

MINISTERUL EDUCAȚIEI ȘI ȘTIINȚEI AL REPUBLICII MOLDOVA
UNIVERSITATEA TEHNICĂ A MOLDOVEI
CATEDRA "AUTOMATICĂ ȘI TEHNOLOGII INFORMAȚIONALE"

Discutat și aprobat
la ședința Consiliului Metodic al Facultății
"Calculatoare, Informatică și Microelectronică"
Decan F.C.I.M. _____ V.Șontea
" ____ " _____ 1999

INDICAȚII METODICE

la lucrările de laborator la disciplina "Matematica discretă în inginerie"
pentru studenții anului 1, specialitățile "Tehnologii informaționale" și "Calculatoare"

(Elaborat de Victor Beșliu)

Chișinău 1999

Grafuri

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 existență 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 aceeași perioadă apare și cunoscuta problemă despre patru culori.

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

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

NOȚIUNI GENERALE

1. Definiția grafului

Se numește graf, ansamblul format dintr-o mulțime finită X și o aplicație F a lui X în X . Se notează $G = (X, F)$. Numărul elementelor mulțimii X determină ordinul grafului finit. Dacă $\text{card } X = n$, graful $G = (X, F)$ se numește *graf finit de ordinul n* . Elementele mulțimii X se numesc *vârfurile* grafului. Geometric, vârfurile unui graf le reprezentăm prin puncte sau cerușe. Perechea de vârfuri (x, y) se numește *arc*; vârful x se numește originea sau extremitatea inițială a arcului (x, y) , iar vârful y se numește 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 numește *vârf izolat*, iar dacă este extremitatea a mai mult de două arce - *nod*. Un arc pentru care extremitatea inițială coincide cu cea finală se numește *bucă*.

Arcele unui graf le mai notăm și cu u_1, u_2, \dots , iar mulțimea arcelor grafului o notăm cu U . Se observă că mulțimea U a tuturor arcelor unui graf determină complet aplicația F , precum și reciproc, aplicația F determină mulțimea 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 și au o extremitate comună. Două vârfuri se zic *adiacente* dacă sunt distincte și sunt unite printr-un arc.

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

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

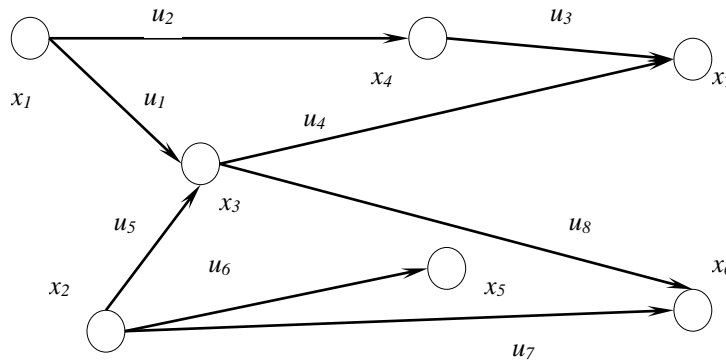


Fig. 1. Exemplu de graf

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

Un drum elementar ce trece prin toate vârfurile grafului se numește drum hamiltonian. Un drum simplu ce conține toate arcele grafului se numește drum eulerian. Un drum finit pentru care vârful inițial coincide cu vârful terminal se numește circuit.

Graful obținut din graful inițial suprimând cel puțin un vârf al acestuia precum și toate arcele incidente cu el se numește subgraf. Graful obținut suprimând cel puțin un arc se numește graf parțial.

Un graf se numește complet dacă oricare ar fi x și 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 $G = (X,F)$ și $x \in X$. Mulțimea C_x formată din toate vârfurile $x_i \in X$ pentru care există un circuit ce trece prin x și x_i se numește componentă tare conexă a lui G corespunzătoare vârfului x .

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

Noțiunile introduse sunt valabile pentru grafurile orientate.

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

Cele două concepte de graf orientat și 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 și invers, dacă se precizează orientarea.

Unui graf orientat simetric i se poate asocia un graf neorientat, legătura dintre două vârfuri x și y realizată de cele două arce (x,y) și (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 și $x \in X$. Se numește 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 cociclomantic și număr ciclomantic

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

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

iar numărul ciclomantic numărul

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

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

Teorema. Fie $G = (X,U)$ un multigraf, iar $G_I = (X,U_I)$ un multigraf obținut din G adăugând o muchie. Dacă $x, y \in X$ și $[x,y]$ este muchia care se adaugă la mulțimea muchiilor grafului G , atunci:

(1) dacă $x = y$ sau x și y sunt unite printr-un lanț

$$r(G_I) = r(G), s(G_I) = s(G) + 1$$

(2) în caz contrar

$$r(G_I) = r(G) + 1, s(G_I) = s(G).$$

Demonstrația este evidentă.

Consecință. Numerele cociclomantic și ciclomantic sunt nenegative.

3. Număr cromatic. Grafuri planare. Arbori

Un graf $G = (X,U)$ se zice ca este *graf p -cromatic* dacă vârfurile lui se pot colora cu p -culori distincte astfel ca două vârfuri adiacente să nu fie de aceeași culoare. *Cel mai mic* număr întreg și pozitiv pentru care graful este *p -cromatic* se numește *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 extremitățile lor. Un graf planar determină regiuni numite *fețe*. O față este o regiune mărginită de muchii și care nu are în interior nici vârfuri, nici muchii. Conturul unei fețe este format din muchiile care o mărginesc. Două fețe sunt adiacente dacă conturile 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 și numai dacă nu are cicluri de lungime impară.

Se numește arbore oricare graf conex fără bucle și cicluri.

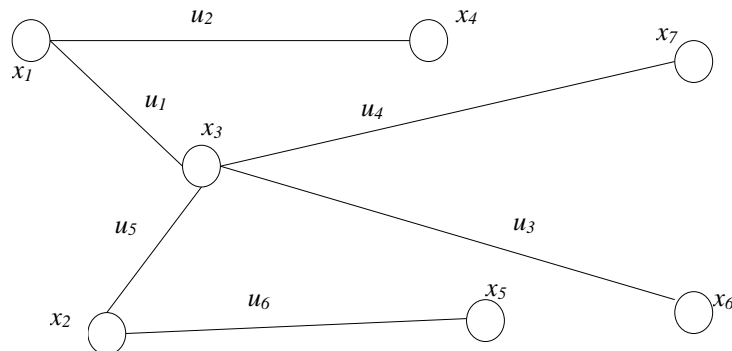


Fig. 2. Exemplu de arbore

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

LUCRAREA DE LABORATOR Nr. 1

TEMA: PĂSTRAREA GRAFURILOR ÎN MEMORIA CALCULATORULUI

1. SCOPUL LUCRĂRII:

- Studiarea metodelor de definire a unui graf: matrice de incidență, matrice de adiacență, liste;
- Elaborarea unor proceduri de introducere, extragere și transformare a diferitor forme de reprezentare internă a grafurilor cu scoaterea rezultatelor la display și imprimantă.

2. NOTE DE CURS

Metode de reprezentare a grafului

Există trei metode de bază de definire a unui graf:

1. *Matricea de incidență;*
2. *Matricea de adiacență;*
3. *Lista de adiacență (incidență).*

Vom face cunoștință cu fiecare dintre aceste metode.

Matricea de incidență

Este o matrice de tipul $m \times n$, în care m este numărul de muchii sau arce (pentru un graf orientat), iar n este numărul vârfurilor. La intersecția 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 și 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 incidență (v.fig.1)

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

În limbajul Pascal matricea de incidență poate fi redată printr-un tablou bidimensional $m \times n$:

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

Matricea de adiacență

Este o matrice pătrată $n \times n$, 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 față 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 adiacență (v.fig.1)

În limbajul Pascal matricea de adiacență 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 și în acest caz memoria calculatorului este utilizată nu prea eficient din care cauză matricea de adiacență ca și matricea de incidență 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 facilități algoritmului respectiv.

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

Lista de adiacență și lista de incidență

Lista de adiacență 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 incidență se definește 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 și în timpul procesării. Pentru a lua în considerație lungimea variabilă a liniilor vor fi utilizate variabile dinamice și pointeri. Vom exemplifica pentru un graf cu n vârfuri. Deoarece fiecare element al listei conține numere de vârfuri este evident să considerăm că vom avea un șir de variabile dinamice de tip INTEGER care se vor afla în relația respectivă de precedare (succedare). Această relație se va realiza prin pointeri, uniți împreună cu variabila de tip întreg în înregistrarea (Pascal: record). Pentru a păstra indicatorii de intrare în aceste șiruri se va folosi un tablou unidimensional de indicatori de lungime n . În calitate de simbol de terminare a șirului se va utiliza un simbol care nu a fost folosit la numerația vârfurilor (de exemplu 0), care va fi introdus în calitate de variabilă de tip întreg al ultimului bloc.

De exemplu, lista de adiacență (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:

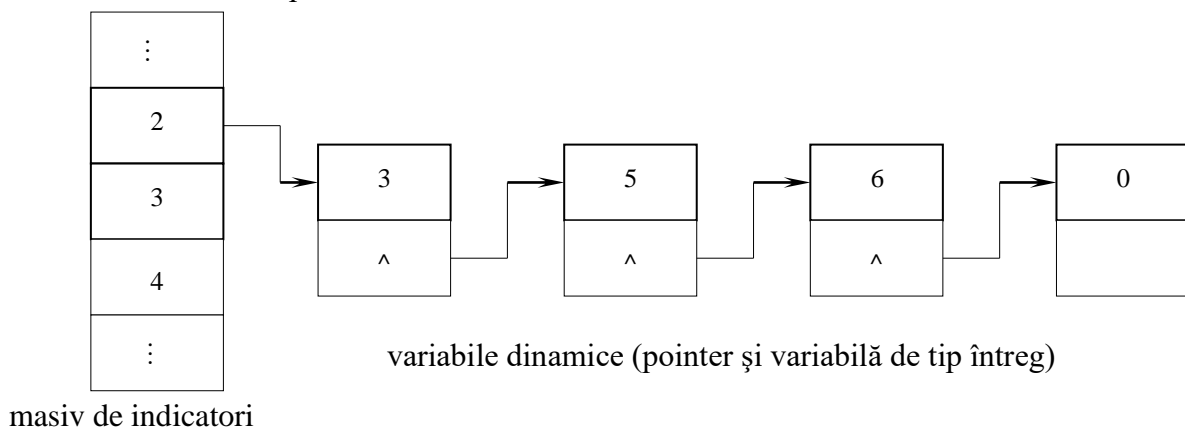


Fig. 5. Reprezentarea internă a listei de adiacență

Va fi necesar de realizat următoarea declarație:

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

Lista de adiacență (și 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. Elaborați procedura introducerii unui graf în memoria calculatorului în formă de matrice de incidență, matrice de adiacență și listă de adiacență cu posibilități de analiză a corectitudinii.
2. Elaborați proceduri de transformare dintr-o formă de reprezentare în alta.
3. Folosind procedurile menționate elaborați programul care va permite:
 - introducerea grafului reprezentat sub oricare din cele trei forme cu posibilități de corecție a datelor;
 - păstrarea grafului în memoria externă în formă de listă de adiacență;
 - extragerea informației într-una din cele trei forme la imprimantă și display.

4. ÎNTREBĂRI DE CONTROL

1. Care sunt metodele de bază de reprezentare a unui graf?
2. Descrieți 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 și a diferitor forme de păstrare și prelucrare a datelor.
- Elaborarea procedurii de căutare în adâncime.

2. NOTE DE CURS

Structuri de date: liste

Fiecare tip de listă definește o mulțime de șiruri finite de elemente de tipul declarat. Numărul de elemente care se numește lungimea listei poate varia pentru diferite liste de același tip. Lista care nu conține nici un element se va numi vidă. Pentru listă sunt definite noțiunile începutul, sfârșitul listei și respectiv primul și ultimul element, de asemenea elementul curent ca și predecesorul și succesorul elementului curent. Element curent se numește acel unic element care este accesibil la momentul dat.

Structuri de date : fire de așteptare

Firele de așteptare (FA, rând, coadă, șir de așteptare) 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).

Operațiile de bază cu firele de așteptare:

- 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).

Operațiile 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 mulțime de structuri fiecare din care va consta dintr-un obiect de bază numit *vârf* sau *rădăcina arborelui* dat și o listă de elemente din mulțimea 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.

Operațiile de bază pentru arbori vor fi:

- Formarea unui arbore trivial;
- Alegerea sau înlocuirea rădăcinii arborelui;
- Alegerea sau înlocuirea listei rădăcinilor subarborilor;
- Operațiile 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 și se termină cu vârful vizitat în momentul dat, poate fi realizat un algoritm nerecursiv de forma:

```
Vizitează rădăcina arborelui și 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 vizitați  
      THEN vizitează fiul mai mare al lui  $p$  și introduce-l în S  
      ELSE BEGIN  
        elimină vârful  $p$  din stiva S  
        IF  $p$  are frați THEN vizitează pe fratele lui  $p$  și introduce-l în stiva S  
      END  
  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 relație de ordine pe mulțimea tuturor vârfurilor grafului, iar mulțimea vârfurilor adiacente cu un vârf arbitrar al grafului este de asemenea ordonată:

```
WHILE va exista cel puțin un vârf care nu a fost vizitat DO  
  BEGIN  
    fie  $p$  - primul din vârfurile nevizitate;  
    vizitează vârful  $p$  și 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  $z$  și introduce-l în stiva S;  
          END  
        ELSE elimină vârful  $p$  din stiva S  
      END  
    END  
  END
```

În cazul în care se va lucra cu un graf conex arbitrar cu relația de ordine lipsă, nu va mai avea importanță ordinea de parcurgere a vârfurilor. Propunem un algoritm care utilizează mai larg posibilitățile stivei, cea ce face programul mai efektiv 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 inițial și 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 și introduce-le în stivă;  
    END
```

3. SARCINA DE BAZĂ

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

4. ÎNTREBĂRI DE CONTROL

1. Definiți structurile principale de date: liste, fire de așteptare, stive, arbori.
2. Care sunt operațiile definite pentru aceste structuri de date?
3. Care este principiul de organizare a prelucrării elementelor în firele de așteptare și în stive?
4. Definiți noțiunea de parcurgere a grafului în adâncime.
5. Ce fel de structuri de date se vor utiliza în căutarea în adâncime?
6. Exemplificați utilizarea algoritmului de căutare în adâncime.

LUCRAREA DE LABORATOR Nr. 3

TEMA: ALGORITMUL DE CĂUTARE ÎN LĂRGIME

1. SCOPUL LUCRĂRII:

- Studiarea 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 și parcurgerea în adâncime, va garanta vizitarea fiecărui vârf al grafului exact o singură dată, însă principiul va fi altul. După vizitarea vârfului inițial, 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 ș.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 așteptare *O1* și *O2*.

Se vor forma două fire de așteptare vide O_1 și O_2 ;

Introduce rădăcina în FA O_1 ;

WHILE cel puțin unul din firele de așteptare O_1 sau O_2 nu va fi vid DO

IF O_1 nu este vid THEN

BEGIN

 fie p vârful din topul FA O_1 ;

 vizitează vârful p eliminându-l din O_1 ;

 vizitează pe toți fiii lui p în FA O_2 , începând cu cel mai mare;

END

ELSE

 în calitate de O_1 se va lua FA O_2 , care nu este vid,

 iar în calitate de O_2 se va lua FA vid O_1 ;

Vom nota că procedura parcurgerii grafului în lărgime permite să realizăm arborele de căutare și în același timp să construim acest arbore. Cu alte cuvinte, se va rezolva problema determinării unei rezolvări sub forma vectorului (a_1, a_2, \dots) 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 și O_2 vide;

Introdu vârful inițial în FA O_1 și marchează-l;

WHILE FA O_1 nu este vid DO

BEGIN

 vizitează vârful din topul FA O_1 și elimină-l din FA;

 IF există vârfuri nemarcate adiacente cu vârful dat THEN introdu-le în FA O_2 ;

END

3. SARCINA DE BAZĂ

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

4. ÎNTREBĂRI DE CONTROL

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

LUCRAREA DE LABORATOR Nr. 4

TEMA: ALGORITMUL DETERMINĂRII GRAFULUI DE ACOPERIRE

1. SCOPUL LUCRĂRII:

- Studiarea algoritmului de determinare a grafului de acoperire și elaborarea programelor care vor realiza acest algoritm.

2. NOTE DE CURS

Noțiuni de graf de acoperire

Fie H un subgraf care conține 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 numește graf aciclic orice graf care nu conține cicluri. Pentru un graf arbitrar G cu n vârfuri și m muchii sunt echivalente următoarele afirmații:

1. G este arbore;
2. G este un graf conex și $m = n - 1$;
3. G este un graf aciclic și $m = n - 1$;
4. oricare două vârfuri distincte (diferite) ale lui G sunt unite printr-un lanț 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 obținut va conține exact un ciclu.

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

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

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

Numărul $s(G) = m(G) - n(G) + k(G)$ se numește *rang ciclic* sau număr *ciclotomic* al grafului G . Numărul $r(G) = n(G) - k(G)$ – *rang cociclotomic* sau număr *cociclotomic*.

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

Este adevărată următoarea afirmație: orice subgraf a unui graf arbitrar G se conține într-un graf de acoperire a grafului G .

Algoritmul de determinare a grafului de acoperire

Există mai mulți 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. Esența algoritmului constă în aceea că folosind două fire de așteptare în unul din care sunt înscrise (pe rând) numerele vârfurilor adiacente cu vârfurile din celălalt FA (ca și în cazul căutării în lărgime), vor fi eliminate muchiile dintre vârfurile unui FA și toate muchiile în afară de una dintre fiecare vârf al FA curent și vârfurile din FA precedent. În cazul în care ambele FA vor deveni vide procedura se va termina.

Pentru a nu admite ciclarea și 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 sfârșit, în caz contrar în calitate de vârf inițial se va lua oricare din vârfurile nemarcate.

Descrierea algoritmului:

1. Se vor declara două FA (FA_1 și FA_2) vide.
2. Se va lua în calitate de vârf inițial un vârf arbitrar al grafului.
3. Se va introduce vârfurile inițiale în firul de așteptare vid FA_1 și se va marca acest vârf.
4. Se vor introduce în FA_2 toate vârfurile adiacente cu vârfurile din FA_1 și 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 vârfurile din FA_1 .
7. Se vor schimba cu numele FA_1 și FA_2 (FA_1 va deveni FA_2 și invers).
8. Dacă există cel puțin un vârf nemarcat se va lua în calitate de vârf inițial oricare din acestea și se va trece la p.1, altfel
9. STOP.

Graful obținut este graful de acoperire.

3. SARCINA DE BAZĂ

1. Elaborați organigrama algoritmului și programul procedurii de determinare a grafului de acoperire cu posibilități de pornire a procedurii din oricare vârf al grafului.
2. Utilizând procedurile de introducere a grafului în memoria CE din lucrarea Nr. 1, elaborați un program cu următoarele facilități:
 - introducerea grafului care este dat sub formă de matrice de incidență, adiacență sau listă de adiacență;
 - determinarea grafului de acoperire, pornind de la un vârf arbitrar;
 - extragerea informației la display sau imprimantă în oricare din formele numite.

4. ÎNTREBĂRI DE CONTROL

1. Ce este un graf aciclic și prin ce se deosebește el de un arbore?
2. Definiți noțiunile de arbore și graf de acoperire.
3. Care vor fi transformările ce vor fi efectuate într-un graf arbitrar pentru a obține graful de acoperire?
4. Care este esența algoritmului de determinare a grafului de acoperire?
5. Evidențiați 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:

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

2. NOTE DE CURS

Noțiuni de drum minim

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

Vârfurile x_0 și x_r se numesc extremitățile drumului D . Lungimea unui drum este dată de numărul de arce pe care le conține. Dacă vârfurile x_0, x_1, \dots, x_r sunt distincte două câte două drumul D este elementar.

Adeseori, fiecărui arc (muchii) i se pune în corespondență un număr real care se numește *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 lungimea a unui drum suma ponderilor arcelor care formează acest drum. Drumul care unește două vârfuri concrete și are lungimea cea mai mică se va numi *drum minim* iar lungimea drumului minim vom numi *distanță*. Vom nota distanța dintre x și t prin $d(x, t)$, evident, $d(x,x)=0$.

Algoritmul lui Ford pentru determinarea drumului minim

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

1. Fiecărui vârf x_j al grafului G se va atașa un număr foarte mare $H_j(\infty)$. Vârfului inițial i se va atașa $H_i = 0$;
2. Se vor calcula diferențele $H_j - H_i$ 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 micșorarea distanței dintre vârful inițial și 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 distanța de la vârful inițial până la vârful dat x_i .

3. Acest pas presupune stabilirea secvenței de vârfuri care va forma drumul minim. Se va pleca de la vârful final x_j spre cel inițial. 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 relație se va alege la opțiune.

Algoritmul Bellman - Calaba

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

Etapă inițială presupune atașarea grafului dat G a unei matrice ponderate de adiacență, 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 $i = 1, 2, \dots, n-1, j = 1, 2, \dots, n; i < j$;
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. Elaborați procedura introducerii unui graf ponderat;
2. Elaborați procedurile determinării drumului minim;
3. Realizați un program cu următoarele funcții:
 - introducerea grafului ponderat cu posibilități de analiză sintactică și semantică și de corectare a informației;
 - determinarea drumului minim;
 - extragerea informației la display și printer (valoarea drumului minim și succesiunea vârfurilor care formează acest drum).

4. ÎNTREBĂRI DE CONTROL

1. Ce se numește graf ponderat?
2. Definiți noțiunea de distanță.
3. Descrieți etapele principale ale algoritmului Ford.
4. Care sunt momentele principale în algoritmul Bellman-Calaba?
5. Prin ce se deosebește 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 REȚEA DE TRANSPORT

1. SCOPUL LUCRĂRII:

- Studiarea noțiunilor de bază legate de rețelele de transport;
- Programarea algoritmului Ford-Fulkerson pentru determinarea fluxului maxim într-o rețea de transport.

2. NOTE DE CURS

Rețele de transport

Un graf orientat $G = (X, U)$ se numește *rețea de transport* dacă satisface următoarele condiții:

- 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 și există drumuri de la a la b în G ;

d) s-a definit o funcție $c: U \rightarrow \mathbb{R}$ astfel încât $c(u) \geq 0$ pentru orice arc u din U .

Vârful a se numește intrarea rețelei, vârful b se numește ieșirea rețelei, iar $c(u)$ este capacitatea arcului u .

O funcție $f: U \rightarrow \mathbb{R}$ astfel încât $f(u) \geq 0$ pentru orice arc u se numește *flux* în rețeaua de transport G cu funcția de capacitate c , care se notează $G = (X, U, c)$, dacă sunt îndeplinite următoarele două condiții:

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

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

Pentru orice mulțime de vârfuri $A \subseteq U$ vom defini o *tăietură* $w_-(A) = \{(x, y) \mid x \notin A, y \in A, (x, y) \in U\}$, adică mulțimea arcelor care intră în mulțimea A de vârfuri.

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

Este justă afirmația: 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 rețea de transport $G = (X, U, c)$ cu intrarea a și ieșirea b valoarea maximă a fluxului la ieșire 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 ieșirea b a unei rețele de transport $G = (X, U, c)$, unde capacitatea c ia numai valori întregi:

1. Se definește fluxul inițial având componente nule pe fiecare arc al rețelei, adică $f(u) = 0$ pentru orice arc $u \in U$;
2. Se determină lanțurile 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 ieșirea b , atunci fluxul f_b obținut la pasul curent nu este maxim. Se va considera atunci un lanț format din vârfurile etichetate (ale căror

etichete au respectiv semnele + sau -) care unește pe a cu b și care poate fi găsit ușor urmărind etichetele vârfurilor sale în sensul de la b către a .

Dacă acest lanț este v , să notăm cu v^+ mulțimea arcelor (x, y) , unde marcajul lui y are semnul "+", deci care sunt orientate în sensul de la a către b și cu v_- mulțimea 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^+ și vom micșora cu e fluxul pe fiecare arc $u \in v_-$, obținând la ieșire un flux egal cu $f_b + e$. Se repetă aplicarea pasului 2 cu fluxul nou obținut.

Dacă prin acest procedeu de etichetare nu putem marca ieșirea b , fluxul f_b are o valoare maximă la ieșire, iar mulțimea arcelor care unesc vârfurile marcate cu vârfurile care nu au putut fi marcate constituie o tăietură de capacitate minimă (demonstrați că se va ajunge în această situație după un număr finit de pași).

3. SARCINA DE BAZĂ

1. Realizați procedura introducerii unei rețele de transport cu posibilități de verificare a corectitudinii datelor introduse;
2. În conformitate cu algoritmul Ford-Fulkerson elaborați procedura determinării fluxului maxim pentru valori întregi ale capacităților arcelor;
3. Elaborati programul care va permite îndeplinirea următoarelor deziderate:
 - introducerea rețelei de transport în memorie;
 - determinarea fluxului maxim pentru rețeaua concretă;
 - extragerea datelor obținute (fluxul maxim și fluxul fiecărui arc) la display și printer.

4. ÎNTREBĂRI DE CONTROL

1. Ce se numește rețea de transport?
2. Formulați noțiunile de flux și capacitate.
3. Ce este un arc saturat? Dar un drum saturat?
4. Ce se numește flux complet? Ce este un flux maxim?
5. Definiți noțiunea de tăietură.
6. Formulați teorema Ford-Fulkerson.
7. Descrieți algoritmul de determinare a fluxului maxim.
8. Demonstrați că algoritmul se va opri după un număr finit de pași.

Bibliografie

1. T. Bânzaru ș.a. Matematici speciale. București, E.D.P., 1981
2. Gh.Mihoc, N.Micu. Teoria probabilităților și statistica matematica. București, E.D.P., 1980