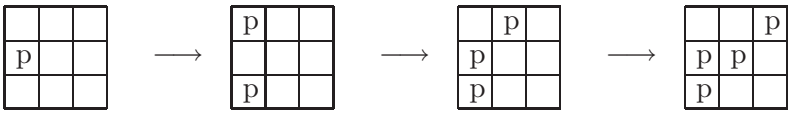
Tentamen Algoritmiek
Dinsdag 5 augustus 2008, 10.00 – 13.00 uur

Geef een duidelijke toelichting bij al je antwoorden.

Opgave 1. We bekijken het volgende eenpersoonsspelletje. Het wordt gespeeld op een m bij n bord (met $m, n > 1$). In de beginsituatie staat op precies één vakje een pion. Een zet bestaat uit het weghalen van een pion naar keuze, en het tegelijkertijd neerzetten van twee pionnen op aangrenzende (horizontaal/verticaal) lege velden. Als een vakje met een pion erop geen twee aangrenzende lege velden heeft kan er niet met die pion gespeeld worden. De bedoeling is dat aan het eind op op een na alle vakjes een pion staat.

Voorbeeld van een serie van drie opeenvolgende zetten ($m = n = 3$):



begin (A) (B)

Toelichting: vanuit stand (A) zijn er in totaal vier zetten mogelijk; je kunt de pion uit de eerste rij weghalen en dan heb je drie aangrenzende lege velden en (dus) drie manieren waarop je de twee nieuwe pionnen kunt neerzetten. De pion op de tweede rij heeft twee lege buurvelden, dus die levert één mogelijke zet. De pion in de derde rij heeft slechts één leeg buurvakje, dus die levert geen zet op.

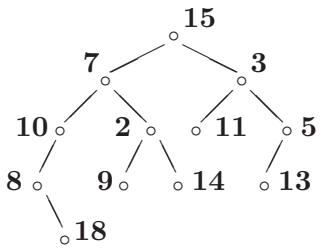
a. Wat zijn voor dit spel toestanden en acties (voor algemene m en n)?
b. (i) Geef (een goede bovengrens voor) het aantal mogelijke toestanden zoals in **a** gedefiniëerd. Leg uit hoe je aan dit aantal komt. (ii) Zijn hierbij nog illegale toestanden (d.w.z. toestanden die bij het spelen van het spel volgens de regels niet bereikt kunnen worden)? Zo ja, geef een voorbeeld.
c. We bekijken het geval $m = n = 3$. Teken nu dat deel van de toestand-actie-ruimte dat alle toestanden (en verbindende acties) bevat die in hooguit twee zetten uit toestand (B) verkregen kunnen worden.
d. Geef een opeenvolging van zetten die leidt van (B) naar een eindtoestand (op acht vakjes een pion en één vakje leeg).

Opgave 2. Gegeven een binaire boom met ingang `wortel`. Een knoop van de boom ziet er uit als hieronder links is aangegeven. Bij aanvang hebben de `vader`-velden nog geen waarde.

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

Voorbeeld:

Bij de knopen staat de waarde van het info-veld vermeld.

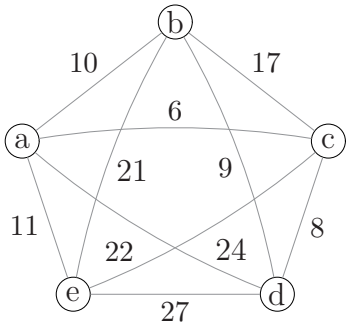


a. Schrijf een recursieve C++-functie `void init(knoop* wortel)`, die de `vader`-velden van alle knopen op NULL zet.
b. We gaan nu voor elke knoop de `vader`-pointer laten wijzen naar zijn ouder in de boom. In de wortel is de `vader`-pointer NULL. Schrijf hiertoe een recursieve C++-functie

`void ouders(knoop* wortel)`.
c. Neem aan dat alle `vader`-velden gevuld zijn zoals in **b.** bedoeld. Schrijf een niet-recursieve C++-functie `knoop* oom(knoop* p)`, die een pointer naar de oom van de knoop waar p naar wijst retourneert (indien die bestaat; anders NULL). De wortel van de boom is hierbij onbekend. In de voorbeeldboom: de oom van 14 is 10; 18 en 3 hebben geen oom, etc.

Opgave 3. Het handelsreizigersprobleem (Traveling Salesman Problem) luidt: gegeven een complete, ongerichte graaf met n knopen en met gewichten op de takken. Geef een Hamiltonkring met minimaal totaalgewicht.

a. Beschrijf een *backtracking* algoritme voor het probleem met algemene n en licht dit toe aan de hand van onderstaand voorbeeld. Teken ook een gedeelte (ongeveer een kwart) van de bijbehorende state space tree bij het voorbeeld, en geef daarin de volgorde aan waarin de knopen bekeken worden.



De kring abedca is een Hamiltonkring met totaalgewicht $10+21+27+8+6 = 72$. Dit is *niet* minimaal.

b. Zowel bij backtracking als bij branch and bound worden oplossingen stap voor stap gegenereerd. Leg uit hoe best-fit-first *branch and bound* werkt voor minimalisatieproblemen in het algemeen. Geef daarbij duidelijk aan wat de verschillen zijn met backtracking en waarom branch and bound (meestal) sneller een minimale oplossing zal vinden. Geef ook een mogelijk nadeel van branch en bound t.o.v. backtracking.
c. Beschrijf nu een best-fit-first branch and bound algoritme dat het handelsreizigersprobleem oplost. Doe dit aan de 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 het te verwachten totaalgewicht gebruik je voor deeloplossingen?

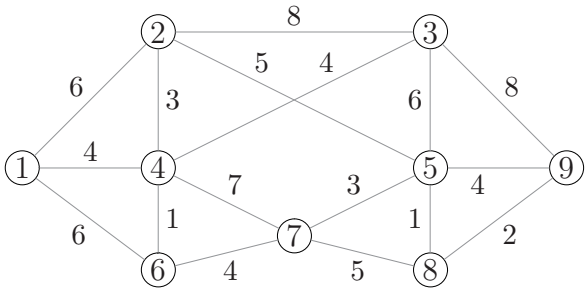
Opgave 4. Gegeven een array A ($A[1], \ldots, A[n]$, met $n \geq 2$), dat n *verschillende* gehele getallen bevat. Gevraagd worden de index van de grootste waarde en de index van de kleinste waarde uit A.

a. Geef een eenvoudig iteratief algoritme (in pseudocode of C++) voor dit probleem dat slechts één for-loop gebruikt.

We gaan nu verdeel en heers gebruiken om het probleem op te lossen.
b. Geef een decrease by two algoritme (in pseudocode of C++) voor het probleem. Neem hierbij aan dat n even is. Er moet hiertoe een recursieve functie `selectie(i,gr,kl)` worden geschreven die de index `gr` van de grootste en de index `kl` van de kleinste waarde oplevert uit het deelarray $A[1], \ldots, A[\mathtt{i}]$, met `i` even.
c. Geef een divide-and-conquer algoritme (in pseudocode of C++) voor het probleem waarbij n een 2-macht is. Verdeel hierbij het array in twee gelijke delen. Er moet dus een recursieve functie `selecteer(i,j,gr,kl)` worden geschreven die de index van de grootste en de kleinste waarde oplevert uit het deelarray $A[\mathtt{i}], \ldots, A[\mathtt{j}]$, ter lengte een 2-macht.

Opgave 5. 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). Geef ook de uiteindelijke boom van kortste paden 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 en licht je antwoord toe.

Veel succes!

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