Proiectarea algoritmilor

Tema de casă nr. 1

Sortarea topologică

Cuprins

1 Consideraţii generale

Se aplică la secvenţe cu elemente din mulţimi parţial ordonale. Exemplu de relaţie de ordine este parţială: $a_1 < a_0, a_1 < a_2 < a_3$.

Problema constă în a crea o listă liniară care să fie compatibilă cu relaţia de ordine, adică, dacă $a_i < a_j$, atunci a_i va precede pe a_j în lista finală.

Pentru exemplul nostru, lista liniară finală va putea fi (a_1, a_0, a_2, a_3), sau (a_1, a_2, a_0, a_3), sau (a_1, a_2, a_3, a_0).

Definiţie: Fie $(S, \leq)$ o mulţime parţial ordonată finită şi $a = (a_0, a_1, \ldots, a_{n-1})$ o liniarizare a sa. Spunem că secvenţa a este *sortată topologic*, dacă $\forall i, j : a_i < a_j \Rightarrow i < j$.

1.1 Mulţimi parţial ordonate finite şi digrafuri aciclice

Există o legătură strânsă între mulţimile parţial ordonate finite şi digrafurile aciclice (digrafuri fără circuite numite pe scurt dag-uri).

Un *graf* este o pereche $G = (V, E)$, unde V este o mulţime ale cărei elemente sunt numite *vârfuri*, iar E este o mulţime de perechi neordonate $\{u, v\}$ de vârfuri, numite *muchii*.

Un *digraf* este o pereche $D = (V, A)$, unde V este o mulţime de vârfuri, iar A este o mulţime de perechi ordonate (u, v) de vârfuri, numite arce.

Orice mulţime parţial ordonată $(S, \leq)$ defineşte un dag $D = (S, A)$, unde există arc de la a la b, dacă $a < b$ şi nu există $c \in S$ cu proprietatea $a < c < b$.

Reciproc, orice dag $D = (V, A)$ defineşte o relaţie de ordine parţială $\leq$ peste V, dată prin: $u \leq v$, dacă există un drum de lungime ≥ 0 de la u la v.

De fapt, $\leq$ este închiderea reflexivă şi tranzitivă a lui A (se mai notează $\leq = A^*$).

Sortarea topologică a unui dag constă într-o listă liniară a vârfurilor astfel încât dacă există arc de la u la v, atunci u precede pe v în listă, pentru oricare două vârfuri u şi v.

Vârfurile care candidează pentru primul loc în lista sortată topologic au proprietatea că nu există arce incidente spre interior (care sosesc în acel vârf) şi se numesc *surse*.

1.2 Reprezentarea digrafurilor prin liste de adiacenţă

Reprezentarea prin liste de adiacenţă exterioară:

Digraful D este reprezentat printr-o structură asemănătoare cu cea de la matricele de adiacenţă.

Matricea de adiacenţă este înlocuită cu un tablou unidimensional de n liste liniare, implementate prin liste simplu înlănţuite şi notate cu `D.a[i]` pentru $i = 0, \ldots, n-1$.

Lista `D.a[i]` conţine vârfurile destinatare ale arcelor care pleacă din i (= lista de adiacenţă exterioară).

Reprezentarea prin liste de adiacenţă interioară:

Lista `D.a[i]` conţine vârfurile surse ale arcelor care sosesc în i (= lista de adiacenţă interioară).

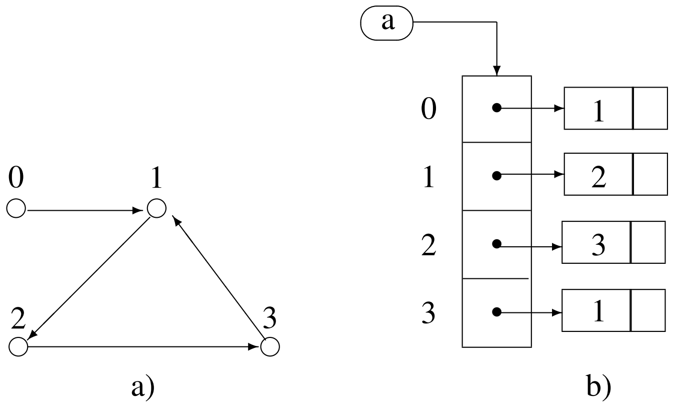


Figura 1: Digraf reprezentat prin liste de adiacenţă exterioară înlănţuite

2 Algoritm BFS

2.1 Descrierea algoritmului

Presupunem că pentru dag-ul D sunt create atât listele de adiacenţă interioară, cât şi cele de adiacenţă exterioară. Listele de adiacenţă interioară vor fi utilizate la determinarea vârfurilor sursă (vârfuri fără predecesori); acestea au listele de adiacenţă interioară vide.

1. Iniţializează coada cu vârfurile sursă.
2. Extrage un vârf u din coadă pe care-l adaugă la lista sortată parţial.
3. Elimină din reprezentarea (acum parţială) a lui D vârful u şi toate arcele (u, v).
4. Dacă pentru un astfel de arc lista de adiacenţă interioară a vârfului v devine vidă, atunci v va fi adăugat la coadă.
5. Repetă paşii 2-4 până când coada devine vidă.

Extindem structura `D`, care reprezintă digraful *D*, cu tabloul `np[1..n]`.

- `D.np[u]` conţine numărul predecesorilor vârfului *u*.
- `L` este lista care conţine varfurile digrafului *D* în ordine topologică.

```
procedure sortareTopologicaBFS(D,np)
   coadaVida(C) //initializeaza coada C
   // insereaza in C varfurile fara predecesori
   for u ← 0 to D.n-1 do
       if D.np[u]=0 then insereaza(C,u)
   // construieste lista varfurilor (afiseaza) in ordine topologica
   for k ← 0 to D.n-1 do
       if esteVida(C)
          then return ("Graful contine cicluri")
       u ← elimina(C)
       inserează(L, u) // inserarea se face la sfarsitul listei
       p ← D.a[u]
       while p≠NULL do
          v ← p->elt //v este un succesor imedial al lui u
          D.np[v] ← D.np[v]-1
          if D.np[v]=0
             then insereaza(C,v)
          p ← p->succ
end
```

2.2 Complexitatea timp

- Identificarea vârfurilor fără predecesori: $O(n)$
- Ştergerea muchiilor: $O(m)$
- Afisarea vârfurilor: $O(n)$

Timp total : $O(n+m)$

3 Sarcini de lucru şi barem de notare

Sarcini de lucru:

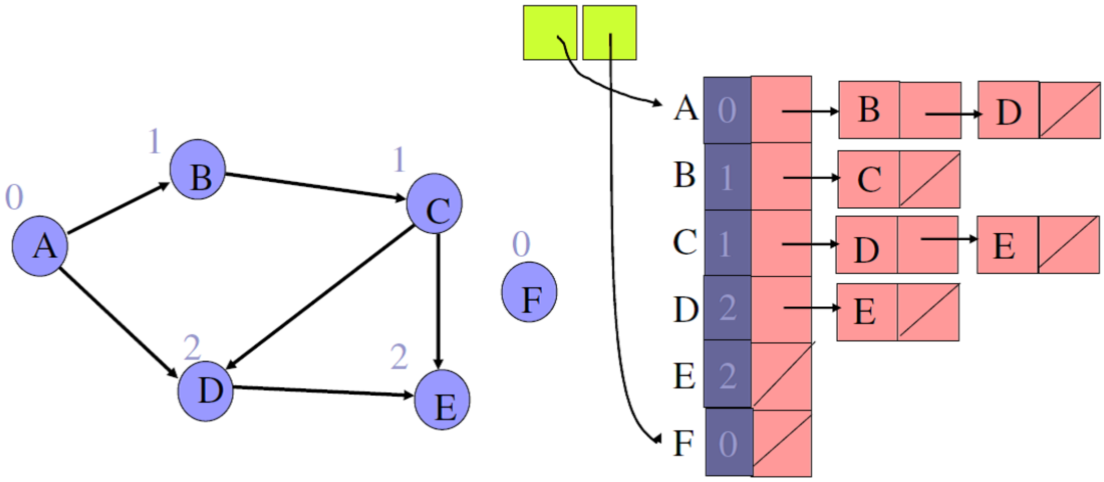
1. Scrieţi un program C/C++ care sortează topologic vârfurile unui digraf aciclic $D = (V,A)$.

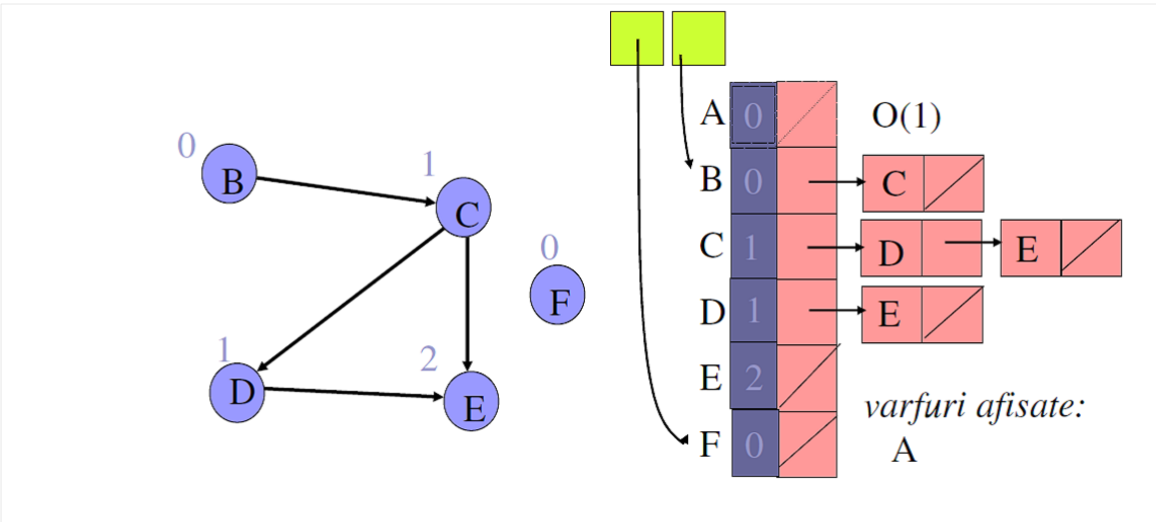
Barem de notare:

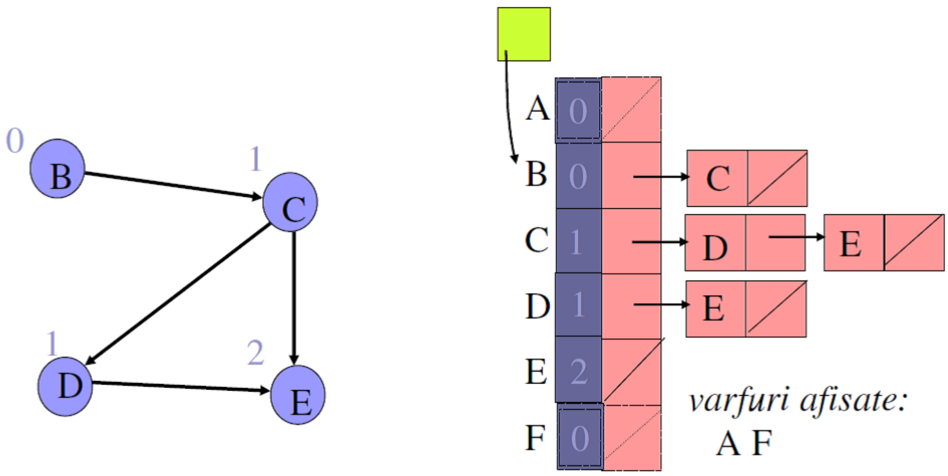
1. Construirea listelor de adiacenţă: 3p
2. Implementarea algoritmului de sortare topologică : 6p
3. Baza: 1p

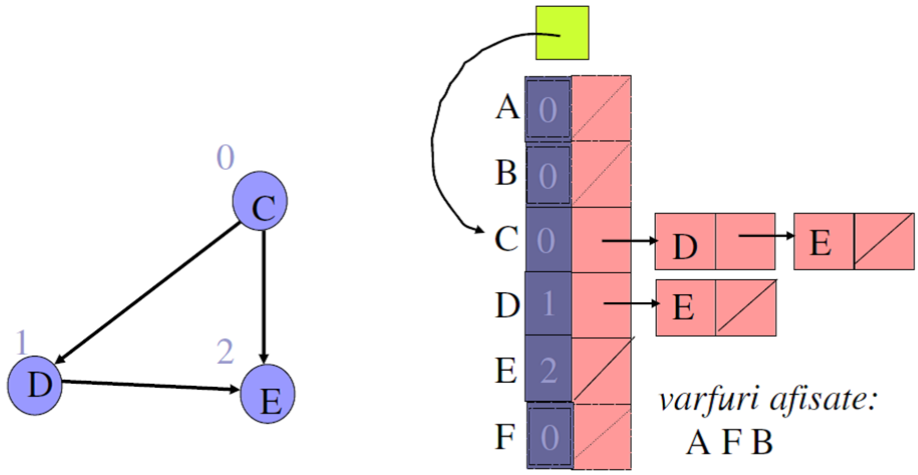
Bibliografie

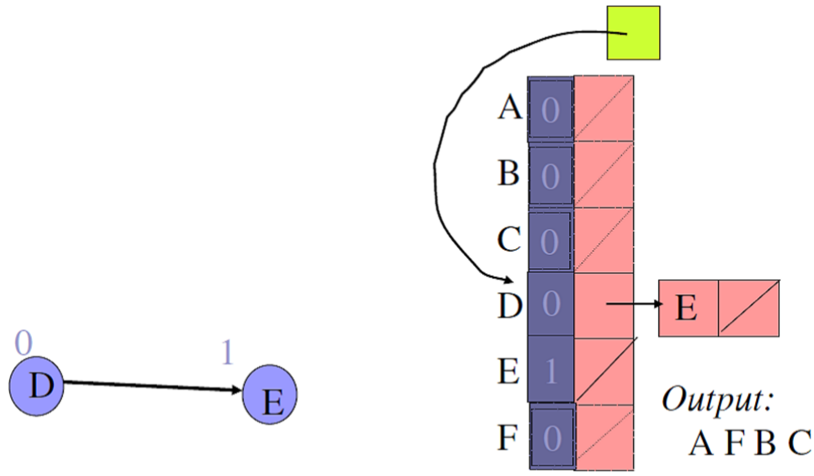
[1] Lucanu, D. şi Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.

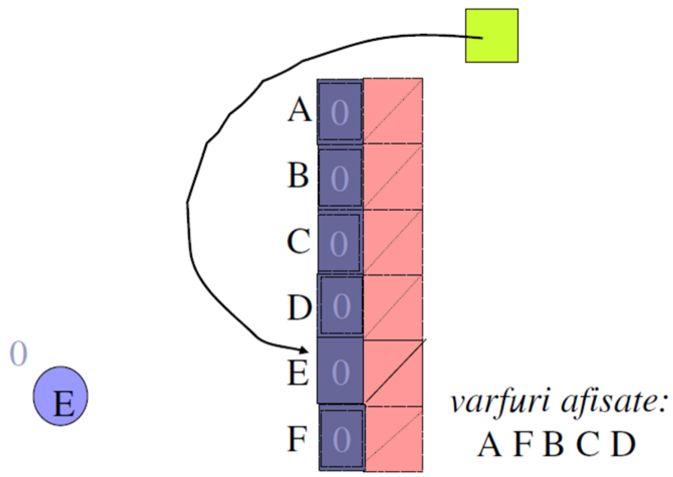

A B C D E F
0 1 1 2 2 0


O(1)
varfuri afisate:
A

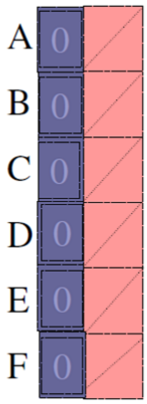

varfuri afisate:
A F


0
C
1
D
2
E
A 0
B 0
C 0
D 1
E 2
F 0
D
E
E
varfuri afisate:
A F B


0
D
1
E
A 0
B 0
C 0
D 0
E 1
F 0
E
Output:
A F B C


0
E
A 0
B 0
C 0
D 0
E 0
F 0
varfuri afisate:
A F B C D

FINAL!



varfuri afisate:
A F B C D E