MINISTERUL EDUCAŢIEI ŞI ŞTIINŢEI AL REPUBLICII MOLDOVA
UNIVERSITATEA TEHNICĂ A MOLDOVEI
CATEDRA "TEHNOLOGII INFORMAŢIONALE"

Discutat şi aprobat
la şedinţa Consiliului Metodic al Facultăţii
“Calculatoare, Informatică şi Microelectronică”
Decan F.C.I.M._____V.Şontea
"___"_____1999

INDICAŢII METODICE

la lucrările de laborator la disciplina "Matematica discretă în inginerie"
pentru studenţii anului 1, specialităţile "Tehnologii inofrmaţionale" şi “Calculatoare”

(Elaborat de Victor Beşliu)

Chişinău 1999

INTRODUCERE

Anul 1736 este considerat pe bună dreptate de început pentru teoria grafurilor. În acel an L.Euler a rezolvat problema despre podurile din Königsberg, stabilind criteriul de existenţă în grafuri a unui circuit special, denumit astăzi ciclu Euler. Acestui rezultat i-a fost hărăzit să fie mai bine de un secol unicul în teoria grafurilor. Doar la jumătatea secolului XIX inginerul G.Kirchof a elaborat teoria arborilor pentru cercetarea circuitelor electrice, iar matematicianul A.Caly a rezolvat problema enumerării pentru trei tipuri de arbori. În aceeaşi perioadă apare şi cunoscuta problemă despre patru culori.

Avându-şi începuturile în rezolvarea unor jocuri distractive (problema calului de şah şi reginelor, "călătoriei în jurul Pământului" despre nunţi şi haremuri, etc.), astăzi teoria grafurilor s-a transformat într-un aparat simplu şi accesibil, care permite rezolvarea unui cerc larg de probleme. Grafuri întâlnim practic peste tot. Sub formă de grafuri pot fi reprezentate sisteme de drumuri şi circuite electrice, hărţi geografice şi molecule chimice, relaţii dintre oameni şi grupuri de oameni.

Foarte fertile au fost pentru teoria grafurilor ultimile trei decenii, ceea ce a fost cauzat de creşterea spectaculoasă a domeniilor de aplicaţii. În termeni grafo-teoretici pot fi formulate o mulţime de probleme legate de obiecte discrete. Astfel de probleme apar la proiectarea circuitelor integrate şi sistemelor de comandă, la cercetarea automatelor finite, circuitelor logice, schemelor-bloc ale programelor, în economie şi statistică, chimie şi biologie, teoria orarelor şi optimizarea discretă. Teoria grafurilor a devenit o parte componentă a aparatului matematic al ciberneticii, limbajul matematicii discrete.

NOŢIUNI GENERALE

1. Definiţia grafului

Se numeşte graf, ansamblul format dintr-o mulţime finită X şi o aplicaţie F a lui X în X. Se notează $G = (X,F)$. Numărul elementelor mulţimii X determină ordinul grafului finit. Dacă $card\ X = n$, graful $G = (X,F)$ se numeşte *graf finit de ordinul n*. Elementele mulţimii X se numesc *vârfurile* grafului. Geometric, vârfurile unui graf le reprezentăm prin puncte sau cerculeţe. Perechea de vârfuri (x,y) se numeşte *arc*; vârful x se numeşte originea sau extremitatea iniţială a arcului (x,y), iar vârful y se numeşte extremitatea finală sau terminală. Un arc (x,y) îl reprezentăm geometric printr-o săgeată orientată de la vârful x la vârful y.

Dacă un vârf nu este extremitatea nici unui arc el se numeşte *vârf izolat*, iar dacă este extremitatea a mai mult de două arce - *nod*. Un arc pentru care extremitatea iniţială coincide cu cea finală se numeşte *buclă*.

Arcele unui graf le mai notăm şi cu $u_1, u_2,...$, iar mulţimea arcelor grafului o notăm cu U. Se observă că mulţimea U a tuturor arcelor unui graf determină complet aplicaţia F, precum şi reciproc, aplicaţia F determină mulţimea U a arcelor grafului. Un graf G poate fi dat fie prin ansamblul (X,F) fie prin ansamblul (X,U).

Două *arce* se zic *adiacente* dacă sunt distincte şi au o extremitate comună. Două *vârfuri* se zic *adiacente* dacă sunt distincte şi sunt unite printr-un arc.

Un arc (x,y) se spune că este *incident* cu vârful x *spre exterior* şi este *incident* cu vârful y *spre interior*.

Fie $G = (X,F)$ şi $x \in X$. Mulţimea tuturor arcelor incidente cu x spre exterior (interior) se numeşte semigradul exterior (interior) a lui x şi se notează d^+x (d_x). Dacă pentru un vârf x, $d^+x=0$ sau $d\_x=0$ atunci el se numeşte vârf terminal

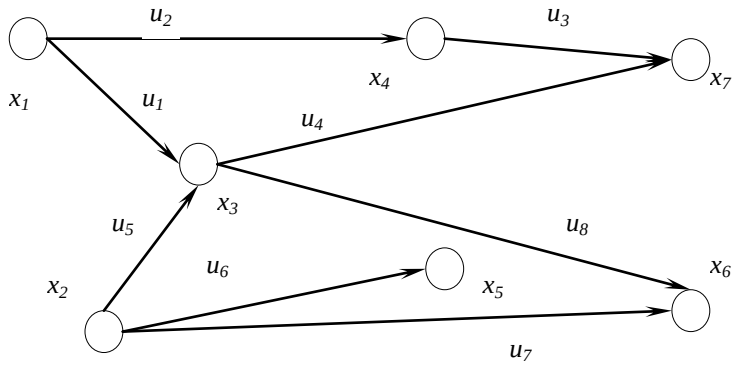

Fig. 1. Exemplu de graf

Într-un graf $G = (X,U)$ se numeşte drum un şir de arce $(u_1,...,u_k)$, astfel încât extremitatea terminală a fiecărui arc u_I coincide cu extremitatea iniţială a arcului următor u_{i+1}. Un drum care foloseşte o singură dată fiecare arc al său se numeşte drum simplu. Un drum care trece o singură dată prin fiecare vârf al său se numeşte drum elementar. Lungimea unui drum este numărul de arce din care este compus drumul.

Un drum elementar ce trece prin toate vârfurile grafului se numeşte drum hamiltonian. Un drum simplu ce conţine toate arcele grafului se numeşte drum eulerian. Un drum finit pentru care vârful iniţial coincide cu vârful terminal se numeşte circuit.

Graful obţinut din graful iniţial suprimând cel puţin un vârf al acestuia precum şi toate arcele incidente cu el se numeşte subgraf. Graful obţinut suprimând cel puţin un arc se numeşte graf parţial.

Un graf se numeşte complet dacă oricare ar fi x şi y din X există un arc de la x la y sau de la y la x.

Un graf $G = (X,F)$ se zice tare conex dacă pentru orice $x, y \in X$ (x diferit de y) există un drum de la x la y sau că oricare pereche de vârfuri x, y cu x diferit de y se află pe un circuit.

Fie $G = (X,F)$ şi $x \in X$. Mulţimea C_x formată din toate vârfurile $x_i \in X$ pentru care există un circuit ce trece prin x şi x_i se numeşte componentă tare conexă a lui G corespunzătoare vârfului x.

Componentele tari conexe ale unui graf $G = (X,F)$ constituie o partiţie a lui X.

Noţiunile introduse sunt valabile pentru grafurile orientate.

În cazul în care orientarea arcelor nu are nici o importanţă graful se va numi *neorientat*. Arcele se vor numi *muchii*, drumul - *lanţ*, circuitul - *ciclu*. La fel se introduce noţiunea de lanţ elementar şi simplu, lanţ Euler şi hamiltonian. Graful tare conex se va numi conex.

Cele două concepte de graf orientat şi graf neorientat se pot sprijini în practică unul pe altul. De la un graf orientat se poate trece la omologul său neorientat când se abordează o problemă ce nu presupune orientarea şi invers, dacă se precizează orientarea.

Unui graf orientat simetric i se poate asocia un graf neorientat, legătura dintre două vârfuri *x* şi *y* realizată de cele două arce *(x,y)* şi *(y,x)* de sensuri contrarii înlocuindu-se cu muchia *[x,y]*. De asemenea un graf neorientat poate fi identificat cu mai multe grafuri orientate înlocuind fiecare muchie cu două arce orientate în sens opus.

Fie $G = (X,U)$ un graf neorientat şi $x \in X$. Se numeşte gradul vârfului *x* numărul muchiilor care au o extremitate în vârful *x*. Se notează $g(x)$. Un vârf este izolat dacă $g(x) = 0$.

2. Număr cociclomatic şi număr ciclomatic

Fie $G = (X,F)$ un graf neorientat cu *n* vârfuri, *m* muchii şi *p* componente conexe. Numim număr cociclomatic asociat grafului *G* numărul

$$r(G) = n - p$$

iar numărul ciclomatic numărul

$$s(G) = m - r(G) = m - n + p.$$

Se numeşte multigraf un graf neorientat în care există perechi de vârfuri unite prin mai multe muchii. Se numeşte *q* - graf un multigraf pentru care numărul maxim de muchii ce unesc două vârfuri este *q*.

Teorema. Fie $G = (X,U)$ un multigraf, iar $G_1 = (X,U_1)$ un multigraf obţinut din G adăugând o muchie. Dacă $x, y \in X$ şi $[x,y]$ este muchia care se adaugă la mulţimea muchiilor grafului *G*, atunci:

(1) dacă $x = y$ sau *x* şi *y* sunt unite printr-un lanţ

$$r(G_1) = r(G),\ s(G_1) = s(G) + 1$$

(2) în caz contrar

$$r(G_1) = r(G) + 1,\ s(G_1) = s(G).$$

Demonstraţia este evidentă.

Consecinţă. Numerele cociclomatic şi ciclomatic sunt nenegative.

3. Număr cromatic. Grafuri planare. Arbori

Un graf $G = (X,U)$ se zice ca este *graf p-cromatic* daca vârfurile lui se pot colora cu p-culori distincte astfel ca două vârfuri adiacente să nu fie de aceeaşi culoare. *Cel mai mic* număr întreg şi pozitiv pentru care graful este *p-cromatic* se numeşte *număr cromatic*.

Un graf se zice că este *planar* dacă poate fi reprezentat pe un plan astfel ca două muchii să nu aibă puncte comune în afară de extremităţile lor. Un graf planar determină regiuni numite *feţe*. O faţă este o regiune mărginită de muchii şi care nu are în interior nici vârfuri, nici muchii. Conturul unei feţe este format din muchiile care o mărginesc. Două feţe sunt adiacente dacă contururile au o muchie comună. S-a demonstrat că numărul cromatic al unui graf planar este patru.

Cu privire la numărul cromatic s-a demonstrat următoarea

Teorema (König). Un graf este bicromatic dacă şi numai dacă nu are cicluri de lungime impară.

Se numeşte arbore oricare graf conex fără bucle şi cicluri.

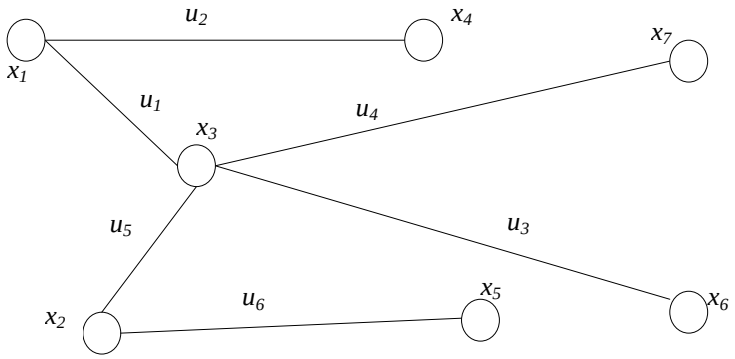

Fig. 2. Exemplu de arbore

Se numeşte arborescenţă un graf orientat fără circuite astfel ca să existe un vârf şi numai unul care nu e precedat de nici un alt vârf (rădăcina) şi orice alt vârf să fie precedat de un singur vârf.

LUCRAREA DE LABORATOR Nr. 1

TEMA: PĂSTRAREA GRAFURILOR ÎN MEMORIA CALCULATORULUI

1. SCOPUL LUCRĂRII:

- Studierea metodelor de definire a unui graf: matrice de incidenţă, matrice de adiacenţă, liste;
- Elaborarea unor proceduri de introducere, extragere şi transformare a diferitor forme de reprezentare internă a grafurilor cu scoaterea rezultatelor la display şi imprimantă.

2. NOTE DE CURS

Metode de reprezentare a grafului

Există trei metode de bază de definire a unui graf:

1. *Matricea de incidenţă;*
2. *Matricea de adiacenţă;*
3. *Lista de adiacenţă (incidenţă).*

Vom face cunoştinţă cu fiecare dintre aceste metode.

Matricea de incidenţă

Este o matrice de tipul *m*x*n*, în care *m* este numărul de muchii sau arce (pentru un graf orientat), iar *n* este numărul vârfurilor. La intersecţia liniei *i* cu coloana *j* se vor considera valori de *0* sau *1* în conformitate cu următoarea regulă:

- *1* - dacă muchia *i* este incidentă cu vârful *j* (dacă arcul *i* "intră" în vârful *j* în cazul unui graf orientat);
- *0* - dacă muchia (arcul) *i* şi vârful *j* nu sunt incidente;
- *-1* - numai pentru grafuri orientate, dacă arcul *i* "iese" din vârful *j*.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
u_1	-1	0	1	0	0	0	0
u_2	-1	0	0	1	0	0	0
u_3	0	0	0	-1	0	0	1
u_4	0	0	-1	0	0	0	1
u_5	0	-1	1	0	0	0	0
u_6	0	-1	0	0	1	0	0
u_7	0	-1	0	0	0	1	0
u_8	0	0	-1	0	0	1	0

Fig. 3. Exemplu de matrice de incidenţă (v.fig.1)

Este uşor de observat că această metodă este de o eficacitate mică în sensul utilizării memoriei calculatorului: fiecare linie conţine doar două elemente diferite de zero (o muchie poate fi incidentă cu nu mai mult de două vârfuri).

În limbajul Pascal matricea de incidenţă poate fi redată printr-un tablou bidimensional *m*x*n*:

```
Matr_Incd: array [1..LinksCount, 1..TopsCount] of integer;
```

Matricea de adiacenţă

Este o matrice pătrată *n*x*n,* aici *n* este numărul de vârfuri. Fiecare element poate fi *0*, dacă vârfurile respective nu sunt adiacente, sau 1, în caz contrar. Pentru un graf fără bucle putem observa următoarele:

- diagonala principală este formată numai din zerouri;
- pentru grafuri neorientate matricea este simetrică faţă de diagonala principală.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0	0	1	1	0	0	0
x_2	0	0	1	0	1	1	0
x_3	0	0	0	0	0	1	1
x_4	0	0	0	0	0	0	1
x_5	0	0	0	0	0	0	0
x_6	0	0	0	0	0	0	0
x_7	0	0	0	0	0	0	0

Fig. 4. Exemplu de matrice de adiacenţă (v.fig.1)

În limbajul Pascal matricea de adiacenţă poate fi reprezentată în modul următor:

```
Matr_Ad :array [1..TopsCount,1..TopsCount] of byte,
```

sau

```
Matr_Ad :array [1..TopsCount,1..TopsCount] of boolean;
```

După cum este lesne de observat şi în acest caz memoria calculatorului este utilizată nu prea eficace din care cauză matricea de adiacenţă ca şi matricea de incidenţă se vor utiliza de obicei doar în cazul în care se va rezolva o problemă concretă pentru care reprezentarea grafului în această formă aduce unele facilităţi algoritmului respectiv.

Pentru păstrarea grafurilor în memoria calculatorului (în deosebi, memoria externă) se va utiliza una din posibilităţile de mai jos.

Lista de adiacenţă şi lista de incidenţă

Lista de adiacenţă este o listă cu *n* linii (după numărul de vârfuri *n*), în linia cu numărul *i* vor fi scrise numerele vârfurilor adiacente cu vârful *i*.

Lista de incidenţă se defineşte analogic cu deosebirea că în linia *i* vor fi scrise numerele muchiilor (arcelor) incidente cu vârful *i*.

Reprezentarea grafurilor prin intermediul acestor liste permite utilizarea mai eficace a memoriei calculatorului, însă aceste forme sunt mai complicate atât în realizare, cât şi în timpul procesării. Pentru a lua în consideraţie lungimea variabilă a liniilor vor fi utilizate variabile dinamice şi pointeri.

Vom exemplifica pentru un graf cu *n* vârfuri. Deoarece fiecare element al listei conţine numere de vârfuri este evident să considerăm că vom avea un şir de variabile dinamice de tip INTEGER care se vor afla în relaţia respectivă de precedare (succedare). Această relaţie se va realiza prin pointeri, uniţi împreună cu variabila de tip întreg în înregistrarea (Pascal: record). Pentru a păstra indicatorii de intrare în aceste şiruri se va folosi un tablou unidimensional de indicatori de lungime *n*. În calitate de simbol de terminare a şirului se va utiliza un simbol care nu a fost folosit la numeraţia vârfurilor (de exemplu *0*), care va fi introdus în calitate de variabilă de tip întreg al ultimului bloc.

De exemplu, lista de adiacenţă (fig.1.1):

1 - 3, 4, 0
2 - 3, 5, 6, 0
3 - 6, 7, 0
4 – 7, 0
5 - 0
6 - 0
7 - 0

va avea următoarea reprezentare internă:

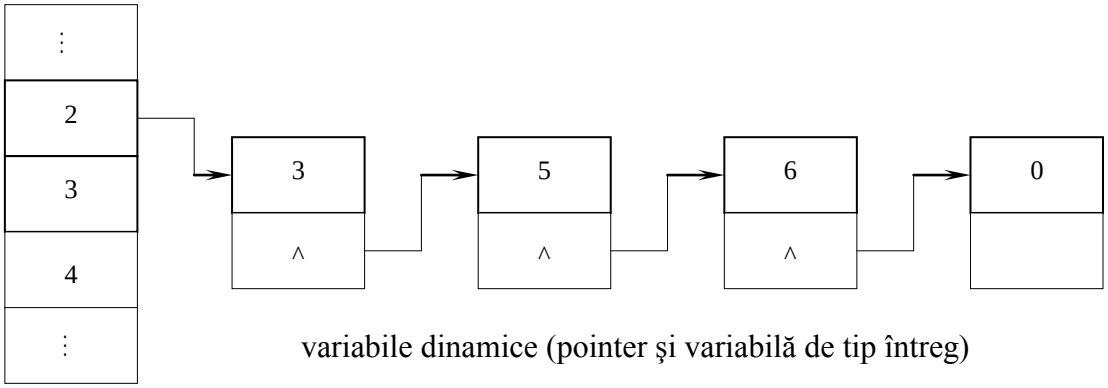

Fig. 5. Reprezentarea internă a listei de adiacenţă

Va fi necesar de realizat următoarea declaraţie:

```
Type
        BlockPtr = ^BlockType;
```

```
        BlockType = record;
        Number : integer;
        Point : BlockPtr;
        end;
Var In_Ptr :array [1..TopsCount] of BlockPtr;
```

Lista de adiacenţă (şi realizarea reprezentării interne) se va implementa cu ajutorul procedurii New (BlockPtr_Type_Variable), în care BlockPtr_Type_Variable este o variabilă de tip BlockPtr.

3. SARCINA DE BAZĂ

1. Elaboraţi procedura introducerii unui graf în memoria calculatorului în formă de matrice de incidenţă, matrice de adiacenţă şi listă de adiacenţă cu posibilităţi de analiză a corectitudinii.
2. Elaboraţi proceduri de transformare dintr-o formă de reprezentare în alta.
3. Folosind procedurile menţionate elaboraţi programul care va permite:
   - introducerea grafului reprezentat sub oricare din cele trei forme cu posibilităţi de corecţie a datelor;
   - păstrarea grafului în memoria externă în formă de listă de adiacenţă;
   - extragerea informaţiei într-una din cele trei forme la imprimantă şi display.

4. ÎNTREBĂRI DE CONTROL

1. Care sunt metodele de bază de reprezentare a unui graf?
2. Descrieţi fiecare din aceste metode.
3. Cum se vor realiza aceste metode în limbajul Pascal?

LUCRAREA DE LABORATOR Nr. 2

TEMA: ALGORITMUL DE CĂUTARE ÎN ADÂNCIME

1. SCOPUL LUCRĂRII:

- Studierea algoritmilor de căutare în graf şi a diferitor forme de păstrare şi prelucrare a datelor.
- Elaborarea procedurii de căutare în adâncime.

2. NOTE DE CURS

Structuri de date: ***liste***

Fiecare tip de listă defineşte o mulţime de şiruri finite de elemente de tipul declarat. Numărul de elemente care se numeşte lungimea listei poate varia pentru diferite liste de acelaşi tip. Lista care nu conţine nici un element se va numi vidă. Pentru listă sunt definite noţiunile începutul, sfârşitul listei şi respectiv primul şi ultimul element, de asemenea elementul curent ca şi predecesorul şi succesorul elementului curent. Element curent se numeşte acel unic element care este accesibil la momentul dat.

Structuri de date : ***fire de aşteptare***

Firele de aşteptare (*FA*, rând, coadă, şir de aşteptare) se vor folosi pentru a realiza algoritmul de prelucrare a elementelor listei în conformitate cu care elementele vor fi eliminate din listă în ordinea în care au fost incluse în ea (primul sosit - primul servit).

Operaţiile de bază cu firele de aşteptare:

- Formarea unui FA vid;
- Verificare dacă FA nu este vid;
- Alegerea primului element cu eliminarea lui din FA;
- Introducerea unei valori noi în calitate de ultim element al FA.

Structuri de date: ***stive***

Stiva se utilizează pentru a realiza algoritmul de prelucrare a elementelor după principiul "ultimul sosit - primul prelucrat" (LIFO).

Operaţiile de bază cu stivele sunt următoarele:

- Formarea unei stive vide;
- Verificare la vid;
- Alegerea elementului din topul stivei cu sau fără eliminare;
- Introducerea unui element nou în topul stivei.

Structuri de date - ***arbori***

Se va defini o mulţime de structuri fiecare din care va consta dintr-un obiect de bază numit *vârf* sau *rădăcina arborelui* dat şi o listă de elemente din mulţimea definită, care (elementele) se vor numi *subarbori* ai arborelui dat. Arborele pentru care lista subarborilor este vidă se va numi *arbore trivial*, iar rădăcina lui - *frunză*.

Rădăcina arborelui se va numi tatăl vârfurilor care servesc drept rădăcini pentru subarbori; aceste vârfuri se vor mai numi copiii rădăcinii arborelui: rădăcina primului subarbore se va numi *fiul cel mai mare,* iar rădăcina fiecărui subarbore următor în listă se va numi frate.

Operaţiile de bază pentru arbori vor fi:

- Formarea unui arbore trivial;
- Alegerea sau înlocuirea rădăcinii arborelui;
- Alegerea sau înlocuirea listei rădăcinilor subarborilor;
- Operaţiile de bază care sunt valabile pentru liste.

Căutare în adâncime

La căutarea în adâncime (parcurgerea unui graf în sens direct, în preordine) vârfurile grafului vor fi vizitate în conformitate cu următoarea procedură recursivă:

mai întâi va fi vizitată rădăcina arborelui q, apoi, dacă rădăcina arborelui nu este frunză - pentru fiecare fiu p al rădăcinii q ne vom adresa recursiv procedurii de parcurgere în adâncime pentru a vizita vârfurile tuturor subarborilor cu rădăcina p ordonate ca fii ai lui q.

În cazul utilizării unei stive pentru păstrarea drumului curent pe arbore, drum care începe din rădăcina arborelui şi se termină cu vârful vizitat în momentul dat, poate fi realizat un algoritm nerecursiv de forma:

Vizitează rădăcina arborelui şi introdu-o în stiva vidă *S*;
WHILE stiva *S* nu este vidă DO
    BEGIN
    fie *p* - vârful din topul stivei *S*;
    IF fiii vârfului *p* încă nu au fost vizitaţi
    THEN vizitează fiul mai mare al lui *p* şi introduce-l în *S*

```
            ELSE BEGIN
                elimină vârful p din stiva S
                IF p are fraţi THEN vizitează pe fratele lui p şi introduce-l în stiva S
            END
    END
```

Acest algoritm poate fi modificat pentru a putea fi utilizat la parcurgerea tuturor vârfurilor unui graf arbitrar. În algoritmul de mai jos se va presupune că este stabilită o relaţie de ordine pe mulţimea tuturor vârfurilor grafului, iar mulţimea vârfurilor adiacente cu un vârf arbitrar al grafului este de asemenea ordonată:

```
WHILE va exista cel puţin un vârf care nu a fost vizitat DO
    BEGIN
    fie p - primul din vârfurile nevizitate;
    vizitează vârful p şi introduce-l în stiva vidă S;
        WHILE stiva S nu este vidă DO
            BEGIN
            fie p - vârful din topul stivei S;
            IF m vârfuri ale lui p sunt vârfuri adiacente nevizitate
                THEN BEGIN
                fie z primul vârf nevizitat din vârfurile adiacente cu p;
                parcurge muchia (p,z), vizitează vârful z şi introduce-l în stiva S;
                END
            ELSE elimină vârful p din stiva S
            END
    END
```

În cazul în care se va lucra cu un graf conex arbitrar cu relaţia de ordine lipsă, nu va mai avea importanţă ordinea de parcurgere a vârfurilor. Propunem un algoritm care utilizează mai larg posibilităţile stivei, cea ce face programul mai efectiv în sensul diminuării timpului de calcul necesar. De exemplu, acest algoritm în varianta recursivă este pe larg utilizat în programele de selectare globală în subdirectori (cazul programelor antivirus).

```
Introdu în stivă vârful iniţial şi marchează-l;
WHILE stiva nu este vidă DO
    BEGIN
    extrage un vârf din stivă;
    IF există vârfuri nemarcate adiacente cu vârful extras
        THEN marchează-le şi introduce-le în stivă;
    END
```

3. SARCINA DE BAZĂ

1. Elaboraţi procedura căutării în adâncime într-un graf arbitrar;
2. Elaboraţi un program cu următoarele posibilităţi:
   - introducerea grafului în calculator,
   - parcurgerea grafului în adâncime,
   - vizualizarea rezultatelor la display şi imprimantă.

4. ÎNTREBĂRI DE CONTROL

1. Definiţi structurile principale de date: liste, fire de aşteptare, stive, arbori.
2. Care sunt operaţiile definite pentru aceste structuri de date?

3. Care este principiul de organizare a prelucrării elementelor în firele de aşteptare şi în stive?
4. Definiţi noţiunea de parcurgere a grafului în adâncime.
5. Ce fel de structuri de date se vor utiliza în căutarea în adâncime?
6. Exemplificaţi utilizarea algoritmului de căutare în adâncime.

LUCRAREA DE LABORATOR Nr. 3

TEMA: ALGORITMUL DE CĂUTARE ÎN LĂRGIME

1. SCOPUL LUCRĂRII:

- Studierea algoritmului de căutare în lărgime;
- Elaborarea programului de căutare în lărgime.

2. NOTE DE CURS

Algoritmul de căutare în lărgime

Parcurgerea grafului în lărgime, ca şi parcurgerea în adâncime, va garanta vizitarea fiecărui vârf al grafului exact o singură dată, însă principiul va fi altul. După vizitarea vârfului iniţial, de la care va începe căutarea în lărgime, vor fi vizitate toate vârfurile adiacente cu vârful dat, apoi toate vârfurile adiacente cu aceste ultime vârfuri ş.a.m.d. până vor fi vizitate toate vârfurile grafului. Evident, este necesar ca graful să fie conex. Această modalitate de parcurgere a grafului (în lărgime sau postordine), care mai este adesea numită parcurgere în ordine orizontală, realizează parcurgerea vârfurilor de la stânga la dreapta, nivel după nivel.

Algoritmul de mai jos realizează parcurgerea în lărgime cu ajutorul a două fire de aşteptare *O1* şi *O2*.

Se vor forma două fire de aşteptare vide *O1* şi *O2*;

Introduce rădăcina în FA *O1*;

WHILE cel puţin unul din firele de aşteptare *O1* sau *O2* nu va fi vid DO

    IF *O1* nu este vid THEN

        BEGIN
            fie *p* vârful din topul FA *O1*;
            vizitează vârful *p* eliminându-l din *O1*;
            vizitează pe toţi fiii lui *p* în FA *O2*, începând cu cel mai mare;

        END

    ELSE
    în calitate de *O1* se va lua FA *O2*, care nu este vid,
    iar în calitate de *O2* se va lua FA vid *O1*;

Vom nota că procedura parcurgerii grafului în lărgime permite să realizăm arborele de căutare şi în acelaşi timp să construim acest arbore. Cu alte cuvinte, se va rezolva problema determinării unei rezolvări sub forma vectorului (a_1, a_2,...) de lungime necunoscută, dacă este cunoscut că există o rezolvare finită a problemei.

Algoritmul pentru cazul general este analogic cu cel pentru un graf în formă de arbore cu o mică modificare care constă în aceea că fiecare vârf vizitat va fi marcat pentru a exclude ciclarea algoritmului.

Algoritmul parcurgerii grafului în lărgime:

Se vor defini două FA O_1 şi O_2 vide;

Introdu vârful iniţial în FA O_1 şi marchează-l;

```
WHILE FA O1 nu este vid DO
BEGIN
        vizitează vârful din topul FA O1 şi elimină-l din FA;
        IF există vârfuri nemarcate adiacente cu vârful dat THEN introdu-le în FA O2;
END
```

3. SARCINA DE BAZĂ

1. Elaboraţi procedura care va realiza algoritmul de parcurgere a grafului în lărgime;
2. Folosind procedurile din lucrările precedente, elaboraţi programul care va permite:
   - introducerea grafului în calculator;
   - parcurgerea grafului în lărgime;
   - extragerea datelor la display şi printer.

4. ÎNTREBĂRI DE CONTROL

1. În ce constă parcurgerea arborelui şi a grafului în lărgime?
2. Care este diferenţa dintre parcurgerea în lărgime a unui arbore şi a unui graf arbitrar?
3. Ce fel de structuri de date se vor utiliza în algoritmul de căutare în lărgime?
4. Exemplificaţi algoritmul căutării în lărgime.

LUCRAREA DE LABORATOR Nr. 4

TEMA: ALGORITMUL DETERMINĂRII GRAFULUI DE ACOPERIRE

1. SCOPUL LUCRĂRII:

- Studierea algoritmului de determinare a grafului de acoperire şi elaborarea programelor care vor realiza acest algoritm.

2. NOTE DE CURS

Noţiune de graf de acoperire

Fie H un subgraf care conţine toate vârfurile unui graf arbitrar G. Dacă pentru fiecare componentă de conexitate a lui G subgraful H va defini un arbore atunci H se va numi graf de acoperire (scheletul sau carcasă) grafului G. Este evident că graful de acoperire există pentru oricare graf: eliminând ciclurile din fiecare componentă de conexitate, adică eliminând muchiile care sunt în plus, vom ajunge la graful de acoperire.

Se numeşte graf aciclic orice graf care nu conţine cicluri. Pentru un graf arbitrar G cu n vârfuri şi m muchii sunt echivalente următoarele afirmaţii:

1. G este arbore;
2. G este un graf conex şi $m = n - 1$;
3. G este un graf aciclic şi $m = n - 1$;
4. oricare două vârfuri distincte (diferite) ale lui G sunt unite printr-un lanţ simplu care este unic;
5. G este un graf aciclic cu proprietatea că, dacă o pereche oarecare de vârfuri neadiacente vor fi unite cu o muchie, atunci graful obţinut va conţine exact un ciclu.

Consecinţă: numărul de muchii pentru un graf arbitrar G, care va fi necesar a fi eliminate spre a obţine un graf de acoperire nu depinde de ordinea eliminării lor şi este egal cu

$$m(G)-n(G)+k(G),$$

unde *m(G), n(G)* şi *k(G)* sunt numărul de muchii, vârfuri şi componente conexe, respectiv.

Numărul $s(G) = m(G)-n(G)+ k(G)$ se numeşte *rang ciclic* sau număr *ciclomatic* al grafului *G*. Numărul $r(G) = n(G)-k(G)$ – *rang cociclomatic* sau *număr cociclomatic.*

Deci, $s(G)+r(G)=m(G)$.

Este adevărată următoarea afirmaţie: orice subgraf a unui graf arbitrar *G* se conţine într-un graf de acoperire a grafului *G*.

Algoritmul de determinare a grafului de acoperire

Există mai mulţi algoritmi de determinare a grafului de acoperire. Algoritmul de mai jos nu este un algoritm-standard, ci este unul elaborat în bază algoritmului de căutare în lărgime. Esenţa algoritmului constă în aceea că folosind două fire de aşteptare în unul din care sunt înscrise (pe rând) numerele vârfurilor adiacente cu vârfurile din celălalt FA (ca şi în cazul căutării în lărgime), vor fi eliminate muchiile dintre vârfurile unui FA şi toate muchiile în afară de una dintre fiecare vârf al FA curent şi vârfurile din FA precedent. în cazul În care ambele FA vor deveni vide procedura se va termina.

Pentru a nu admite ciclarea şi ca să fim siguri că au fost prelucrate toate componentele conexe se va utiliza marcarea vârfurilor. Dacă după terminarea unui ciclu ordinar nu au mai rămas vârfuri nemarcate procedura ia sfârşit, în caz contrar în calitate de vârf iniţial se va lua oricare din vârfurile nemarcate.

Descrierea algoritmului:

1. Se vor declara două FA (FA_1 şi FA_2) vide.
2. Se va lua în calitate de vârf iniţial un vârf arbitrar al grafului.
3. Se va introduce vârful iniţial în firul de aşteptare vid FA_1 şi se va marca acest vârf.
4. Se vor introduce în FA_2 toate vârfurile adiacente cu vârfurile din FA_1 şi se vor marca. Dacă FA_2 este vid se va trece la p.7, în caz contrar - la p. 4.
5. Se vor elimina toate muchiile care leagă vârfurile din FA_2.
6. Pentru toate vârfurile din FA_2 vor fi eliminate toate muchiile în afară de una care leagă vârful dat cu vârfurile din FA_1.
7. Se vor schimba cu numele FA_1 şi FA_2 (FA_1 va deveni FA_2 şi invers).
8. Dacă există cel puţin un vârf nemarcat se va lua în calitate de vârf iniţial oricare din acestea şi se va trece la p.1, altfel
9. STOP.

Graful obţinut este graful de acoperire.

3. SARCINA DE BAZĂ

1. Elaboraţi organigrama algoritmului şi programul procedurii de determinare a grafului de acoperire cu posibilităţi de pornire a procedurii din oricare vârf al grafului.
2. Utilizând procedurile de introducere a grafului în memoria CE din lucrarea Nr. 1, elaboraţi un program cu următoarele facilităţi:
   - introducerea grafului care este dat sub formă de matrice de incidenţă, adiacenţă sau listă de adiacenţă;
   - determinarea grafului de acoperire, pornind de la un vârf arbitrar;

- extragerea informaţiei la display sau imprimantă în oricare din formele numite.

4. ÎNTREBĂRI DE CONTROL

1. Ce este un graf aciclic şi prin ce se deosebeşte el de un arbore?
2. Definiţi noţiunile de arbore şi graf de acoperire.
3. Care vor fi transformările ce vor fi efectuate într-un graf arbitrar pentru a obţine graful de acoperire?
4. Care este esenţa algoritmului de determinare a grafului de acoperire?
5. Evidenţiaţi etapele de bază ale algoritmului de determinare a grafului de acoperire.

LUCRAREA DE LABORATOR Nr. 5

TEMA: ALGORITMI DE DETERMINARE A DRUMULUI MINIM

1. SCOPUL LUCRĂRII:

- Studierea algoritmilor de determinare a drumurilor minime într-un graf.
- Elaborarea programelor de determinare a drumului minim într-un graf ponderat.

2. NOTE DE CURS

Noţiune de drum minim

Pentru un graf orientat $G = (X,U)$ se va numi *drum* un şir de vârfuri $D = (x_0, x_1,..., x_r)$ cu proprietatea că (x_0, x_1), (x_1, x_2),..., (x_{r-1}, x_r) aparţin lui U, deci sunt arce ale grafului şi extremitatea finală a arcului precedent coincide cu extremitatea iniţială a arcului următor.

Vârfurile x_0 şi x_r se numesc extremităţile drumului D. Lungimea unui drum este dată de numărul de arce pe care le conţine. Dacă vârfurile $x_0, x_1,..., x_r$ sunt distincte două câte două drumul D este elementar.

Adeseori, fiecărui arc (muchii) i se pune în corespondenţă un număr real care se numeşte *ponderea* (lungimea) arcului. Lungimea arcului (x_i, x_j) se va nota $w(i,j)$, iar în cazul în care un arc este lipsă ponderea lui va fi considerată foarte mare (pentru calculator cel mai mare număr pozitiv posibil). În cazul grafurilor cu arce ponderate (grafuri ponderate) se va considera lungime a unui drum suma ponderilor arcelor care formează acest drum. Drumul care uneşte două vârfuri concrete şi are lungimea cea mai mică se va numi *drum minim* iar lungimea drumului minim vom numi *distanţă*. Vom nota distanţa dintre x şi t prin $d(x, t)$, evident, $d(x,x)=0$.

Algoritmul lui Ford pentru detrminarea drumului minim

Permite determinarea drumului minim care începe cu un vârf iniţial x_i până la oricare vârf al grafului G. Dacă prin L_{ij} se va nota ponderea arcului (x_i, x_j) atunci algoritmul conţine următorii paşi:

1. Fiecărui vârf x_j al grafului G se va ataşa un număr foarte mare $H_j(\infty)$. Vârfului iniţial i se va ataşa $H_o = 0$;
2. Se vor calcula diferenţele $H_j - H_i$ pentru fiecare arc (x_i, x_j). Sunt posibile trei cazuri:
   a) $H_j - H_i < L_{ij}$,
   b) $H_j - H_i = L_{ij}$,
   c) $H_j - H_i > L_{ij}$.

Cazul "*c*" permite micşorarea distanţei dintre vârful iniţial şi x_j din care cauză se va realiza $H_j = H_i + L_{ij}$.

Pasul 2 se va repeta atâta timp cât vor mai exista arce pentru care are loc inegalitatea "c". La terminare, etichetele H_i vor defini distanţa de la vârful iniţial până la vârful dat x_i.

3. Acest pas presupune stabilirea secvenţei de vârfuri care va forma drumul minim. Se va pleca de la vârful final x_j spre cel iniţial. Predecesorul lui x_j va fi considerat vârful x_i pentru care va avea loc $H_j - H_i = L_{ij}$. Dacă vor exista câteva arce pentru care are loc această relaţie se va alege la opţiune.

Algoritmul Bellman - Calaba

Permite determinarea drumului minim dintre oricare vârf al grafului până la un vârf, numit vârf final.

Etapa iniţială presupune ataşarea grafului dat *G* a unei matrice ponderate de adiacenţă, care se va forma în conformitate cu următoarele:

1. $M(i,j) = L_{ij}$, dacă există arcul (x_i, x_j) de pondere L_{ij};
2. $M(i,j) = \infty$, unde ∞ este un număr foarte mare (de tip întreg maximal pentru calculatorul dat), dacă arcul (x_i, x_j) este lipsă;
3. $M(i,j) = 0$, dacă $i = j$.

La etapa a doua se va elabora un vector V_0 în felul următor:

1. $V_{0(i)} = L_{in}$, dacă există arcul (x_i, x_n), unde x_n este vârful final pentru care se caută drumul minim, L_{in} este ponderea acestui arc;
2. $V_{0(i)} = \infty$, dacă arcul (x_i, x_n) este lipsă;
3. $V_{0(i)} = 0$, dacă $i = j$.

Algoritmul constă în calcularea iterativă a vectorului *V* în conformitate cu următorul procedeu:

1. $V_{k(i)} = min\{V_{k-1}; L_{ij}+V_{k-1(j)}\}$, unde $i = 1, 2, \ldots, n - 1, j = 1, 2, \ldots, n;\ i<>j;$
2. $V_{k(n)} = 0$.

Când se va ajunge la $V_k = V_{k-1}$ - STOP.

Componenta cu numărul *i* a vectorului V_k cu valoarea diferită de zero ne va da valoarea minimă a drumului care leagă vârful *i* cu vârful *n*.

3. SARCINA DE BAZĂ

1. Elaboraţi procedura introducerii unui graf ponderat;
2. Elaboraţi procedurile determinării drumului minim;
3. Realizaţi un program cu următoarele funcţii:
   - introducerea grafului ponderat cu posibilităţi de analiză sintactică şi semantică şi de corectare a informaţiei;
   - determinarea drumului minim;
   - extragerea informaţiei la display şi printer (valoarea drumului minim şi succesiunea vârfurilor care formează acest drum).

4. ÎNTREBĂRI DE CONTROL

1. Ce se numeşte graf ponderat?

2. Definiţi noţiunea de distanţă.
3. Descrieţi etapele principale ale algoritmului Ford.
4. Care sunt momentele principale în algoritmul Bellman-Calaba?
5. Prin ce se deosebeşte algoritmul Ford de algoritmul Bellman-Calaba?
6. Cum se va stabili succesiunea vârfurilor care formează drumul minim?

LUCRAREA DE LABORATOR Nr. 6

TEMA: DETERMINAREA FLUXULUI MAXIM ÎNTR-O REŢEA DE TRANSPORT

1. SCOPUL LUCRĂRII:

- Studierea noţiunilor de bază leagate de reţelele de transport;
- Programarea algoritmului Ford-Fulkerson pentru determinarea fluxului maxim într-o reţea de transport.

2. NOTE DE CURS

Reţele de transport

Un graf orientat $G = (X, U)$ se numeşte *reţea de transport* dacă satisface următoarele condiţii:

a) există un vârf unic a din X în care nu intră nici un arc sau $d\_(a)=0$;
b) există un vârf unic b din X din care nu iese nici un arc sau $d^+(a)=0$;
c) G este conex şi există drumuri de la a la b în G;
d) s-a definit o funcţie c: $U \to R$ astfel încât $c(u) \geq 0$ pentru orice arc u din U.

Vârful a se numeşte intrarea reţelei, vârful b se numeşte ieşirea reţelei, iar $c(u)$ este capacitatea arcului u.

O funcţie $f: U \to R$ astfel încât $f(u) \geq 0$ pentru orice arc u se numeşte *flux* în reţeaua de transport G cu funcţia de capacitate c, care se notează $G = (X, U, c)$, dacă sunt îndeplinite următoarele două condiţii:

a) Condiţia de conservare a fluxului: Pentru orice vârf x diferit de a şi b suma fluxurilor pe arcele care intră în x este egală cu suma fluxurilor pe arcele care ies din x.
b) Condiţia de mărginire a fluxului: Există inegalitatea $f(u) \leq c(u)$ pentru orice arc $u \in U$.

Dacă $f(u) = c(u)$ arcul se numeşte *saturat*. Un *drum* se va numi *saturat* dacă va conţine cel puţin un arc saturat. Fluxul, toate drumurile căruia sunt saturate se va numi *flux complet*. Cel mai mare dintre fluxurile complete se numeşte *flux maxim*.

Pentru orice mulţime de vârfuri $A \in U$ vom defini o *tăietură* $w\_(A) = \{(x, y) \mid x \notin A, y \in A, (x,y \in U\}$, adică mulţimea arcelor care intră în mulţimea A de vârfuri.

Prin $w^+(A)$ vom nota mulţimea arcelor care ies din mulţimea A de vârfuri.

Este justă afirmaţia: suma $f(u)$ pentru $u \in w^+(A)$ este egală cu suma $f(u)$ pentru arcele $u \in w\_(A)$. Această valoare comună se va nota f_b.

Algoritmul Ford-Fulkerson

Are loc următoarea **teoremă** (Ford-Fulkerson):

Pentru orice reţea de transport $G = (X, U, c)$ cu intrarea a şi ieşirea b valoarea maximă a fluxului la ieşire este egală cu capacitatea minimă a unei tăieturi, adică:

$$max\ f_b = min\ c(w\_(A)).$$

În baza acestei teoreme a fost elaborat următorul algoritm de determinare a fluxului maxim (Ford-Fulkerson) la ieşirea b a unei reţele de transport $G = (X, U, c)$, unde capacitatea c ia numai valori întregi:

1. Se defineşte fluxul iniţial având componente nule pe fiecare arc al reţelei, adică $f(u) = 0$ pentru orice arc $u \in U$;

2. Se determină lanţurile nesaturate de la a la b pe care fluxul poate fi mărit, prin următorul procedeu de etichetare:

a) Se marchează intrarea a cu [+];

b) Un vârf x fiind marcat, se va marca:

cu [+x] oricare vârf y nemarcat cu proprietatea că arcul $u = (x, y)$ este nesaturat, adică $f(u)<c(u)$;

cu [-x] - orice vârf y nemarcat cu proprietatea că arcul $u = (x, y)$ are un flux nenul, adică $f(u)>0$.

Dacă prin acest procedeu de marcare se etichetează ieşirea b, atunci fluxul f_b obţinut la pasul curent nu este maxim. Se va considera atunci un lanţ format din vârfurile etichetate (ale căror etichete au respectiv semnele + sau -) care uneşte pe a cu b şi care poate fi găsit uşor urmărind etichetele vârfurilor sale în sensul de la b către a.

Dacă acest lanţ este v, să notăm cu v^+ mulţimea arcelor (x, y), unde marcajul lui y are semnul "+", deci care sunt orientate în sensul de la a către b şi cu $v\_$ mulţimea arcelor (x, y), unde marcajul lui y are semnul "-", deci care sunt orientate în sensul de la b către a.

Determinăm cantitatea:

$e = min\ \{min(c(u) - f(u)), min\ f(u)\}.\ u \in v^+,\ u \in v\_$

Din modul de etichetare rezultă $e > 0$.

Vom mări cu e fluxul pe fiecare arc u din v^+ şi vom micşora cu e fluxul pe fiecare arc $u \in v\_$, obţinând la ieşire un flux egal cu f_b+e. Se repetă aplicarea pasului 2 cu fluxul nou obţinut.

Dacă prin acest procedeu de etichetare nu putem marca ieşirea b, fluxul f_b are o valoare maximă la ieşire, iar mulţimea arcelor care unesc vârfurile marcate cu vârfurile care nu au putut fi marcate constituie o tăietură de capacitate minimă (demonstraţi că se va ajunge în această situaţie după un număr finit de paşi).

3. SARCINA DE BAZĂ

1. Realizaţi procedura introducerii unei reţele de transport cu posibilităţi de verificare a corectitudinii datelor introduse;
2. În conformitate cu algoritmul Ford-Fulkerson elaboraţi procedura determinării fluxului maxim pentru valori întregi ale capacităţilor arcelor;
3. Elaboraţi programul care va permite îndeplinirea următoarelor deziderate:

- introducerea reţelei de transport în memorie;
- determinarea fluxului maxim pentru reţeaua concretă;
- extragerea datelor obţinute (fluxul maxim şi fluxul fiecărui arc) la display şi printer.

4. ÎNTREBĂRI DE CONTROL

1. Ce se numeşte reţea de transport?
2. Formulaţi noţiunile de flux şi capacitate.
3. Ce este un arc saturat? Dar un drum saturat?
4. Ce se numeşte flux complet? Ce este un flux maxim?
5. Definiţi noţiunea de tăietură.
6. Formulaţi teorema Ford-Fulkerson.
7. Descrieţi algoritmul de determinare a fluxului maxim.
8. Demonstraţi că algoritmul se va opri după un număr finit de paşi.

Bibliografie

1. T. Bânzaru ş.a. Matematici speciale. Bucureşti, E.D.P., 1981
2. Gh.Mihoc, N.Micu. Teoria probabilităţilor şi statistica matematica. Bucureşti, E.D.P., 1980