Tentamen Algoritmiek
Dinsdag 3 augustus 2010, 10.00 – 13.00 uur

Geef een duidelijke toelichting bij al je antwoorden.

Opgave 1. Het tweepersoonsspel Misère-Nim wordt hetzelfde gespeeld als Nim, behalve dat degene die de laatste zet doet *verliest* in plaats van wint. We bekijken hier de volgende variant.
Bij aanvang van het spel ligt er één stapel met $n > 2$ lucifers op tafel. Er zijn twee spelers, Bert en Giovanni, die om de beurt een zet doen. Een zet bestaat hier uit het verdelen van één van de op tafel liggende stapels lucifers in twee *ongelijke* stapels (d.w.z. twee stapels die verschillende aantallen lucifers bevatten). Het spel is afgelopen zodra de speler die aan de beurt is geen stapel meer kan vinden die in twee ongelijke stapels kan worden verdeeld. Hij heeft dan gewonnen, aangezien zijn tegenstander de laatste zet heeft gedaan. Er wordt afgesproken dat Giovanni begint. Merk op dat na elke zet het aantal stapels met één toeneemt.

Voorbeeld met $n = 17$. Stel dat Giovanni begint en de stapel verdeelt in een stapel van 11 en een stapel van 6. Bert kiest vervolgens één van de twee stapels, bijvoorbeeld die van 6. Die kan hij op twee manieren verdelen: in stapels van 5 en 1 en in stapels van 4 en 2. Stel hij kiest voor de tweede mogelijkheid. Dan is Giovanni weer aan de beurt, enzovoort. Een mogelijk spelverloop is dan: 17 $\longrightarrow$ 11,6 $\longrightarrow$ 11,4,2 $\longrightarrow$ 8,3,4,2 $\longrightarrow$ 6,2,3,4,2 $\longrightarrow$ etc.

a. Wat zijn voor dit spel toestanden en acties (voor algemene n)? Geef ook aan hoe eindstanden er uitzien.
b. (i) Hoeveel zetten kunnen er maximaal gedaan worden totdat één der spelers gewonnen heeft? Hoe moet men dan spelen (geef een spelverloop) en wat wordt in dat geval de eindstand?
(ii) Dezelfde vraag als **(i)**, maar nu voor het minimale aantal zetten.
c. Ga na of het spel voor het geval $n = 7$ winnend of verliezend is voor Giovanni. Teken daartoe de toestand-actie-ruimte voor $n = 7$, uitgaande van de beginsituatie, waarbij Giovanni begint. Geef bij *elke* toestand aan of deze winnend is voor Giovanni of voor Bert (bij perfect spel van beide spelers) Geef ook aan hoe Giovanni danwel Bert moet spelen om te winnen. Licht je antwoord toe.
d. Voor $n = 8$ is het spel winnend voor Giovanni, de beginnende speler. Beredeneer, zonder de hele toestand-actie-ruimte te tekenen, hoeveel (en welke) winnende beginzetten er zijn voor Giovanni. Je mag hierbij resultaten uit de toestand-actie-ruimte van $n = 7$ gebruiken, maar geef wel duidelijk uitleg. Je kunt bijvoorbeeld gebruiken dat bepaalde standen speltechnisch hetzelfde zijn. Leg dan wel uit waarom.

Opgave 2. Op zeker moment heeft taxibedrijf FAST te Leiden n taxi's beschikbaar, terwijl er ook precies n klanten (elk op een andere locatie) staan te wachten om afgehaald te worden. Het taxibedrijf weet van elke taxi hoe lang deze erover doet om een klant te bereiken. Dit is opgeslagen in een tweedimensionaal array `tijd`, waarin `tijd`$[i][j]$ het aantal minuten is dat taxi i moet rijden om bij klant j te komen. Elke taxi moet precies één klant ophalen. Om zijn naam eer aan te doen wil het taxibedrijf natuurlijk de totale tijd die het kost om alle klanten af te halen (= de totale wachttijd van alle klanten samen) minimaliseren. Er wordt dus een toewijzing gezocht van taxi's aan klanten, met minimale totale tijd.

Voorbeeld met $n = 4$

	klant 1	klant 2	klant 3	klant 4
taxi A	13	35	7	6
taxi B	8	17	10	9
taxi C	14	18	12	6
taxi D	15	23	9	12

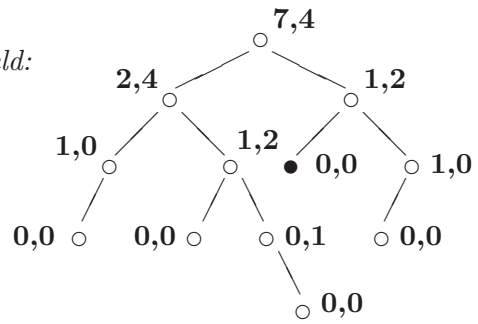
De totale tijd benodigd om alle klanten te bereiken in de toewijzing A3, B1, C2, D4 is 45 minuten. Dit is niet minimaal.

a. Beschrijf in woorden een *exhaustive search* algoritme voor dit toewijzingsprobleem (voor algemene n).
b. Beschrijf in woorden een *backtracking* algoritme voor bovenstaand probleem met algemene n en licht het toe aan de hand van het voorbeeld. Teken hiertoe ook een gedeelte (ongeveer een kwart) van de bijbehorende state-space-tree, en geef daarin de volgorde aan waarin de knopen (=deeloplossingen) bekeken worden.
c. Zowel bij backtracking als bij branch and bound worden oplossingen stap voor stap opgebouwd. 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 and bound.
d. Beschrijf in woorden een best-fit-first branch and bound algoritme dat bovenstaand 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 totale aantal minuten gebruik je voor deeloplossingen?

Opgave 3. Gegeven een binaire boom met ingang `wortel`. Een knoop van de boom ziet er uit als hieronder links is aangegeven. De `aantallinks`-velden en de `aantalrechts`-velden zijn bij aanvang van deze opgave nog niet geïnitialiseerd.

```
struct knoop {
   knoop* links;
   knoop* rechts;
   int aantallinks;
   int aantalrechts;
}; // knoop
```

Voorbeeld:



a. Schrijf een *recursieve* C++-functie `void initialiseer(knoop* wortel)`, die alle `aantallinks`-velden en `aantalrechts`-velden op `0` zet.
b. Schrijf een *recursieve* C++-functie `void nakomelingen(knoop* wortel)`, die in elke knoop het `aantallinks`-veld vult met het aantal knopen in de linkersubboom van die knoop, en het `aantalrechts`-veld met het aantal knopen in diens rechtersubboom.
In bovenstaand voorbeeld wordt bij elke knoop de inhoud van het `aantallinks`-veld en het `aantalrechts`-veld gegeven.

We willen nu een gegeven nieuwe knoop op een speciale manier aan de boom toevoegen. Zoals gebruikelijk voegen we de nieuwe knoop ergens onderaan als blad toe. We zoeken de juiste plek door, beginnend in de wortel, naar de aantallen knopen in de linker- resp.

rechtersubboom te kijken. We lopen naar links als de linkersubboom minder (of evenveel) knopen bevat dan de rechtersubboom, en anders naar rechts. Zo gaan we door totdat we de knoop kunnen toevoegen. Voor de voorbeeldboom betekent dit dat een nieuwe knoop zal worden toegevoegd links onder de zwarte knoop.

c. Schrijf een *niet-recursieve* C++-functie `toevoegen(knoop* wortel, knoop* nieuw)`, die de nieuwe knoop in de boom opbergt volgens bovenstaande methode. 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 `aantallinks`-velden en `aantalrechts`-velden van de onderweg tegengekomen knopen worden aangepast. We nemen aan dat de boom niet leeg is.

Opgave 4. Gegeven een array A ($A[0], \ldots, A[n-1]$, met $n \geq 1$), dat n *verschillende* gehele getallen bevat. Verder is gegeven dat er een index p met $0 \leq p < n$ bestaat zodat A stijgend is tot index p, en daarna dalend. Deze p is dus de index van het maximum.
Voorbeeld: als $A = 3, 6, 9, 11, 8, 2$, dan is $p = 3$. *Randgevallen:* als $A = 7, 5, 3, 2, 1$, dan is $p = 0$; als $A = 5$ (dus bestaat uit 1 element), dan $p = 0$; als $A = 4, 9$, dan $p = 1$.

a. Geef een decrease-by-one algoritme in C++ voor het vinden van de index p. Hiervoor dient een recursieve functie `int bergtop`(i, A) geschreven te worden die de index van het maximum oplevert op het deelarray $A[0], \ldots, A[i]$. Je moet gebruikmaken van wat je weet over A (eerst stijgend, dan dalend) en stoppen zodra je zeker weet dat de index p gevonden is.
b. Geef een decrease-by-half algoritme (in C++) voor het bepalen van de index p en leg uit waarom je methode werkt, dus waarom je steeds maar één helft hoeft te bekijken. Je mag hier aannemen dat n een 2-macht is. Er moet dus een recursieve C++-functie `int berg`(l, r, A) worden geschreven die het probleem oplost voor het deelarray $A[l], \ldots, A[r]$ ter lengte een 2-macht.

Veel succes !

Puntenverdeling: 1: 25; 2: 30; 3: 25; 4: 20