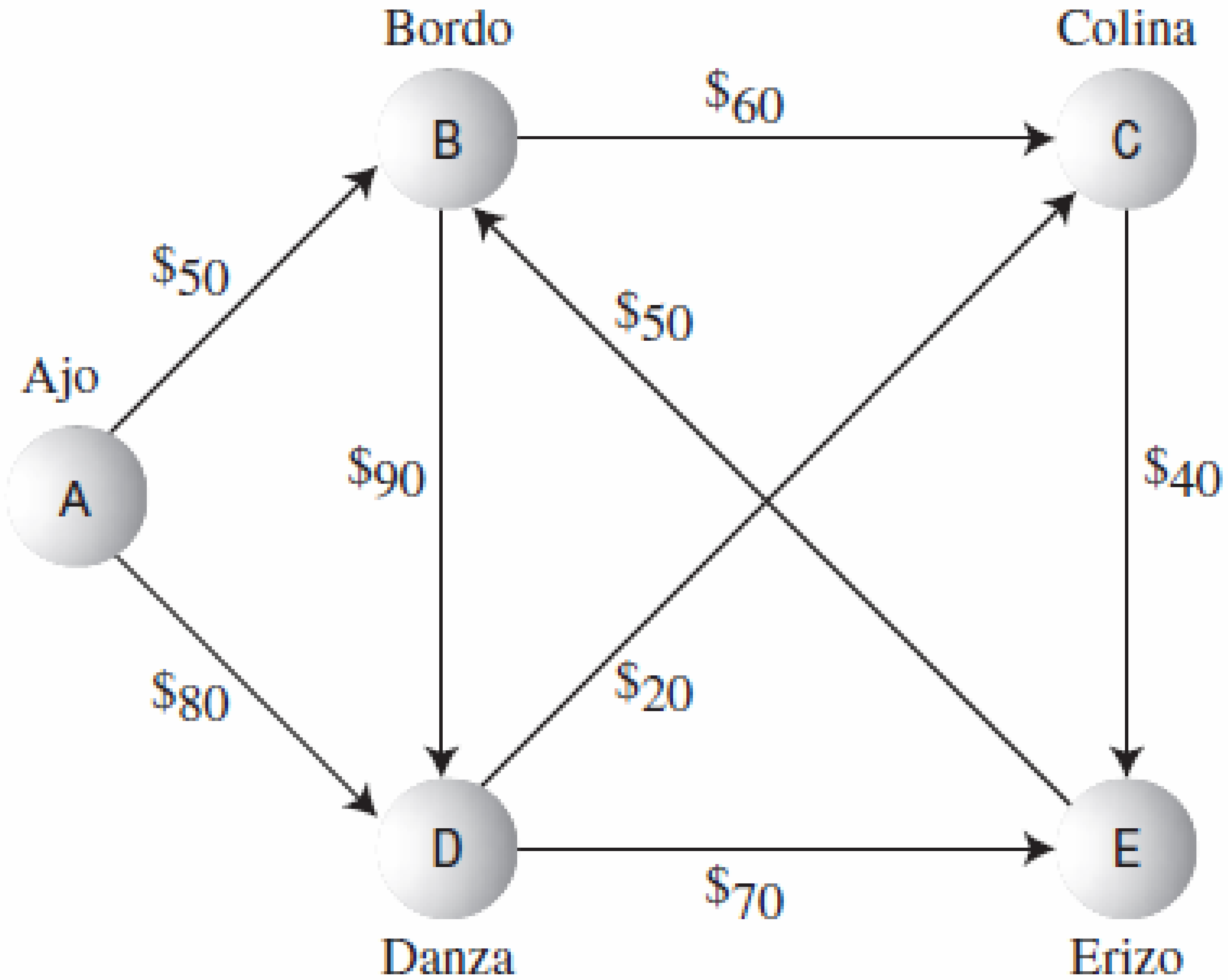
ALGORITMUL LUI FLOYD

Șl. Dr. Ing. Șerban Radu
Departamentul de Calculatoare
Facultatea de Automatică și Calculatoare

Problema drumului de lungime minimă din orice vârf

- Problema se referă la aflarea **costului minim** din **orice vârf** către **oricare alt vârf**, folosind muchii multiple
- Aceasta se numește problema drumului de lungime minimă din orice vârf (**all-pairs shortest path** problem)

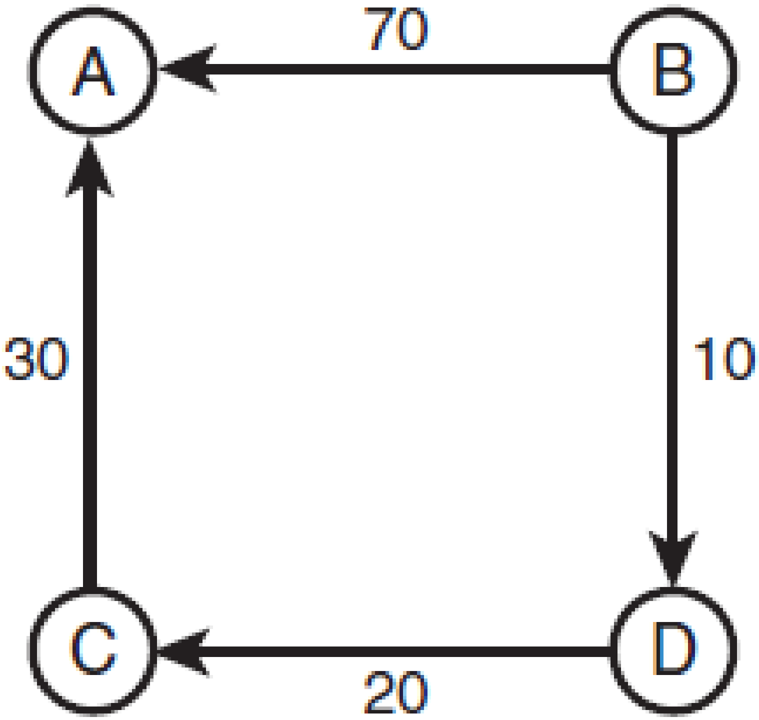


Bordo
Colina
B
C
$60
$50
$50
Ajo
A
$90
$40
$80
$20
D
E
$70
Danza
Erizo

	A	B	C	D	E
A	—-	50	100	80	140
B	—-	—-	60	90	100
C	—-	90	—-	180	40
D	—-	110	20	—-	60
E	—-	50	110	140	—-

Algoritmul lui Floyd

- Algoritmul lui Warshall reprezintă o modalitate rapidă de a crea un tabel care indică vârfurile în care se poate ajunge dintr-un anumit vârf, într-unul sau mai mulți pași
- O abordare similară pentru grafuri ponderate este folosită de algoritmul lui Floyd, descoperit de Robert Floyd în 1962



	A	B	C	D
A				
B	70			10
C	30			
D			20	

Observații

- Matricea de adiacență indică costurile tuturor căilor cu o singură muchie
- Se dorește extinderea acestei matrici pentru a indica costurile tuturor căilor, indiferent de lungimea lor
- De exemplu, se poate ajunge de la B la C cu un cost de 30 (10 de la B la D și 20 de la D la C)

Observații

- Similar algoritmului lui Warshall, se modifică matricea de adiacență
- Se examinează fiecare celulă de pe fiecare rând
- Dacă există o pondere pozitivă, de exemplu 30 la intersecția liniei C cu coloana A, atunci se analizează coloana C, deoarece C reprezintă linia unde se află 30

Observații

- Dacă se găsește o celulă în coloana C, de exemplu 40 la linia D, atunci există o cale de la C la A cu o pondere de 30 și o cale de la D la C cu o pondere de 40
- Se poate deduce că există o cale cu două muchii de la D la A, cu o pondere de 70

Observații

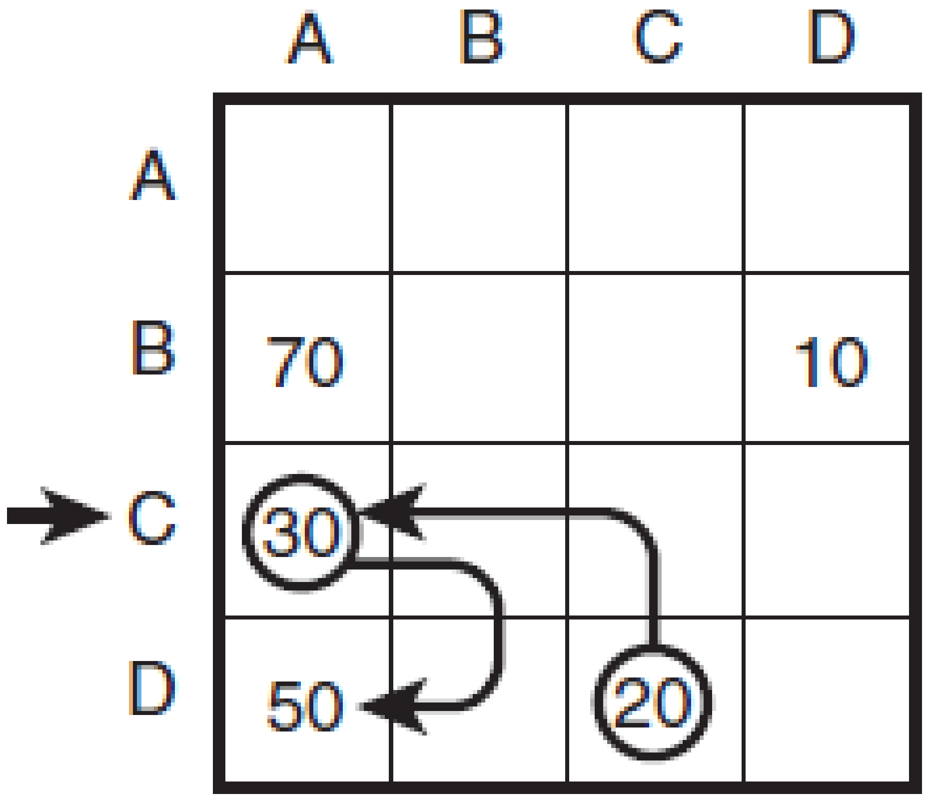
- Linia A este vidă
- Pe linia B este 70 în coloana A și 10 în coloana D, dar coloana B este vidă, deci muchiile care încep din B nu pot fi combinate cu nicio muchie care se termină în B

Observații

- În linia C se află 30 pe coloana A
- În coloana C se află 20 pe linia D
- Muchia de la C la A are o pondere de 30
- Muchia de la D la C are o pondere de 20
- Se obține calea de la D la A cu ponderea de 50

a)

$y = 2, x = 0, z = 3$



$C \rightarrow A$ & $D \rightarrow C$

30 20

so $D \rightarrow A$

50

Observații

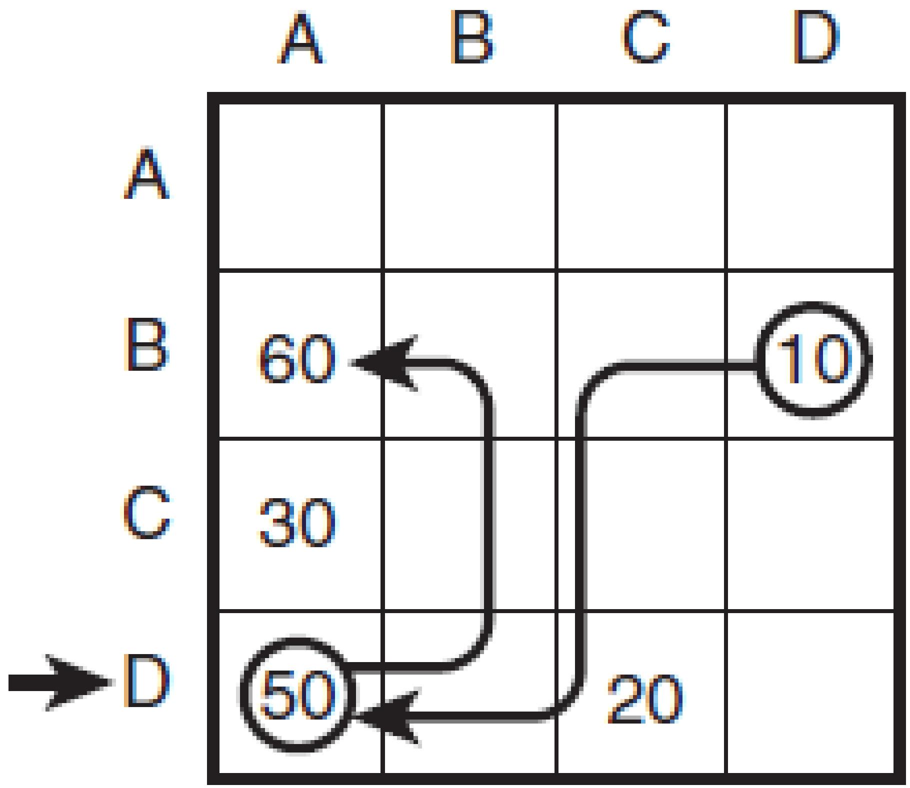
- Linia D arată o situație interesantă – se poate micșora un cost existent deja
- Pe linia D există 50 în coloana A
- Pe linia B există 10 în coloana D
- Există o cale de la B la A cu costul 60
- Cu toate acestea, există deja costul 70 pe linia B, în coloana A

Observații

- Deoarece 60 e mai mic decât 70, se înlocuiește 70 cu 60
- În cazul căilor multiple de la un vârf la altul, tabelul indică calea de cost minim

b)

$y = 3, x = 0, z = 1$



$D \rightarrow A$ & $B \rightarrow D$

50 10

so $B \rightarrow A$

60

c)

$y = 3, x = 2, z = 1$

	A	B	C	D
A				
B	60		30	10
C	30			
D	50		20	

$D \rightarrow C$ & $B \rightarrow D$

20 10

so $B \rightarrow C$

30

Observații

- Implementarea algoritmului lui Floyd este similară algoritmului lui Warshall
- În locul inserării valorii 1 în tabel, cum se procedează în algoritmul lui Warshall, când se găsește o cale cu două muchii, se adaugă costul căii cu două muchii și se inserează suma costurilor celor două muchii în tabel

Observații

- Algoritmul lui Floyd se bazează pe utilizarea unei matrice **A** a distanțelor minime, ale cărei valori sunt calculate în mai multe etape
- Inițial:
- $A[i,j] = cost[i,j]$, pentru orice $i \neq j$
- $A[i,j] = 0$, pentru $i=j$
- $A[i,j] = \infty$, dacă nu există arcul (i,j)

Observații

- Calculul distanțelor minime se face în **n** iterații
- La iterația **k**, **A[i,j]** va avea ca valoare cea mai mică distanță între **i** și **j**, pe căi care nu conțin vârfuri numerotate peste **k** (exceptând capetele **i** și **j**)
- Se utilizează formula:
- $A_k[i,j] = \min (A_{k-1}[i,j], A_{k-1}[i,k] + A_{k-1}[k,j])$

Observații

- Deoarece
- $A_k[i,k] = A_{k-1}[i,k]$ și
- $A_k[k,j] = A_{k-1}[k,j]$
- nicio intrare cu unul din indici egal cu **k** nu se modifică la iterația **k**
- Se poate realiza calculul cu o singură copie a matricei A

```
AlgoritmFloyd() {
pentru toate liniile i execută
   pentru toate coloanele j execută
                  A[i,j] ← cost[i,j]
   pentru toate liniile i execută
                  A[i,i] ← 0
   pentru k de la 1 la n execută
      pentru toate liniile i execută
         pentru toate coloanele j execută
            dacă A[i,k] + A[k,j] < A[i,j] atunci
                           A[i,j] ← A[i,k] + A[k,j]
}
```

Observații

- Pentru a păstra căile minime, se utilizează un tablou adițional **P**, unde **P[i,j]** reprezintă acel vârf **k**, care a condus la distanța minimă **A[i,j]**
- Dacă **P[i,j] = 0**, atunci arcul **(i,j)** este calea minimă între **i** și **j**

Observații

- Pentru a afișa vârfurile intermediare aflate pe calea cea mai scurtă între **i** și **j**, se poate utiliza algoritmul:
- Cale(i,j) {
- k ← P[i,j]
- dacă (k != 0) atunci {
- Cale(i,k)
- Scrie nodul k
- Cale(k,j) }
- }

Concluzii

- Într-un graf ponderat, fiecare muchie are asociat un număr, numit pondere
- Ponderile pot reprezenta distanțe, costuri, timpi sau alte mărimi
- Arborele minim de acoperire, într-un graf ponderat, minimizează ponderile muchiilor necesare pentru a conecta toate vârfurile

Concluzii

- Pentru determinarea arborelui minim de acoperire al unui graf, putem utiliza o coadă cu priorități
- Problema drumului minim într-un graf neponderat presupune determinarea numărului minim de muchii dintre două vârfuri

Concluzii

- Rezolvarea problemei drumului minim, în cazul grafurilor ponderate, se poate face utilizând algoritmul lui Dijkstra
- Rezolvarea problemei drumului de lungime minimă din orice vârf înseamnă găsirea costurilor totale ale muchiilor între toate perechile de vârfuri ale unui graf; această problemă se rezolvă folosind algoritmul lui Floyd