De Algoritmiek-toets van 30 september 2008
De toets is genormeerd op 20 punten, de uitslag van de toets
is natuurlijk de helft van de behaalde punten.

Opgave 1: Tertiaan, 5 punten.

(1a) Het element met rang 2/3(n+1) van onderen gerekend,
is precies het element met rang n/3 van bovenaf gerekend.
Je kunt voor deze "duotertiaan" dus exact de methode Tertiaan
toepassen, waarbij je alleen de richting van de ordening
omdraait. Dus <= wordt >=, < wordt >.

Het is niet voldoende om de elementen van A eerst in volgorde
om te keren. Wat je moet omkeren is de ordening waarmee de
Tertiaan werkt, niet de posities van de elementen.

Wat ook absoluut NIET kan is "de parameter (n+1)/3 in Tertiaan
veranderen in 2n/3". De aanroep van Tertiaan heeft die
parameter niet, en over de werking van Tertiaan mag je niet
zomaar aannemen dat je daarin met een zo simpele wijziging
de werking verandert (zonder de complexiteit aan te tasten).
Op deze manier doorredenerend zou ik ook algemene k-Selectie
met n-1 vergelijkingen kunnen doen, immers: "Minimum kan met
n-1 vergelijkingen, en dat is eigenlijk 1-Selectie. Als ik
dus in de Minimum-procedure de 1 in k verander, krijg ik
precies k-Selectie in n-1 vergelijkingen." Als je tot je
door laat dringen hoe hier geredeneerd wordt, snap je wel dat
dat echt niet zo kan.

Goede oplossing: 2 punten.

(1b) Je kunt de tertiaan gebruiken om een gegarandeerd
goede splitsing te krijgen, vergelijkbaar met de mediaan-
der-medianen-methode. Probleem is dat je nog n vergelijkingen
nodig hebt voor de Split, en dan in recursie moet gaan op
2n/3 elementen. En de meetkundige reeks met factor 2/3 heeft
toch echt som 3, waarmee het totaal op 3(beta+1)n uitkomt.
Als je deze oplossing noemde kon je een punt verdienen, en
met een berekening van de complexiteit plus de constatering
dat de oplossing niet aan de gevraagde complexiteit voldoet,
een tweede punt.

Wat echter veel simpeler is, is de rij aan te vullen met
dummy elementen zo, dat het gevraagde element de tertiaan wordt.
Voorbeeld: stel je wilt de 9e uit 20 elementen hebben.
Van 25 elementen is de 9e de tertiaan, want ceil((25+1)/3 is 9.
Voeg daarom aan je rij 5 dummies toe die gelden als +oneindig,
maw deze nieuwe elementen zijn groter dan alle bestaande.
Van de 25 elementen bereken je de tertiaan.
Omdat je met de tertiaan ook de duotertiaan kunt vinden (volgens
(1a)) hoef je nooit meer dan n/2 elementen toe te voegen om de
gezochte waarde tot tertiaan of duotertiaan te maken.

3 punten voor deze oplossing.

Opgave 2: Zwaarste stijgende deelrij, 8 punten.
Bij deze opgave waren er meerdere kandidaten die niet wisten
en niet uit de omschrijving begrepen wat een deelrij was. De
omschrijving met een stijgende rij indices staat ook in het
boek, en bij de bespreking va de LCS is dit op het bord voor-
gedaan. Belangrijk is, dat de getallen van een deelrij NIET
aansluitend in A hoeven te staan. Zo'n aansluitend rijtje
noemen we een SEGMENT en dat wordt niet door een rijtje van
n' indices gekarakteriseerd, maar door slechts twee indices
namelijk een begin- en eindpunt in A.

(2a) "De grootste zit er altijd in", dit is NIET waar. Kijk
maar naar de rij 1 2 3 20 12 14 16.
Als je 20 in de deelrij opneemt, kun je niet 12, 14 en 16 nemen
maar toch is dat wel beter. De beste rij is 1 2 3 12 14 16
met gewicht 48.

Een dergelijk voorbeeld leverde 2 punten op, mits het om een
voorbeeld ging waarbij er geen ZSD met het maximum bestond.

(2b) De definitie K(i) is het maximale gewicht van een stijgende
deelrij van $A[1..i]$.
Het is erg moeilijk om deze definitie te gebruiken voor een
OSP redenering. Een deelrij van A[1..i] kan A[i] WEL of NIET
hebben. Als hij hem NIET heeft, is de deelrij natuurlijk een
zwaarste deelrij van A[1..i-1]. Maar als je A[i] WEL in de
rij hebt, weet je niet, of de zwaarste deelrij van A[1..i-1]
wel met dit element te combineren is. Je mag namelijk alleen

een deelrij van A[1..i-1] VOOR A[i] plakken, als het laatste element van die deelrij kleiner dan A[i] is.
Je hebt er dus NIET voldoende aan om te weten hoe zwaar de ZSD van A[1..i-1] is; en dit gaat ook op voor het Zwaarste Stijgende Segment.

Het antwoord NEE plus argumentatie leverde 1 punt.

(2c) De definitie L(i) is het maximale gewicht van een stijgende deelrij die eindigt met $A[i]$.
Hoe bepaal ik zo'n rij? Een deelrij die eindigt met A[i] heeft VOOR A[i] een van de eerdere getallen, zeg A[k] waar k<i, en begint dan met een deelrij die eindigt met A[k]. Hoe zwaarder des te beter! Maar A[k] moet wel kleiner dan A[i] zijn anders mag de deelrij eindigend met A[k] er niet voor.
Als er geen eerdere elementen zijn kleiner dan A[i], vormt A[i] zelf de zwaarste stijgende deelrij (met lengte 1) die met zichzelf eindigt. Als we het maximum van de lege verzameling even 0 nemen geldt dus
L(i) = A[i] + max_{k<i: A[k]<A[i]} L(k).

Deze karakterisering van L in zichzelf leverde 2 punten op.
De OSP eigenschap vraagt je wel, om L te karakteriseren in termen van L. Dat je K (uit opgave 2b) of het eindantwoord in termen van L kunt uitdrukken, mag je dus niet als OSP zien.

(2d) Het bepalen van L(i), als de kleinere L(k) al bekend zijn:

```
ZSD(i):
  L[i] = A[i] ; keuze[i] = 0 ;
  for k=i-1 downto 1
    if ( A[k] < A[i] and L[k]+A[i] > L[i] )
      { L[i] = L[k]+A[i] ; keuze[i] = k }
```

Het totale algoritme:

```
for i=1 to n { ZSD(i) }
```

bestaat uit twee geneste lussen en kost O(n^2) tijd.
Met de opgeslagen rij keuze[] kun je de rij zelf terugzoeken.

Dit algoritme leverde 3 punten op.

Opgave 3: Afstanden en BFS.
Een deel van deze opgave ging meer over de combinatoriek van paden dan over algoritmen.

(a) Te bewijzen |d[x] - d[y]| <= delta <= d[x] + d[y].
Dit volgt uit het driemaal toepassen van de driehoeks-
ongelijkheid. Die zegt dat voor elk drietal a,b,c geldt:
d(a,c) <= d(a,b) + d(b,c).
((Bewijs hiervoor: er bestaat een ab-pad van lengte d(a,b)
en een bc-pad van lengte d(b,c). De concatenatie hiervan
is een ac-pad van lengte d(a,b)+d(b,c). Het bestaan van
dit pad toont aan d(a,c) <= d(a,b) + d(b,c).))
Toegepast met
abc = xys geeft d(x,s) <= d(x,y)+d(y,s)
oftewel d[x] - d[y] <= delta;
abc = yxs geeft d(y,s) <= d(y,x)+d(x,s)
oftewel d[y] - d[x] <= delta;
Samen precies |d[x]-d[y]| <= delta.
abc = xsy geeft d(x,y) <= d(x,s) + d(s,y)
oftewel delta <= d[x]+d[y].

Deze redenering, of iets wat er een voldoende lange LCS mee
heeft, leverde 2 punten.

(b) Bewering: in een boom geldt gelijkheid delta = d[x]+d[y].
Dit is onjuist. Teken maar eens een boom met s in de wortel,
een kind t van s, kind u van t, en x en y kinderen van u.
Knopen x en y zitten nu op diepte 3, maar hebben onderlinge
afstand slechts 2.

Dit tegenvoorbeeld had bijna iedereen en leverde 1 punt.

(c) Een boomstructuur heeft DRIE eigenschappen:
(1) samenhangend; (2) n-1 kanten (3) geen cykels.
Nu is het zo, dat TWEE van deze eigenschappen voldoende zijn
om de derde te impliceren. Ik baseer mijn algoritme even op
de eigenschappen 1 en 2 (om niet aan cykel-detectie te hoeven
doen). Mijn algoritme doet een BFS en
(1) telt ondertussen de kanthelften door bij elke bezochte
knoop v een globale teller op te hogen met |Adj[v]|.
(2) checkt na afloop van een BFS vanuit 1 punt de samenhang
door te kijken of er nog witte knopen zijn.
(3) Een extra truuk nog om aan de O(n) tijd te komen:
zodra de kanthelftenteller over de 2n-2 gaat spring
ik uit de lus met antwoord NEE.

Een paar dingen die mis kunnen gaan (en dus ook gingen):

- Als je op cykels test door te kijken of een gevonden knoop nog wit is, moet je de vader van een knoop apart behandelen: die vind je namelijk altijd zwart, maar deze wijst niet op een cykel.
- Als je bij de BFS geen cykel vindt, ben je niet klaar! Check of de graaf samenhangend is (is alles bezocht?).
- Als je de hele BFS afmaakt, gebruik je O(n+m) tijd en dit kan te veel zijn! Je moet dus de BFS afbreken bij het vinden van een cykel, om bij (d) te kunnen scoren.

Een goed algoritme leverde 2 punten; eentje die een beetje brak was opgeschreven maar wel het Vader-probleem onderkende ook.

(d) Een BFS is normaal n+m werk, wat voor een boom O(n) is. Zou ik een BFS afmaken op een non-boom dan kan ik wel tot n^2 werk verspillen... maar zodra ik 2n-1 kanthelften heb gezien is het zeker dat de graaf geen boom is, en termineert mijn algoritme.

Een redenering die ik een paar keer tegenkwam: "Mijn oplossing doet BFS, met kleine extra's, en gebruikt DUS OOK O(n)." Dat kan natuurlijk niet! Je moet in je argument wel aandacht geven aan het afbreken, want juist daardoor kom je op O(n), ook in een dichte graaf.

Een goede redenering leverde 2 punten. Had je wel in je oplossing het afbreken ziteen maar dit niet beargumenteerd, dan 1 pt.