Tentamen Algoritmiek
Dinsdag 2 augustus 2011, 10.00 – 13.00 uur

Geef een duidelijke toelichting bij al je antwoorden.
Puntenverdeling: 1: 22; 2: 20; 3: 25; 4: 21; 5: 12

Opgave 1. We bekijken een variant van het tweepersoonsspel Nim. Bij het begin van het spel liggen er twee stapels lucifers op tafel. De ene stapel heeft m lucifers, de andere n. Er zijn twee spelers, Hans en Grietje, die om de beurt een zet doen. Er zijn twee soorten zetten: 1. het wegnemen van één of twee lucifers van één van de twee stapels, of 2. het wegnemen van *evenveel* lucifers van elk van beide stapels. Het spel is afgelopen zodra er geen lucifers meer op tafel liggen. De speler die de laatste lucifer(s) heeft weggehaald heeft gewonnen. Er wordt afgesproken dat Hans begint.
Merk op dat, als er nog slechts één stapel op tafel ligt, dit spel reduceert tot "gewoon" Nim.
Voorbeeld van een mogelijk spelverloop, met $m = 15$ en $n = 19$:

```
       H         G         H         G         H         G         H
15,19 ---> 9,13 ---> 9,11 ---> 5,7 ---> 4,7 ---> 4,5 ---> 4,4 ---> 0,0
```

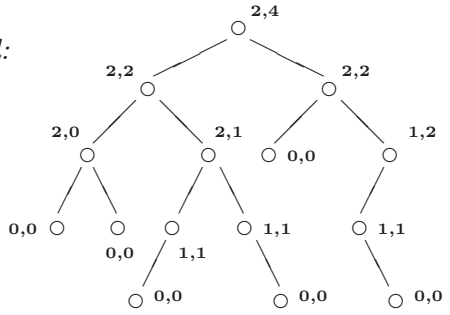
Hans begint en neemt van beide stapels 6 lucifers weg, vervolgens neemt Grietje er 2 van de grootste stapel; in de laatste zet wint Hans.

a. Wat zijn voor dit spel toestanden en acties (voor algemene m en n)?
b. (i) Hoeveel zetten kunnen er maximaal gedaan worden totdat één der spelers gewonnen heeft?
(ii) Dezelfde vraag als **(i)**, maar nu voor het minimale aantal zetten.
c. Teken de toestand-actie-ruimte voor het geval $m = 2$ en $n = 4$, uitgaande van de beginsituatie, waarbij Hans begint. Geef bij *elke* toestand aan of deze winnend is voor Hans of voor Grietje. Toestanden die in één zet te winnen zijn hoef je verder niet uit te werken. Verder hoef je toestanden die (speltechnisch) hetzelfde zijn maar één keer uit te werken. Geef wel een verwijzing naar de overeenkomstige toestand.
d. Bekijk het speciale geval waarin we een stapel van ℓ en een stapel van $\ell + 2$ (met $\ell > 2$) lucifers hebben. Beredeneer of het spel nu gewonnen is voor Hans (de beginnende speler) of voor Grietje. Motiveer je antwoord en geef aan hoe diegene kan winnen.

Opgave 2. Gegeven een binaire boom met ingang `wortel`. Een knoop van de boom ziet er uit als hieronder links is aangegeven. De `kind`-velden en de `aantaleen`-velden zijn bij aanvang van deze opgave nog niet geïnitialiseerd.

```
struct knoop {
   knoop* links;
   knoop* rechts;
   int kind;
   int aantaleen;
}; // knoop
```

Voorbeeld:



-vervolg Opgave 2-

a. Schrijf een *recursieve* C++-functie `void initialiseer(knoop* wortel)`, die van alle knopen in de (sub-)boom met ingang `wortel` het `kind`-veld vult met het aantal kinderen dat de betreffende knoop heeft.
b. Schrijf een *recursieve* C++-functie `void enigkind(knoop* wortel)`, die in elke knoop het `aantaleen`-veld vult met het aantal knopen uit de subboom met de betreffende knoop als wortel, die precies één kind hebben. (Dat aantal is dus inclusief de knoop zelf als die één kind heeft.) In bovenstaand voorbeeld wordt bij elke knoop de inhoud van het `kind`-veld en het `aantaleen`-veld gegeven.

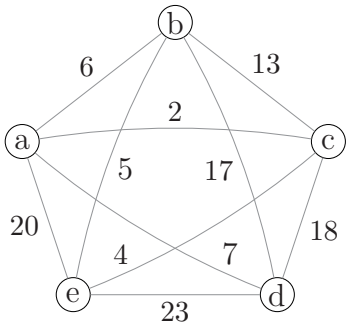
We willen nu een gegeven nieuwe knoop op een speciale manier aan de boom toevoegen. Zoals gebruikelijk voegen we de nieuwe knoop als blad toe. We willen dat nu echter doen onder een knoop die zelf maar één kind heeft. Als zo'n knoop niet bestaat voegen we de nieuwe knoop niet toe. We zoeken zo'n knoop door, beginnend in de wortel, de `eenkind`-velden te gebruiken om de juiste kant op te lopen.

c. Schrijf een *niet-recursieve* C++-functie `void voegtoe(knoop* wortel, knoop* nieuw)`, die de nieuwe knoop in de boom opbergt op bovenstaande manier. De pointer `nieuw` is een pointer naar de nieuwe knoop, waarvan de vier velden nog gevuld moeten worden. Tijdens het zoeken naar de juiste plek om toe te voegen moeten ook de `kind`-velden en `aantaleen`-velden van de onderweg tegengekomen knopen worden aangepast indien nodig. We nemen aan dat de boom niet leeg is.

Opgave 3. Gegeven een array A dat n (≥ 2, even) gehele getallen $A[0], A[1], \ldots, A[n-1]$ bevat. Er is verder gegeven dat op de even posities (dus $0, 2, 4, \ldots$) positieve (> 0) getallen staan, en op de oneven posities negatieve (< 0) getallen. Bovendien zijn de positieve getallen onderling oplopend gesorteerd, en geldt hetzelfde voor de negatieve getallen. Een voorbeeld van zo'n rijtje is $2, -8, 3, -5, 7, -3, 9, -1$.
Het array moet zo gereorganiseerd worden dat alle getallen na afloop oplopend gesorteerd zijn. Maak bij het reorganiseren gebruik van de speciale vorm van het array; er mogen bij **a.**, **b.** en **c.** *geen vergelijkingen* tussen array-elementen worden gedaan. Houd je bovendien aan de aanwijzingen bij de afzonderlijke onderdelen. Het voorbeeldrijtje ziet er na afloop van de reorganisatie als volgt uit: $-8, -5, -3, -1, 2, 3, 7, 9$

a. Geef een decrease-by-two algoritme dat het array A reorganiseert zoals hierboven aangegeven. Schrijf hiertoe een *recursieve* C++-functie `void reorganiseer(A,i)`, die de reorganisatie uitvoert op $A[0], A[1], \ldots, A[i]$.
b. We nemen aan dat n een 2-macht is. Geef een divide-and-conquer algoritme in C++ voor bovenstaand probleem. Het array dient hiervoor in twee gelijke delen te worden verdeeld. Schrijf hiertoe een *recursieve* C++-functie `void reorganiseer2(A,links,rechts)` die het probleem oplost voor het deelarray A[`links`], ..., `A`[`rechts`] ter lengte een 2-macht, gebruikmakend van verwisselingen.
c. Stel dat gegeven is dat het array A alleen getallen -42 en +42 bevat, en er dus bijvoorbeeld voor $n = 6$ uitziet als $42, -42, 42, -42, 42, -42$. Schrijf een eenvoudige *iteratieve* C++-functie `void reorganiseer3(A,n)` die het array reorganiseert en dat daarbij zo weinig mogelijk verwisselingen gebruikt. Hoeveel verwisselingen doet je algoritme?

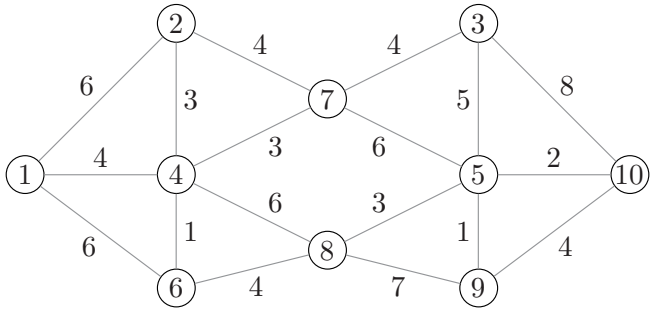
Opgave 4. Het handelsreizigersprobleem (Traveling Salesman Problem) luidt: gegeven een complete, ongerichte graaf met n knopen en met gewichten op de takken. Geef een Hamiltonkring met minimaal totaalgewicht.



De kring abdec is een Hamiltonkring met totaalgewicht $6+17+23+4+2 = 52$. Dit is *niet* minimaal.

a. Leg uit hoe best-fit-first *branch and bound* werkt voor minimalisatieproblemen in het algemeen. Geef daarbij o.a. aan hoe (deel)oplossingen gegenereerd worden, wat met branch bedoeld wordt en wat met bound, wat best-fit-first betekent, wanneer gesnoeid wordt, enzovoort.
b. Beschrijf nu een concreet best-fit-first branch and bound algoritme dat ons probleem oplost. Pas het algoritme 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 het te verwachten totaalgewicht gebruik je voor deeloplossingen? Geef een duidelijke uitleg.

Opgave 5. Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.
Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop 1. Geef voor elke stap van het algoritme duidelijk aan welke knoop erbij wordt gekozen in U (= verzameling knopen waarvan de kortste afstand vanaf 1 bekend is) en welke labels door die keuze veranderen en hoe. Doe dit elke stap als volgt: Kies knoop ...; label knoop ... wordt ..., label knoop ... wordt ..., etcetera.
Geef ook de uiteindelijke boom van kortste paden met daarin de bijbehorende lengtes van de kortste paden vanaf 1 !



Veel succes !