ÕÍÍ¸
³ Tentamen Algoritmiek, 4 juni 1999, 9.00-12.00 uur ³
ÔÍÍ¾

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

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

1. Gegeven een geheel getal n (n>0, even). Verder hebben we een rij met n+2 stoelen. De eerste twee stoelen zijn leeg, op de volgende n stuks zitten om en om n/2 meisjes en n/2 jongens.
Onder een duo-verplaatsing zullen we hier verstaan het verplaatsen van twee naast elkaar gezeten kinderen naar de twee open stoelen, waarbij hun onderlinge volgorde wordt verwisseld.
De bedoeling is om door een opeenvolging van duo-verplaatsingen in een eindsituatie te komen waarbij alle jongens bij elkaar zitten en alle meisjes, en met de twee lege stoelen aan een van de twee uiteinden van de rij.

Voorbeeld voor n=6.
Uitgaande van de beginsituatie - - j m j m j m zijn er vijf verschillende duo-verplaatsingen mogelijk, en die leveren de volgende vervolgsituaties: m j - - j m j m (verplaats de kinderen die op de derde en de vierde stoel zitten), j m j - - m j m (verplaats de kinderen die op de vierde en de vijfde stoel zitten), m j j m - - j m, j m j m j - - m en m j j m j m - - .
Een voorbeeld van een opeenvolging van duo-verplaatsingen vanuit de beginsituatie is:

```
                  (3,4)                (6,7)                (4,5)
- - j m j m j m ---> m j - - j m j m ---> m j j m j - - m --->
m j j - - j m m
```

De (i,i+1) boven de pijlen betekent: verplaats de kinderen van stoelen i en i+1.

(a) Wat zijn voor dit probleem toestanden en acties (algemene n)?
(b) Geef een redelijke bovengrens (leg uit!) voor het aantal toestanden. Het antwoord 3**(n+2) wordt hier niet goed gerekend.
(c) Vanuit de begintoestand kun je door drie duo-verplaatsingen in de situatie m j j - - j m m komen. Teken nu het gedeelte van de toestand-actie graaf voor het geval n=6 vanuit m j j - - j m m, waarbij je slechts twee acties diep kijkt. De vervolgtoestand m j j m j - - m hoef je daarbij niet verder te bekijken.
Geef ook een serie acties die leidt tot een gewenste eindsituatie.
(d) Geef een geschikte C++-datastructuur (klasse) toestand die alle informatie van een stand kan bevatten (voor algemene (const) n).
(e) Schrijf nu een C++-functie void volgende(toestand stand), die de (directe) vervolgtoestanden van een gegeven situatie stand afdrukt (voor algemene n). Neem aan dat een member-functie stand.drukaf(), die de situatie stand op het beeldscherm afdrukt, beschikbaar is.
(f) Beschrijf kort hoe je met behulp van de methode backtracking de kortste serie zetten van begintoestand naar eindtoestand kunt vinden.

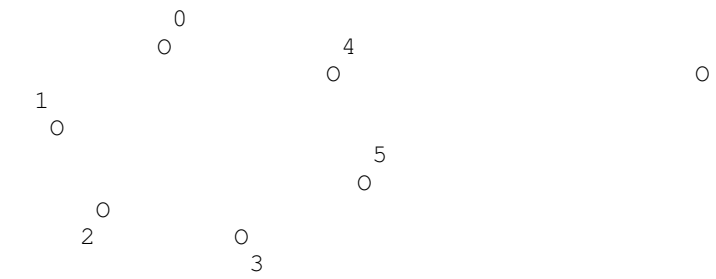
2. Gegeven een ongerichte graaf G met n knopen (genummerd 0 t/m n-1).
De graaf heeft geen lussen of meervoudige takken.
We zullen G representeren met behulp van de adjacency-list representatie, in C++ gedefinieerd als:

```
struct buur {
    int knoopnummer;
    buur* volgende;
}; // buur
```

```
typedef buur* graafje[n];  // n is een gegeven constante
```

De buurlijsten hoeven niet gesorteerd te zijn.

(a) Laat zien hoe de adjacency-list representatie er uitziet voor nevenstaande graaf (teken een plaatje).



(b) Schrijf een C++-functie int buren(graafje G, int num), die het aantal buren oplevert van de knoop met knoopnummer num (0<=num<n).

Een (ongerichte) graaf heet regulier als elke knoop evenveel buren heeft, en regulier van graad aantal (>0) als elke knoop aantal buren heeft. Bovenstaande voorbeeldgraaf is regulier van graad 3.

Zij nu gegeven dat de beschouwde graaf G regulier is van graad aantal.

(c) Veronderstel dat iemand voor de grap aan onze graaf geknoeid heeft, met als gevolg dat er ,,n tak verdwenen is.
Schrijf een C++-functie herstel(graafje& G, int aantal), die eerst de twee knopen opzoekt waartussen een tak is weggehaald en vervolgens de betreffende tak weer aanbrengt.

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

```
struct knoop {
    int info;
    knoop* links;
    knoop* rechts;
    bool rechterbroer;
    knoop* extra;
};  // knoop
```

De extra-pointers en de rechterbroer-velden hebben bij aanvang van deze opgave nog geen waarde.

(a) Schrijf een RECURSIEVE C++-functie void init(knoop* wortel) die alle extra-pointers op NULL zet en alle rechterbroer-velden op false.

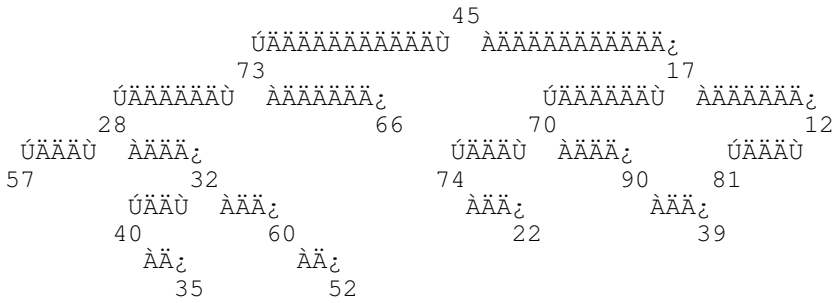
vervolg opgave 3.

(b) Laten de rechterbroer-velden en de extra-velden geinitialiseerd zijn zoals in (a) bedoeld.
Schrijf nu een RECURSIEVE C++-functie void broeders(knoop* wortel) die in elke knoop de rechterbroer-velden true maakt als de betreffende knoop een rechterbroer heeft (en anders false), en bovendien de extra-velden als volgt aanpast. Als de betreffende knoop een rechterbroer heeft moet de extra-pointer naar die rechterbroer wijzen, indien niet, dan moet de extra-pointer naar de vader wijzen (NULL als die niet bestaat).
(c) Gegeven is nu een pointer wijzer die wijst naar een knoop N in de boom. De wortel van de boom is niet bekend.
(i) Schrijf een C++-functie knoop* vindpa(knoop* wijzer) die een pointer naar de vader van N oplevert (NULL als N geen vader heeft).
(ii) Schrijf een C++-functie knoop* vindpeen(knoop* wijzer) die de wortel van de boom oplevert.

4. Deze opgave gaat over binaire bomen. De verschillende onderdelen staan

echter los van elkaar.

(a) Zoals bekend bestaat er een 1-1-correspondentie tussen geordende bossen en binaire bomen. Geef nu het geordende bos dat correspondeert met onderstaande binaire boom.



(b) Geef de inhoud van de knopen van bovenstaande binaire boom in postorde (LRW) volgorde.
(c) Geef de binaire ZOEKboom die resulteert als de volgende getallen in de aangegeven volgorde aan een aanvankelijk lege zoekboom worden toegevoegd.

45, 73, 17, 28, 66, 70, 12, 57, 32, 60, 74, 90, 81, 40, 52, 22, 39, 35

Toevoegen gaat via het gebruikelijke toevoegalgoritme voor binaire zoekbomen.
(d) Hoe ziet de COMPLETE binaire ZOEKboom eruit die precies dezelfde sleutels bevat als de boom uit (c)?
(e) Hoe hoog kan een VOLLE binaire boom met 9 knopen minimaal resp. maximaal zijn? Geef van beide een voorbeeld en bepaal tevens het aantal volle binaire bomen met minimale resp. maximale hoogte.

Veel succes!

(8U