INFORMATICA INSTITUUT, FACULTEIT WISKUNDE EN INFORMATICA, UU.
IN ELEKTRONISCHE VORM BESCHIKBAAR GEMAAKT DOOR DE $\mathcal{TBC}$ VAN A−Eskwadraat.
HET COLLEGE INFOAL WERD IN 2004/2005 GEGEVEN DOOR HANS BODLAENDER.

Algoritmiek (INFOAL)
14 december 2004

U heeft twee uur de tijd voor dit tentamen. Schrijf uw naam op alle blaadjes die U inlevert. Bij elk onderdeel staat het aantal punten dat u voor dit onderdeel kunt halen aangegeven. Schrijf duidelijk en helder. Wanneer u een gedeelte van een opgave niet heeft kunnen aantonen, dan mag u dat resultaat toch gebruiken in een later gedeelte van die opgave. Dit geldt met name bij opgave 3(iii): gebruik als basis van uw pseudocode de recurrente betrekking die u zelf koos bij onderdeel (ii).

Opgave Datastructuren voor grafen *(2 punt)*

Stel we hebben een ongerichte graaf $G = (V, E)$. Schrijf $n = |V|$, en $m = |E|$. De *graad* van een knoop v is het aantal buren van $v : d(v) = |\{w \in V | \{v, w\} \in E\}|$. Stel we willen de maximum graad van de knopen in G uitrekenen: $D(G) = \max_{v \in V} d(v)$

(i) Hoeveel tijd (in O-notatie, als functie van n en m) kost het om $D(G)$ uit te rekenen, als de graaf G gegeven is met de adjacency list datastructuur? Leg (beknopt, maar duidelijk uit hoe dit gaat. *(1 punt)*

(ii) Hoeveel tijd (in O-notatie, als functie van n en m) kost het om $D(G)$ uit te rekenen, als de graaf G gegeven is met de adjacency matrix datastructuur? Leg (beknopt, maar duidelijk) uit hoe dit gaat. *(1 punt)*

Opgave Minimum opspannende bomen *(1.25 punt)*

(i) Geef een korte maar duidelijke beschrijving van het algoritme van Prim. (Maximaal 10 regels.) *(1 punt)*

(ii) Noem de algoritmische techniek waarvan Prim's algoritme gebruik maakt: divide and conquer, simplification, dynamisch programmeren of greedy algoritmen. *(0.25 punt)*

Opgave Dynamisch programmeren *(3.5 punt)*

We willen het volgende probleem oplossen met dynamisch programmeren. We hebben n voorwerpen $v_1, \ldots, v_n$. Van ieder voorwerp v_i zijn gegeven een waarde w_i en een gewicht g_i. Waardes en gewichten zijn positieve integers. Ook gegeven zijn twee niet-negatieve integers W en G. We willen testen of er een deelverzameling van de voorwerpen is met totaalwaarde precies W en totaalgewicht precies G ois.
We definieren, voor $0 \leq i \leq, 0 \leq K \leq W, 0 \leq L \leq G : A(i, K, L)$ true, dan en slechts dan als er een deelverzameling van de voorwerpen $\{v_1, \ldots v_i\}$ met totaalwaarde precies K en een totaalgewicht precies L is.

(i) Wat is $A(0, 0, 0)$? Wat is $A(), K,))$ als $K > 0$? Wat is $A(0, 0, L$ als $L > 0$? En wat is $A(0, K, L)$ als $K > 0$ en $L > 0$? *(0.5 punt)*

(ii) Welke van de volgende recurrente betrekkingen voor A is correct? Leg duidelijk uit waarom dit de correcte recurrente betrekking voor A is. We nemen hieronder aan dat $i > 0$.*(1.5 punt)*

$$A(i, K, L) = \begin{cases} A(i-1, K, L) & \text{als } w_i > K \text{of } g_i > L \\ A(i-1, K, L) \vee A(i, K - w_i, L - g_i) & \text{als } w_i \leq K \text{en } g_i \leq L \end{cases} \quad (1)$$

$$A(i,K,L) = \begin{cases} A(i-1,K,L) & \text{als } w_i > K \text{of } g_i > L \\ A(i-1,K,L) \vee A(i-1,K-w_i,L-g_i) & \text{als } w_i \leq K \text{en } g_i \leq L \end{cases} \tag{2}$$

$$A(i,K,L) = \begin{cases} A(i-1,K,L) & \text{als } w_i > K \text{of } g_i > L \\ A(i-1,K,L) \wedge A(i,K-w_i,L-g_i) & \text{als } w_i \leq K \text{en } g_i \leq L \end{cases} \tag{3}$$

$$A(i,K,L) = \begin{cases} A(i-1,K,L) & \text{als } w_i > K \text{of } g_i > L \\ A(i-1,K,L) \wedge A(i-1,K-w_i,L-g_i) & \text{als } w_i \leq K \text{en } g_i \leq L \end{cases} \tag{4}$$

(ii) Geef een dynamisch programmeer algoritme dat het probleem oplost. Hoeveel tijd (in O-notatie) gebruikt uw algoritme? *(1.5 punt)*

Opgave Greedy en divide and conquer. *(1.75 punt)*

We hebben n voorwerpen, met gewichten $w_1, w_2, \ldots, w_n$. We willen zo veel mogelijk voorwerpen meenemen, maar we kunnen in het totaal maar B gewicht meenemen. D.w.z., we zoeken een deelverzameling van de voorwerpen met totaalgewicht hooguit B en zo groot mogelijke cardinaliteit.

(i) Beschouw het volgende greedy algoritme voor dit probleem.
Soteer de voorwerpen op niet dalend gewicht. Neem nu aan $w_1 \geq w_2 \geq \ldots w_n$.
$tot = 0; A = \emptyset$.
for $i = 1$ **to** n **do**
  if $(tot + w_i > B)$ **then** return A; stop
  else $tot = tot + w_i; A = A \cup \{w_i\}$
(We doen dus voorwerpen bij de antwoordverzameling in volgorde van niet-dalend gewicht tot het eerste voorwerp dat er niet meer bij past.)
Bewijs dat dit algoritme het probleem correct oplost. *(1.25 punt)*

(ii) Geef een algoritme dat het probleem in $O(n)$ tijd oplost. (Hint: gebruik een divide and conquer techniek.) *(1.5 punt)*

Opgave Implementatie van een scheduling-heuristiek *(1.5 punt)*

Er zijn een aantal taken $t_1, \ldots, t_k$ die gescheduled moeten worden. Elke taak gebruikt éé tijdstap, en heeft een relaese date r_i. De taak komt op tijdstip r_i beschikbaar, kan niet voor r_i gescheduled worden. Er kan maar 1 taak per tijdstap worden uitgevoerd. De volgende heuristiek wordt gebruikt om de taken te schedulen:

for $i = 1$ **to** k **do**
  Schedule taak t_i op het eerste tijdstip $\geq r_i$ waarop nog geen andere taak gescheduled is.

De taken en release dates zijn van te voren bekend; hierbij zijn de taken *niet* op release date gesorteerd.
In deze opgave kijken we hoe deze heuristiek efficiënt geimplementeerd kan worden. Stel taken en release dates zijn gegeven. Leg uit hoe we het uiteindelijke kunnen vinden in $O(k \log^* k)$ tijd.
(Als u dit niet lukt, voor deelpunten: in hoeveel tijd kunt u deze heuristiek implementeren?)