(10U -1-

ÕÍÍ¸
³ Tentamen Algoritmiek, 5 juni 2001, 10.00-13.00 uur ³
ÔÍÍ¾

OPMERKING. Geef een duidelijke toelichting bij al je antwoorden!!!

1. We bekijken het volgende eenpersoonsspel, dat gespeeld wordt met n witte en n zwarte schijven. In de beginsituatie liggen de schijven in een rij beginnend (van links naar rechts) met een witte schijf, dan een zwarte, dan weer een witte, dan weer een zwarte, etcetera. Aan het eind van de rij laten we twee plekken open.

   Beginsituatie voor n=4: W Z W Z W Z W Z - -

   De bedoeling van het spel is om via opeenvolgende duo-verplaatsingen zo snel mogelijk de situatie te bereiken waarin van links naar rechts eerst alle n witte schijven achter elkaar liggen, dan de twee open posities en tenslotte alle n zwarte schijven.

   Eindsituatie voor n=4: W W W W - - Z Z Z Z

   Onder een duo-verplaatsing (hieronder ook wel zet genoemd) wordt verstaan het verplaatsen van twee naast elkaar liggende schijven naar de twee open plekken, waarbij de onderlinge volgorde van de verplaatste schijven behouden blijft.

   Als voorbeeld een opeenvolging van drie duo-verplaatsingen uitgaande van de beginsituatie voor n=4:

   W Z W Z W Z W Z - - ÄÄÄÄÄ> W Z W Z - - W Z W Z ÄÄÄÄÄ>

   W - - Z Z W W Z W Z ÄÄÄÄÄ> W W W Z Z - - Z W Z

(a) Wat zijn voor dit spel toestanden en acties (voor algemene n) ?
(b) Geef een redelijke bovengrens voor het aantal mogelijke toestanden zoals in (a) gedefinieerd (3**(2n+2) is niet voldoende) en leg uit hoe je aan dat aantal komt.
(c) (i) Beschouw het geval n=3. De begintoestand, W Z W Z W Z - -, heeft vijf directe vervolgstanden, waarvan we er ,,n zullen uitwerken, namelijk W Z - - W Z W Z.
Teken dat deel van de toestand-actie-graaf voor n=3 dat alle toestanden (en verbindende acties) bevat die in hooguit twee zetten uit W Z - - W Z W Z verkregen kunnen worden. De begintoestand hoeft daarbij niet nogmaals bekeken te worden als vervolgstand.
(ii) Geef een opeenvolging van zetten die leidt van begintoestand naar eindtoestand voor n=3.

   In de onderdelen (d) en (e) bekijken we het spel weer voor algemene n.

(d) Geef een geschikte C++-datastructuur (klasse) toestand die alle informatie van een spelsituatie kan bevatten (voor algemene (const) n).
(e) Schrijf nu een C++-functie void vervolg(toestand stand) die de directe vervolgtoestanden van een gegeven spelsituatie stand afdrukt (weer voor algemene waarde van n). Directe vervolgstanden zijn die standen die via ,,n zet vanuit de gegeven situatie bereikt kunnen worden.
Neem aan dat een member-functie stand.drukaf(), die de situatie stand op het beeldscherm afdrukt, beschikbaar is.

. -2-

2. Veronderstel dat je met de auto via een gegeven route van plaats 0 naar plaats n wilt rijden. Onderweg kom je om de zoveel kilometer een benzinestation tegen, en wel op de plaatsen genummerd 0, 1, 2, ..., n-1, n. Bij benzinestation j (= het benzinestation op plaats j) kun je je tank geheel vol laten gooien tegen een vast bedrag benzine[j]. Dit bedrag is onafhankelijk van de hoeveelheid benzine die nog in de tank aanwezig is, maar de bedragen verschillen per benzinestation.

Gegeven is dat een volle tank altijd precies genoeg is om twee stations verder te rijden. Dus als je bij benzinestation i tankt, dan kun je daarna doorrijden naar plaats i+1 of plaats i+2, maar niet verder. Je start met een lege tank, dus in de beginplaats 0 moet je tanken om verder te komen. Het doel is uiteraard om zo goedkoop mogelijk van 0 in n te komen. De reis bestaat dan uit een opsomming van de achtereenvolgende stopplaatsen. In deze opgave bekijken we een paar manieren waarop we dit probleem op kunnen lossen.

(a) Geef in woorden een algoritme om de goedkoopste reis te construeren, dat gebruik maakt van de methode backtracking.

Een andere oplossingsmethode is om top-down met gebruik van recursie tot een oplossing te komen. Voor het gemak nemen we aan dat we alleen op zoek zijn naar de minimale kosten van een reis, en niet naar die reis zelf.

(b) Laat geld(i) het bedrag aangeven dat je aan benzine uitgeeft om zo goedkoop mogelijk vanuit plaats 0 in plaats i te komen. Dan geeft geld(n) de gevraagde minimale kosten.
Leg uit dat geld(i) aan de volgende recurrente betrekking voldoet:

```
geld(0) = 0
geld(1) = benzine[0]
geld(i) = minimum ( benzine[i-1] + geld(i-1),
                    benzine[i-2] + geld(i-2) )     als i>1
```

(c) Geef in pseudocode of in C++ een functie die geld(n) recursief berekent. Waarom is deze methode niet efficient?

De derde oplossingsmethode maakt gebruik van dynamisch programmeren. Hierbij wordt een array gebruikt dat bottom up (van klein naar groot) wordt ingevuld. In dit geval is dat een eendimensionaal array geld, waarbij geld[i] het minimale bedrag is dat je aan benzine uitgeeft om vanuit plaats 0 in plaats i te komen. De geld[i]'s voldoen dan uiteraard aan een recurrente betrekking als boven, met geld[i] i.p.v. geld(i). De gevraagde oplossing komt uiteindelijk in geld[n] te staan.

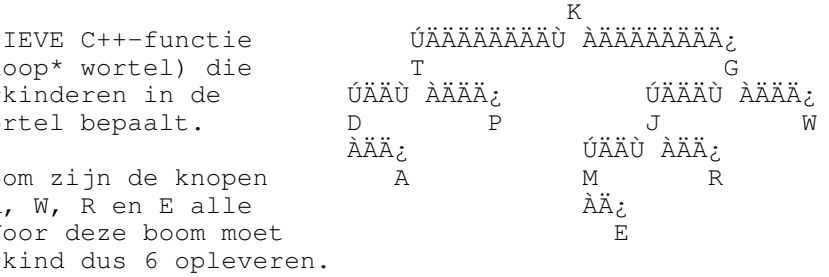
(d) Geef in pseudocode of in C++ een algoritme dat dit uitvoert. Hoeveel stappen doet dit algoritme?

3. Gegeven een binaire boom, toegankelijk via de pointer wortel van type knoop*, die karakters bevat:

```
struct knoop {
    char info;
    knoop* links;
    knoop* rechts;
};  // knoop
```

(a) Schrijf een RECURSIEVE C++-functie int rechterkind(knoop* wortel) die het aantal rechterkinderen in de boom met ingang wortel bepaalt.

In nevenstaande boom zijn de knopen met daarin G, P, A, W, R en E alle rechterkinderen. Voor deze boom moet de functie rechterkind dus 6 opleveren.



Veronderstel nu dat er twee extra velden zijn toegevoegd in elke knoop, te weten kindrechts en kindlinks, die het aantal rechterkinderen resp. het aantal linkerkinderen in de subboom met die knoop als wortel gaan bevatten. Een knoop ziet er nu dus zo uit:

```
struct knoop {
    char info;
```

```
        knoop* links;
        knoop* rechts;
        int kindrechts;
        int kindllinks;
    };  // knoop
```

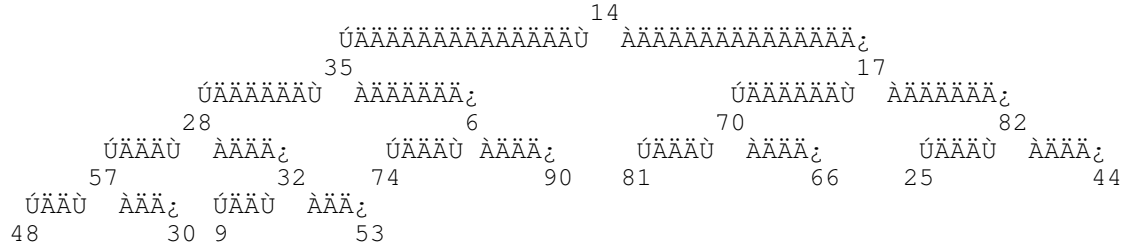
(b) Schrijf een RECURSIEVE C++-functie void kinderen(knoop* wortel), die in elke knoop de twee nieuwe velden met de juiste waarden vult. Neem aan dat deze velden al op nul geinitialiseerd zijn. De functie uit (a) mag niet worden aangeroepen.

(c) Veronderstel nu dat de boom een binaire zoekboom is en dat de velden kindrechts en kindlinks met de juiste waarden gevuld zijn.
Waar in de boom en hoe veranderen de kindrechts- en kindlinks-velden als een nieuwe letter aan de boom wordt toegevoegd?
Leg duidelijk uit hoe je efficient (worst case O(h), met h de hoogte van de boom) een karakter letter kunt toevoegen, terwijl tevens de kindrechts- en kindlinks-velden (indien noodzakelijk) worden aanpast.
Van tevoren weet je niet of letter al in de boom zit of niet.

4. Deze opgave gaat over binaire bomen. De onderdelen (a) t/m (d) staan echter los van elkaar.

(a) (i) Hoeveel knopen kan een complete binaire boom met hoogte 5 maximaal hebben? En hoeveel minimaal?
Geef bij beide gevallen een voorbeeldboom.
(ii) Hoeveel knopen kan een volle binaire boom met hoogte 5 maximaal hebben? En hoeveel minimaal?
Geef bij beide gevallen een voorbeeldboom.

(b) Geef de inhoud van de knopen van onderstaande binaire boom in postorde (LRW) volgorde.



(c) Bepaal de interne padlengte van bovenstaande binaire boom.

(d) Breng bovenstaande complete binaire boom uit in hoopstructuur door toepassing van de methode heapify. Laat duidelijk de achtereenvolgende stappen zien.

VEEL SUCCES !

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

(8U