Tentamen Algoritmiek
Dinsdag 5 juni 2007, 10.00 – 13.00 uur

Geef een duidelijke toelichting bij je antwoorden. Veel succes!

Opgave 1. We bekijken het tweepersoonsspelletje Grundy's Game. Het wordt gespeeld met n munten ($n > 1$, geheel). Deze liggen bij aanvang op één stapel, boven op elkaar, op tafel. Er zijn twee spelers, Jip en Janneke, die om de beurt een zet doen. Ze spreken af dat Jip begint. Een zet bestaat hier uit het verdelen van één van de op tafel liggende stapels munten in twee *ongelijke* stapels (d.w.z. twee stapels die verschillende aantallen munten bevatten). De speler die geen stapel meer kan vinden die in twee ongelijke stapels kan worden verdeeld verliest. Merk op dat na elke zet het aantal stapels met één toeneemt.

Voorbeeld met $n = 15$. Stel dat Jip de stapel verdeelt in een stapel van 9 en een stapel van 6. Janneke kiest vervolgens één van de twee stapels, bijvoorbeeld die van 6. Die kan ze op twee manieren verdelen: in stapels van 5 en 1 of in stapels van 4 en 2. Ze kiest bijvoorbeeld voor de tweede mogelijkheid. Dan is Jip weer aan de beurt, etcetera. Een mogelijk spelverloop is dan: $15 \longrightarrow 9,6 \longrightarrow 9,4,2 \longrightarrow 5,4,4,2 \longrightarrow 3,2,4,4,2 \longrightarrow$ etc.

a. Wat zijn voor dit spel toestanden en acties (voor algemene n)?
b. Hoeveel zetten kunnen maximaal gedaan worden totdat een der spelers gewonnen heeft (algemene n)? Leg je antwoord uit.
c. Teken de toestand-actie-ruimte voor $n = 7$, uitgaande van de beginsituatie.
d. Geef bij elke toestand aan of deze winnend is voor Jip of voor Janneke. (Een toestand is winnend voor een speler als die speler altijd kan winnen, ongeacht wat de tegenstander doet.) Is het spel voor $n = 7$ winnend voor Jip of voor Janneke?

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 $\texttt{quality}[i][j]$. 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	3	5	8	4
fabrikant B	4	7	9	2
fabrikant C	3	5	7	8
fabrikant D	7	3	4	9

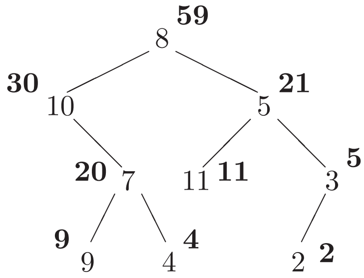
De toewijzing A1, B2, C3, D4 heeft kwaliteit 26. Dit is *niet* maximaal.

a. Beschrijf een *exhaustive search* algoritme dat dit toewijzingsprobleem oplost.
b. Beschrijf ook een *backtracking* algoritme voor het probleem en licht dit toe aan de hand van het voorbeeld. Is dit backtracking algoritme beter dan het exhaustive search algoritme uit **a.**?
c. Beschrijf een best-fit-first *branch and bound* algoritme dat bovenstaand problem oplost. Doe dit aan hand van het voorbeeld: pas het algoritme daarop toe 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`, die gehele getallen bevat. Een knoop van de boom ziet er uit als hieronder links is aangegeven.

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

Voorbeeld:



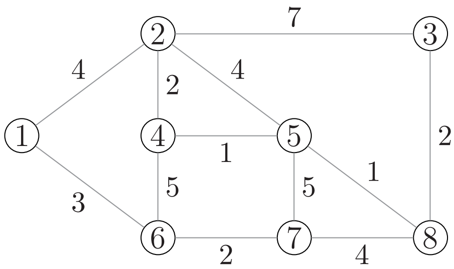
a. Schrijf een *recursieve* C++-functie `void init(knoop* wortel)`, die de som-velden van alle knopen uit de boom op nul zet.
b. Laten alle som-velden op nul geïnitialiseerd zijn. Schrijf nu een *recursieve* C++-functie `optellen(knoop* wortel)`, die in elke knoop het som-veld vult met de som van alle info-waarden uit de subboom met die knoop als wortel.
Voorbeeld: in de boom hier rechts boven is bij elke knoop de inhoud van het som-veld dikgedrukt aangegeven.

Opgave 4. Gegeven een array A dat n verschillende gehele getallen $A[1], A[2], \ldots, A[n]$ bevat, oplopend gesorteerd. We gebruiken verdeel en heers om een index i te vinden waarvoor geldt dat $A[i] = i$. Als er niet zo'n index bestaat moet 0 worden geretourneerd; als er meerdere zijn moet één daarvan (het maakt niet uit welke) worden teruggegeven.

a. Geef een divide-and-conquer algoritme (in pseudocode of C++) voor het vinden van een index i met $A[i] = i$. Verdeel hierbij het array in twee ongeveer gelijke delen.
b. Geef een decrease-by-half algoritme (in pseudocode of C++) voor het bepalen van deze index. Leg uit waarom het werkt. Gebruik daarbij wat gegeven is over het array.

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

a. Geef de C++-representatie van de adjacency-list voor grafen met gewichten op de takken en laat zien hoe deze er uitziet voor onderstaande graaf. Teken een plaatje.



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

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

Bijlage bij het tentamen Algoritmiek (5 juni 2007)

Het algoritme van Dijkstra

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array `pad` dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is `pad`$[v]$ = de lengte van een kortste pad van s naar v
// bij aanvang is `pad`[s] = 0; `pad`$[v] = \infty$ voor de andere knopen
$U := \emptyset$; // U bevat de knopen waarvoor de kortste afstand vanuit s bekend is
while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met `pad`$[v^*]$ minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if `pad`$[v^*]+$ `gewicht`$(v^*, v) <$ `pad`$[v]$ **then**
            `pad`$[v] :=$ `pad`$[v^*]+$ `gewicht`(v^*, v);
        fi
    od
od