# Schets antwoorden Tentamen Algoritmiek 2003/2004

**Vraag 1 (i)** Begonnen wordt met een stroming die 0 is over alle pijlen. De volgende stap wordt herhaald tot geen verbeterend pad meer gevonden wordt: maak het rest-netwerk $G_f$, en vind in dit netwerk een pad van $s$ naar $t$ (een verbeterend pad). Over dit pad sturen we zo veel mogelijk extra stroming van $s$ naar $t$.

**Vraag 1 (ii)** Een adjacency list, omdat deze datastructuur efficienter is voor het vinden van paden.

**Vraag 2** Veel studenten hadden deze vraag verkeerd beantwoord, omdat ze een oplossing gaven die alleen werkte voor ongerichte grafen. Het probleem op gerichte grafen is wat lastiger.

Er zijn verschillende oplossingen mogelijk. Het handigst is om een kleine aanpassing te doen bij het algoritme dat een topologische sortering van $G$ zoekt, voor gerichte acyclische grafen.

**Vraag 3 (i)** Beide zijn correct. De eerste omdat OPT($i$,$i$) niets hoeft uit te rekenen: matrix $A_i$ kan gewoon opgeleverd worden. De tweede formuleert precies het gegeven uit de vraagstelling van de opgave.

**Vraag 3 (ii)** Dit is een kleine variant van wat behandeld is in het college voor wanneer we de gewone matrixvermenigvuldiging nemen. De basisgevallen zijn als in 3(i). Verder, als $j > i + 1$:

$$OPT(i, j) =$$

$$\min\{OPT(i, k)+OPT(k+1, j)+c_0 \cdot \min(d_{i-1} \cdot d_i \cdot d_{i+1}, (\max\{d_{i-1}, d_i, d_{i+1}\})^{2.81}) | i \leq k \leq j\}$$

**Vraag 4** Volgt de standaardtechnieken.

**Vraag 5 (i)** Voor alle steden $v$, $w$, $x$: $d(v, w) \leq d(v, x) + d(x, w)$.

**Vraag 5 (ii)** $u$ wordt toegevoegd direct na $w$. Stel $x$ was de knoop die na $w$ op de cycle kwam. Dan wordt de kant $(w, x)$ op de cycle vervangen door kanten $(w, u)$ en $(u, x)$. De toename van de lengte is dus $d(w, u) + d(u, x) - d(w, x)$, en vanwege de driehoeksongelijkheid is dit kleiner of gelijk aan $2d(w, u) = 2d(u, w)$.

**Vraag 5 (iii)** De verzameling kanten $F^*$ is precies de verzameling kanten die gekozen worden door het algoritme van Prim.

**Vraag 5(iv)** Als we een kant $e$ weghalen uit een optimale tour $T_{opt}$, dan krijgen we een opspannende boom. De lengte van $T_{opt} - e$ is dus groter of gelijk aan de som van de lengtes van alle kanten in de minimum opspannende boom $F^*$, en dus is de lengte van $T_{opt}$ dat ook.

**Vraag 5(v)** We moeten laten zien dat de lengte van de tour die gemaakt wordt door de heuristiek hooguit twee keer de lengte van een optimale tour is. Uit vraag 5(ii) en 5(iii) volgt dat de lengte van de tour hooguit twee keer de lengte van een minimum opspannende boom is, en vanwege 5(iv) is dat hooguit twee keer de lengte van de optimale tour.

**Vraag 6** Bij een Find-operatie gaan we nogmaals het zoekpad langs en hangen alle knopen direct onder de wortel van de boom. Daardoor gaan volgende Find's voor die knopen sneller.

**Vraag 7 (i) en (ii)** Een *greedy* algoritme lost dit probleem in de gewenste tijd op. (Het algoritme kan overigens ook gezien worden als een voorbeeld van *simplificatie*.)

- linkerkanthuidiginterval = $x_1$;
- rechterkanthuidiginterval = $x_1 + 1$;
- for $i = 2$ to $n$
- do
  if $x_i >$ rechterkanthuidiginterval
  then
  - Lever het interval [linkerkanthuidiginterval,rechterkanthuidiginterval] op.
  - linkerkanthuidiginterval = $x_i$;
  - rechterkanthuidiginterval = $x_i + 1$;
- Lever het interval [linkerkanthuidiginterval,rechterkanthuidiginterval] op.

Met andere woorden: we gaan van klein naar groot door de rij getallen, en beginnen steeds een nieuw interval wanneer een getal niet meer in het vorige interval past.

Dat het algoritme correct is kan je bijv. zo inzien: er moet een interval zijn dat $x_1$ bevat. Dat interval naar links schuiven kan nooit helpen om een betere oplossing te vinden. Dus, een optimale oplossing bevat een interval $[x_1, x_1 + 1]$. De rest van de intervallen moeten alle $x_i$'s overdekken die niet in dit interval liggen, en daar kunnen we steeds hetzelfde argument gebruiken.

## 3