Tentamen Algoritmiek
Dinsdag 1 juni 2010, 10.00 – 13.00 uur

Geef een **duidelijke toelichting** bij al je antwoorden.

Puntenverdeling: 1: 20; 2: 15; 3: 25; 4: 30; 5: 10

Opgave 1. We bekijken het volgende probleem. Twee personen, Bonny en Clyde geheten, moeten in een lange rij met n huizen kranten bezorgen. Aangezien ze daar eigenlijk niet veel zin in hebben, maken ze er een wedstrijdje van. Ze spreken af dat ze om de beurt 1 of 2 kranten bezorgen, en wel als volgt: degene die aan de beurt is gooit ofwel 1 krant in een (nog lege) brievenbus, ofwel 2 kranten in twee naast elkaar gelegen (nog lege) brievenbussen. De wedstrijd is afgelopen als er in elke brievenbus precies 1 krant ligt. Degene die de laatste krant(en) bezorgt, heeft gewonnen. Er wordt afgesproken dat Bonny begint.

Een mogelijk spelverloop voor $n = 8$:

```
                  B                    C                    B
- - - - - - - - ⟶ - k k - - - - - ⟶ - k k - k - - - ⟶ - k k - k k k -
```

Clyde is vervolgens aan de beurt, en het is duidelijk dat hij uiteindelijk zal gaan winnen. Merk op dat een stand als `- k k - - k k k` speltechnisch (winnend voor wie aan de beurt is of niet) hetzelfde is als `k k - - k k - k`, en ook als `k - - k - k` (*).

a. Wat zijn voor dit spel toestanden en acties (voor algemene n)?
b. Beredeneer dat de gevallen $n = 2$, $n = 3$ en $n = 4$ winnend zijn voor degene die begint.
c. Ga na of het spel voor het geval $n = 5$ winnend of verliezend is voor Bonny. Teken daartoe de toestand-actie-ruimte met $n = 5$, uitgaande van de beginsituatie, waarbij Bonny begint. Geef bij elke toestand aan of deze winnend is voor Bonny of voor Clyde (bij perfect spel van beide spelers). Geef ook aan hoe Bonny danwel Clyde moet spelen om te winnen. Licht je antwoord toe.
Opmerking. Toestanden die speltechnisch (zie (*)) hetzelfde zijn hoef je maar één keer uit te werken. Geef wel aan waarom zo'n stand dan winnend is voor Bonny/Clyde.
d. Toon aan dat het spel voor algemene n winnend is voor degene die begint. Geef hiertoe een winnende strategie voor de beginnende speler (dus niet alleen een winnende beginzet).

Opgave 2. We bekijken het volgende toewijzingsprobleem. Gegeven zijn n producten die moeten worden gemaakt, en n fabrikanten die deze kunnen leveren. De kwaliteit van een product hangt af van de fabrikant die dat product maakt, en wordt weergegeven als een geheel getal tussen 1 en 10: fabrikant i maakt product j met kwaliteit `quality`$[i][j]$. De bedoeling is nu om een toewijzing te vinden van de producten aan de fabrikanten (één fabrikant per product en één product per fabrikant) met maximale totale kwaliteit.

Voorbeeld:

	product 1	product 2	product 3	product 4
fabrikant A	4	5	7	4
fabrikant B	2	5	9	5
fabrikant C	8	5	6	3
fabrikant D	9	8	4	6

De toewijzing A4, B3, C2, D1 heeft kwaliteit 27. Dit is *niet* maximaal.

a. Geef een *gretig* algoritme voor dit toewijzingsprobleem en pas het toe op het voorbeeld. Levert je algoritme een toewijzing met maximale totale kwaliteit op?
b. Beschrijf in woorden een best-fit-first *branch and bound* algoritme dat bovenstaand probleem oplost. Pas het algoritme vervolgens toe op het voorbeeld en teken de bijbehorende state-space-tree. Geef daarin ook aan in welke volgorde de knopen bekeken worden. Wat voor afschatting van de te verwachten totale kwaliteit gebruik je voor deeloplossingen?

Opgave 3. Gegeven een binaire boom met ingang `wortel`. Een knoop van de boom ziet er uit zoals hieronder is aangegeven.

```
struct knoop {
   knoop* links;
   knoop* rechts;
   knoop* opa;
   int info;
}; // knoop
```

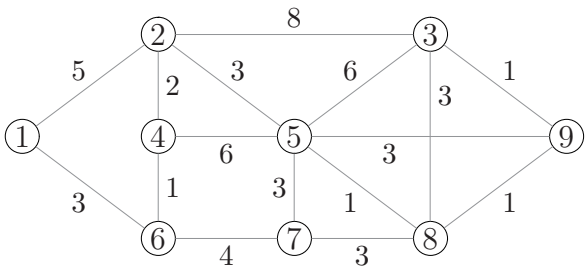
a. Schrijf een *recursieve* C++-functie `void initialiseer(knoop* wortel)`, die alle `opa`-velden op `NULL` zet.
b. Neem aan dat de `opa`-velden gevuld zijn zoals in **a.** bedoeld. Schrijf nu een *recursieve* C++-functie `void grootvader(knoop* wortel)`, die in elke knoop de `opa`-pointer laat wijzen naar diens opa (dus de vader van zijn vader). Indien de knoop geen opa heeft moet het `opa`-veld NULL blijven.
c. Gegeven is een pointer `wijzer` die naar een knoop in de boom wijst. De wortel van de boom is nu niet bekend. Schrijf een niet-recursieve C++-functie `int root(knoop* wijzer)` die de waarde van het info-veld van de wortel oplevert. We nemen aan dat de knoop waar `wijzer` naar wijst op nivo 2 of lager zit, en dus een opa heeft.

Opgave 4. Gegeven een oplopend gesorteerd array A ($A[0], \ldots, A[n-1]$, met $n \geq 2$), dat n gehele getallen bevat. Verder is bekend dat $A[0] = 1$ en $A[n-1] = m < n$.

a. Leg uit waarom je zeker weet dat het array een getal bevat dat minstens tweemaal in A voorkomt.
b. Geef een decrease-by-one algoritme dat een waarde oplevert die minstens tweemaal voorkomt. Schrijf hiertoe een recursieve C++-functie `dubbel1(i,A)`.
c. Geef een divide-and-conquer algoritme voor het probleem. Verdeel hierbij het array in twee gelijke delen. Neem aan dat n een 2-macht is. Hier moet dus een recursieve C++-functie `dubbel2(l,r,A)` worden geschreven die het probleem oplost voor het deelarray $A[\mathtt{l}], \ldots, A[\mathtt{r}]$ ter lengte een 2-macht.

d. Geef ten slotte een decrease-by-half algoritme dat een element oplevert dat minstens tweemaal voorkomt. Neem ook hier aan dat n een 2-macht is. Schrijf hiervoor een C++-functie `dubbel3(l,r,A)`. Leg uit waarom je methode werkt, dus waarom je steeds maar maximaal één helft verder hoeft te bekijken.

Opgave 5. Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.



Illustreer de werking van het algoritme van Dijkstra door het toe te passen op bovenstaande graaf, beginnend in knoop 1. Geef in een aparte toelichting voor elke stap van het algoritme duidelijk de labels van de nog niet gekozen knopen, en welke knoop erbij wordt gekozen in U (= verzameling knopen waarvan de kortste afstand vanaf 1 bekend is). Geef ten slotte ook de boom van kortste paden.

Veel succes !