8 Grafuri - continuare

8.1 Obiective

Scopul acestui laborator este de a prezentata traversarea grafurilor în adâncime si aplicaţiile acestei operaţi.

8.2 Noţiuni eoretice

8.2.1 Explorarea in adâncime

La explorarea în adâncime se marchează vizitarea nodului iniţial, după care se vizitează în adâncime, recursiv, fiecare nod adiacent. După vizitarea tuturor nodurilor ce pot fi atinse din nodul de start, parcurgerea se consideră încheiată. Dacă rămân noduri nevizitate, se alege un nou nod şi se repetă procedeul de mai sus.

Algoritmul de bază este următorul:

```
enum { NEVIZITAT, VIZITAT };
void dfs( int nrNoduri, int nodSursa )
{
        int vizitate[ nrNoduri ]; /* marcarea nodurilor vizitate */
        Stiva ST; /* stiva nodurilor - intregi */
        int i, v, w; /* noduri */

        initializeaz\u a( ST );
        for ( i = 0; i < nrNoduri; i++ ) /* marc\u am toate nodurile ca nevizitate */
                vizitate[ i ] = NEVIZITAT;
        vizitate[ nodSursa ] = VIZITAT; /* marc\u am nodul sursa ca vizitat */
        proces\u am informa\c tia pt nodSursa;
        introducere in stiv\u a( nodSursa, ST );
        /* nodSursa va fi primul nod scos din stiva */
        while( ! goal\u a( ST ))
        {
                v = con\c tinutul din v\^ arful stivei ST;
                w = urm\u atorul nod adiacent lui v nevizitat;
                if ( exis\u a w )
                {
                        vizitate[ w ] = VIZITAT;
                        proces\u am informa\c tia pt w;
                        introducere in stiv\u a( w, ST );
                }
                else pop(ST); /*  se \c sterge nodul v din v\^ arful stivei ST  */
        }
}
```

O prezentare a paşilor relevanţi a algoritmului de trasare în adâncime sunt prezentaţi in figura 8.1.

8.2.2 Marcaje de timp şi colorarea nodurilor

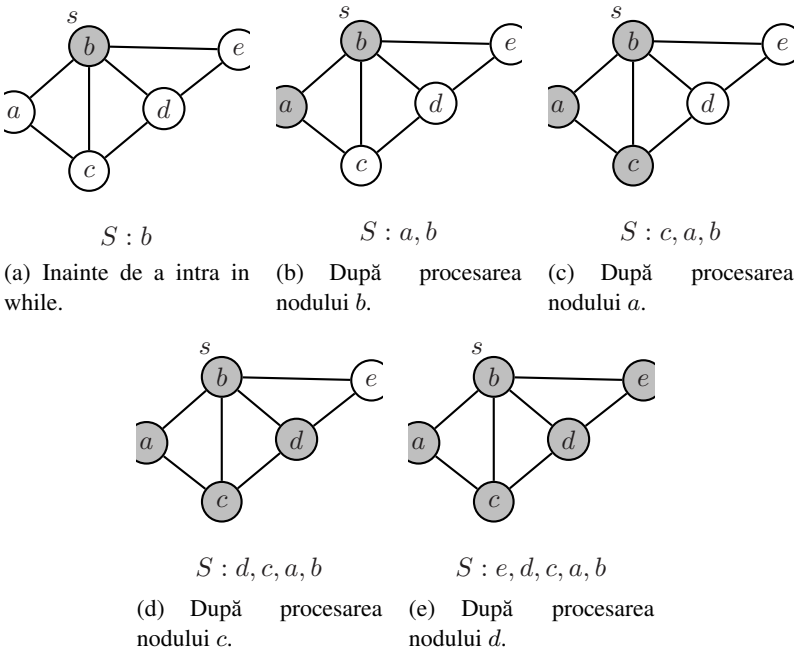
Parcurgerea în adâncime este folosita şi pentru a asocia fiecărui nod două marcaje de timp, pentru a ţine minte când acest nod a fost descoperit (într-un şir d[v]) şi când am parcurs toţi vecini nodului (informaţie ţinută într-un şir f[v]). Aceste valori sunt utile pentru mulţi algoritmi pe grafuri, cât şi folositoare pentru a trasa comportamentul parcurgeri în adâncime.

Asociem fiecărui nod şi o culoare în funcţie de etapa de procesare în care se află: ALB dacă încă nu a fost procesat, GRI dacă a fost abordat, NEGRU dacă şi vecini săi au fost procesaţi.

Algoritmul pentru explorarea in adâncime cu aceste valori introduse, este următorul:

```
DFS(G)
        for each vertex u in V [G]
                do color[u] <- ALB
                π[u] <- NIL
        time <- 0
        for each vertex u in V [G]
                do if color[u] = ALB
                then DFS-VISIT(u)

DFS-VISIT(u)
        color[u] <- GRI // Nodul alb u tocmai a fost vizitat.
```



$S : b$

(a) Inainte de a intra in while.

$S : a, b$

(b) După procesarea nodului b.

$S : c, a, b$

(c) După procesarea nodului a.

$S : d, c, a, b$

(d) După procesarea nodului c.

$S : e, d, c, a, b$

(e) După procesarea nodului d.

Figure 8.1: O trasare a căutari în adâncime pe un graf.

```
time <- time +1
d[u] time
for each v in Adj[u] // exploram muchia(u, v).
        do if color[v] = ALB
        then π[v] <- u
        DFS-VISIT(v)
color[u] NEGRU // Coloram u cu negru; am terminat cu acest nod.
f [u] -> time <- time +1
```

8.2.3 Etichetarea muchiilor

Muchiile unui graf pot fi clasificate in 4 categori, în funcţie de arbori generaţi ca urmare a parcurgeri in adâncime:

- **Muchii ale arborelui**. Muchia (u, v) este de acest tip daca nodul v a fost descoperit parcurgând muchia (u, v).
- **Muchii înapoi**. Sunt toate acele muchii (u, v) unde u este un descendent a lui v în arborele format de parcurgerea in adâncime.
- **Muchiile inainte** sunt muchiile (u, v) care nu aparţin arborelui care conecteaza un nod u de un descendent v.
- **Muchiile transversale** sunt toate celelalte muchii.

Algoritmul de parcurgere in adâncime poate fi modificat pentru a permite etichetarea muchiilor în funcţie de categoria lor. Asta se face urmărind culoarea nodului v în momentul in care se parcurge muchia (u, v) în algoritmul de parcurgere în adâncime.

- ALB indică o muchie a arborelui.
- GRI indică o muchie înapoi.
- NEGRU indică o muchie înainte sau transversala.

8.2.4 Sortarea topologică

Sortarea topologică a unui graf direcţionat $G = (V, E)$ este o ordonare lineara a tuturor nodurilor aşa incât dacă G conţine o muchie (u, v), atunci u apare înaintea lui v în ordonare. (dacă graful nu este aciclic atunci această ordonare este imposibilă). O sortare topologică a unui graf poate fi văzută ca o ordonare a nodurilor într-o linie orizontală aşa incât toate muchile direcţionate să meargă de la stânga la dreapta.

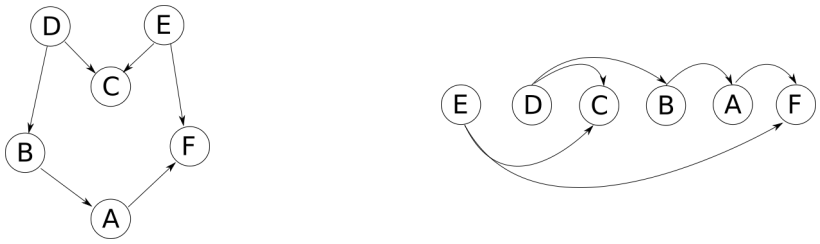


Figure 8.2: O vizualizare a sortari topologice pe un graf

```
SORTARE-TOPOLOGICA(G)
        apelam DFS(G) pentru a calcula timstamp-urile de fnal f[v] pentru fiecare nod v
        cand un nod este terminat, el este inserat intr-o lista inlantuita
        returnam lista inlantuita cand toate nodurile au fost terminate
```

8.3 Mersul lucrării

8.3.1 Probleme Obligatorii

1. Să se implementeze o aplicaţie care să parcurgă arborele în adâncime si apoi sa printeze marcajele de timp pentru fiecare nod.
2. Să se implementeze o aplicaţie care să parcurgă arborele în adâncime si apoi sa printeze toate muchiile în funcţie de categoria lor.

8.3.2 Probleme Optionale

1. Să se implementeze o aplicaţie care să sorteze topologic nodurile unui graf.
2. Să se implementeze o aplicaţie care să determine dacă un graf conţine sau nu cicluri.

8.3.3 Probleme Extra

1. Să se implementeze o aplicaţie care să găsească elementele puternic conexe a unui graf.
2. Să se implementeze o aplicaţie care să găsească punctele de articulaţie a unui graf neorientat.