3	5	8	4
4	11	13	1
5	1	6	4
6	9	12	2

   Geef van de volgende mogelijke keuzen uit deze reeks aanvragen welke optimaal zijn (als er al een optimale bij zit), en wat er mis is met de andere.

   (5,3), (3,6,2), (1,4), (2,3,4), (2,3,6).

We sorteren de aanvragen op hun eindtijdstip, d.w.z. bij de eerder genoemde condities op s_i, f_i en v_i veronderstellen we nu ook nog:

- $f_i \leqslant f_{i+1}$ voor $1 \leqslant i < n$

Verder bepalen we een vierde rij $p_i, \ldots, p_n$ waarin p_i de (index van de) laatste aan aanvraag i voorafgaande aanvraag geeft die niet met aanvraag i overlapt. $p_i = 0$ als er niet ~~en~~ een dergelijke niet overlappende aanvraag bestaat. In een formule

- $p_i = \max\{j \mid f_j \leqslant s_i\}$

2. De gesorteerde versie van de reeks aanvragen die we in 4.1 bekeken is als volgt.

i	s_i	f_i	v_i
1	0	4	2
2	1	6	4
3	5	8	4
4	2	11	7
5	9	12	2
6	11	13	1

   Bepaal de rij $p_1, \ldots, p_6$ die bij deze reeks aanvragen behoort.

Het volgende stukje pseudocode gaat uit van vier gegeven rijen: een rij met starttijden (s), een rij met eindtijden (f), een rij met opbrengsten (v) en en een rij met laatste

niet overlappende voorgangers (p). De methode bepaalt de maximaal te behalen opbrengst kijkend naar alleen de eerste k aanvragen, uitgaande van de veronderstelling dat de eindtijden opklimmend gesorteerd staan.

```
double highestprofit (double [ ] s, f, v; int [ ] p; int k):
  if (k==1)
    return v[1]
  else
   {
    double opt1 = v[k] + highestprofit(s,f,v,p,p[k]);
    double opt2 = highestprofit(s,f,v,p,k-1);
    return max(opt1, opt2)
   }
```

De aanroep `highestprofit (s,f,v,p,n)` zal de maximaal te behalen opbrengst bij de gegeven n aanvragen bepalen.

3. Als we `highestprofit(s,f,v,p,6)` uitvoeren, met s, f, v, en p zoals in 4.2, dan onstaat een boomstructuur van aanroepen van de methode met daarin

   `highestprofit(s,f,v,p,p[6])`,

   `highestprofit(s,f,v,p,5)`,

   ... en zo verder.

   Teken deze boomstructuur.

4. Wat is in het slechtste geval de asymptotische tijdscomplexiteit van

   `highestprofit(s,f,v,p,n)`?

5. hp_k is de hoogst haalbare opbrengst uit de eerste k aanvragen. Het is duidelijk dat $hp_1 = v_1$, maar wat is de uitdrukking die hp_k definieert voor $k > 1$?

6. Kunt u een methode `highestprofits` schrijven die in lineaire tijd een array vult waarin voor iedere k de hoogst haalbare opbrengst uit de eerste k aanvragen te vinden is?

7. Als de aanvragen niet op eindtijd maar op begintijd gerangschikt zouden zijn, wat zou u aanpassen om opnieuw een correcte methode te verkrijgen?

Opgave 5

20 pt

Geef van elk van de volgende uitspraken aan of ze waar of niet waar zijn. Onderbouw uw antwoorden.

1. Een direct pad in een gewogen gerichte graph G is een pad zonder cykels. Het algortime van Dijkstra kan worden toegepast om in een dergelijke graph de afstand tussen knopen te vinden gemeten langs directe paden, zelfs als sommige gewichten van kanten negatief zijn.

2. Een direct pad in een gewogen gerichte graph G is een pad zonder cykels. Het algortime van Floyd kan worden toegepast om in een dergelijke graph de afstand tussen knopen te vinden gemeten langs directe paden, zelfs als sommige gewichten van kanten negatief zijn.

3. $(\sum_{i=1}^{n} i)^2 = \sum(i^3)$.

4. Laat G en G' twee ongerichte gewogen graphen zijn, die maar in één opzicht verschillen: knopen en kanten zijn identiek, maar het gewicht van elke kant in G' is het kwadraat van het gewicht van dezelfde kant in G. We hebben van G de minimum spanning tree T bepaald. De overeenkomstige boom T' in G' is dan een minimum spanning tree voor G'.

5. Laat $P(n)$ en $Q(n)$ twee problemen zijn die wederzijds polynomiaal tot elkaar reduceerbaar zijn. Veronderstel dat we weten dat een oplossing van $P(n)$ een wordt-case tijdscomplexiteit $T_P(n)$ heeft met $T_P(n) \in \Omega(n^2)$. Dan geldt ook voor de worst-case tijdscomplexiteit $T_Q(n)$ voor een oplossing van Q dat $T_Q(n) \in \Omega(n^2)$.