ASPECTE DIDACTICE ÎN PREDAREA ALGORITMILOR PENTRU DETERMINAREA DRUMURILOR MINIME ÎN GRAFURI

Liubomir CHIRIAC, prof. univ., dr. hab.
Marina BOSTAN, doctorand
Universitatea de Stat Tiraspol

Rezumat. În acest articol sunt examinate abordări didactice privind procesul de predare-învățare al algoritmilor pentru determinarea drumurilor minime în grafuri. Luând în considerație provocările tehnologice din ultima perioadă, autorii menționează că este necesar să se schimbe accentele și paradigmele privind studierea Teoriei Grafurilor în instituțiile superioare de învățământ. Instrumentele didactice propuse de autor vin să soluționeze unele din problemele examinate.
Cuvinte-cheie: teoria grafurilor, drumuri minime, algoritm, predare-învățare.

DIDACTIC ASPECTS IN TEACHING THE ALGORITHMS FOR DETERMINING THE MINIMUM PATHS IN GRAPHS

Abstract. This article examines didactic approaches on teaching-learning the algorithms for determining the minimum paths in graphs. Taking into consideration the technological challenges of the last period, the authors point out that it is necessary to change the emphasis and paradigms regarding the study of the Graph Theory in the higher education institutions. The didactic tools proposed by the author come to solve some of the problems examined.
Keywords: graph theory, shortest paths, algorithm, teaching-learning.

1. Considerente metodice privind aplicațiile TIC la expunerea unor subiecte din cursul universitar de Teoria Grafurilor

Teoria grafurilor, după cum se știe și se recunoaște de specialiștii în domeniu și utilizatorii consacrați, are o mare importanță practică, deoarece prin intermediul algoritmilor și metodelor din teoria respectivă sunt soluționate o serie de probleme care țin de transport, turism, economie, logistică, fizică, biologie, chimie, etc. Luând în considerație provocările tehnologice din ultima perioadă, este necesar să se schimbe accentele și paradigmele privind studierea Teoriei Grafurilor în instituțiile superioare de învățământ.

În mod tradițional, cursul universitar Teoria Grafurilor, predat, în mod special, în universitățile cu profil pedagogic, se studiază utilizându-se un număr limitat de instrumente didactice cu implementarea limitată a tehnologiilor informaționale moderne. Eficiența procesului de predare-învățare-evaluare poate să crească continuu, doar introducând și perfecționând noi instrumente didactice, în conexiune strânsă cu implementarea noilor tehnologii informaționale.

În experimentul desfășurat de noi, privind implementarea tehnologiilor moderne în cursul universitar de Teoria Grafurilor, s-a analizat din mai multe puncte de vedere oportunitățile și avantajele oferite de soft-urile MAPLE 18, DELPHI, TPASACAL, C/C++ [7, 8, 9].

Puncte slabe în procesul de predare-învățare a subiectelor care țin de cursul Teoria Grafurilor

Vom scoate în evidență unele chestiuni didactice privind procesul de predare a Teoriei Grafurilor, care, în viziunea noastră, nu sunt tratate, în virtutea unor circumstanțe obiective ori subiective, destul de profund la această disciplină. Menționăm următoarele conform [1, 3, 5, 6]:

a) În manualele existente, destinate studenților de la universitățile cu profil pedagogic, de exemplu, nu sunt suficiente probleme ce ar conduce la elaborarea modelelor matematice, selectarea metodelor necesare din teoria grafurilor, elaborarea algoritmilor și testarea rezultatelor obținute la calculator. Parcurgerea etapelor fundamentale, care țin de modelarea matematică le va cultiva, în opinia noastră, studenților, deci, ulterior, și elevilor din licee o viziune generală privind integritatea procesului examinat și le va dezvolta o atitudine creatoare vizavi de abordarea acestui proces.

b) La expunerea subiectelor din Teoria Grafurilor, se utilizează tradițional limbajele de programare, care au, evident, anumite avantaje, dar, în același timp, pot fi substituite cu succes de alte softuri matematice moderne, cum ar fi de exemplu: AutoCad, MatLab, Maple, Matematica, etc. Utilizarea softurilor de ultim[generație imprimă lecțiilor care țin de algoritmica grafurilor un suflu nou și, totdată, sunt cu mult mai atractive, ilustrative și, în unele situații, cu mult mai eficiente și mai practice. În exemplele examinate mai jos vom puncta avantajele care țin de utilizarea acestor softuri, luând ca model softul Maple 18.

c) În cadrul predării-învățării-evaluării teoriei grafurilor se presupune însușirea celor mai importante metode și algoritmi din acest domeniu, precum și aplicarea lor directă la soluționarea unor probleme concrete, modelul matematic al cărora este foarte clar și exact. Chiar dacă această abordare este corectă din punctul de vedere al asimilării materiei propuse, acest fapt nu îndeamnă studenții, mai ales cei cu pregătirea peste nivelul mediu, la o atitudine creatoare privind elaborarea, obținerea modelului matematic, prin intermediul căruia ar putea fi rezolvată problema respectivă.

d) Prin intermediul Teoriei Grafurilor în general și a însușirii subiectelor, care se referă la algoritmica grafurilor pentru studenții din universitățile cu profil pedagogic, se poate realiza eficient conexiunea cu alte discipline (turism, teoria comunicațiilor, rețele de transport, etc.) studiate atât de studenți, cât și elevi în cursul liceal (fizică, chimie, biologie, geografie, etc.). Chestiunile care țin de legăturile interdisciplinare pot fi tratate cu succes anume în procesul elaborării, examinării modelelor matematice ale diverselor fenomene cercetate (matematice, informatice, geografice, etc.) prin intermediul implementării softurile moderne. De aceste aspecte metodice nu întotdeauna se ține cont la predarea Teoriei Grafurilor.

Analizând [2, 3, 4, 5, 6], vom încerca să aducem ca exemple unele probleme, ale căror modele matematice sunt elaborate și soluționate cu ajutorul algoritmicii grafurilor,

soluționate manual, ulterior, aplicând limbajul Pascal și, în mod special, softul matematic Maple 18.

2. Chestiuni metodice privind utilizarea softul matematic Maple 18 în Teoria Grafurilor

Softul Maple este un sistem de calcul algebric (CAS) dezvoltat de firma Maplesoft (http://maplesoft.com) [7]. El poate fi utilizat și la soluționarea unor probleme din Teoria Grafurilor. În acest scop se folosesc diverse pachete: Combinatorics, GraphTheory, Linear Algebra, Optimization, Plots, etc.

Pachetele sunt colecții de funcții care permit efectuarea de calcule specifice. Apelarea lor se face cu ajutorul comenzii *with* (nume_pachet). Pentru a apela o anumită funcție dintr-un pachet, se folosește sintaxa: pachet [‘funcție’] (argumente).

De exemplu, la apelarea pachetului *with(**GraphTheory**)*, se obține lista tuturor funcțiilor apelabile care țin de examinarea și cercetarea proprietăților grafurilor.

Pachetul **GraphTheory** reprezintă o colecție de comenzi destinate pentru crearea, construirea, manipularea și testarea grafurilor și determinarea proprietăților acestora.

Comandă de bază pentru crearea unui graf este **Graph**, care poate fi folosită în mai multe forme: Graph(V); Graph(n); Graph(V, E); Graph(L); Graph(V, L); Graph(A); Graph(V, E, A); Graph(D, W, V, E, A); Graph(N), unde parametrii sunt opționale și au următoare semnificație:

V - (opțional) lista nodurilor (numere întregi, simboluri, șiruri de caractere);
n - (opțional) numerele întregi pozitive pentru specificarea nodurilor 1,2,...,n;
E - (opțional) mulțimea muchiilor (arcelor);
L - (opțional) lista sau vectorul nodurilor-vecini;
A - (opțional) matricea de adiacență (costurilor muchiilor);
D - (opțional) simbolul, orientat sau neorientat;
W - (opțional) simbolul, cu costuri sau fără costuri;
N - (opțional) procedura (graful rețelelor).

Tipul fiecărui argument determină elementele unui graf, de aceea ele pot fi punctate în orice ordine.

Un număr întreg ***n*** specifică numărul de noduri și, implicit, vârful etichetelor ***1...n***.

O listă de numere întregi ***V*** este o listă de simboluri sau șiruri de caractere, care specifică nodurile. Fiecare nod trebuie să fie un număr întreg, simbol sau un șir.

O mulțime ***E*** specifică mulțimea de muchii. O muchie neorientată între nodurile ***i*** și ***j*** este introdusă ca o mulțime de două noduri. O muchie orientată din vârful ***A*** la vârf ***B*** este introdusă ca o listă. O muchie ponderată este introdusă ca o muchie, unde ***w*** – greutatea muchiei, care este un număr întreg sau zecimal.

Un vector / listă de noduri ***L*** specifică drumul de la nodurile la vecinii lor. Dacă graful este neorientat, atunci lista / vector de vecini trebuie să fie simetrică.

O matrice ***A*** înseamnă matricea de adiacență. O matrice simetrică este interpretată ca un graf neorientat, iar în cazul grafului orientat orientarea muchiilor este precizată altfel.

O procedură ***N*** reprezintă un graf de rețele. Această opțiune permite conversia dintr-o reprezentare din rețele într-un graf pentru reprezentarea GraphTheory.

Pentru reprezentare grafică a grafului în Maple se utilizează comanda **DrawGraph**, care, de asemenea, poate fi utilizată în mai multe forme: DrawGraph(G); DrawGraph(G, style=s); DrawGraph(G, style=tree, root=v); DrawGraph(G, dimension=d); DrawGraph(G, style=spring); DrawGraph(G, style=spring, redraw); DrawGraph(G, style=spring[constant]); DrawGraph(L, options).

Parametrii utilizați sunt:

G - graful;
s - (opțional) circle, tree, bipartite, spring, planar;
v - (opțional) numele nodului rădăcină;
d - (opțional) numărul întreg 2 sau 3;
L - lista sau mulțimea grafurilor.

Comanda **DrawGraph** afișează vârfurile și muchiile unui graf **G**. În cazul în care **G** are mai puțin de 100 de noduri, sunt, de asemenea, afișate etichetele nodurilor. În cazul în care **G** este un graf ponderat cu mai puțin de 46 de muchii, sunt afișate greutățile acestora.

Opțiunea stilului în comanda **DrawGraph** pentru a afișa graful inițial într-un stil specific. Există câteva stiluri diferite susținute pentru afișarea unui graf: cerc, arbore, bipartit și planar.

Comanda A**llPairsDistance(G)**, unde **G** este graf, returnează o matrice pătrată, în care este prezentată distanța de la nodul ***i*** până la nodul ***j*** din graful **G**, adică cea mai scurtă lungime. Dacă **G** nu este graful orientat, atunci muchiile au costul egal cu **1**. În cazul în care nu există niciun drum între două noduri, atunci distanța este egală cu ∞.

Aceasta procedur[este o implementare a algoritmului **Roy-Floyd** (Bellman Ford), care determină toate drumurile minime între toate perechile de noduri.

Pentru a calcula distanțele sau cele mai scurte trasee dintr-un singur nod la oricare alt nod, utilizăm algoritmul lui **Dijkstra** ori **Roy-Floyd** (Bellman Ford).

Pentru determinarea drumului minim de la nod sursă până la celelalte noduri este prevăzută comanda **DijkstrasAlgorithm (G, s, t)**, unde **G** – este un graf fără costuri negative sau neponderat; **s**, **t** – nodurile grafului **G**.

Dacă **G** este un graf neorientat, atunci se presupune că toate muchiile au o costul **1**. În cazul în care **G** este un graf orientat, **DijkstrasAlgorithm**(, 's' 'G', 't') returnează cea mai scurtă cale ponderată de la vârf **s** la vârf **t** în graful **G**.

Pentru a calcula simultan distanțele dintre toate perechile de noduri, utilizăm comanda **AllPairsDistance**. Pentru a ignora greutățile utilizăm comanda **ShortestPath**.

Comanda **Distance(G, s, t)**, unde **G** – este un graf, **s**, **t** – nodurile grafului **G**, returnează numărul de muchii din cel mai scurt drum de la **s** la **t**. În cazul în care nu există o astfel de cale, rezultatul este infinit (∞).

3. Unele abordări didactice privind predarea compartimentului „Drumuri minime în grafuri"

Procesul de predare a compartimentului „Drumuri minime în graf" se axează atât pe partea teoretică, cât și pe cea practică, care este una foarte semnificativă în dezvoltarea și consolidarea abilităților necesare pentru soluționarea problemelor care țin de algoritmica grafurilor.

Algoritmii pentru determinarea drumurilor minime au multiple aplicații practice și reprezintă clasa de algoritmi pe grafuri cel mai des utilizată:

- Rutare în cadrul unei rețele (de calculatoare, telefonice etc.);
- Găsirea drumului minim dintre două locații (Google Maps, GPS etc.);
- Stabilirea unei agende de zbor în vederea asigurării unor conexiuni optime;
- Asignarea unui peer/server de fișiere în funcție de metricile definite pe fiecare linie de comunicație. [2]

Problema drumurilor minime este aplicată în diferite domenii ca problemă a determinării distanței minime între obiecte și este folosită aproape peste tot la găsirea traseului optim între două obiecte de pe teren (de exemplu, cel mai scurt drum de acasă la școală.), utilizată în sistemul pilot automat, este folosită, de asemenea, pentru a găsi ruta optimă pentru comutare de pachete de informații și multe altele.

Unele dintre cele mai frecvente și populare metode de a găsi cele mai scurte distanțe sunt algoritmii **Dijkstra** (pentru găsirea drumului minim de la un vârf fixat până la toate celelalte) și **Roy-Floyd** (pentru identificarea celei mai scurte căi între toate perechile de vârfuri).

Mai jos vom examina unele aspecte didactice referitor la implementarea tehnologiilor informaționale în predarea algoritmilor **Dijkstra**, și **Roy-Floyd**. [2, 4, 5]

Problema drumurilor minime în grafuri

Pentru un graf orientat $G=(V, A)$, unde V mulțimea nodurilor și A – mulțimea muchiilor grafului G, notăm prin a_{ij} – costul (lungimea) drumului între orice două noduri $(i, j) \in A$. Pentru drumul final $(v_1, v_2, \dots, v_k)$, lungimea poate fi exprimată prin suma: $a_{ij} = \sum_{i=1}^{k-1} a_{v_i v_{i+1}}$. Drumul se numește minimal, dacă lungimea acestuia este minimală dintre toate drumurile determinate pentru noduri i și j. Problema drumului minim include în sine căutarea drumurilor de lungime minimă între perechi de noduri inițiale.

Diversitatea problemei drumurilor minime, poate fi examinată prin prisma următoarelor două aspecte:

- Determinarea drumului minim de la un nod considerat sursă până la fiecare dintre celelalte noduri ale grafului;

- Determinarea drumurilor minime intre toate perechile de vârfuri.

În rezolvarea problemei drumurilor minime, pentru a simplifica calculele, vom considera graful orientat și costul drumului între două noduri a_{ij} un număr întreg pozitiv.

3.1. Abordări didactice privind implementarea algoritmului Dijkstra

Algoritmul Dijkstra a fost descoperit de Edsger W. Dijkstra, informatician olandez, în 1956 și publicat 3 ani mai târziu. Algoritmul lui **Dijkstra** determină câte un drum de cost minim de la vârful sursă, la fiecare dintre celelalte vârfuri ale grafului. La momentul inițial singurul vârf pentru care drumul de cost minim este cunoscut este vârful inițial. Astfel vom utiliza următoarele structuri:

- Un vector a cărui dimensiune este dată de numărul de noduri din graf și care memorează costul drumului de cost minim de la nodul de start la un nod oarecare, drum care trece doar prin vârfuri din mulțimea vârfurilor selectate;
- Un vector care va reține vârfurile pentru care drumul de cost minim este deja calculat;
- Un vector care reține pentru fiecare vârf din graf vârful care îl precede pe drumul de cost minim.

Mai detaliat. Considerăm graful orientat $G=(V, A)$, memorat prin matricea ponderilor. Se cere să se determine pentru $r \in V$ fixat, lungimea minimă a drumului de la r la oricare $y \in V$.

Algoritmul Dijkstra selectează nodurile grafului, în ordinea crescătoare a costului drumului de la nodul r la ele, unul câte unul, într-o mulțime S, care conține inițial numai nodul r. În procesul de prelucrare, se folosesc 3 vectori, care, tradițional, în literatura de specialitate, se notează: D *(lungimea drumului)*, *T (drumurile identificate)* și S *(nodurile selectate, unul câte unul)*. Astfel, pentru o mai bună înțelege de către studenți a celor menționate mai sus, este necesar să se explice următoarele:

- Vectorul D este vectorul lungimii drumurilor de la vârful r la toate celelalte vârfuri ale grafului. Prin intermediul $D[i], unde\ i \in (1, \dots, v)$, se va indica costul drumului găsit la un moment dat, între vârfuri r și i.
- Vectorul T indică drumurile găsite între noduri r și celelalte noduri ale grafului. Pentru aceasta se utilizează o memorare specială a drumurilor, în care pentru nodul i se reține nodul precedent pe unde trece drumul de la r la i. Pentru r se ia valoarea *0*.
- Vectorul S, indică mulțimea nodurilor selectate: $S[i] = 0$, dacă nodul i este neselectat și $S[i] = 1$ dacă nodul i este selectat. Vectorul S se mai numește și vector caracteristic.

Evidențiem următorii pași care se referă la realizarea algoritmului:

Pasul 1. Inițial mulțimea S se consideră vidă. Nodul r este adăugat mulțimii S și, în felul acesta, avem $S[r] = 1$. Astfel:

1.1. În vectorul ***D***, de pe linia ***r***, a matricei ***A***, se trec toate costurile drumurilor de la vârful ***r*** până la fiecare vârf ***i*** grafului;

1.2. Pentru toate nodurile ***i***, având un cost al drumului de la **r** la ele, finit, se atribuie $T(i) = r$.

Pasul 2. Secvența se execută de ***n-1***, realizând următoarele:

1.3. Se examinează nodurile neselectate. Se găsește nodul aflat la distanța minimă în raport cu ***r***. Acest nod se selectează și se include în mulțimea ***S***. Notăm acest vârf prin intermediul ***poz***.

1.4. Se actualizează în vectorul ***D*** costul drumurilor de la ***r*** până la nodurile neselectate, ***j***, utilizând, în acest scop, ca nod intermediar, nodul selectat, ***poz***, procedând în felul următor:

a) se compară costul existent în vectorul ***D[j]***, cu suma dintre costul existent ***D[poz]*** și nodul selectat ***A[poz, j]***.

b) în cazul în care suma este mai mică, elementul din ***D*** corespunzător nodului pentru care se face actualizarea, ***j***, reține suma calculată conform relației: ***D[j] = D[poz]+A[poz, j]***.

c) simultan, dacă suma calculată este mai mică în comparație cu costul existent, valoarea din ***T*** corespunzător aceluiași vârf ia valoarea vârfului selectat: ***T[j] = poz***. Acest fapt înseamnă că drumul trece prin acest vârf.

Pasul 3. Cu excepția lui ***r***, pentru fiecare vârf al grafului se trasează drumul de la ***r*** la vârful respectiv.

Să examinăm cum se aplică algoritmul respectiv la soluționarea unei probleme concrete.

Problema 1. Determinarea drumurilor de cost minim de la un vârf fixat până la toate celelalte vârfuri.

În orașul Chișinău există una dintre cele mai dinamice companii de transport aerian ”Chișinău Air” din Sud-Estul Europei. Din Chișinău, unde se află sediul central, zboară zilnic avioane cu marfă spre Roma, Londra și Lisabona. Deoarece cheltuielile erau mari, administrația companiei aeriene a decis să reducă din cheltuieli identificând cele mai economice rute din Chișinău către fiecare din celelalte orașe.

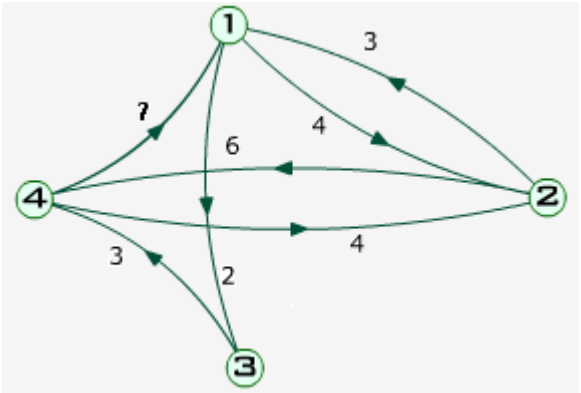


Figura 1. Graful orientat cu 4 noduri și 7 arce

Un absolvent al facultății FMTI de la Universitatea de Stat Tiraspol, Mihai Fluturașu, care a fost angajat recent la companie, a primit drept sarcină să găsească o soluție la problema respectivă. În acest scop, administrația companiei i-a pus la dispoziție o hartă pe care sunt marcate orașele respective Chișinău (1), Roma (2), Londra (3) și Lisabona (4). De asemenea, pe hartă sunt marcate prin săgeţi cursele aeriene existente între oraşe. Sensul săgeţii indică sensul de zbor, iar numărul specificat pe săgeată reprezintă costul zborului.

Tânărul absolvent trebuie să elaboreze un algoritm, care să determine pentru fiecare din orașele Roma (2), Londra (3) și Lisabona (4) costul minim pentru a transporta mărfuri de la sediul central din Chișinău, precum şi o rută de transport pentru care se obţine costul minim.

Soluție. Deoarece Mihai Fluturașu a studiat la universitate cu pasiune Teoria Grafurilor, a elaborat modelul matematic și a aplicat ulterior algoritmul lui Dijkstra ca să determine drumuri de cost minim de la nodul 1 la toate celelalte noduri: 2, 3, și 4. Astfel, Mihai, ca să verifice soluția identificată de el, a propus administrației companiei câteva metode de rezolvare a problemei formulate. Să le urmărim:

Metoda 1. ***Soluția manuală.***

Pentru graful dat a fost construită matricea ponderilor: $A_0 = \begin{pmatrix} 0 & 4 & 2 & \infty \\ 3 & 0 & \infty & 6 \\ \infty & \infty & 0 & 3 \\ 7 & 4 & \infty & 0 \end{pmatrix}$.

Pasul 1. Completăm tabelul (D,S,T) cu valorile inițiale: D – lungimea drumului de la vârful 1 până la 2, 3, 4; S – mulțimea nodurilor selectate *r=1*; T – drumurile găsite de la vârful 1 la celelalte vârfuri (notăm 1 sau 0).

	1	**2**	**3**	**4**
D	0	4	2	∞
S	1	0	0	0
T	0	1	1	0

Pasul 2. Căutăm cel mai apropiat nod de la vârful 1. Este nodul 3 cu $D[3]=2$. Notăm: *poz=3*. Examinăm drumurile de la vârful 1 până la vârfurile *j*=2,4, luând în considerație tabelul precedent prin formulă:$D[j] \leftarrow D[poz] + A[poz, j]$, dacă $D[j] > D[poz] + A[poz, j]$ și $T[j] = poz$. Obținem:

j=2: $D[2] = 4 < D[3] + A[3,2] = 2 + \infty = \infty$
Nu se schimbă nimic.

j=4: $D[4] = \infty > D[3] + A[3,4] = 2 + 3 = 5$
Deci, se modifică $D[4] = 5$ și $T[4] = 3$.

	1	**2**	**3**	**4**
D	0	4	2	5
S	1	0	1	0
T	0	1	1	3

Pasul 3. După nodul 3, cel mai apropiat nod este nodul 2, unde $D[2] = 4$

Notăm: $poz = 2$ și examinăm drumurile până la nodurile $j = 3, 4$.

Obținem:

j=3:	$D[3] = 2 < D[2] + A[2,3] = 4 + \infty = \infty$ Nu se schimbă nimic.
j=4:	$D[4] = 5 < D[2] + A[2,4] = 4 + 6 = 10$ Nu se schimbă nimic

	1	**2**	**3**	**4**
D	0	4	2	5
S	1	1	1	0
T	0	1	1	3

Pasul 4. După nodurile 3 și 2, cel mai apropiat nod de la vârful 1 este nodul 4 cu $D[4] = 5$.

Notăm: $poz = 4$ și examinăm drumurile până la nodurile $j = 2, 3$. Obținem:

j=2:	$D[2] = 4 < D[4] + A[4,2] = 5 + 4 = 9$ Nu se schimbă nimic.
j=3:	$D[3] = 2 < D[4] + A[4,3] = 5 + \infty = \infty$ Nu se schimbă nimic.

Observăm că nu s-a făcut nicio îmbunătățire.

	1	**2**	**3**	**4**
D	0	4	2	5
S	1	1	1	1
T	0	1	1	3

Astfel, am obținut distanțe minime de la vârful 1 până la celelalte vârfuri cu drumurile respective pentru graful inițial.

	Distanța	Drumul
$1 \rightarrow 2$	D[2]=4	Deoarece $T[2] = 1$, obținem $T[1]$=0 și drumul =>$1 \rightarrow 2$.
$1 \rightarrow 3$	D[3]=2	Deoarece $T[3] = 1$, obținem $T[1]$=0 și drumul =>$1 \rightarrow 3$
$1 \rightarrow 4$	D[4]=5	Deoarece $T[4] = 3$, obținem $T[3] = 1, T[1]$=0 și drumul care trece prin vârful 3 este: =>$1 \rightarrow 3 \rightarrow 4$

În aceste condiții putem construi graful îmbunătățit:

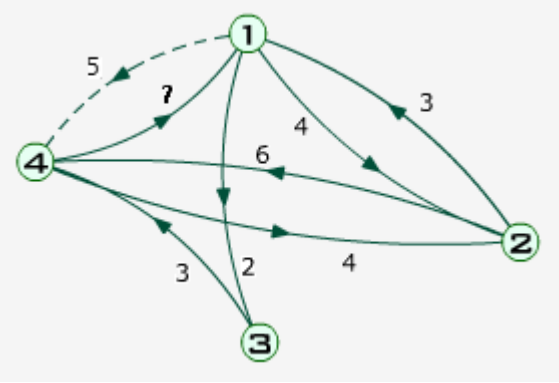


Figura 2. Graful orientat îmbunătățit

Răspuns. *Astfel, costul minim pentru a transporta mărfuri de la sediul central din Chișinău până la Roma (nodul 2) este de 4 (unități), ruta de transport este zborul direct 1→2. De la Chișinău până la Londra (nodul 3) este de 2 (unități), ruta de transport la fel este zborul direct 1→3. Și de la Chișinău până la Lisabona (nodul 4) este de 5(unități), ruta de transport trece prin nodul intermediar 3 (orașul Londra): 1→3→4.*

Metoda 2. ***Rezolvare în aplicația Maple 18:***

Se apelează pachetul **GraphTheory**: *with*(*GraphTheory*)

Se definește graful cu 4 noduri și costurile date:

$G4 := Graph(\{[[1,2],4],[[1,3],2],[[2,1],3],[[2,4],6],[[3,4],3],[[4,1],7],[[4,2],4]\})$

Graph 1: a directed weighted graph with 4 vertices and 7 arc(s)

Se construiește graful definit:

DrawGraph(*G4*);

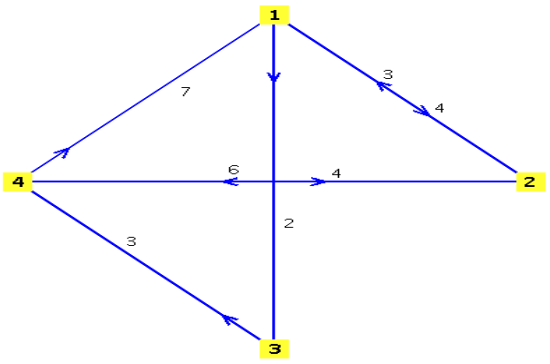


Figura 3. Graful orientat G4

Apelăm procedura **DijkstraAlgorithm** pentru aflarea drumurilor de cost minim de la vârful 1 până la celelalte: *DijkstrasAlgorithm*(*G4*, 1)

$[[[1],0],[[1,2],4],[[1,3],2],[[1,3,4],5]]$

Observăm că obținem aceleași costuri minime:

De la nodul Chișinău la nodul Roma costul este de 4 (unități); de la Chișinău la Londra costul este de 2 (unități); de la Chișinău la Lisabona costul este de 5 (unități).

Apelăm procedura **Distance (G4, s, t)** pentru a returna numărul de muchii din cel mai scurt drum de la nodul *s* la nodul *t:* *Distance*(*G4*, 1, 2) 1

A fost returnat numărul de muchii din cel mai scurt drum de la **1** la **2 care este de 1.** **Altfel spus, este rută directă de la Chișinău la Roma.**

$$Distance(G4, 1, 3)\ 1$$

A fost returnat numărul de muchii din cel mai scurt drum de la 1 la 3 care este de 1. Altfel spus, este rută directă de la Chișinău la Londra.

$$Distance(G4, 1, 4)\ 2$$

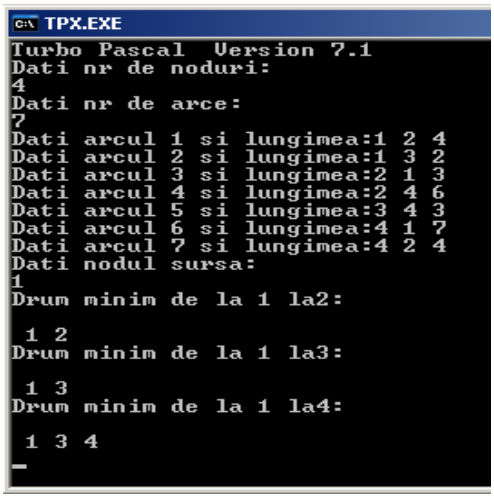
A fost returnat numărul de muchii din cel mai scurt drum de la 1 la 4 care este de 2. Ruta directă de la Chișinău la Lisabona nu există. După cum s-a menționat anterior, la Lisabona se poate de ajuns doar prin orașul intermediar Londra (nodul 3).

Metoda 3. ***Rezolvare în programul Pascal:***

```
Program DDijkstra;
Const nmax=15;
      inf=maxint div 2;
Var c:array[1..nmax,1..nmax] of integer;
    k,i,j,arc,m,n,x,y,z,xp:integer;
    s,d,prec:array[1..nmax] of integer;
    g:boolean;
Procedure Min(var k:integer);
 var m,i:integer;
Begin
     m:=inf*2;
     For i:=1 to n do
        If (s[i]=0) and (d[i]<m) then
        begin
        m:=d[i]; k:=i;
        end;
end;
Procedure Drum(i:integer);
Begin
If i<>0 then
 begin
 drum(prec[i]); write(i:2);
 end else writeln;
end;
Begin
 writeln('Dati nr de noduri:'); readln(n);
 For i:=1 to n do
 For j:=1 to n do c[i,j]:=inf;
 For i:=1 to n do c[i,i]:=0;
 writeln('Dati nr de arce:'); readln(arc);
 For i:=1 to arc do
  begin
   write('Dati arcul ',i,' si lungimea:');
   readln(x,y,z);
   c[x,y]:=z;
  end;
  writeln('Dati nodul sursa:');
  read(xp);
  For i:=1 to n do
  begin
   d[i]:=c[xp,i];
   s[i]:=0;
   If c[xp,i]<inf then prec[i]:=xp else
prec[i]:=0;
  end;
  s[xp]:=1;
  prec[xp]:=0;
  g:=true; x:=0;
  repeat
  min(k);
  x:=x+1;
  If (d[k]=inf) or (x=n)
  then g:=false else
  begin
   s[k]:=1;
   For j:=1 to n do
    If (s[j]=0) and (d[j]>d[k]+c[k,j]) then
    begin
     d[j]:=d[k]+c[k,j];
     prec[j]:=k;
     end;
    end;
   until not(g);
   For i:=1 to n do
    If i<>xp then
     If d[i]=inf then
      begin
       write('Nu exista drum de la ', xp,'
la',i);
       writeln;
       end else
       begin
```

```
      writeln('Drum minim de la ',xp,' la', i,':');
      drum(i);
    writeln;
  end;
  end.
```

Soluțiile obținute pot fi vizualizate mai jos:



Sesizăm că se obțin același rezultate.

3.2. Abordări didactice privind implementarea Algoritmul Roy-Floyd

Algoritmul a fost publicat sub forma recunoscută în prezent de către Robert Floyd în anul 1962. În principiu, este asemănător algoritmului publicat de către Bernard Roy în 1959. Stephen Warshall a avut, de asemenea, o contribuție importantă în crearea acestuia, astfel dezvoltându-se algoritmul Floyd-Warshall.

Mai jos prezentăm algoritmul respectiv.

$$f: V \times V \to R_+,\ f((x,y)) = \begin{cases} c_{x,y} & x \neq y,\ (x,y) \in A \\ \infty & x \neq y,\ (x,y) \notin A \\ 0 & x = y \end{cases}$$ Fie $\boldsymbol{G = (V, A)}$ un graf orientat. Ataşăm fiecărui arc $(\boldsymbol{x, y}) \in \boldsymbol{A}$ o pondere (un cost): $c_{x,y} > 0$. Definim pe mulţimea $V \times V$ o funcţie, astfel:

Funcţia este reţinută de o matrice, numită matricea ponderilor forma 1. Să examinăm următoarea problemă generală.

Problema. *Fiind dat un graf orientat $G = (V, A)$, memorat prin matricea ponderilor în forma 1, se cere să se determine, pentru orice două noduri* ***x,y*** *din* $\boldsymbol{V}$, *lungimea minimă a drumului de la nodul x la nodul y. Prin lungimea unui drum înţelegem suma costurilor arcelor care-l alcătuiesc.*

Soluție: Pentru a rezolva problema respectivă vom lua în considerație următoarele:

- *În matricea ponderilor vor fi incluse numai lungimea drumurilor directe între două noduri. Nu este permis ca un drum între două noduri să treacă printr-un alt nod.*
- *Pentru arce inexistente se reţine valoarea, $+\infty$, (o valoare considerată foarte mare).*
- *Matricea inițială a ponderilor în forma 1 o vom nota prin* $\boldsymbol{A_0}$.

Să descriem procesul de implementare al algoritmul **Roy-Floyd**.

Scopul algoritmului este să construim matricele ponderilor A_0, A_1, A_2, ... , $\boldsymbol{A_n}$.

După fiecare pas, matricele vor reţine lungimea optimă a drumurilor între oricare două noduri, drumuri care pot trece prin nodurile intermediare 1, 2, ..., n. Algoritmul continuă în acest mod, prin eventuale îmbunătăţiri succesive ale lungimii. În final, Matricea $\boldsymbol{A_n}$ va conține costul drumurilor optime între oricare două vârfuri. Pentru a obține acest rezultat vom aplica relația:

$$\boldsymbol{A_k(i,j)=min(A_{k-1}(i,j),A_{k-1}(i,k)+A_{k-1}(k,j)),}$$

$k = 1,2, \dots, n$, unde n – dimensiunea matricei, şi oricare două vârfuri $i, j \in V$.

Pasul 1. La început, încercăm să obţinem drumuri mai scurte, între oricare două vârfuri $i, j \in V$, permiţând ca acestea să poată trece prin nodul 1. Aceasta înseamnă că pentru $\forall i, j \in V$ se face comparaţia: $A(i, j) > A(i, 1) + A(1, j)$, adică se compară dacă lungimea drumului direct (care nu trece prin alte noduri) este mai mare decât cea a drumului care trece prin nodul 1.

Astfel, pentru nodul intermediar *k=1*, examinăm toate drumurile de tipul: *i→1→j*. Efectuând operațiile: $\boldsymbol{A_1(i,j)=min(A_0(i,j),A_0(i,1)+A_0(1,j)),}$ construim matricea nouă $\boldsymbol{A_1}$.

După acest pas, matricea va reţine lungimea optimă a drumurilor între oricare două noduri, drumuri care pot trece prin nodul l.

Pasul 2. În continuare vom obține drumuri mai scurte, între oricare două noduri $i, j \in V$, permiţând ca acestea să poată trece şi prin nodul intermediar 2. Astfel, pentru nodul intermediar *k=2*, identificăm drumurile de tipul*: i→2→j, și* examinăm relația:

$$\boldsymbol{A_2(i,j)=min(A_1(i,j),A_1(i,2)+A_1(2,j)),}$$

În așa mod vom construi matricea nouă $\boldsymbol{A_2}$.

După acest pas, matricea va reţine lungimea optimă a drumurilor între oricare două noduri, drumuri care pot trece prin nodurile intermediare 1 și 2. Algoritmul continuă în acest mod, prin eventuale îmbunătăţiri succesive ale lungimii.

Pasul 3. Evident, după $\boldsymbol{n}$ paşi, matricea ponderilor va reţine costurile drumurilor optime de la i la j, $i, j \in \{1, 2, 3, \dots, n\}$, drumuri care pot trece prin n noduri, adică, va reţine drumurile optime pe ansamblul grafului orientat.

Notă. *Astfel, aplicând algoritmul descris mai sus, aflăm costul drumurilor optime între oricare două vârfuri. Dar, în contextul respectiv, este necesar să cunoaştem şi pe unde trece un astfel de drum. Astfel, se pune următoarea problemă: fiind dată matricea drumurilor optime şi fiind date două vârfuri i şi j, se cere să se reconstituie nodurile prin care trece unul din drumurile optime între i şi j (pot exista mai multe de aceeaşi lungime, minimă).*

Dacă drumul optim între i şi j trece prin nodul intermediar k, atunci:

$$A(i, j) = A(i, k + 1) + A(k + 1, j)$$

Dacă există vârful k, astfel încât $A(i, j) = A(i, k + 1) + A(k + 1, j)$, atunci k se găseşte pe unul din drumurile optime de la i la j.

Să examinăm următoarea problemă concretă.

<u>Problema 2.</u> Determinarea drumurilor de cost minim între oricare două vârfuri.

Mihai Fluturaşu, absolventul facultăţii FMTI de la Universitatea de Stat Tiraspol, este cunoscut de către companiile mari de transport din Chişinău ca un bun informatician, care poate determina cu exactitate costul minim al rutelor pentru a transporta mărfurile. Graţie acestei reputaţii, Mihai a fost angajat la aeroportul din Chişinău. Lui Mihai i s-a pus la dispoziţie o hartă pe care sunt marcate aeroporturile din oraşele Chişinău (1), Roma (2), Londra (3) şi Lisabona (4). De asemenea pe hartă sunt marcate prin săgeţi cursele aeriene existente între oraşe. Sensul săgeţii indică sensul de zbor, iar numărul specificat pe săgeată reprezintă costul zborului.

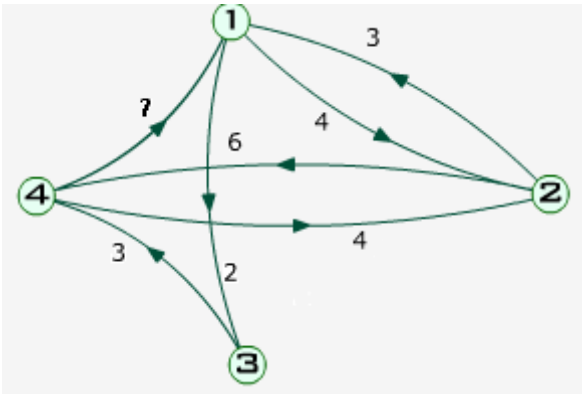


Figura 4. Graful orientat cu 4 noduri şi 7 arce

Tânărului absolvent i s-a cerut să elaboreze un algoritm de găsire a drumurilor de cost minim între oricare două aeroporturi menţionate mai sus, precum şi rutele de transport pentru care se obţine costul minim.

Soluţie. Tânărul absolvent a elaborat modelul matematic şi ulterior a aplicat algoritmul lui **Roy-Floyd** ca să determine drumurile de cost minim intre oarecare două noduri. Ca să verifice soluţia identificată, Mihai a propus administraţiei aeroportului câteva metode de rezolvare a problemei formulate. Să le examinăm:

Metoda 1. ***Soluţia manuală***

Alcătuim matricea ponderilor de forma 1, conform relaţiei:

$$A_0 = \begin{pmatrix} 0 & 4 & 2 & \infty \\ 3 & 0 & \infty & 6 \\ \infty & \infty & 0 & 3 \\ 7 & 4 & \infty & 0 \end{pmatrix}.$$

Construim matricea ulterioară, aplicând relaţia:

$A_k(i,j)=\min(A_{k-1}(i,j),A_{k-1}(i,k)+A_{k-1}(k,j))$, $k = 1,2,\dots,n$, unde n – dimensiunea matricei.

Vom examina toate drumurile care trec prin nodurile intermediare: k=1,2,3,4.

<u>Pasul 1.</u> Fie nodul intermediar $k=1$.

Examinăm toate drumurile de tipul: $i \to 1 \to j$.

Astfel avem : $2 \to 1 \to 3$, $4 \to 1 \to 2$, $4 \to 1 \to 3$.

$a_1(2,3)=\min(a_0(2,3), a_0(2,1)+ a_0(1,3))=\min(\infty,3+2)=5 \Rightarrow a(2,3)=5$.

$a_1(4,2)=\min(a_0(4,2), a_0(4,1)+ a_0(1,2))=\min(4,7+4)=4$. Nu se schimbă nimic.

$a_1(4,3)=\min(a_0(4,3), a_0(4,1)+ a_0(1,3))=\min(\infty,7+2)=9$. Deci, $a_1(4,3)=9$.

Am obținut matricea $A_1 = \begin{pmatrix} 0 & 4 & 2 & \infty \\ 3 & 0 & 5 & 6 \\ \infty & \infty & 0 & 3 \\ 7 & 4 & 9 & 0 \end{pmatrix}$.

Pasul 2. Considerăm nodul intermediar $k=2$.

Identificăm drumurile de tipul: $i \to 2 \to j$.

Astfel, vom examina drumurile: $1 \to 2 \to 4$; $4 \to 2 \to 1$; $1 \to 2 \to 3$; $4 \to 2 \to 3$.

$a_2(1,4)=\min(a_1(1,4), a_1(1,2)+ a_1(2,4))=\min(\infty,4+6)=10$. Deci, $a_2(1,4)=10$.

$a_2(4,1)=\min(a_1(4,1), a_1(4,2)+ a_1(2,1))=\min(7,7)=7$. Nu se schimbă nimic.

$a_2(1,3)=\min(a_1(1,3), a_1(1,2)+ a_1(2,3))=\min(2,4+5)=2$. Nu se schimbă nimic.

$a_2(4,3)=\min(a_1(4,3), a_1(4,2)+ a_1(2,3))=\min(9,4+5)=9$. Nu se schimbă nimic.

Obținem matricea $A_2 = \begin{pmatrix} 0 & 4 & 2 & 10 \\ 3 & 0 & 5 & 6 \\ \infty & \infty & 0 & 3 \\ 7 & 4 & 9 & 0 \end{pmatrix}$.

Pasul 3. Co nsiderăm nodul intermediar k=3.

Găsim drumurile care trec prin vârful 3: $i \to 3 \to j$.

Astfel, avem drumurile: $1 \to 3 \to 4$ și $2 \to 3 \to 4$.

Deci, $a_3(1,4)=\min(a_2(1,4), a_2(1,3)+ a_2(3,4))=\min(10,2+3)=5$. Deci $a_3(1,4)=5$.

$a_3(2,4)=\min(a_2(2,4), a_2(2,3)+ a_2(3,4))=\min(6,5+3)=6$. Nu se schimbă nimic.

Obținem matricea $A_3 = \begin{pmatrix} 0 & 4 & 2 & 5 \\ 3 & 0 & 5 & 6 \\ \infty & \infty & 0 & 3 \\ 7 & 4 & 9 & 0 \end{pmatrix}$.

Pasul 4. Considerăm nodul intermediar k=4.

Examinăm drumurile care trec prin vârful 4: $i \to 4 \to j$.

Astfel, vom cerceta: $2 \to 4 \to 3$; $1 \to 4 \to 3$; $3 \to 4 \to 2$; $3 \to 4 \to 1$.

În acest caz, avem:

$a_4(3,2)=\min(a_3(3,2), a_3(3,4)+ a_3(4,2))=\min(\infty,3+4)=7$; $a_4(3,2)=7$.

$a_4(3,1)=\min(a_3(3,1), a_3(3,4)+ a_3(4,1))=\min((\infty,3+7)=10$; $a_4(3,1)=10$.

$a_4(1,3)=\min(a_3(1,3), a_3(1,4)+ a_3(4,3))=\min((2,2+9)=2$; Nu se îmbunătățește nimic.

$a_4(2,3)=\min(a_3(2,3), a_3(2,4)+ a_3(4,3))=\min((5,6+9)=5$; Nu se îmbunătățește nimic.

Obținem matricea $A_4 = \begin{pmatrix} 0 & 4 & 2 & 5 \\ 3 & 0 & 5 & 6 \\ 10 & 7 & 0 & 3 \\ 7 & 4 & 9 & 0 \end{pmatrix}$.

În baza matricei A_4 construim graful îmbunătățit:

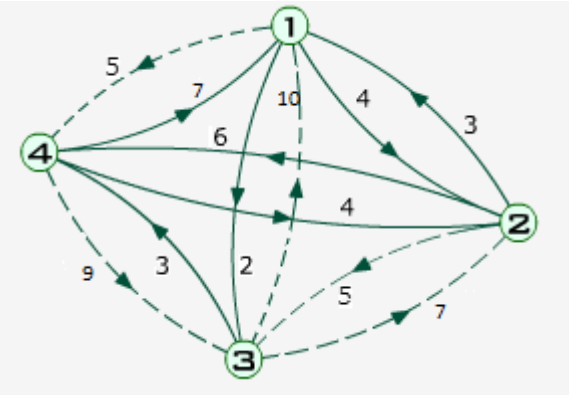


Figura 5. Graful orientat îmbunătățit

Răspuns. Luând în considerare matricea A_4 și graful îmbunătățit, putem scrie următoarele răspunsuri privind drumurilor de cost minim între oricare două aeroporturi menționate mai sus, precum şi rutele de transport pentru care se obţine costul minim.

	Chișinău (1)	**Roma (2)**	**Londra (3)**	**Lisabona (4)**
Chișinău (1)	0	Cost: 4 (unități) Drumul: *1→2*	Cost: 2 (unități) Drumul: *1→3*	Cost: 5 (unități) Drumul: *1→3→4*
Roma (2)	Cost: 3 (unități) Drumul: *1→2*	0	Cost: 5 (unități) Drumul: *2→1→3*	Cost: 6 (unități) Drumul: 2→*4*
Londra (3)	Cost: 10 (unități) Drumul: *3→4→1*	Cost: 7 (unități) Drumul: *3→4→2*	0	Cost: 3 (unități) Drumul: 3→*4*
Lisabona (4)	Cost: 7 (unități) Drumul: 4→*1*	Cost: 4 (unități) Drumul: 4→*2*	Cost: 9 (unități) Drumul: *4→1→3*	0

Observație. *Astfel, costurile minime pentru a transporta mărfuri de la aeroportul din Chișinău până la Roma (nodul 2) este 4 (unități), Londra (nodul 3) este 2 (unități), Lisabona (nodul 4) este 5(unități), coincid cu răspunsurile obținute la soluționarea Problemei 1.*

Metoda 2. ***Soluționarea problemei 2 în Maple 18***

Se apelează la pachetul **GraphTheory**: *with*(*GraphTheory*)

Se definește graful:

```
G4 := Graph({[[1, 2], 4], [[1, 3], 2], [[2, 1], 3], [[2, 4], 6], [[3, 4], 3], [[4, 1], 7], [[4, 2],
    4]})
```

Graph 1: a directed weighted graph with 4 vertices and 7 arc(s)

Se construiește graful definit cu ajutorul comenzei **DrawGraph**: *DrawGraph*(*G4*);

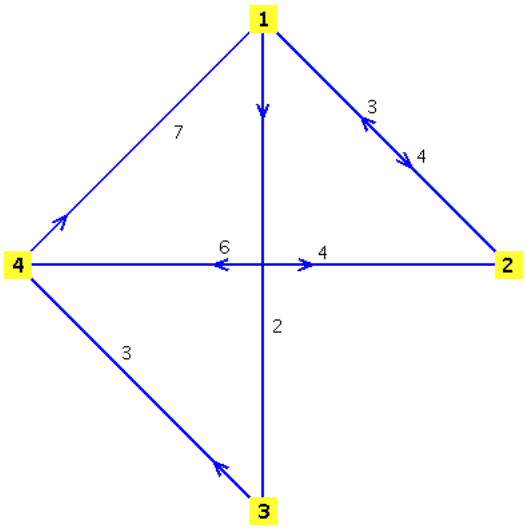


Figura 5. Graful orientat G4

Se aplică comanda **AllPairsDistance** pentru a obține valorile costurilor a tuturor drumurilor: *AllPairsDistance*(*G4*)

$$\begin{bmatrix} 0 & 4 & 2 & 5 \\ 3 & 0 & 5 & 6 \\ 10 & 7 & 0 & 3 \\ 7 & 4 & 9 & 0 \end{bmatrix}$$

Se construiește graful pentru o nouă matrice obținută: *DrawGraph*(*G4*);

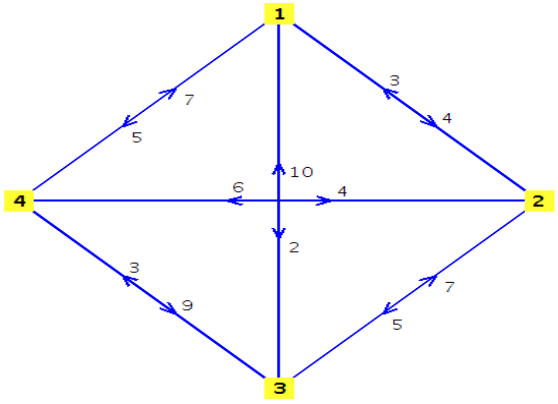


Figura 6. Graful orientat G4 îmbunătățit

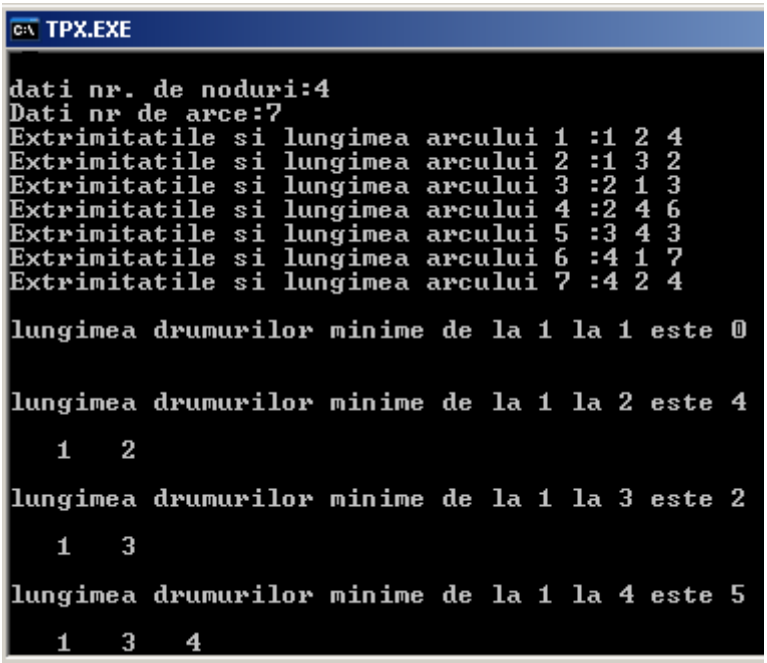
Metoda 3. ***Rezolvare în programul Pascal:***

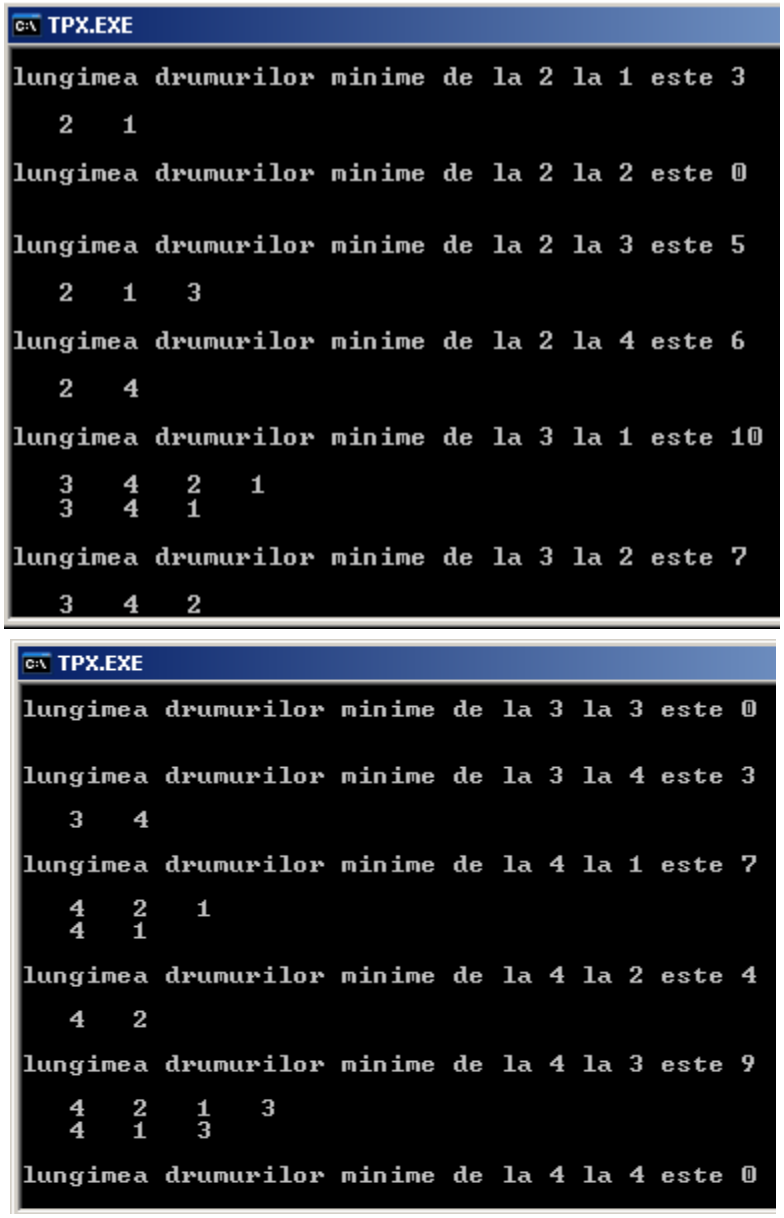
```
Program RoyMin;
Const nmax= 20;
     inf=maxint div 2;
Type Multime=Set of 1..nmax;
Var c:array[1..nmax,1..nmax] of word;
   {c-initial matricea drumurilor}
   d:array[1..nmax,1..nmax] of Multime;
   dr:array[1..nmax] of
1..nmax;n,m,i,j,k,lg:word;

Procedure Initc;{Inițializarea matricei
costurilor C}
 Var i,j,x,y,z:word;
```

```
begin
 write('dati nr. de noduri:'); readln(n);
 For i:=1 to n do
 begin
  for j:=1 to n do  c[i,j]:=inf;
 c[i,i]:=0;
 end;
 write('Dati nr de arce:'); readln(m);
 For i:=1 to m do
 begin
  write('Extrimitatile si lungimea arcului
',i,' :');
  readln(x,y,z); c[x,y]:=z;
```

```
  end; end;
Procedure Initd; {Inițializarea matricei
D}
  Var i,j:word;
  begin
   For i:=1 to n do
   For j:=1 to n do
    If (c[i,j]<inf) and (i<>j) then d[i,j]:=[i]
       else d[i,j]:=[];
    end;
Procedure Drum(i,j:integer);
 {Genereaza in vectorul dr un drum
minim de la i la j pornind de la nodul j}
 Var k:word;
 Begin
  If i<>j then
  begin
  for k:=1 to n do
   if k in d[i,j] then
   begin
    lg:=lg+1;     dr[lg]:=k;      drum(i,k);
    lg:=lg-1;
   end; end else
   begin
    writeln;
   for k:=lg downto 1 do write(dr[k]:4);
    end; end;
    Procedure Afisare; {Afisarea
rezultatelor}
    Var i,j:word;
    Begin
     for i:=1 to n do
      for j:=1 to n do
       begin
        writeln;
   if c[i,j]=inf then
    writeln('Nu exista drum intre ',i,' si ', j)
  else       begin
  writeln('Lungimea drumurilor minime
de la ',i,' la ',j,' este ',c[i,j]);
        if i<>j then begin
         lg:=1;      dr[1]:=j;       drum(i,j);
         end; readln;
  end; end;
end;
Begin
  initc;
  initd;
  for k:=1 to n do
   for i:=1 to n do
    for j:=1 to n do
     if c[i,j]>c[i,k]+c[k,j] then
     begin
      c[i,j]:=c[i,k]+c[k,j];
     d[i,j]:=d[k,j];
     end else
      if c[i,j]=c[i,k]+c[k,j] then
d[i,j]:=d[i,j]+d[k,j];
      afisare;
End.
```

Soluțiile obținute pot fi vizualizate mai jos:





Observăm că obținem aceleași rezultate.

4. Concluzii și sinteze

Astfel, mai întâi de toate, algoritmii Dijkstra și Roy-Floyd trebuie să fie explicați și demonstrați teoretic, ulterior, se recomandă să fie aplicați la soluționarea problemelor concrete, care presupune elaborarea modelului matematic, în conformitate cu etapele descrise mai jos. În acest sens, se formulează o problemă practică, unde se cere determinat drumul minim de la nodul sursă până la orice alt nod al grafului examinat ori drumul minim între oricare două noduri ale grafului. În acest scop, studenții trebuie, întâi de toate, să alcătuiască modelul matematic (matricea ponderilor), apoi să aplice algoritmul Dijkstra ori Roy-Floyd, folosind următoarele instrumente didactice:

1) Implementarea manuală a algoritmului. În acest sens, se completează tabelul (D,S,T) și sunt realizate etapele după algoritmului Dijkstra ori se efectuează pașii corespunzători conform algoritmului Roy-Floyd. Studenții, în cooperare cu profesorul, clarifică fiecare etapă de implementare a algoritmului pentru a înțelege

procedeele de parcurgere a fiecărui pas. Obținând soluțiile problemei, este necesar de analizat și de verificat rezultatele obținute. În contextul respectiv, se poate trece la implementarea următoarei etape.

2) Verificarea rezultatelor obținute prin intermediul softului MAPLE. În felul acesta studenții pot verifica și analiza corectitudinea rezultatelor obținute. După ce studenții se conving de faptul că rezultatele sunt corecte, se poate trece la etapa elaborării unui program Dijkstra și Roy-Floyd.
3) Elaborarea unui program pentru implementarea algoritmilor Dijkstra și Roy-Floyd este o etapă esențială în fundamentarea cunoștințelor teoretice și practice. Profesorul, în cooperare cu studenții, făcând referință la etapele parcurse anterior, alcătuiesc programul respectiv, care, fiind executat, va livra aceleași rezultate.

Astfel, în urma experimentului desfășurat, urmând abordările didactice descrise mai sus, s-a demonstrat: capacitatea de înțelegere și asimilare a materialului prezentat este cu mult mai profundă comparativ cu metodele tradiționale utilizate.

5. Probleme propuse pentru rezolvare

Problema 1. În imperiul ”Țară fără drumuri” există 5 orașe, așa cum se vede în Figura 7; între unele orașe există un drum direct și din orice oraș se poate ajunge în toate celelalte. Noul împărat vrea să își instaleze curtea în unul dintre orașe, astfel încât să poată ajunge la fiecare dintre celelalte pe drumurile cele mai scurte. El vrea ca aceste drumuri să fie pietruite și, bineînțeles, costul total al acestei lucrări sa fie minim. Ajutați împăratul să poată determina costul minim pentru pietruire între oricare două orașe.

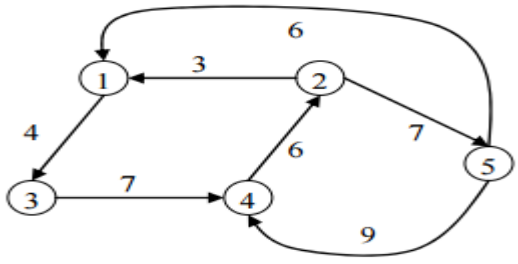


Figura 7. Graful drumurilor din imperiu ”Țară fără drumuri”

Problema 2. Între 4 stații spațiale se stabilesc trasee, după se vede în Figura 8. Se cunosc traseele directe între unele stații și distanța dintre ele. Se cere să se determine drumul minim între oricare două stații din figură. Să se afișeze drumul intre stații.

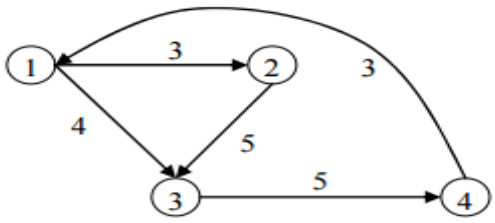


Figura 8. Graful traseelor intre stațiile spațiale

Problema 3. În orașul Chișinău există una dintre cele mai dinamice companii de transport aerian ”Chișinău Air” din Sud-Estul Europei. Din Chișinău, unde se află sediul central,

zboară zilnic avioane cu marfă în Berlin, Londra și Madrid. Deoarece cheltuielile erau mari, administrația companiei aeriene a decis să reducă din cheltuieli, identificând cele mai economice rute din Chișinău către fiecare din celelalte orașe. Un absolvent al facultății FMTI de la Universitatea de Stat Tiraspol, care a fost angajat recent la companie, a primit drept sarcină să găsească o soluție la problema respectivă. În acest scop, administrația companiei i-a pus la dispoziție matricea ponderilor A_0, unde liniile/coloanele 1, 2, 3 și 4 reprezintă respectiv Chișinău (1), Berlin (2), Londra (3) și Madrid (4).

0	10	∞	∞
8	0	5	9
∞	∞	0	3
8	6	∞	0

Cerință: Tânărul absolvent trebuie să elaboreze un algoritm care să determine pentru fiecare din orașele Berlin (2), Londra (3) și Madrid (4) costul minim pentru a transporta mărfuri de la sediul central din Chișinău, precum şi o rută de transport pentru care se obţine costul minim. Să se construiască graful drumurilor.

Bibliografie

1. Tomescu I. Combinatorică şi teoria grafurilor, Ed. Universității din Bucureşti, 1990.
2. Bang-Jensen, Gutin G. Digraphs Theory. Algorithms and Applications. Springer-Verlag, 2007.
3. Bomdy J.A., Murty U.S.R. Graph Theory, Springer, 2007.
4. Ore O. Theory of graphs. In: American Mathematical Society Colloquium Publications, Vol. XXXVIII, American Mathematical Society, 1962. 270 p.
5. Хаггарти Р. Дискретная математика для программистов. Москва: Техносфера, 2003. 320 с., ISBN 5-94836-016-4.
6. Proiectarea algoritmilor 2011-2012. Drumuri minime. [accesat 15.02.2016], http://andrei.clubcisco.ro/cursuri/f/f-sym/2pa/labs2012/Lab%208%20Drumuri%20minime.pdf
7. Computing Science & Discrete Math. Алгоритмы поиска кратчайших путей [vizitat 22.02.2016] http://kvodo.ru/category/algorithms/graph.
8. Drumuri minime în graf [vizitat 23.02.2016]. http://campion.edu.ro/arhiva/www/arhiva_2009/seds/7/index.htm
9. Bârză S., Morgan L-M. Algoritmica grafurilor. București: Ed. Fundației România de Mâine, 2008. 148p. http://www.ocpiilfov.ro/ocpi_ilfov/Man.pdf. [vizitat 15.11.2016] ISBN 978-973-163-147-9.
10. http://www.maplesoft.com/products/maple/new_features/maple18/Graph_Theory.aspx [vizitat 05.09.2016].
11. https://www.embarcadero.com/products/delphi [vizitat 20.03.2017].
12. http://ccm.net/download/download-24152-turbo-pascal [vizitat 20.03.2017].