Tentamen Algoritmiek
Dinsdag 7 augustus 2007, 10.00 – 13.00 uur

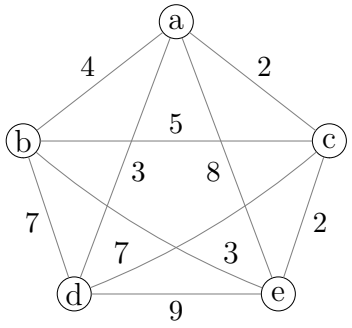
Geef een duidelijke toelichting bij je antwoorden. Veel succes!

Opgave 1. We bekijken een variant op het Nim-spel. Op tafel liggen n (> 1) lucifers. Er zijn twee spelers, Bert en Ernie. De speler die begint mag minimaal 1 en maximaal $n-1$ lucifers van tafel nemen. Vervolgens moeten de spelers om de beurt minimaal 1 lucifer en maximaal twee keer het aantal lucifers dat de tegenstander zojuist heeft gepakt van tafel nemen. De speler die de laatste lucifer(s) wegneemt heeft gewonnen. Neem aan dat Bert begint. *Voorbeeld*: stel dat Ernie op een gegeven moment twee lucifers van tafel neemt. Dan mag Bert in de volgende zet 1, 2, 3 of 4 lucifers pakken, maar niet meer.

a. Wat zijn voor dit spel toestanden en acties (voor algemene n)?
b. Als Bert in de eerste zet $n-1$ lucifers wegneemt, kan Ernie vervolgens in één zet winnen. Hetzelfde is het geval (als $n > 2$ tenminste) als Bert er $n-2$ wegneemt. Leg uit wat (voor algemene n) het minimale aantal is waarvoor geldt: als Bert in de eerste beurt dit aantal lucifers wegneemt, dan kan Ernie vervolgens in één zet winnen. Neem voor het gemak aan dat n een drievoud is.
c. Teken de toestand-actie-ruimte voor $n = 7$, uitgaande van de beginsituatie. Toestanden die in één zet te winnen zijn hoef je niet verder uit te werken. Schrijf er wel bij wie wint.
d. Geef bij *elke* toestand aan of deze winnend is voor Bert of voor Ernie. (Een toestand is winnend voor een speler als deze altijd kan winnen, ongeacht wat de tegenstander doet.) Is het spel met $n = 7$ winnend voor Bert of voor Ernie? Hoe moet hij spelen om te winnen?

Opgave 2. Het handelsreizigersprobleem (Traveling Salesman Problem) luidt: gegeven een complete, ongerichte graaf met n knopen (=steden) en met gewichten (=lengtes) op de takken (=wegen). Geef een Hamiltonkring met minimaal totaalgewicht.

a. Beschrijf een *backtracking* algoritme voor het probleem en licht dit toe aan de hand van onderstaand voorbeeld. (Er wordt geen state space tree gevraagd.)



De kring adebc is een Hamiltonkring met totaalgewicht 3+9+3+5+2 = 22. Deze is echter *niet* optimaal.

b. Beschrijf 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?
c. Formuleer een *gretig* algoritme voor dit probleem. Levert dit algoritme voor elke gewogen graaf een Hamiltonkring met minimaal totaalgewicht? Zo ja, leg uit waarom je dat denkt. Zo nee, geef een klein tegenvoorbeeld.

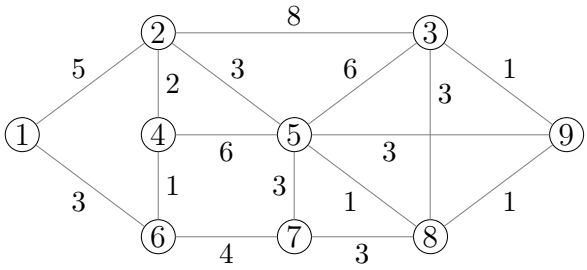
Opgave 3. Gegeven een array A dat n (≥ 2) gehele getallen $A[1], A[2], \ldots, A[n]$ bevat, en een waarde W. Gevraagd worden indices i en j ($i \neq j$) waarvoor $A[i] + A[j] = W$.

a. Geef een brute force algoritme in pseudocode of C++, dat zulke indices oplevert indien ze bestaan (anders krijgen i en j beide de waarde 0).
b. Geef een divide-and-conquer algoritme (in pseudocode of C++) voor bovenstaand probleem. Verdeel hierbij het array in twee gelijke delen. Je mag voor het gemak aannemen dat n een tweemacht is.

Nu een ander probleem. We hebben een array A met n (≥ 1) verschillende gehele getallen $A[1], A[2], \ldots, A[n]$. Verder is gegeven dat er een index p met $1 \leq p \leq n$ bestaat zodat A stijgend is tot index p, en daarna dalend.
Voorbeeld: als $A = 3\ \ 6\ \ 9\ \ 11\ \ 8\ \ 2$, dan is $p = 4$. *Randgevallen*: als $A = 7\ \ 5\ \ 3\ \ 2\ \ 1$, dan $p = 1$; als $A = 5$ (dus bestaat uit 1 element), dan $p = 1$; als $A = 4\ \ 9$, dan $p = 2$.

c. Geef een decrease-by-half algoritme (in pseudocode of C++) voor het bepalen van deze index p en leg uit waarom het werkt.

Opgave 4. 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 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.

Opgave 5. **a.** Wanneer is een binaire boom een heap (=hoopstructuur)?
b. We hebben een heap die allemaal verschillende gehele getallen bevat. Waar bevindt zich de grootste waarde? Waar moet de op een na grootste waarde zich bevinden (meer dan één mogelijkheid)? En waar de op twee na grootste (ook meer dan één mogelijkheid)?
c. Teken de met de rij 51 40 60 34 77 65 46 85 70 20 57 corresponderende complete binaire boom en breng deze m.b.v. *heapify* in hoopstructuur. Laat tussenstappen zien.
d. Leg uit hoe het sorteeralgoritme heapsort werkt. Voer ter illustratie de eerste drie herhaalstappen van heapsort uit op de hoopstructuur die in **c.** verkregen is.

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

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