CZU: 519.17+004:378.147

METODOLOGIA STUDIERII ALGORITMILOR DE PARCURGERE ÎN GRAFURI

Marina BOSTAN, lector asistent
Universitatea de Stat din Tiraspol/Universitatea Pedagogică de Stat „Ion Creangă”
https://orcid.org/0000-0002-1191-9501

Rezumat. În articol este examinată metodologia studierii parcurgerii grafurilor în sistemul universitar. O parcurgere este o tranziție simplă de la vârf la vârf pentru a găsi proprietățile legăturilor (conexiunilor) dintre aceste vârfuri. Există doi algoritmi de parcurgere a grafului - căutarea în adâncime (DFS) și căutarea în lățime (BFS). Pentru studierea mai eficientă a acestor algoritmi sunt examinate parcurgerile respective din mai multe perspective.
Cuvinte cheie: teoria grafurilor, algoritmi de parcurgere, căutaree în lățime, căutare în adâncime.

METHODOLOGY OF THE STUDY OF GRAPH TRAVERSAL ALGORITHMS

Abstract. In the article it examined the methodology of studying graphs in the university system of studying. A traversal is a simple transition from vertex to vertex to find the properties of the links (connections) between these vertices. There are two graph traversal algorithms - depth-first search (DFS) and breadth-first search (BFS). For the more efficient study of these algorithms, the respective paths are examined from several perspectives.
Keywords: graph theory, graph traversal, Breadth first search (BFS), Depth first search (DFS).

1. De ce sunt necesare parcurgerile în grafuri?

Rezolvarea multor probleme care țin de grafuri, presupun cercetarea legăturilor între noduri prin parcurgerea lor. Parcurgerea grafurilor poate fi aplicată în procesul de: verificarea conexității unui graf, determinarea componentelor conexe ale unui graf, determinarea unor lanțuri într-un graf, verificarea dacă graful este bipartit, etc.

Cei mai cunoscuți dintre acești algoritmi sunt *căutarea în adâncime* (Depth first search - DFS) și *căutarea în lățime* (Breadth first search – BFS), care se consideră fundamentali pentru elaborarea altor algoritmi ce se aplică la soluționarea problemelor practice.

Ideea cheie a parcurgerii unui graf este de a eticheta fiecare vârf prima dată când este vizitat și de a stoca informații despre acele vârfuri în care nu au fost vizitate toate muchiile.

În procesul de parcurgere a grafului, fiecare dintre vârfuri acestuia se va afla în una dintre cele trei stări:

- *Nedeschis* – starea inițială a vârfului,
- *Deschis* – se găsește vârful, dar marginile incidente cu acesta nu sunt analizate,
- *Procesat(Etichetat)* – sunt vizitate toate muchiile incidente cu acest vârf.

Este clar că fiecare vârf al grafului acceptă secvențial toate aceste stări. Inițial, doar un singur vârf este deschis – începutul parcurgerii grafului.

Notă: metodele propuse parcurg toate vârfurile conținute în aceeași componentă de conectivitate cu vârful inițial. Prin urmare, oricare dintre algoritmii de parcurgere poate fi utilizat pentru a determina conectivitatea(conexitatea) unui graf.

Parcurgerea în lățime a fost inventată de Konrad Zuse în 1945 în teza de doctor care a fost respinsă, totuși aceasta lucrare nu a fost publicată până 1972.(however, this paper was not published till 1972). În 1959 algoritmul de parcurgere a grafurilor în lățime a fot reinventat de către profesorul american de matematică și informatică Edward Forrest Moore. L-a folosit pentru a găsi cea mai scurtă cale de ieșire dintr-un labirint.

Parcurgerea în adâncimea fost folosită pe scară largă începând cu sfârșirul anului 1950, în special în programele din domeniul intelegenței artificiale. Robert Endre Tarjan, informatician și mathematician American, a descoperit mai mulți algoritmi de parcurgere, a elaborat un algoritm liniar pentru determinarea componentelor tare conexe și a fost primul care a propus un algoritm liniar pentru sortarea topologică.

Algoritmii de parcurgere a grafurilor în lățime și adâncime permit realizarea traversării a unui graf neorientat sau orientat.

2. Algoritmul parcurgerii grafurilor în lățime (BFS)

Această metodă se bazează pe următoarea tehnică. Fie un graf $G = (X, U)$ cu ***n*** noduri şi un nod de plecare ***ns*** numit şi nod sursă - căutarea în lățime explorează sistematic muchiile grafului G pentru a „descoperi” fiecare nod accesibil din ***ns***.

Algoritmul calculează distanţa (cel mai mic număr de muchii) de la ***ns*** la toate vârfurile accesibile lui. El produce un „arbore de lăţime” cu rădăcina în ***ns***, care conţine toate nodurile accesibile.

Pentru fiecare nod **v** accesibil din **ns**, calea din arborele de lăţime de la **ns** la **v** corespunde *"celui mai scurt drum"* de la **ns** la **v**, adică conţine un număr minim de muchii.

Parcurgerea în lățime se face astfel:

- se parcurge vârful de start;
- ulterior, se parcurg vecinii vârfului de start în ordine crescătoare;
- apoi vecinii nevizitați ai acestora în ordinea crescătoare etc.;
- până când sunt vizitate toate vârfurile accesibile;

Practic, pentru a stabili ordinea de vizitare se folosește o coadă, iar pentru a stabili dacă un vârf a fost sau nu vizitat se foloseşte un vector caracteristic.

Algoritmul este:

- adăugăm în coadă vârful inițial și îl vizităm;
- cât timp coada este nevidă;
- extragem un element din coadă;
- determinăm vecinii nevizitați ai vârfului extras, îi vizităm și îi adăugăm în coadă;
- eliminăm elementul din coadă.

Observație. Dacă graful nu este conex, în urma parcurgerii nu se vor vizita toate vârfurile. [1]

Exemplu 1. Să se realizeze parcurgerea în lățime (BFS) pentru graful neorientat reprezentat mai jos începând cu nodul 7.

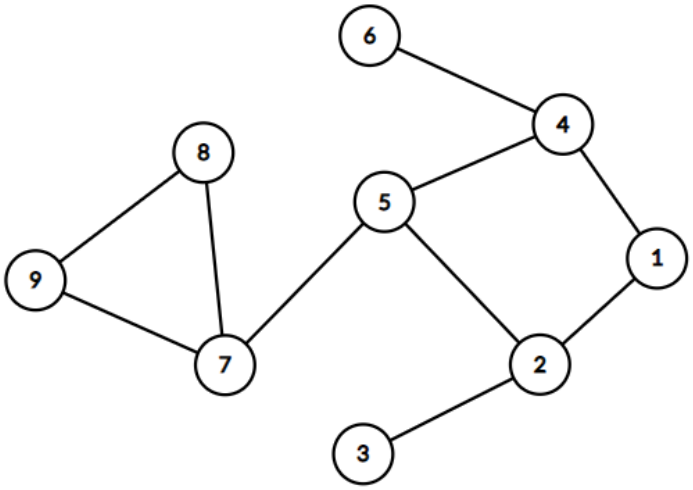

Figura 1. Graful neorientat cu 9 noduri și 10 muchii

Vârfurile grafului au fost parcurse în ordinea: **7, 5, 8, 9, 2, 4, 1, 3, 6**.

În urma parcurgerii în lățime, muchiile folosite pentru parcurgere formează un arbore. Acest arbore este graf parțial al grafului dat (dacă graful este conex), și se numește **arbore parțial de parcurgere**. Pentru graful de mai sus, arborele de parcurgere pornind din vârful **7** este:

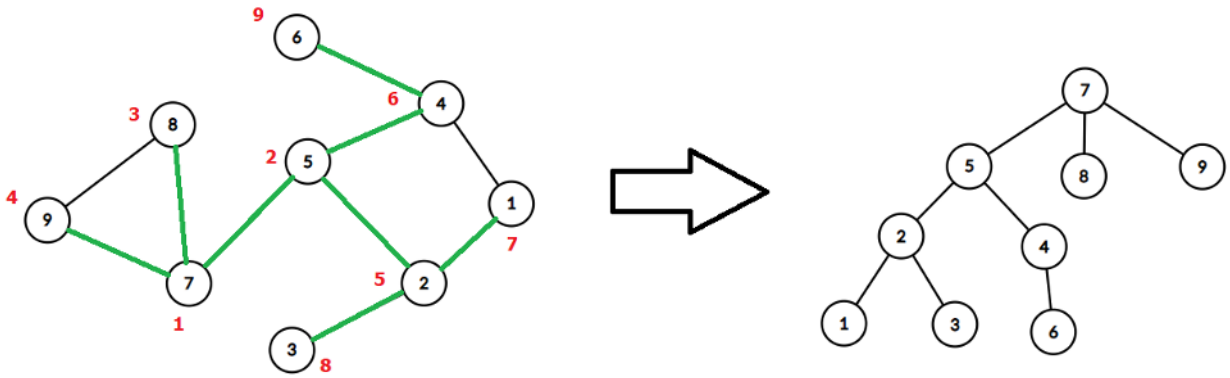

Figura 2. Parcurgerea grafului în lățime

3. Algoritmul parcurgerii grafurilor în adâncime (DFS)

Parcurgerea în adâncime reprezintă explorarea "naturală" a unui graf neorientat. Este foarte asemănătoare cu modul în care un turist vizitează un oraș în care sunt obiective turistice (vârfurile grafului) și căi de acces între obiective (muchiile). Vizitarea orașului va avea loc din aproape în aproape: se pleacă de la un obiectiv de pornire, se continuă cu un obiectiv învecinat cu acesta, apoi unul învecinat cu al doilea, etc.

Parcurgerea în adâncime se face astfel:

- Se începe cu un vârf inițial x, care este în acest moment vârf curent;
- Vârful x se vizitează. Se determină primul său vecin nevizitat y (cel mai mic după numărul de ordine) al lui x, care devine vârf curent;
- Apoi se vizitează primul vecin nevizitat al lui y, şi aşa mai departe, mergând în adâncime, până când ajungem la un vârf care nu mai are vecini nevizitați. Când ajungem într-un astfel de vârf, ne întoarcem la „părintele” acestuia – vârful din care am ajuns în acesta;
- Dacă acest vârf mai are vecini nevizitați, alegem următorul vecin nevizitat al său și continuam parcurgerea în același mod;
- Dacă nici acest vârf nu mai are vecini nevizitați, revenim în vârful său părinte și continuăm în același mod, până când toate vârfurile accesibile din vârful de start sunt vizitate [1].

Exemplu 2. Să se realizeze parcurgerea în adâncime (DFS) pentru grafului neorientat din Figura 1începând cu nodul 7.

Vârfurile grafului au fost parcurse în ordinea: **7, 5, 2, 1, 4, 6, 3, 8, 9**.

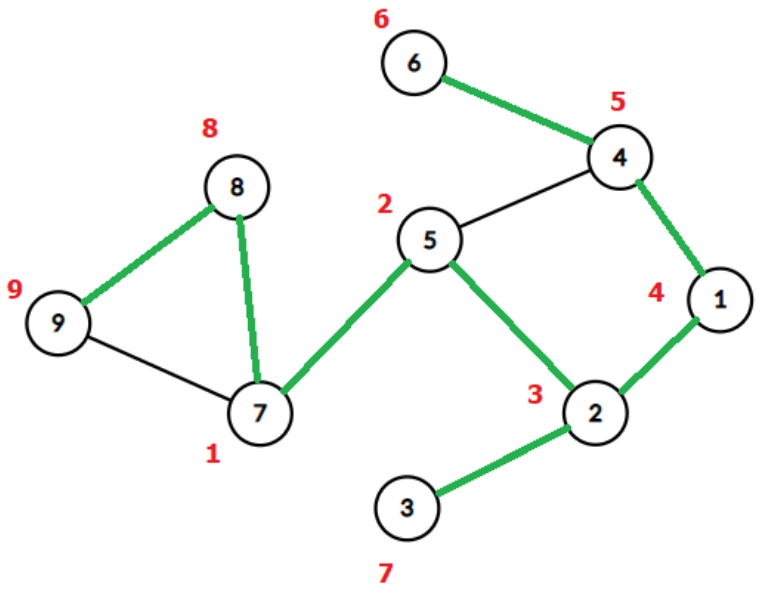

Figura 3. Parcurgerea grafului în adâncime

Pentru înțelegerea mai bună a algoritmilor propunem să exerseze în mod interactiv utilizând aplicațiile online.

4. Algoritmii DFS și BFS în aplicații online

4.1. *Platforma campion.edu.ro – Teoria grafurilor*

Platforma campion.edu.ro include nu doar surse teoretice dar și aplicații interactive pentru exersare. Interfața aplicației pentru parcurgerea grafurilor neorientate este următoare:

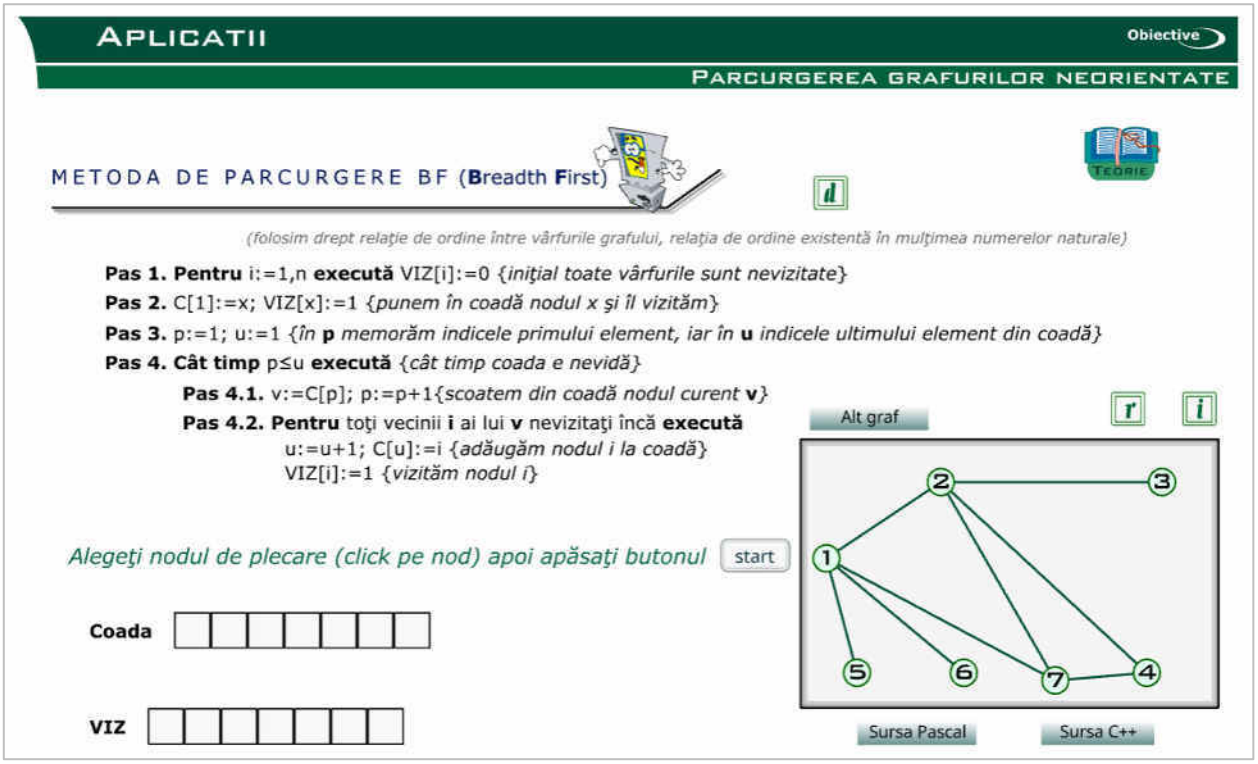

Figura 4. Interfața aplicației Parcurgerea grafului neorientat în lățime

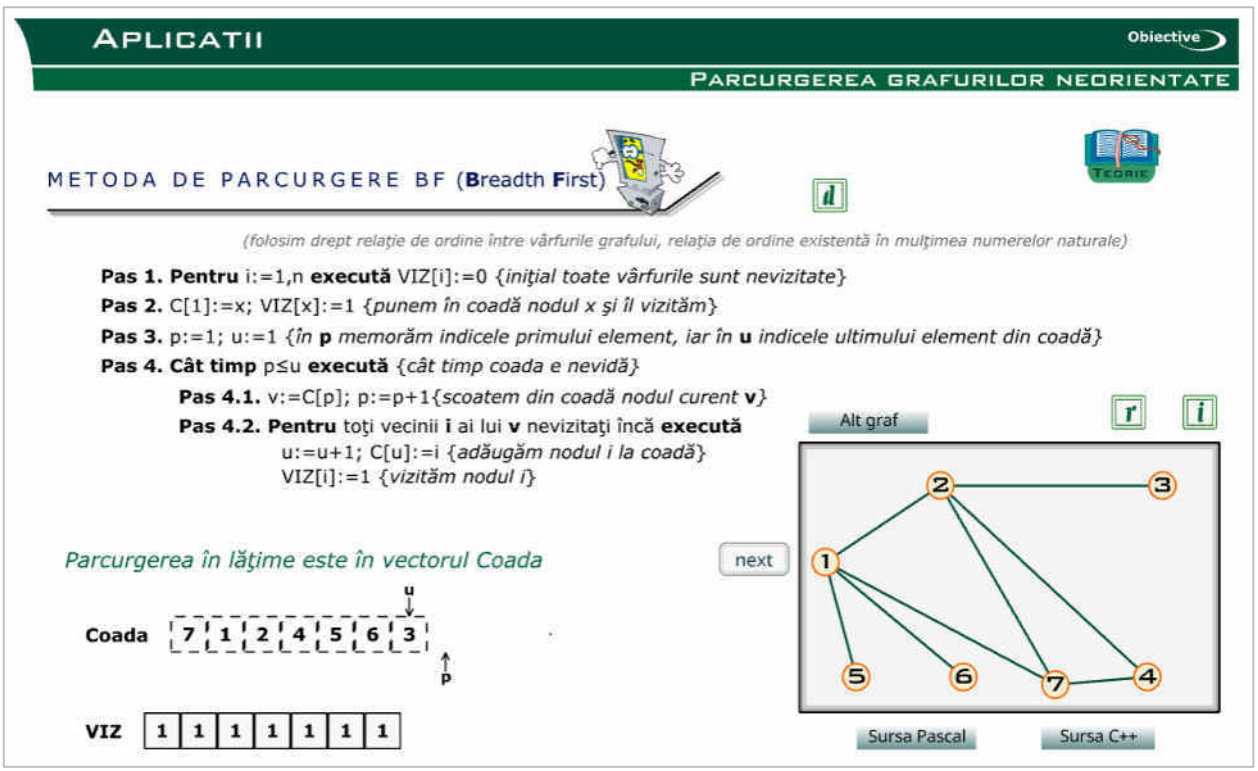

Figura 5. Rezultatul parcurgerii grafului neorientat în lățime

Pe un desen este reprezentat un graf arbitrar neorientat, în care alegem nodul de la care va începe parcurgerea(în lățime sau în adâncime) prin efectuarea unui simplu click pe acest nod. Dup apăsarea butonului *Start* se observă că acesta se transformă într-un buton *Next* pe care îl vom folosi pentru a vedea fiecare pas al algoritmului. Urmărim vectorii ***VIZ*** și ***C***

(respectiv ***ST***) pentru a înțelege pașii algoritmului. La fiecare apăsarea butonului *Next* observăm nodurile vizitate și respectiv nodurile-vecini analizate. Butonul *r* servește pentru reluarea algoritmului. Butonul *d* se folosește pentru a obține descrierea algoritmului. Butoanele *Sursa Pascal, Sursa C++* vă permite să vedeți sursa Pascal respectiv sursa C++ a algoritmului. La apăsarea butonului *Alt graf* se schimbă (generează) graful și se reia execuția pas cu pas.

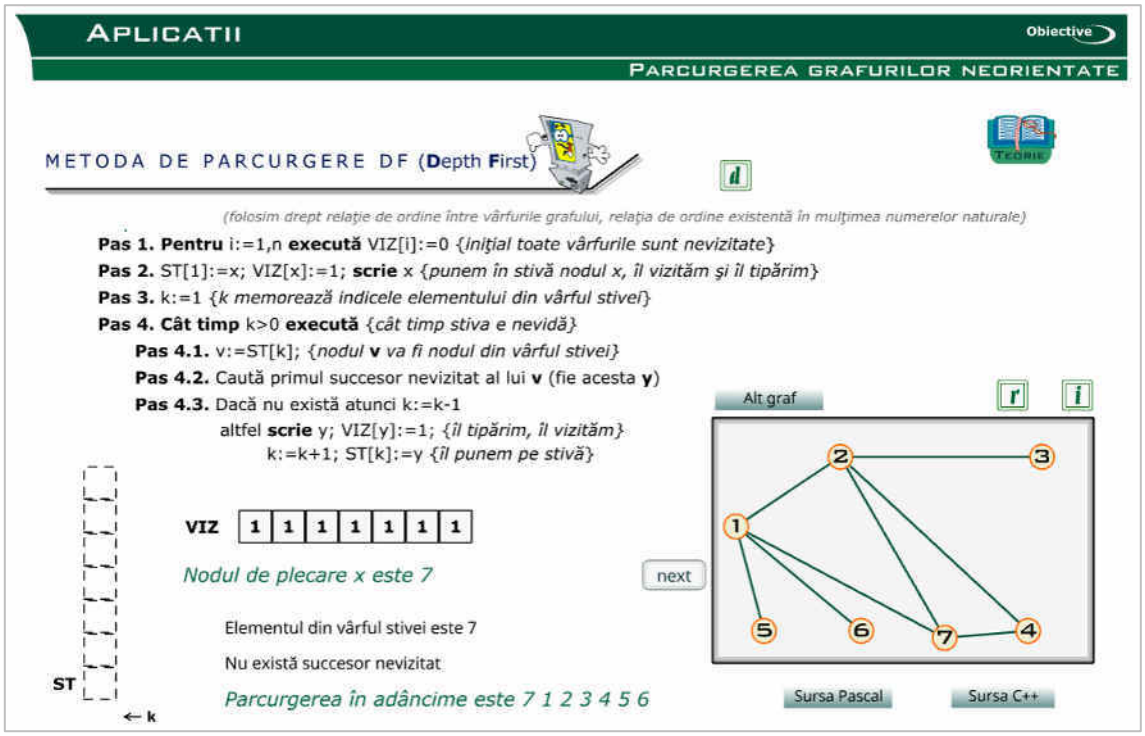

Figura 6. Rezultatul parcurgerii grafului neorientat în adâncime

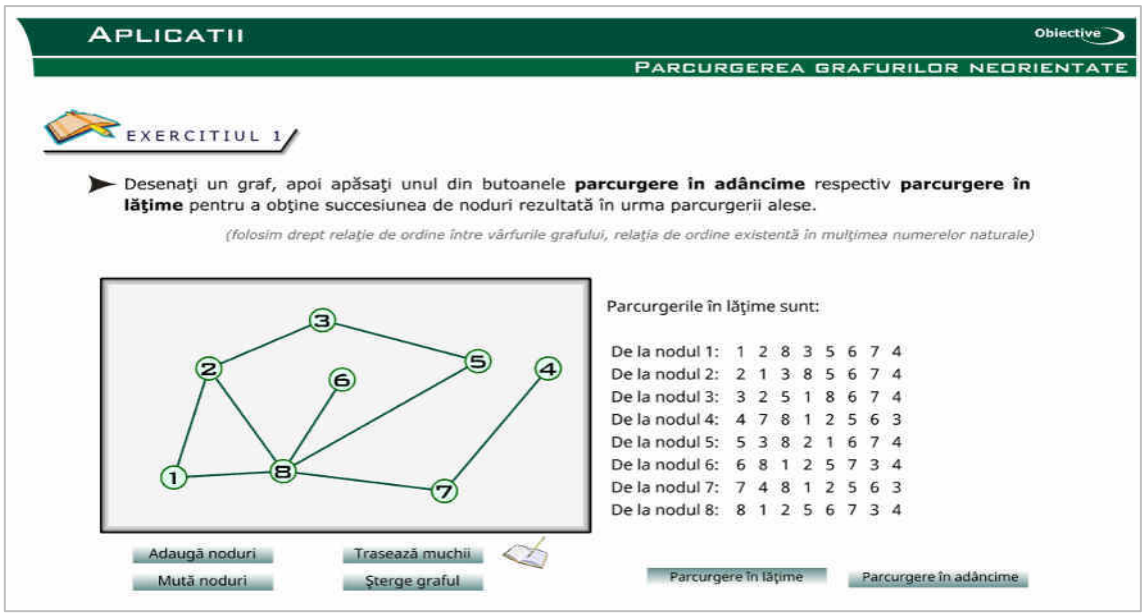

Figura 7. Implementarea algoritmilor de parcurgere pentru graful desenat de utilizator

Platforma campion.edu.ro suplimentar dispune și de aplicație unde utilizatorul poate să introducă(deseneze) graful neorientat dorit, cu restricția - numărul maximum de 8 noduri, pentru care va fi afișată succesiunea de noduri rezultată pentru toate nodurile ca noduri de pornire în urma parcurgerii alese.

4.2. *Platforma graphonline.ru*

Platforma graphonline.ru este un proiect online creat în anul 2015 și dezvoltat pe parcurs. Platforma vizează crearea și vizualizarea online a grafurilor, căutarea drumurilor minime într-un graf. De asemenea oferă posibilitatea de a obține graful introducând datele matricei de adiacență și multe altele. Platforma oferă accesul la interfață în 11 limbi, inclusiv limba engleză și limba rusă.

Interfața aplicației **Graph Online** include instrumentele pentru crearea grafului, inserarea nodurilor(vârfurilor) și muchiilor(arcelor), ștergerea obiectelor, setarea unor parametrilor de afișarea grafurilor, implementarea diferitor algoritmilor și arată în felul următor:

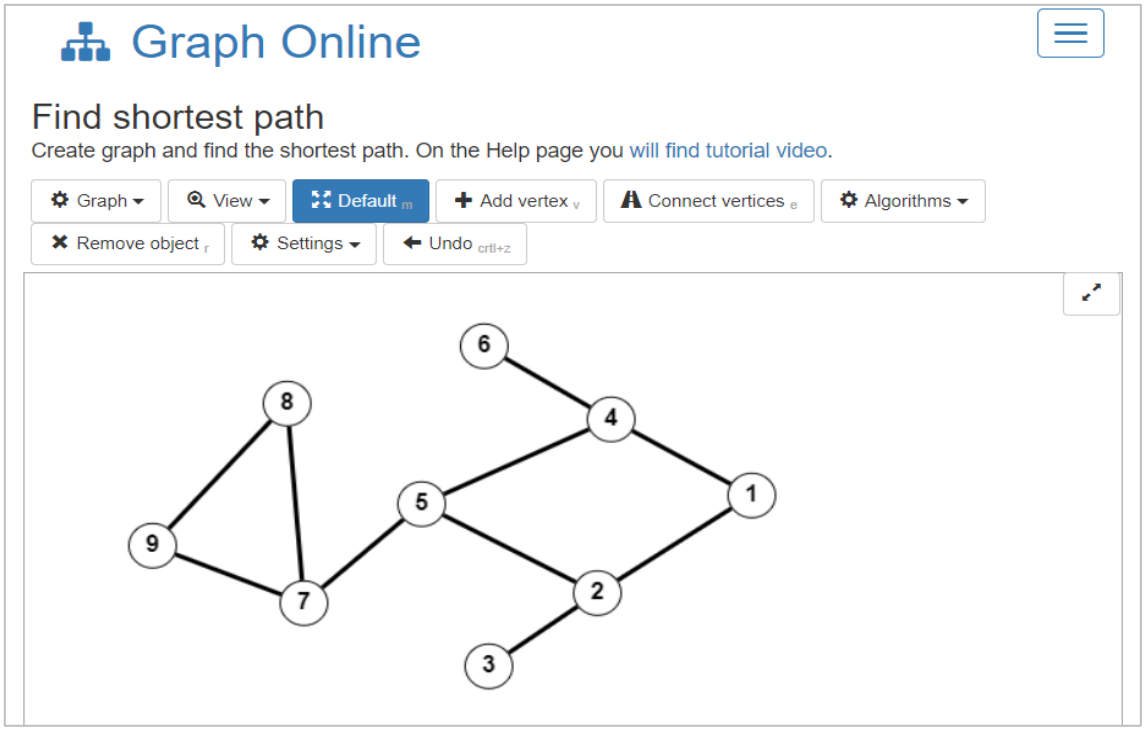

Figura 8. Interfața aplicației Graph Online pe platforma graphonline.ru

Pentru implementarea algoritmilor de parcurgerea, din meniul alegem *Algorithms* și parcurgere în lățime – *BFS*, pe desen alegem nodul de pornire și aplicația automat în mod interactiv prezintă vizitarea nodurilor din graf și afișează parcurgerea obținută în final.

Pentru parcurgerea în adâncime din meniul *Algorithms* alegem parcurgerea în adâncime – *DFS*.

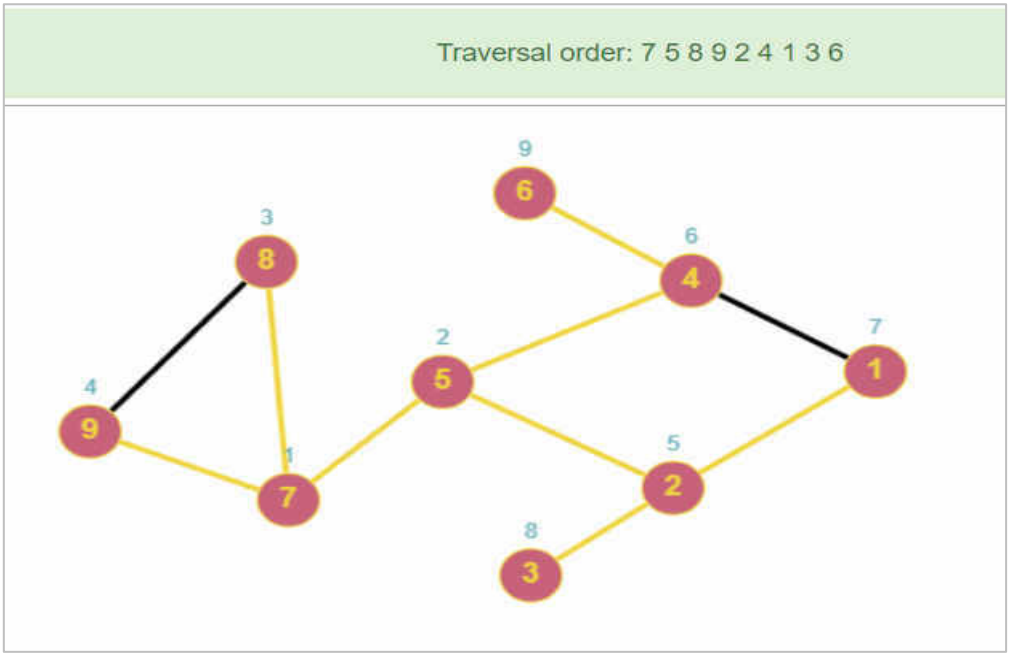

Figura 9. Implementarea algoritmului BFS în aplicația Graph Online

5. Implementarea algoritmilor în programare (limbajul C++)

Exemplu 3. Elaborați un program(Pascal sau C++) care realizează parcurgerea unui graf în lăţime (BFS). (https://www.onlinegdb.com/online_c++_compiler)

```
#include <iostream>
using namespace std;
int main(){
    int viz[30],n,i,j,u,v,p,k,a[20][20],c[30];
    cout<<"Dati numarul de varfuri n = ";cin>>n;
    for(i=1;i<=n-1;i++)
        for(j=i+1;j<=n;j++) {
        cout<<"a["<<i<<","<<j<<"]= ";cin>>a[i][j];
        a[j][i]=a[i][j];
        }
    cout<<"Dati varful de plecare ";cin>>k;
    for(j=1;j<=n;j++) viz[j]=0;
    c[1]=k;p=1;u=1;viz[k]=1;
     while(p<=u) {
         v=c[p];
         for(j=1;j<=n;j++) {
         if( (a[v][j]==1) && (viz[j]==0) )  {
         u++;
         c[u]=j;
         viz[j]=1;
         }
     }
    p++;
    }
     cout<<"Lista varfurilor in parcugerea in latime : "<<endl;
    cout<<k<<" ";
    for(j=2;j<=u;j++) cout<<c[j]<<" ";
     getchar();
  }
```

Rezolvarea pentru graful neorientat din Exemplul 1 este :

```
Dati numarul de varfuri n = 9
a[1,2]= 1   a[1,3]= 0   a[1,4]= 1   a[1,5]= 0   a[1,6]= 0   a[1,7]= 0   a[1,8]= 0   a[1,9]= 0
a[2,3]= 1   a[2,4]= 0   a[2,5]= 1   a[2,6]= 0   a[2,7]= 0   a[2,8]= 0   a[2,9]= 0
a[3,4]= 0   a[3,5]= 0   a[3,6]= 0   a[3,7]= 0   a[3,8]= 0   a[3,9]= 0
a[4,5]= 1   a[4,6]= 1   a[4,7]= 0   a[4,8]= 0   a[4,9]= 0
a[5,6]= 0   a[5,7]= 1   a[5,8]= 0   a[5,9]= 0
a[6,7]= 0   a[6,8]= 0   a[6,9]= 0
a[7,8]= 1   a[7,9]= 1
a[8,9]= 1
Dati varful de plecare 7
Lista varfurilor in parcugerea in latime: 7 5 8 9 2 4 1 3 6
```

Exemplu 4. Elaborați un program(Pascal sau C++) care realizează parcurgerea unui graf în adâncime (DFS). (https://www.onlinegdb.com/online_c++_compiler).

```
#include <iostream>
using namespace std;
int s[30],n,m,i,j,p,a[20][20];
void df(int k)
{
    int l;
    cout<<k<<" ";
    s[k]=1;
    for(l=1;l<=n;l++)
    if( (a[k][l]==1) && (s[l]==0) )
    df(l);
    return;
}

int main()
{
    cout<<"Dati numarul de varfuri n = ";cin>>n;
    for(i=1;i<=n-1;i++)
    for(j=i+1;j<=n;j++)
{  cout<<"a["<<i<<","<<j<<"]= ";cin>>a[i][j];
        a[j][i]=a[i][j];
        }

    cout<<"Dati varful de plecare ";cin>>p;
    cout<<" Lista varfurilor in parcugerea in adincime : ";
    df(p);
    getchar();
}
```

Rezolvarea pentru graful neorientat din Exemplul 2 este:

```
Dati numarul de varfuri n = 9
a[1,2]= 1   a[1,3]= 0   a[1,4]= 1   a[1,5]= 0   a[1,6]= 0   a[1,7]= 0   a[1,8]= 0   a[1,9]= 0
```

a[2,3]= 1 a[2,4]= 0 a[2,5]= 1 a[2,6]= 0 a[2,7]= 0 a[2,8]= 0 a[2,9]= 0
a[3,4]= 0 a[3,5]= 0 a[3,6]= 0 a[3,7]= 0 a[3,8]= 0 a[3,9]= 0
a[4,5]= 1 a[4,6]= 1 a[4,7]= 0 a[4,8]= 0 a[4,9]= 0
a[5,6]= 0 a[5,7]= 1 a[5,8]= 0 a[5,9]= 0
a[6,7]= 0 a[6,8]= 0 a[6,9]= 0
a[7,8]= 1 a[7,9]= 1
a[8,9]= 1
Dati varful de plecare **7**
Lista varfurilor in parcugerea in latime : **7 5 2 1 4 6 3 8 9**

Concluzii

Algoritmii din Teoria Grafurilor ne oferă posibilitatea realizării unor modele matematice intuitive prin intermediul cărora se pot soluționa probleme din diverse domenii teoretice și practice. Parcurgerea grafurilor este un instrument eficient de cercetare a structurii de date de tip graf. Utilizarea tehnologiilor informaționale, în special cele interactive, în studierea algoritmilor de parcurgere grafurilor joacă un rol important în formarea și dezvoltarea competențelor profesionale și competențelor digitale a studenților. Studierea parcurgerilor BFS și DFS contribuie esențial la dezvoltarea spiritului de observație și gândirii spațiale ale studenților.

Articol realizat în cadrul proiectului de cercetări științifice „Metodologia implementării TIC în procesul de studiere a științelor reale în sistemul de educație din Republica Moldova din perspectiva inter/transdisciplinarității (concept STEAM)”, inclus în „Program de stat” (2020-2023), Prioritatea IV: Provocări societale, cifrul 20.80009.0807.20, cu suportul financiar oferit de Agenția Națională pentru Dezvoltare și Cercetare

Bibliografie

1. CHIRIAC, L.; BOSTAN, M. *Elemente de teoria grafurilor*. 2022 (în curs de editare).
2. CORLAT, S.; GREMALSCHI, A. *Grafuri. Metodologia predării în cadrul instruirii de performanță la disciplinele Matematică & Informatică*. Chișinău, 2014. 158 p. ISBN 978-9975-7-122-2.
3. ERICKSON, J. Algorithms. 472 p. ISBN 978-1-792-6448-2. disponibil: https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE-2up.pdf (accesat la 25.09.2022).
4. *Graphviser learn*. https://learn.graphviser.app/?locale=ro#section-6 (accesat la 13.08.2022).
5. *Graph Online*. https://graphonline.ru/en/ (accesat 20.09.2022)
6. *Teoria grafurilor*. http://campion.edu.ro/arhiva/www/arhiva_2009/seds/17/index.htm (accesat 14.09.2022)
7. ДДОЛЬНИКОВ, В. Л., ЯКИМОВА, О. П. Основные алгоритмы на графах. Ярославль, 2011. 80 с. ISBN 978-5-8397-0855-6. disponibil: http://www.lib.uniyar.ac.ru/edocs/iuni/20110210.pdf (accesat 10.08.2022).