Grafuri orientate. Aplicaţii

Analiza algoritmilor

-laborator-

conf. dr. ing. Ciprian-Bogdan Chirilă

Cuprins

- Problema drumurilor minime cu origine unică
  - Algoritmul lui Djikstra.
- Problema drumurilor minime corespunzătoare tuturor perechilor de noduri
  - Algoritmul lui Floyd.
- Traversarea grafurilor orientate
- Grafuri orientate aciclice
  - Sortare topologică.

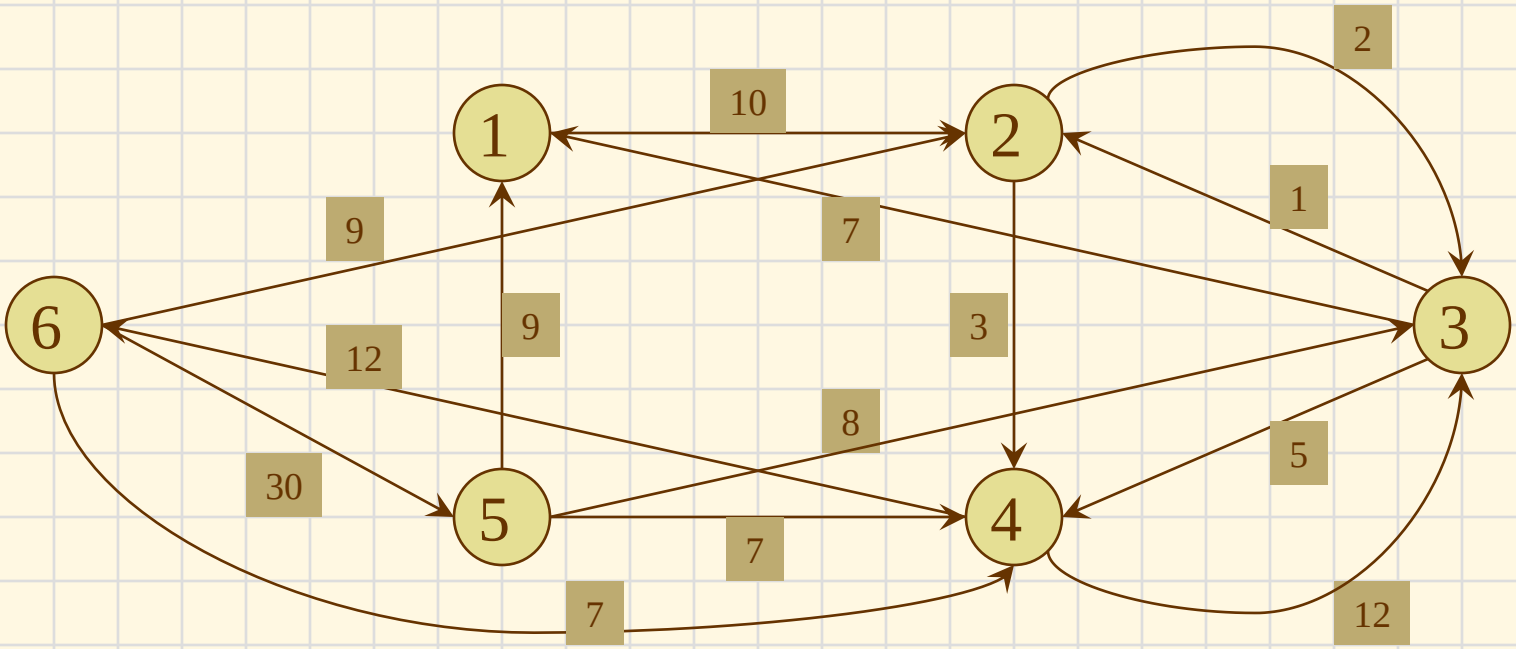
Problema drumurilor minime cu origine unică

- G=(N,A) graf ponderat orientat
- fiecare arc(i,j) are un Cost[i,j]
- M - mulţimea nodurilor din N pentru care distanţa minimă până la origine este cunoscută
- D[i] - lungimea celui mai scurt drum de la nodul i la origine

Problema drumurilor minime cu origine unică (algoritm)

1. Se selectează un nod x din mulţimea N\M cât mai aproape de origine şi se introduce în mulţimea M;
2. Pentru fiecare nod y din N\M se verifică dacă nu există un drum mai scurt de la origine la el care să treacă prin x;
3. Dacă N\M<>{} se execută pasul 1.

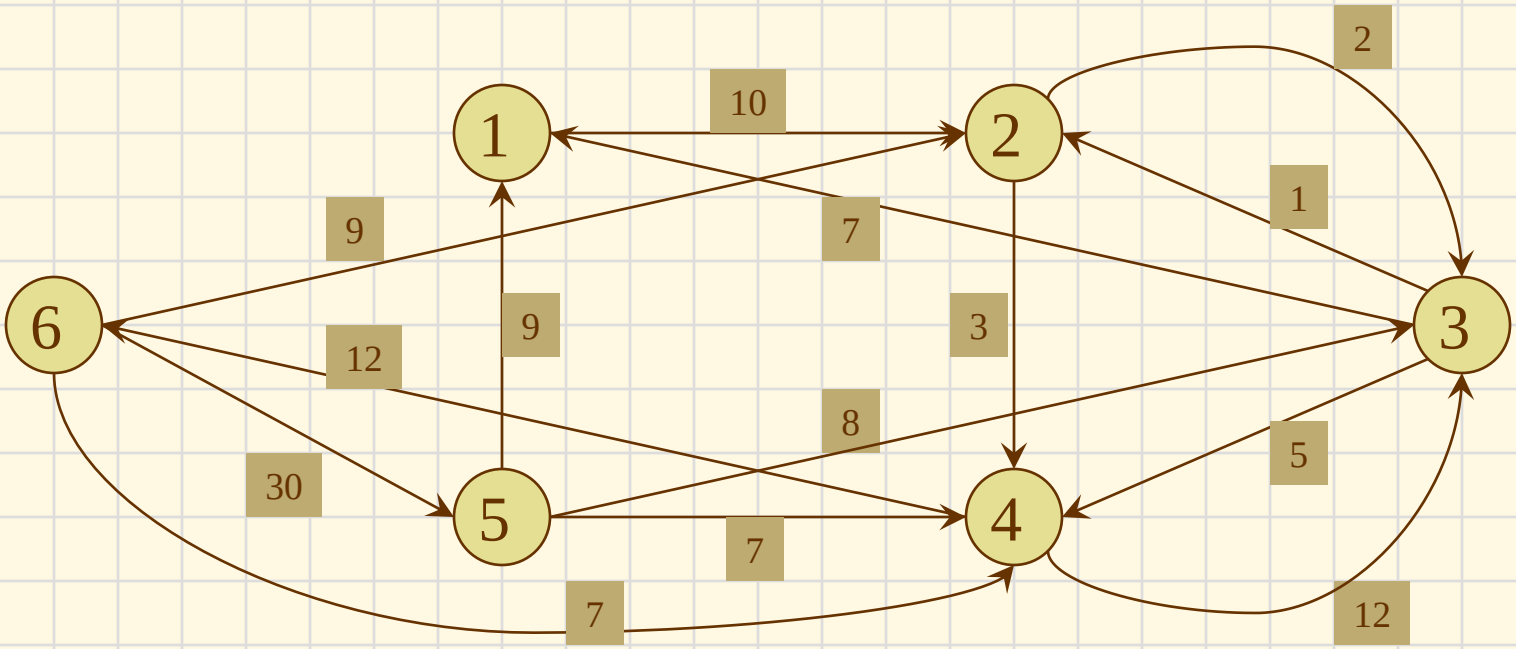
Algoritmul lui Djikstra (demo 1)



x=1; M={1}; N\M={2, 3, 4, 5, 6}

y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	∞	∞	∞	∞
D[x]+C[x,y]:	0	10	∞	∞	∞	∞
$D[y]_{final}$:	0	10	∞	∞	∞	∞

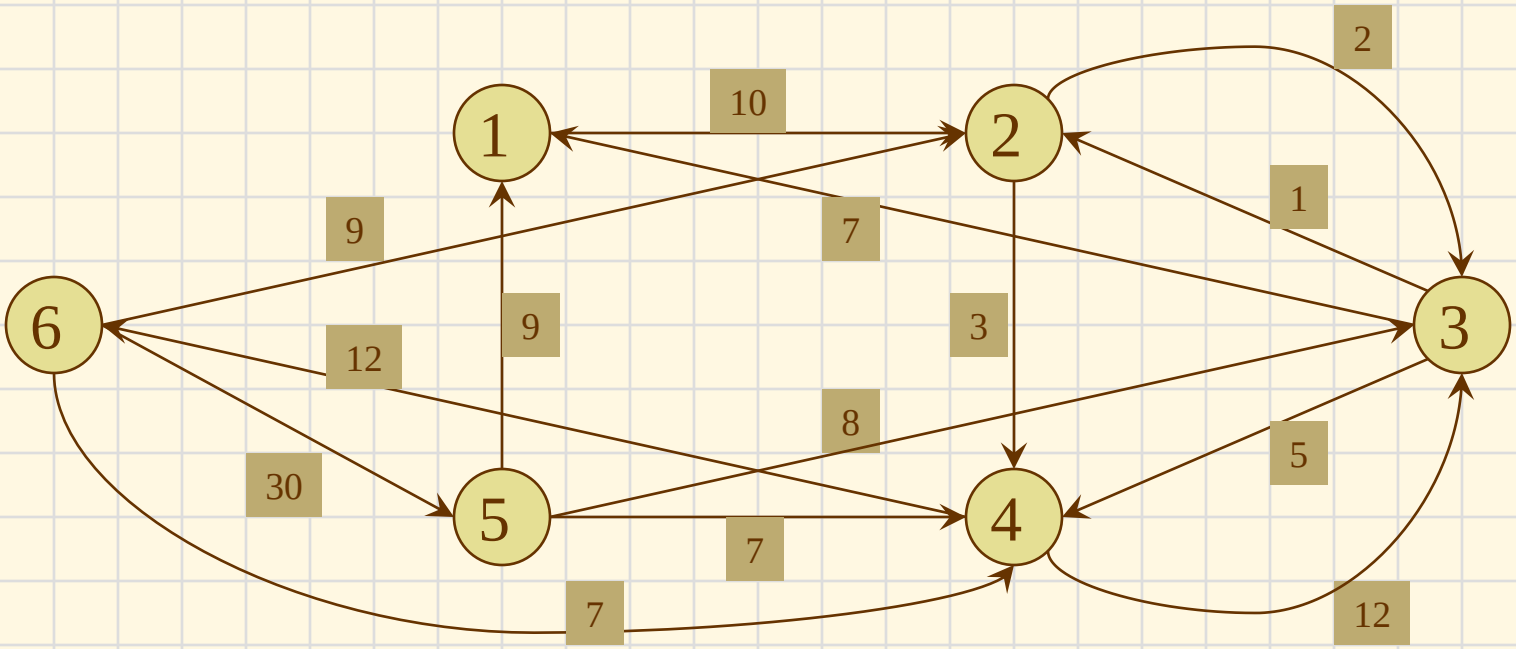
Algoritmul lui Djikstra (demo 2)



x=2; M={1,2}; N\M={3, 4, 5, 6}

y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	∞	∞	∞	∞
D[x]+C[x,y]:	0	10	12	13	∞	∞
$D[y]_{final}$:	0	10	12	13	∞	∞

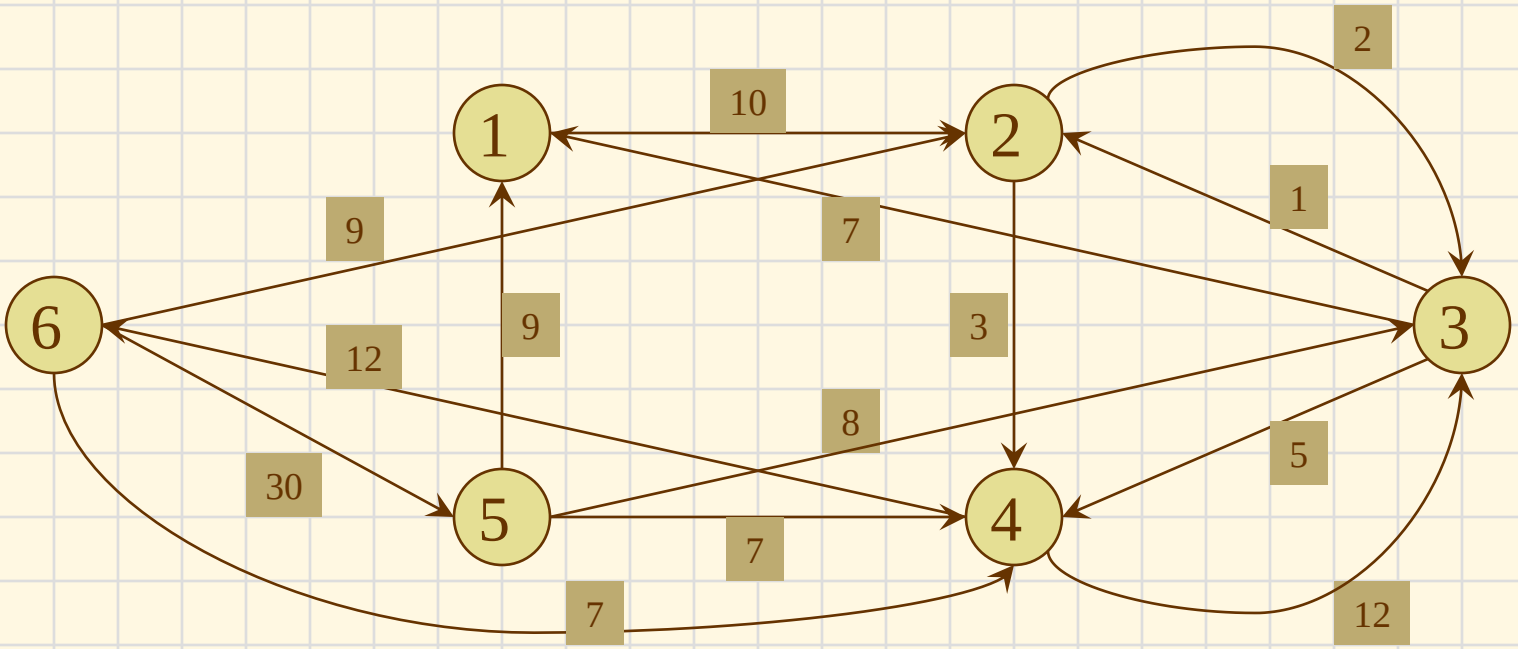
Algoritmul lui Djikstra (demo 3)



x=3; M={1,2,3}; N\M={4, 5, 6}

y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	12	13	∞	∞
D[x]+C[x,y]:	0	10	12	17	∞	∞
$D[y]_{final}$:	0	10	12	13	∞	∞

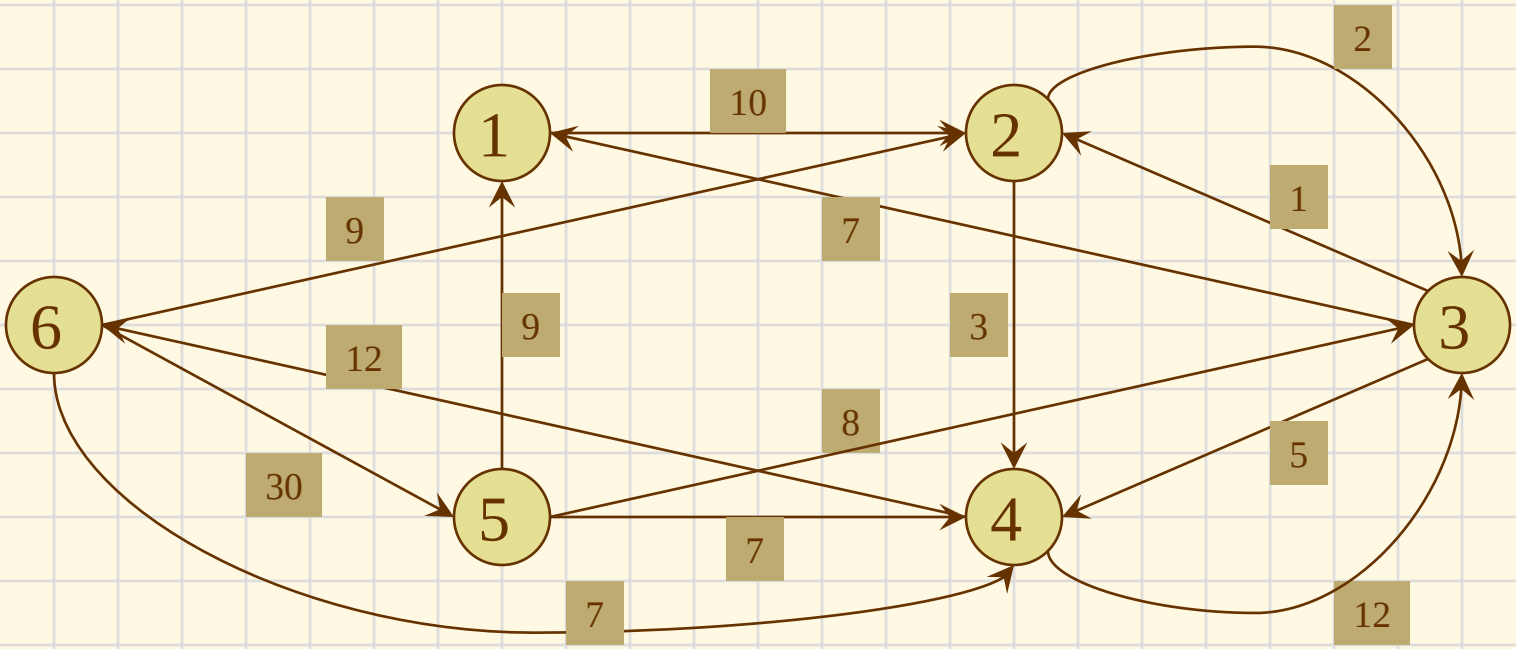
Algoritmul lui Djikstra (demo 4)



x=4; M={1,2,3,4}; N\M={5, 6}

y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	12	13	∞	∞
D[x]+C[x,y]:	0	10	12	13	∞	25
$D[y]_{final}$:	0	10	12	13	∞	25

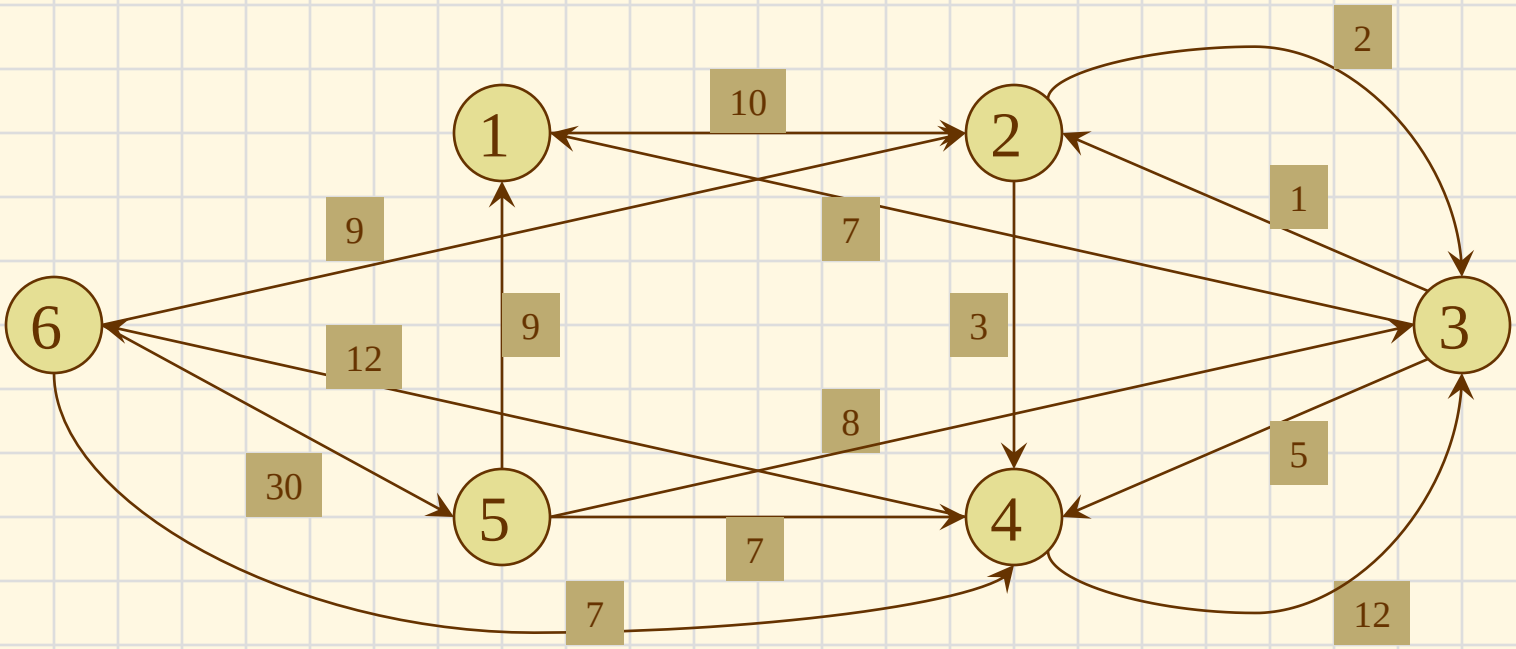
Algoritmul lui Djikstra (demo 5)



x=6; M={1,2,3,4,6}; N\M={5}

y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	12	13	∞	25
D[x]+C[x,y]:	0	10	12	13	55	25
$D[y]_{final}$:	0	10	12	13	55	25

Algoritmul lui Djikstra (demo 6)



x=5; M={1,2,3,4,5,6}; N\M={}

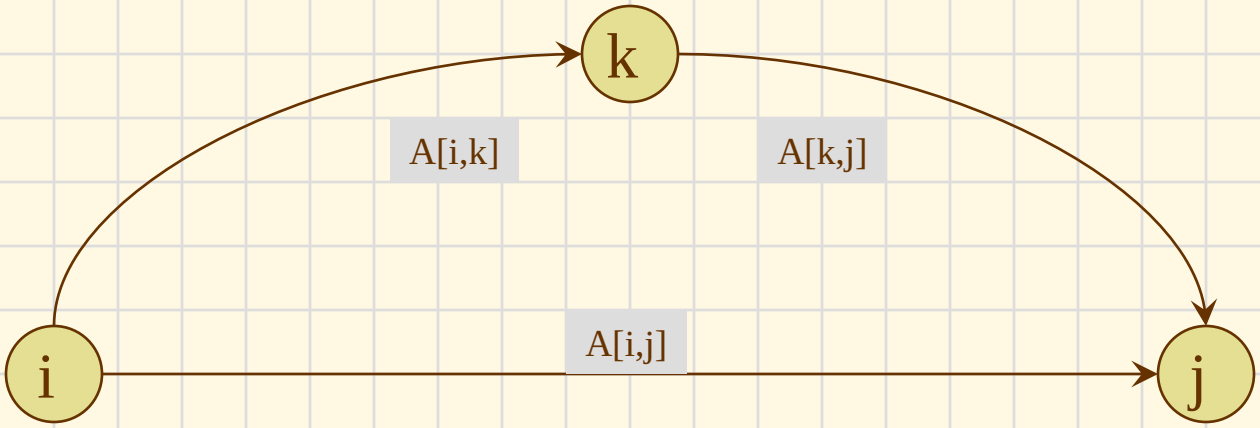
y:	1	2	3	4	5	6
$D[y]_{iniţial}$:	0	10	12	13	55	∞
D[x]+C[x,y]:	0	10	12	13	55	25
$D[y]_{final}$:	0	10	12	13	55	25

Algoritmul lui Djikstra (pseudocod)

```
procedure Dijkstra;
{Date de intrare: Graful orientat ponderat G=(N,A) şi nodul  nod_origine}
{Date de ieşire: pentru fiecare nod i, D[i] este lungimea  calculată a drumului minim de la
      nod_origine la i}
begin
* initializează mulţimea M a nodurilor vizitate, introduce nod_origine în M
* pentru toate nodurile i, iniţializeaza lungimea drumului minim până la i (D[i]) cu lungimea
      arcului direct de la nod_origine la i
while ( * mai există noduri în N-M ) do
begin
 * alege un nod x aparţinând mult imii N - M astfel încât D[x] să fie minim şi îl adaugă pe x
      la M;
 for * fiecare nod y din N-M do
  * test dacă drumul de la nod_origine la y folosind ca punct intermediar nodul x este mai
      scurt decât cel memorat  anterior, şi dacă da, actualizează D[y] în forma
  D[y] := min(D[y],D[x] + COST[x,y]);
 end
end; {Dijkstra}
```

Problema drumurilor minime corespunzătoare tuturor perechilor de noduri

- G=(N,A) graf ponderat orientat
- fiecare arc(i,j) are un Cost[i,j]
- A – matrice de dimensiune NxN
- A[i,j] – lungimea drumului minim de i la j



Algoritmul lui Floyd pseudocod (1)

```
procedure Floyd;
{Date de intrare: Graful orientat ponderat G=(N,A)}
{Date de ies ire: pentru fiecare pereche de noduri i,j se calculează A[i,j]=lungimea drumului
     minim de la i la j, şi Drum[i,j]=un nod intermediar pe traseu de la i la j}
begin
* iniţializeaza matricea lungimilor drumurilor minime A[i,j]) şi matricea  nodurilor
     intermediare de pe trasee (Drum[i,j]) presupunând că drumul minim  între oricare două
     noduri este arcul direct
for * fiecare nod k al grafului do
 for * fiecare pereche de noduri i,j do
  if (* suma drumurilor de la i la k si de la k la j e mai mica decât drumul de la i la j
      (A[i,k] + A[k,j] < A[i,j] )) then
  begin
   * actualizeaza valoarea drumului minim, prin
   A[i,j] := A[i,k] + A[k,j];
   * memorează punctul de pe traseu, Drum[i,j] := k
  end;
end; {Floyd}
```

Algoritmul lui Floyd pseudocod (2)

```
procedure Traseu(i,j:integer);
{Date de intrare: nodurile i şi j şi matricea Drum calculată de algoritmul lui Floyd}
{Date de ieşire: afişează în ordine nodurile intermediare de pe traseul drumului minim de la
      i la j}
begin
 if ( *există cel puţin un nod k=Drum[i,j] intermediar de la i la j) then
 begin
  Traseu(i,k);
  writeln(k);
  Traseu(k,j);
 end;
end; { Traseu }
```

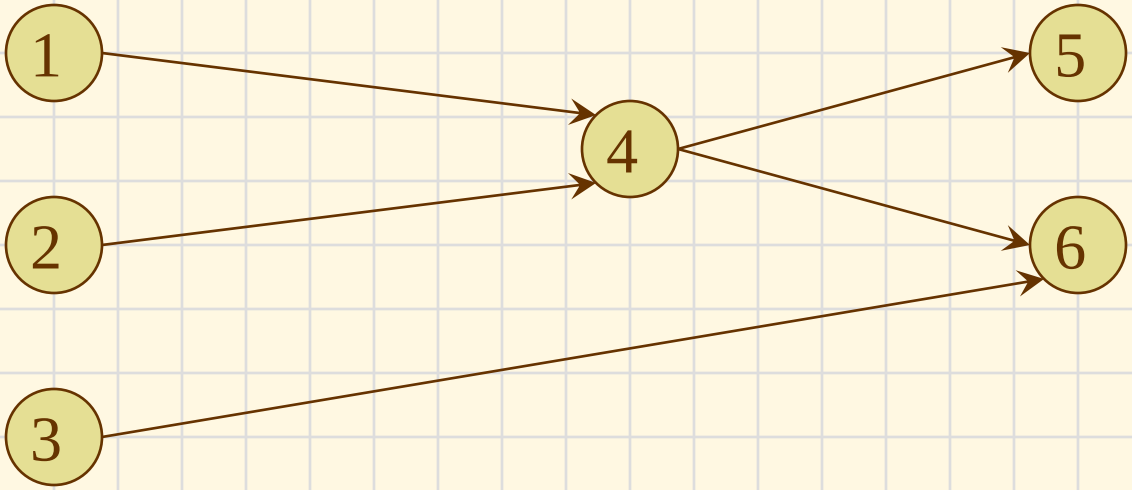
Traversarea grafurilor orientate

- traversare în adâncime
- traversare prin cuprindere
- prezentate în lucrarea precedentă
- se ţine cont de orientarea arcelor traversate

Relaţie de ordine parţială

- reflexivă: a R a;
- tranzitivă: a R b şi b R c atunci a R c;
- antisimetrică: a R b atunci nu b R a;

Grafuri orientate aciclice. Sortare topologică.



- Parcurgere adâncime: 5 4 6 1 2 3;
- Sortare topologică: 3 2 1 6 4 5;
- Altă variantă: 3 2 1 4 6 5;

Aplicaţie

- transportul de călători între n localităţi ale unui judeţ este asigurat cu ajutorul unor autobuze.
- intre oricare două localităţi I şi J există cel mult un autobuz direct, care merge o dată pe zi, având un orar cunoscut (ora de plecare din I şi ora de sosire în J, numere întregi).
- să se găsească timpul total în care un călător poate ajunge dintr-o localitate X în altă localitate Y.
- se consideră că persoana se află în staţia din localitatea X la ora 0 şi trebuie să ajungă în Y până la ora 24 a aceleaşi zile.

Concluzii

- Drumuri minime cu origine unică = determinarea drumurilor minime (lucrarea precedentă);
- Algoritmul lui Floyd($O(N^3)$) = N * Algoritmul Djikstra($O(N^2)$);
- Sortare topologică = căutare în adâncime adaptată
  - planificare de activităţi;