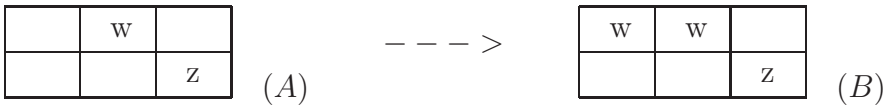
Tentamen Algoritmiek
Dinsdag 3 juni 2008, 10.00 – 13.00 uur

Geef een duidelijke toelichting bij al je antwoorden.

Opgave 1. We bekijken een tweepersoonsspelletje dat gespeeld wordt op een m bij n bord. Bij aanvang van het spel is het bord leeg. Er zijn twee spelers, Suske en Wiske, die om de beurt een zet doen. Suske speelt met zwarte (z) stenen en Wiske met witte (w) stenen. Ze spreken af dat Wiske begint. Een zet is hier het plaatsen door Suske van een zwarte resp. door Wiske van een witte steen op een leeg vakje direct naast (onder, boven, links of rechts (dus niet schuin!)) een steen van de eigen soort. Wanneer er nog geen eigen steen ligt mag de steen op elk willekeurig vakje gelegd worden.
Er wordt doorgespeeld totdat het bord vol is (in dat geval is het remise) of totdat één der spelers niet meer kan zetten (deze heeft dan verloren).

Voorbeeld met $m = 2$ en $n = 3$, waarbij Wiske een zet doet:

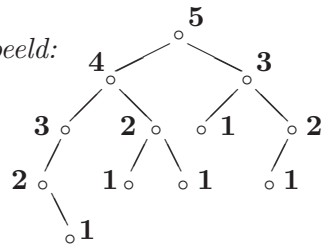


a. Wat zijn voor dit spel toestanden en acties (voor algemene m en n)?
b. Teken de toestand-actie-ruimte voor het geval $m = 2$ en $n = 3$, uitgaande van situatie (A), dus met Wiske aan zet. Geef bij elke toestand aan of deze winnend is voor Suske of voor Wiske, of tot remise leidt (bij perfect spel van beide spelers: d.w.z. beide doen steeds de/een zet die naar het beste resultaat leidt).
c. Is het spel met $m = 2$ en $n = 3$, beginnend in (A), winnend of verliezend voor Wiske, of wordt het remise? Licht je antwoord toe. Indien het winnend is voor een der spelers, geef dan ook aan hoe hij/zij moet spelen om te winnen.
d. Beredeneer, zonder de volledige toestand-actie-ruimte te tekenen, dat als Wiske in de beginsituatie (een leeg bord; $m = 2$ en $n = 3$) haar eerste steen in het middelste vakje van de bovenste rij legt, Suske altijd remise kan halen, hoe goed Wiske ook speelt. Geef daarbij duidelijk aan hoe hij dan moet spelen.

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 `hoogte`-velden nog geen waarde.

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

Voorbeeld:



a. Schrijf een *recursieve* C++-functie `void vulhoogte(knoop* wortel)`, die de `hoogte`-velden van alle knopen vult met de hoogte van de subboom met die knoop als wortel. Neem aan dat een functie `max(x,y)` die het maximum van `x` en `y` bepaalt, beschikbaar is. Voorbeeld: in de boom hier rechts boven is bij elke knoop de inhoud van het `hoogte`-veld dikgedrukt aangegeven.

We noemen een binaire boom k-gebalanceerd als in elke knoop geldt dat de hoogte van de linkersubboom en de hoogte van de rechtersubboom hooguit k verschillen. De voorbeeldboom is 2-gebalanceerd, maar niet 1-gebalanceerd. De lege boom is per definitie k-gebalanceerd.

b. Neem aan dat de `hoogte`-velden gevuld zijn zoals in **a** bedoeld. Schrijf nu een *recursieve* C++-functie `bool gebalanceerd(knoop* wortel, int k)`, die true oplevert als de binaire boom met ingang `wortel` k-gebalanceerd is, en anders false. Je mag aannemen dat een functie `abs(x)` die de absolute waarde van een gegeven getal `x` berekent, beschikbaar is. Merk op dat een lege (sub)boom hoogte 0 heeft.

Opgave 3. In de stad Heelverweg moeten n verschillende nieuwe gebouwen worden gebouwd. Daarvoor zijn n locaties beschikbaar. De bouwkosten zijn afhankelijk van het soort gebouw en de locatie. Deze kosten zijn bekend en zijn opgeslagen in een tweedimensionaal array c, waarin $c[i][j]$ de kosten voor het bouwen van gebouw i op locatie j aangeeft (in miljoenen euro's). Het doel is om de totale bouwkosten te minimaliseren.

Voorbeeld met $n = 4$

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

De gebouwen zijn met a, b, c en d aangegeven, de locaties met $1, 2, 3$ en 4.

De totale kosten van de bebouwing a2, b3, c1, d4 zijn 64. Dit is niet minimaal.

a. Beschrijf een *backtracking* algoritme voor het probleem met algemene n en licht dit toe aan de hand van het voorbeeld. Teken ook een gedeelte (ongeveer een kwart) van de bijbehorende state space tree, en geef daarin de volgorde aan waarin de knopen bekeken worden.
b. Beschrijf een best-fit-first *branch and bound* algoritme dat bovenstaand probleem oplost. Doe dit aan 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 de te verwachten totale bouwkosten gebruik je voor deeloplossingen?
c. Stel dat je van tevoren al een oplossing met bijbehorende waarde kent. Bijvoorbeeld in ons geval de bebouwing a2, b3, c1, d4. Hoe kun je dat gebruiken om je branch and bound algoritme nog iets efficiënter te maken? Leg dit uit aan de hand van het voorbeeld.

Opgave 4. Langs een lang stuk snelweg zijn $n \geq 1$ posities, die steeds 500 meter van elkaar liggen, aangewezen als plekken waar reclameborden mogen worden neergezet. De (verwachte) opbrengst als gevolg van zo'n reclamebord hangt af van de positie waar het staat en is gegeven middels een 1-dimensionaal array: opbrengst$[i]$ $(> 0;\ i = 1, \ldots, n) =$ opbrengst van een reclamebord op plek i. Reclameborden moeten altijd meer dan $(>)$ 1 km van elkaar staan. De bedoeling is om reclameborden zo te positioneren dat de totaalopbrengst maximaal is.

a. Wat is de maximale totaalopbrengst als $n = 1$? En als $n = 2$? En als $n = 3$?
b. Geef een recursieve formulering voor `maxtotaal`(n), de maximale totaalopbrengst. De basisgevallen $(n = 1, 2, 3)$ zijn in **a** berekend.
c. Het probleem kan recursief worden opgelost, maar dat is in dit geval niet efficiënt. Leg uit waarom niet, en geef aan hoe dynamisch programmeren de efficiëntie aanzienlijk

kan vergroten. Bespreek vervolgens (kort) zowel de top down methode als de bottom up methode voor dynamisch programmeren en leg het verschil uit. (Als je **b** niet kon oplossen mag je deze vraag beantwoorden aan de hand van de Fibonaccigetallen.)
d. Geef in pseudocode of in C++ een bottom up dynamisch programmeren algoritme dat de maximale totaalopbrengst berekent. Formuleer daarbij duidelijk wat voor array je gebruikt en geef de recurrente betrekking volgens welke het array gevuld wordt.

Opgave 5. Gegeven een array A ($A[1], \ldots, A[n]$, met $n \geq 2$), dat evenveel oneven als even getallen bevat. De oneven getallen staan op de oneven posities en de even getallen op de even posities. Het array moet —via verwisselingen— zo gereorganiseerd worden dat alle oneven getallen vooraan komen te staan, en alle even getallen achteraan.

a. Geef een eenvoudig iteratief algoritme voor dit probleem dat slechts één for-loop gebruikt. Hoeveel verwisselingen doet je algoritme?

We gaan nu verdeel en heers gebruiken om het array te reorganiseren.
b. Geef een decrease by four algoritme (in pseudocode of C++) voor dit probleem. Neem hierbij aan dat n een 2-voud is. Er moet dus een recursieve functie `hussel(i,j)` worden geschreven die het probleem oplost voor het deelarray $A[i], \ldots, A[j]$ ter lengte een 2-voud.
c. Geef een divide-and-conquer algoritme (in pseudocode of C++) voor dit probleem. Verdeel hierbij het array in twee gelijke delen. Neem aan dat n een 2-macht is. Hier moet dus een recursieve functie `schuffel(i,j)` worden geschreven die het probleem oplost voor het deelarray $A[i], \ldots, A[j]$ ter lengte een 2-macht.

Veel succes!

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