7 Grafuri

7.1 Obiective

Scopul acestui laborator este de a prezentata câteva noţiuni legate de grafuri, modurile de reprezentare şi traversare a lor.

7.2 Noţiuni teoretice

7.2.1 Noţiuni de bază

Graf orientat: Graful orientat sau digraful $G = (V, E)$ este perechea formată din mulţimea V de vărfuri şi mulţimea $E \subset V \times V$ de arce.

Arc: Un arc este o pereche ordonată de vârfuri (v, w), unde v este baza arcului, iar w este vârful arcului. In alţi termeni se spune că w este adiacent lui v.

Graf neorientat: Un graf neorientat sau prescurtat $G = (N, R)$ este perechea formată din mulţimea N de noduri şi mulţimea R de muchii. O muchie este o pereche neordonată $(v, w) = (w, v)$ de noduri. Definiţiile prezentate rămân valabile şi în cazul grafurilor neorientate.

Cale: O cale este o succesiune de vărfuri $v[1], v[2], ..., v[k]$, astfel că există arcele $(v[1], v[2]), (v[2], v[3]), ..., (v[k-1], v[k])$ în mulţimea arcelor E. Lungimea căii este numărul de arce din cale. Prin convenţie, calea de la un nod la el însuşi are lungimea 0.

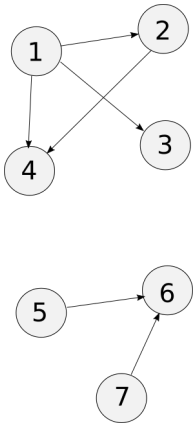
Căi simple: O cale este simplă, dacă toate vârfurile, cu excepţia primului şi ultimului sunt distincte între ele.

Ciclu: Un ciclu este o cale de la un vârf la el însuşi.

Etichetare: Un graf orientat etichetat este un graf orientat în care fiecare arc şi /sau vârf are o etichetă asociată, care poate fi un nume, un cost sau o valoare de un tip oarecare.

Graf conex: Un graf orientat este tare conex, daca oricare ar fi vârfurile v şi w există o cale de la v la w şi una de la w la v.

Subgrafuri: Un graf $G' = (V', E')$ este subgraf al lui G daca $V' \subset V$ şi $E' \subset E$. Se spune că subgraful indus de $V \subset V$ este $G' = (V', E \subset (V' \times V'))$.



Vârfuri :

```
1, 2, 3, 4, 5, 6, 7
```

Arce:

```
{1,2},{1,3},{1,4},{2,4},{5,6},{7,6}
```

Figure 7.1: Un graf format din două elemente conexe, cu listele de vârfuri si arce date

7.2.2 Moduri de reprezentare

Atât grafurile orientate, cât şi cele neorientate se reprezintă frecvent sub două forme: matricea de adiacenţe şi listele de adiacenţe.

Astfel, pentru graful orientat $G = (V, E)$ unde V este mulţimea vârfurilor $V = \{1, 2, \ldots, n\}$, matricea de adiacenţe A va fi definită astfel:

$$A[i][j] = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

Matricea de adiacenţe etichetată A (sau matricea costurilor) va fi definită astfel:

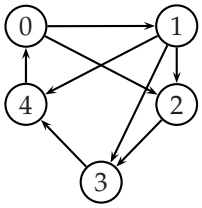
$$A[i][j] = \begin{cases} \text{label of arc } (i,j) & \text{if } (i,j) \in E \\ \text{an arbitrary symbol} & \text{if } (i,j) \notin E \end{cases}$$

Matricea de adiacenţe este simetrică pentru grafuri neorientate şi nesimetrică pentru cele orientate.

Grafurile pot fi reprezentate prin matrici prin următoarea structură:

```
struct graph{
        int n; // numarul de noduri
        int ** m; // matricea de adiacenta
}
```

Reprezentarea prin liste de adicenţe foloseşte mai bine memoria, dar căutarea arcelor este mai greoaie. În această reprezentare, pentru fiecare nod se păstrează lista arcelor către nodurile adiacente. Întregul graf poate fi reprezentat printr-un tablou indexat după noduri, fiecare intrare în tablou conţinând adresa listei nodurilor adiacente. Lista nodurilor adiacente poate fi dinamică sau statică.

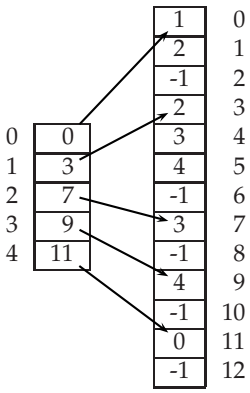


(a) Graf orientat.

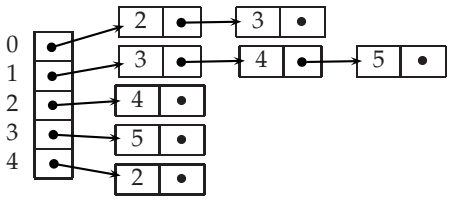
	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	1	1
2	0	0	0	1	0
3	0	0	0	0	1
4	1	0	0	0	0

(b) Matrice de adiacenţa pentru graful din figura Figure 7.2(a)

Figure 7.2: Un graf orientat si matricea sa de adiacenţa



(a) Listă de adiacenţa statică.



(b) Listă de adiacenţa dinamică.

Figure 7.3: Listele de adiacenţa pentru graful din figura 7.2(a)

Explorarea în lăţime

Explorarea în lăţime constă în următoarele acţiuni:

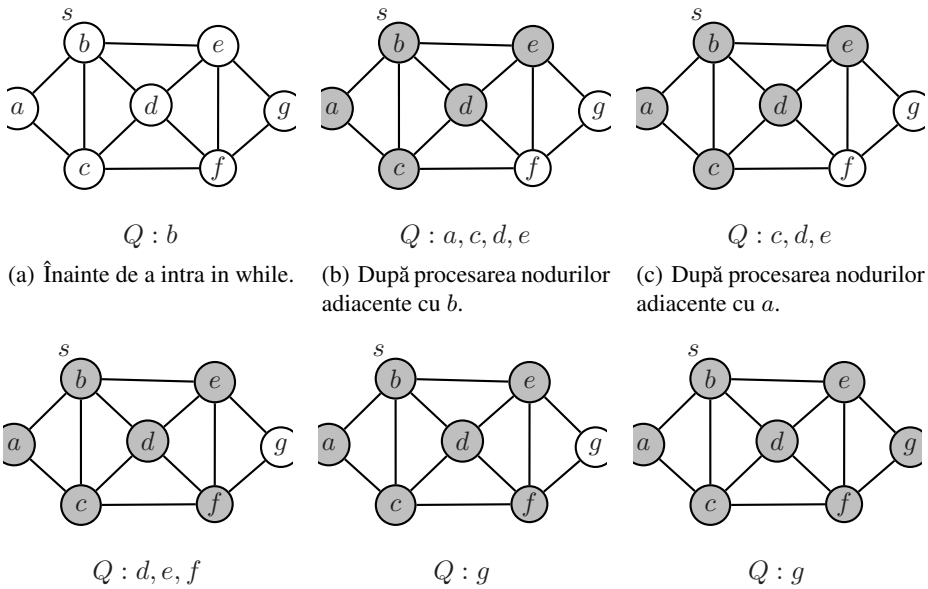
1. Se trece într-o coadă vidă nodul de pornire;
2. Se trece extrage din coadă câte un nod care este prelucrat şi se adaugă toate nodurile adiacente lui neprelucrate;
3. Se repetă pasul 2 până când coada devine vidă.

Algoritmul este următorul:

```
enum { NEVIZITAT, VIZITAT };
void bfs( int nrNoduri, int nodSursa )
{
        int vizitate[ nrNoduri ]; /* marcarea nodurilor vizitate */
        Coada Q; /* coada nodurilor - intregi */
        int i, v, w; /* noduri */

        initializeaza( Q );
        for ( i = 0; i < nrNoduri; i++ ) /* marcam toate nodurile ca nevizitate */
                vizitate[ i ] = NEVIZITAT;
        vizitate[ nodSursa ] = VIZITAT; /* marcam nodul sursa ca vizitat */
        procesam informatia pt nodSursa;
        introducere in coada( nodSursa, Q );
        /* nodSursa va fi primul nod scos din coada */
        while( ! goal\u a( Q ))
        {
                v = scoate din coad\u a( Q );
                for ( fiecare nod w adiacent cu v )
                        if ( vizitate[ w ] == NEVIZITAT )
                        {
                                vizitate[ w ] = VIZITAT;
                                proces\u am informatia pt w;
                                introducere in coada( w, Q );
                        }
        }
}
```

O trasare a paşilor relevanţi pentru cătarea in lăţime este prezentată in figura 7.4.



(a) $Q : b$ — Înainte de a intra in while.

(b) $Q : a, c, d, e$ — După procesarea nodurilor adiacente cu b.

(c) $Q : c, d, e$ — După procesarea nodurilor adiacente cu a.

(d) $Q : d, e, f$ — După procesarea nodurilor adiacente cu c.

(e) $Q : g$ — După procesarea nodurilor adiacente cu d şi e.

(f) $Q : g$ — După procesarea nodurilor adiacente cu f.

Figure 7.4: O trasare a căutari în lăţime pe un graf.

7.3 Mersul lucrării

7.3.1 Probleme Obligatorii

1. Implementaţi bazându-vă pe codul descris in lucrare un program pentru a parcurge un graf citit din fişier în lăţime. Scrieţi in alt fişier nodurile parcurse, în ordine.

2. Citind de la tastatură un graf, si două noduri să se afişeze dacă cele două noduri sunt conectate.

7.3.2 Probleme Optionale

1. Să se implementeze o aplicaţie care să detecteze dacă un graf este bipartit.
2. Să se implementeze o aplicaţie care citind două noduri de la tastatură sa afişeze distanţa minimă dintre ele.

7.3.3 Probleme Suplimentare (pentru punct in plus la nota de la colocviu)

1. Folosiţi parcurgerea în lăţime pentru a detecta existenţa ciclurilor într-un graf.