**Tentamen Algoritmiek**
**Dinsdag 4 augustus 2009, 10.00 – 13.00 uur**

Geef een duidelijke toelichting bij al je antwoorden. **Veel succes!**

**Opgave 1.** (20 punten)
In deze opgave bekijken we het volgende cijferspel. Het wordt gespeeld door twee spelers, in het vervolg Michael (M) en Janet (J) geheten. Bij het begin van het spel liggen de 10 cijfers $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ midden op tafel. De spelers moeten nu om de beurt een cijfer van het midden van de tafel halen en voor zichzelf op tafel neerleggen. Op elk moment kunnen beide spelers precies zien wie welke cijfers in zijn/haar bezit heeft. De speler die het eerst *drie* cijfers in bezit heeft waarvan de som gelijk is aan 15, heeft gewonnen. Merk op dat als je bijvoorbeeld twee of vier cijfers hebt die opgeteld samen 15 zijn, je dan (nog) niet hebt gewonnen: je moet met precies drie cijfers 15 kunnen maken. Het spel is afgelopen wanneer een der spelers zo'n som 15 heeft, of wanneer alle cijfers weggenomen zijn zonder dat er een winnaar is (remise). We nemen aan dat Michael begint.
Voorbeeld van een spelverloop: M neemt 4, J neemt 7, M neemt 6, J neemt 5, M neemt 3, J neemt 8, M neemt 0, J neemt 1, M neemt 9 en wint (want $6 + 0 + 9 = 15$).

**a.** Wat zijn voor dit spel toestanden en acties?
**b.** Teken de toestand-actie-ruimte, uitaande van de toestand direct nadat elke speler drie keer aan de beurt is geweest in bovenstaand voorbeeldspelverloop (dus net nadat J de 8 gepakt heeft). Geef alle vier de directe vervolgstanden, maar werk verder alleen het geval waarbij M de 0 pakt geheel uit.
**c.** Geef bij *elke* toestand, dus ook voor de drie in **b.** bedoelde niet-uitgewerkte directe vervolgstanden (*), aan of deze winnend is voor M of voor J. (Hierbij wordt zoals gebruikelijk steeds verondersteld dat beide speler optimaal spelen, d.w.z. beide doen steeds de/een zet die naar het beste resultaat leidt.) Hoe moet M dan wel J spelen om te winnen? Licht je antwoord toe.
(*) Geef bij deze drie wel duidelijk aan waarom ze winnend zijn voor M/J.
**d.** Wat wordt het spelverloop als M in zijn tweede beurt de 8 pakt (in plaats van de 6) en als we aannemen dat beide spelers optimaal spelen? Beredeneer je antwoord, dus teken hier geen toestand-actie-ruimte. Wie wint in dit geval?
**e.** Als M in zijn tweede beurt de 1 pakt, en J vervolgens de 6, eindigt het spel (als beiden optimaal spelen) in remise. Laat dit zien zonder de hele toestand-actie-ruimte te tekenen.

**Opgave 2.** (15 punten)
Bekijk het partitie-probleem: gegeven een verzameling $V$ van $n$ verschillende gehele getallen $> 0$, met som $S$. We zoeken —indien mogelijk— *een* opsplitsing van $V$ in twee disjuncte deelverzamelingen waarvan de som der elementen even groot is (dus gelijk aan $S/2$). Voorbeeld: $\{6, 8, 4, 5, 1\}$ is op te splitsen in $\{6, 5, 1\}$ en $\{8, 4\}$, beide met som 12.

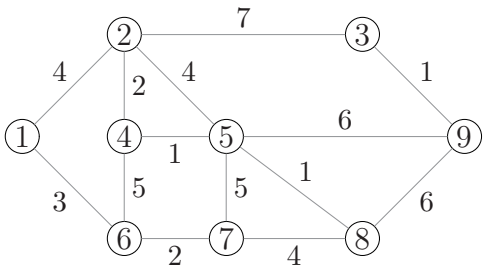
**a.** Geef in woorden een *exhaustive search* algoritme voor dit probleem (met algemene $n$).
**b.** Beschrijf een *backtracking* algoritme voor dit probleem (met algemene $n$). Vermeld daarbij duidelijk hoe (deel)oplossingen stap voor stap gegenereerd worden en wat de restrictie is waaraan deeloplossingen moeten voldoen.
**c.** Illustreer de werking van je backtracking algoritme door het toe te passen op het bovengegeven voorbeeld, en teken de bijbehorende state-space-tree. Geef daarin ook de volgorde aan waarin de knopen bekeken worden.

**Opgave 3.** (25 punten)
Gegeven een array $A$ ($A[0], \ldots, A[n-1]$, met $n \geq 1$), dat gehele getallen bevat. Het array moet —via verwisselingen— zo gereorganiseerd worden dat alle negatieve getallen vooraan komen te staan, en alle niet-negatieve getallen achteraan.

**a.** Geef een decrease by one algoritme in C++ dat het array reorganiseert. Er moet dus een recursieve functie `zetgoed`$(A, i)$ worden geschreven die het probleem oplost voor het deelarray $A[0], \ldots, A[i]$.
**b.** Geef een divide-and-conquer algoritme in C++ dat het array reorganiseert. Verdeel hierbij het array in twee gelijke delen. Neem aan dat $n$ een 2-macht is. Hier moet dus een recursieve functie `reorganiseer`$(A, \ell, r)$ worden geschreven die het probleem oplost voor het deelarray $A[\ell], \ldots, A[r]$ ter lengte een 2-macht.
**c.** Schrijf een eenvoudig (=lineair) algoritme (in C++) voor dit probleem.

**Opgave 4.** (15 punten)
Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.

**a.** Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop 1. Geef voor elke stap van het algoritme duidelijk de labels van de knopen (en hoe die eventueel veranderen), en welke knoop erbij wordt gekozen in $U$ (= verzameling knopen waarvan de kortste afstand vanaf 1 bekend is). Bouw ook de de uiteindelijke boom van kortste paden op met daarin de bijbehorende lengtes van de kortste paden.




**b.** Het algoritme van Dijkstra werkt niet voor grafen met negatieve gewichten. Geef een klein voorbeeld (een graaf met drie of vier knopen is al genoeg) waaruit dit blijkt.

**Opgave 5.** (25 punten)
Gegeven is een binaire boom met ingang `wortel`, die verschillende getallen bevat. Hierin is `wortel` een pointer naar een `knoop`, waarbij:

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

De binaire boom voldoet aan de heapeigenschap als in *elke* knoop geldt dat de waarde die in die knoop is opgeslagen groter is dan de waarde die is opgeslagen in zijn kinderen.

**a.** Schrijf een recursieve C++-functie `bool heap(knoop* wortel)`, die bepaalt of de boom aan de heapeigenschap voldoet.
**b.** Teken de met de rij 44 77 33 50 24 16 61 68 85 40 corresponderende complete binaire boom en breng deze m.b.v. *heapify* in hoopstructuur. Laat tussenstappen zien en geef ook de met het eindresultaat corresponderende rij.
**c.** Leg uit hoe het sorteeralgoritme Heapsort werkt. Voer ter illustratie daarbij de eerste drie stappen van Heapsort uit op de hoopstructuur/rij die in **b.** verkregen is.