Algoritmi Fundamentali - S1

NOTARE: - EXAMEN → 4p (minim nota 5 pt. promovare)
- LABORATOR (3TEME + COLOCVIU) → 5p
  (3TEME: 1,5p; COLOCVIU: 3,5p)
- SEMINAR → 1p (2 TEME)

Cod Teams: oirndlm ! **Platforme TEME:** CSES, Info Arena

$G = (V, E)$

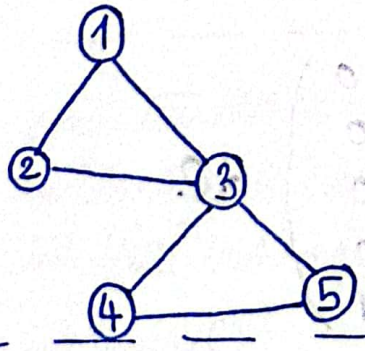
$V \rightarrow$ multimea vârfurilor

$E \rightarrow$ multimea muchiilor

$|V| = n$

$|E| = m$

Grafuri neorientate

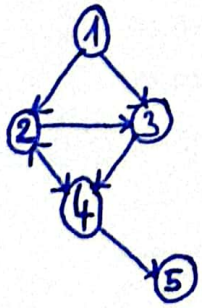


$V = \{1, 2, 3, 4, 5\}$

$E = \{[1,2], [1,3], [2,3], [3,4], [3,5], [4,5]\}$

$\sum_{x \in V} d(x) = 2m$

Grafuri orientate



$V = \{1, 2, 3, 4, 5\}$

$E = \{(1,2), (1,2), (2,3), (2,4), (4,2), (3,4), (4,5)\}$

$\sum_{x \in V} d^-(x) = \sum_{x \in V} d^+(x) = m$

Grade și vecini

$N(x) = \{y \mid [x,y] \text{ sau } [y,x] \in E\}$

$d(x) = |N(x)|$

$N(x) = \{y \mid (x,y) \in E\}$

$\overline{N}(x) = \{y \mid (y,x) \in E\}$

$d^-(x) = |N(x)|$

$d^+(x) = |\overline{N}(x)|$

GRAF CONEX:

G conex $\iff \forall x,y \in V$, $\exists$ lanț de la x la y

MEMORARE GRAF:

1) Matricea de adiacență:

$O(n^2)$

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$ G.N.

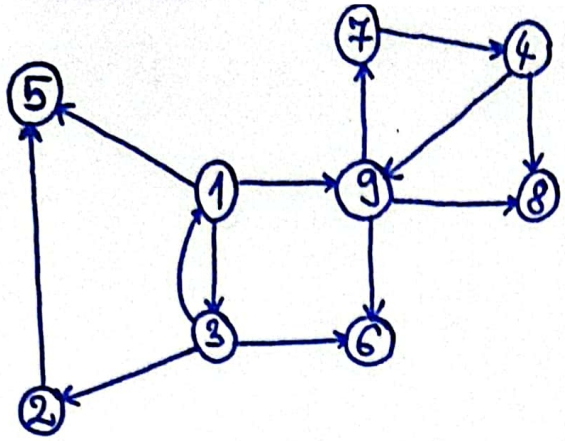
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$ G.O.

2) Listele de adiacență:

$O(n+m)$

$L_1 \to 2,3$
$L_2 \to 1,3$
$L_3 \to 1,2,4,5$
$L_4 \to 3,5$
$L_5 \to 3,4$

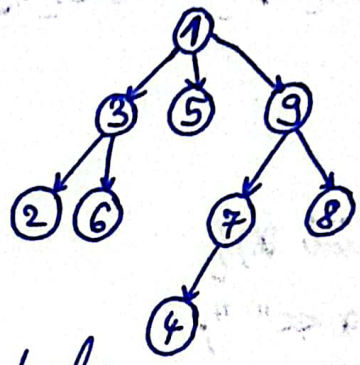
G.N.



BFS:

C: ~~1~~, ~~3~~, ~~5~~, ~~9~~, ~~2~~, ~~6~~, ~~7~~, ~~8~~, ~~4~~

1, 3, 5, 9, 2, 6, 7, 8, 4



v = vector de distanțe (de la 1 la restul nodurilor)

1	2	3	4	5	6	7	8	9
0	2	1	3	1	2	2	2	1

DFS:

S: ~~1~~, ~~3~~, ~~2~~, ~~5~~, ~~6~~, ~~9~~, ~~7~~, ~~4~~, ~~8~~

1, 3, 2, 5, 6, 9, 7, 4, 8

```
const int NMAX = 10^5;
vector<int> G[NMAX+1];
int vis[NMAX+1], d[NMAX+1];
int main(){
    int n, m;
    cin >> n >> m;
    for(i=1; i<=m; i++){
        int x, y;
        cin >> x >> y;
        G[x].push_back(y); }
    BFS(1);
    DFS(1);}
void BFS(int x){                    O(n+m)
    queue<int> q;
    q.push(x);
    d[x] = 0;
    vis[x] = 1;
    while(!q.empty()){
```

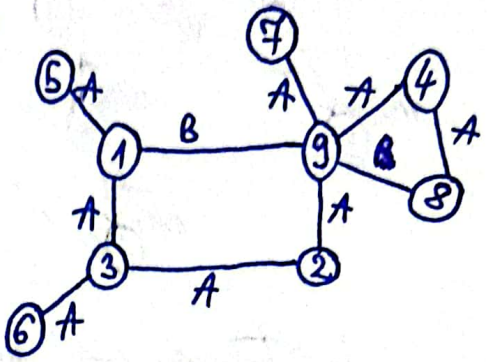
```
x = q.front(); q.pop();
cout << x << " ";
for (auto next : G[x]){
    if (! vis[next]){
        q.push(next);
        vis[next] = 1;
        d[next] = d[x] + 1;}
    }
}

void DFS(int x){
    cout << x << " ";
    vis[x] = 1;
    for (auto next : G[x]){
        if (! vis[next]) DFS(next);
    }
}
```

Algoritmi Fundamentali – S2

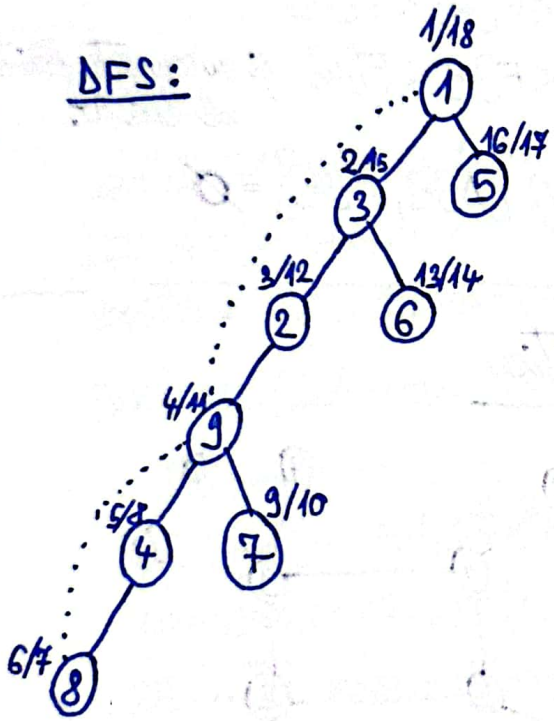
Tipuri de muchii/arce în parcurgerea DFS

1) Graf neorientat:

- muchii de arbore (A)
- muchii de întoarcere (B)



DFS:



$D[i]$ = timpul de descoperire al nodului i

$F[i]$ = timpul de finalizare al nodului i

(u,v) = muchie de întoarcere

$\Leftrightarrow$

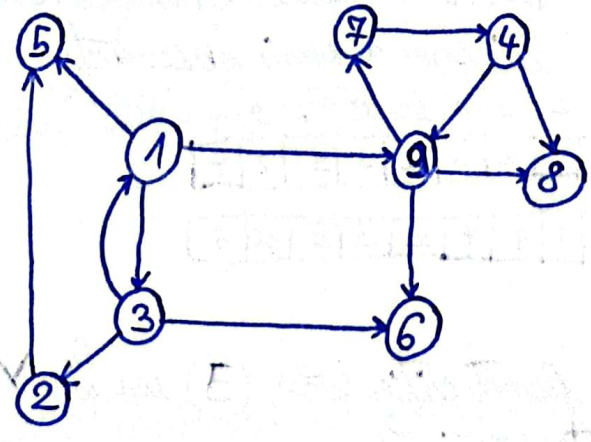
$D[v] < D[u] < F[u] < F[v]$

și v nu e părintele lui u

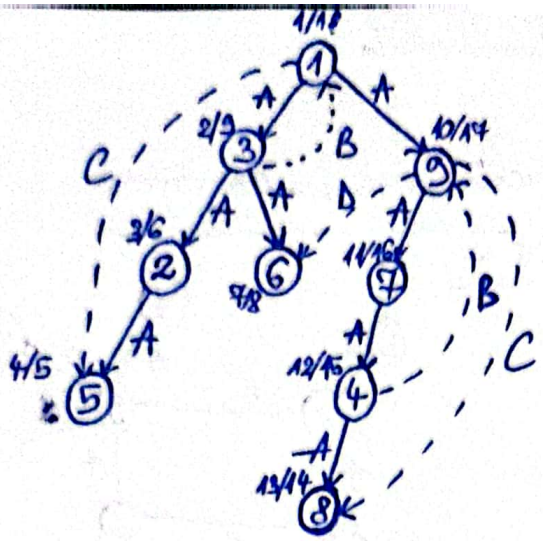
---

2) Graf orientat:

- arce de arbore (A)
- arce de întoarcere (B)
- arce de avansare (C)
- arce de traversare (D)



DFS:



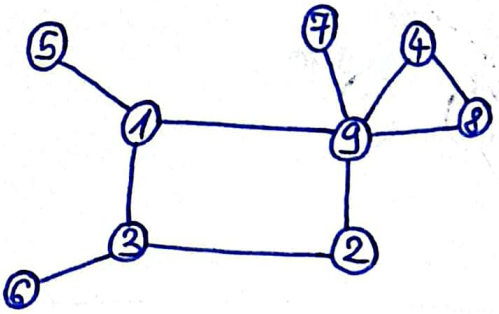
(u,v) = arc de întoarcere $\iff$ $D[v] < D[u] < F[u] < F[v]$

(u,v) = arc de avansare $\iff$ $D[u] < D[v] < F[v] < F[u]$ și u nu este fiul al lui v.

(u,v) = arc de trasversare $\iff [D[u], F[u]] \cap [D[v], F[v]] = \emptyset$

Puncte de articulație și muchii critice

COMPLEXITATE: $O(n+m)$



NIV[i] = nivelul în arborele DF

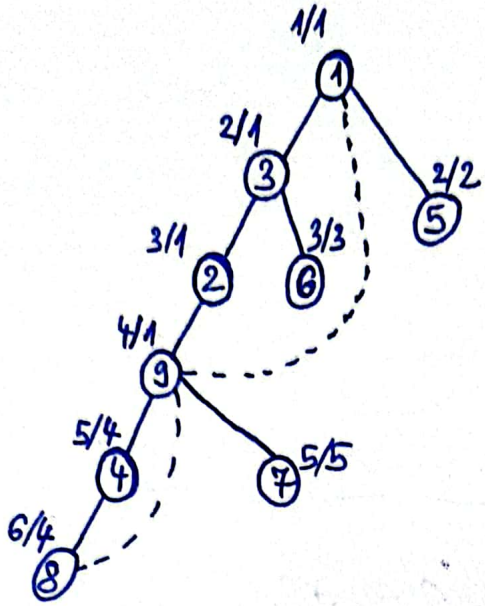
LOW[i] = minimul dintre nivelul său și nivelul la care poate ajunge acesta printr-o muchie de întoarcere a lui sau a unui descendent

	1	2	3	4	5	6	7	8	9
NIV	1	3	2	5	2	3	5	2	4
LOW	1	1	1	4	2	3	5	4	1

X este punct critic $\iff$ ($\exists$) un fiu Y a.î. $LOW[Y] \geq NIV[X]$ (nu este rădăcină)

Rădăcina este punct de articulație $\iff$ are cel puțin 2 fii în arborele DFS.

(X,Y) este muchie critică $\iff LOW[Y] > NIV[X]$

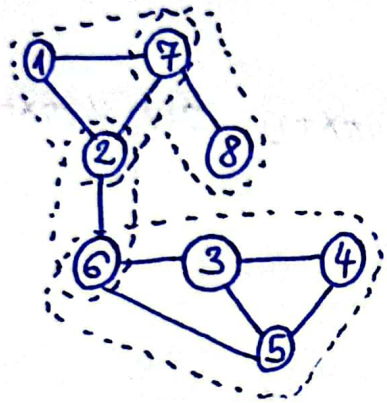


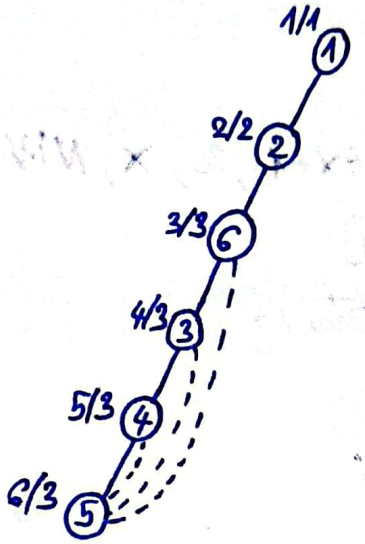
COD:

```
const int NMAX = 125;
int viz[NMAX+1], a[NMAX+1], r, x, NIV[NMAX+1], LOW[NMAX+1]
vector <int> G[NMAX+1]
void DFS(int nod, int tata){
   viz[nod]=1;
   NIV[nod]=NIV[tata]+1;
   LOW[nod]= NIV[NOD];
   for(auto next: G[nod]){
        if(viz[nod]==1){
            if(next!=tata)
                LOW[nod]=min(LOW[nod], NIV[next]);
        }
        else{ dfs(next,nod);
            if(nod==r) x++;
            else{ LOW[nod]=min(LOW[nod], LOW[next]);
                if(LOW[next]≥NIV[nod]) a[nod]=1;}
   }
}
```

```
int main (){
//Citire graf
rc=1;
dfs(rc,0);
if (x>=2) a[rc]=1; }
```

Componente biconexe

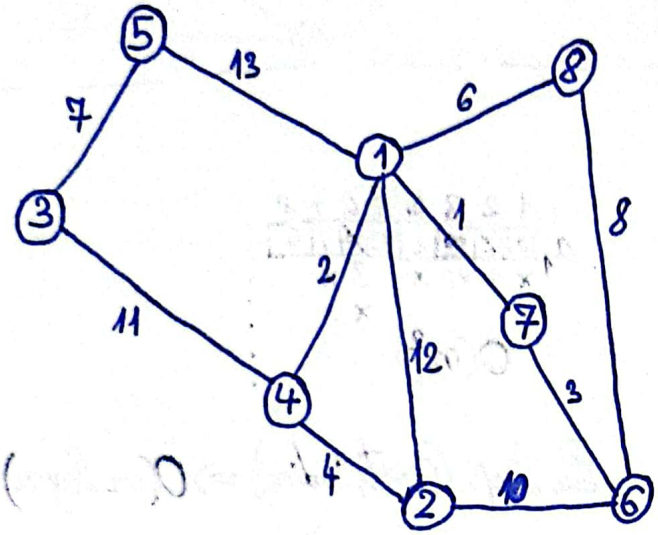




1, 2
(2, 6)
~~6, 3~~
~~3, 4~~
~~4, 5~~
~~5, 3~~
~~5, 6~~

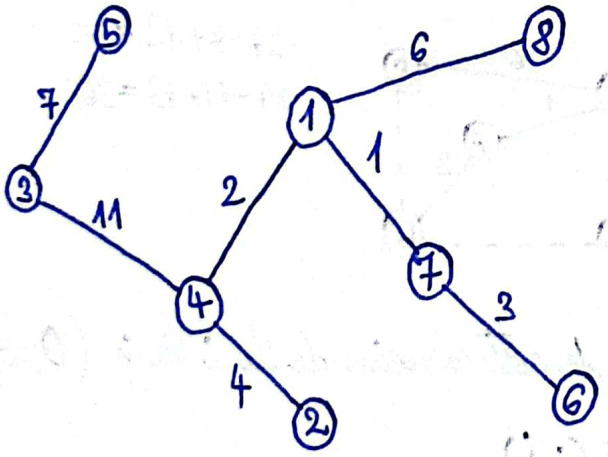
Algoritmi Fundamentali - S3

MST. Second-best MST



KRUSKAL: - sortarea: $(O(m \log m + m))$

[1,7] - 1
[1,4] - 2
[7,6] - 3
[4,2] - 4
[1,8] - 6
[5,3] - 7
[8,6] - 8
[2,6] - 10
[3,4] - 11
[1,2] - 12
[5,1] - 13

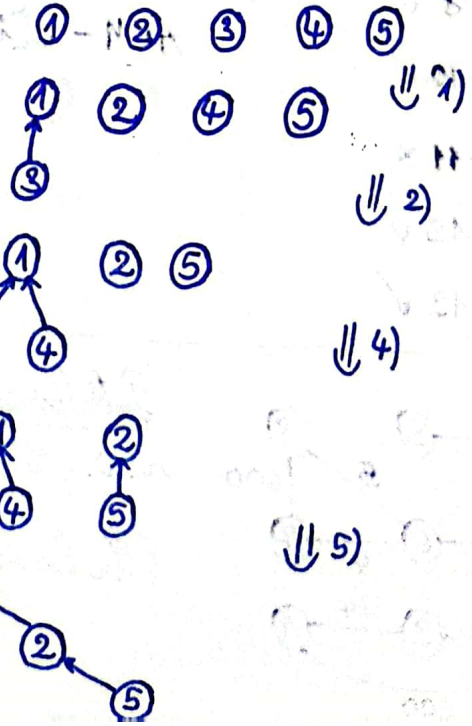


COST = 34

DSU/PĂDURI DE MULȚIMI DISJUNCTE

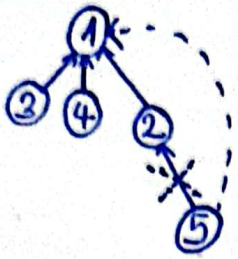
{1},{2},{3},{4},{5}
1XY (unește seturile cu X și Y)
2XY (se află X și Y în același set?)

1) 1 1 3 {1,3},{2},{4},{5}
2) 1 1 4 {1,3,4},{2},{5}
3) 2 2 5 NU
4) 1 2 5 {1,3,4},{2,5}
5) 1 1 5 {1,2,3,4,5}
6) 2 1 5 DA

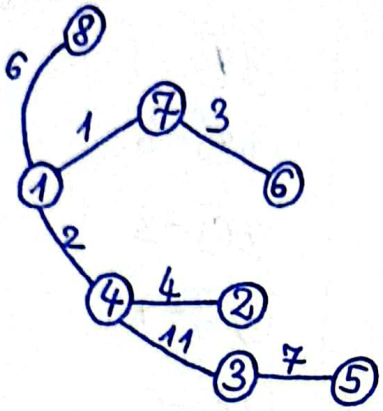


COMPLEXITATE: UNION - O(1)
FIND - O(n)

Dacă facem compresie de drumuri $\Rightarrow O(\log^* n) \simeq O(1)$



PRIM:

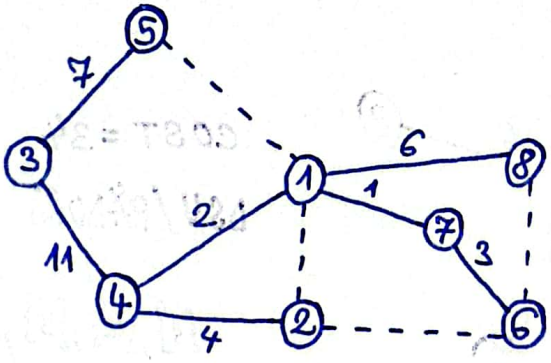


	1	2	3	4	5	6	7	8
d_1	0	12	∞	2	13	∞	1	6
	×	4	11	×		3	×	
						×		

$O(n^2)$

Cu heap ((cost, indice)) $\Rightarrow O(m \log n)$

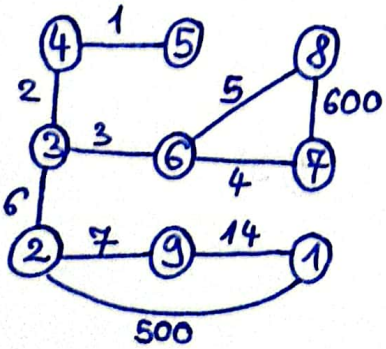
[1,7] - 1 ×
[1,4] - 2 ×
[7,6] - 3 ×
[4,2] - 4 ×
[1,8] - 6 ×
[5,3] - 7 ×
[8,6] - 8 ✓
[2,6] - 10 ✓
[3,4] - 11 ×
[1,2] - 12 ✓
[5,1] - 13 ✓



34 - 6 + 8 = 36
34 - 4 + 10 = 40
34 - 4 + 12 = 42
34 - 11 + 13 = 36

D[i,j] = muchia de cost maxim de la i la j ($O(n^2)$)

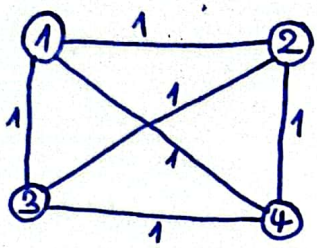
APM - D[i,j] + C(i,j)



Întrebare de examen: Dați valori/costuri muchiilor a.î. să se creeze un APM de cost 42.

(În loc de 500, 600, pot să pun orice nr. > 14)

Ex: Câte APCM-uri pot scoate din graful:



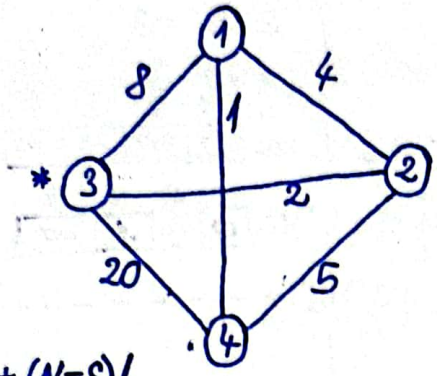
Răspuns: Într-un graf complet $\Rightarrow n^{n-2}$ APM-uri

$n=4 \Rightarrow 4^{4-2} = 4^2 = 16$

Algoritmi Fundamentali - S4

NATURE RESERVE KATTIS:

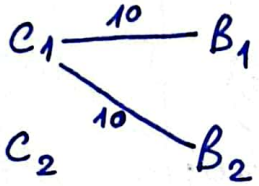
$N = 4$
$M = 6$
$L = 10$
$S = 1$
3



Răspuns: $APM + (N-S)L$
costul stâlpilor

REȚEA 2:

N centrale ≤ 2000
M blocuri ≤ 2000



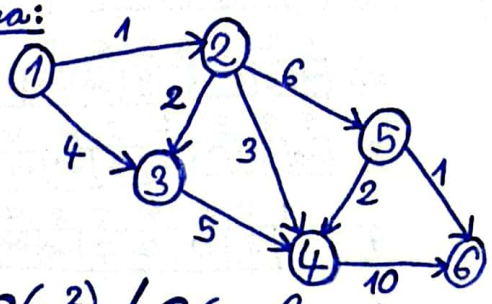
$N = 2$ și $M = 2$

0 0 0 10
10 11 10 0

$$cost(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Dijkstra
Bellman Ford
Roy Floyd

Dijkstra:



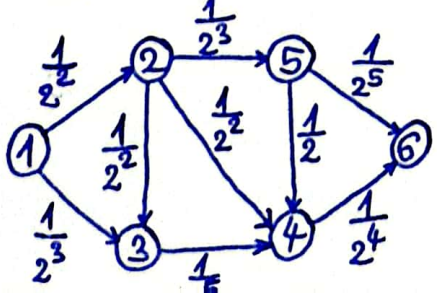
$s = 1$

	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞
1)	0	1	4	∞	∞	∞
2)	0	1	3	4	7	∞

COMPLEXITATE: $O(n^2)$ / $O(m \log n)$

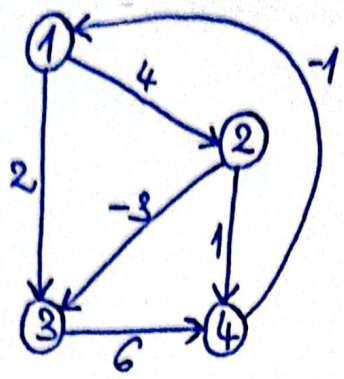
$d[next] > d[nod] + cost(nod, next) \Rightarrow d[next] = d[nod] + cost(nod, next)$

Ex:



Reținem costul ca fiind i pentru $\frac{1}{2^i}$ și apoi facem drum minim.

Bellman Ford:



3 4 6
2 4 1
4 1 -1
1 3 2
1 2 4
2 3 -3

```
struct edge { int x, y, c; }
s = sursa
d[s] = 0
d[x] = ∞ (∀ x ≠ s)
for (int i=1; i<n; i++)
    for(auto e : Edges)
        if (d[e.y] > d[e.x] + e.c)
            d[e.y] = d[e.x] + e.c;
```

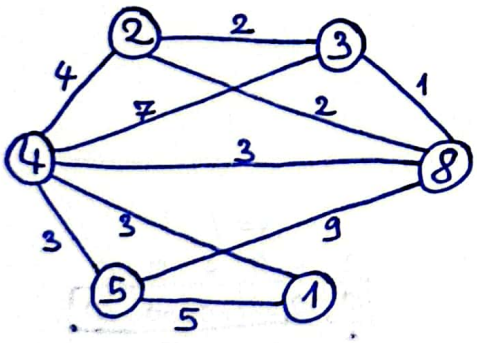
d: | 0 | ∞ | ∞ | ∞ | (1 2 3 4)

1) | 0 | 4 | 1 | ∞ | (1 2 3 4)

2) | 0 | 4 | 1 | 5 | (1 2 3 4)

(n-1 pași în total)

COMPLEXITATE: $O(n \cdot m)$



s = 3

1	2	3	4	5	6	7	8
∞	∞	0	∞	∞	∞	∞	∞

1)

1	2	3	4	5	6	7	8
∞	2	0	7	∞	∞	∞	1

2)

1	2	3	4	5	6	7	8
∞	2	0	4	9	∞	∞	1

3)

1	2	3	4	5	6	7	8
∞	2	0	4	9	∞	∞	1

4)

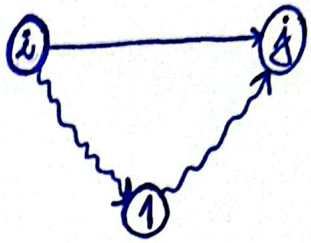
1	2	3	4	5	6	7	8
7	2	0	4	7	∞	∞	1

Roy Floyd:

$D[i][j]$ = drumul de cost minim de la i la j

$D[x][y] = c$

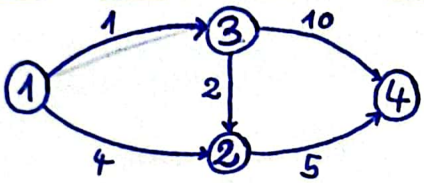
$D[i][i] = 0$



Pseudocod:

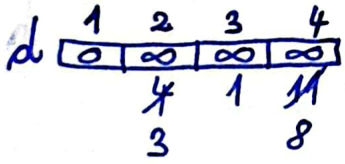
```
for (int k=1; k<=n; k++)
    for(int i=1; i<=n; i++)
        for(int j=1;j<=n;j++)
            D[i][j] = min(D[i][j], D[i][k]+D[k][j])
```

DAG:



COMPLEXITATE: $O(n+m)$

Sort. top.: 1 3 2 4

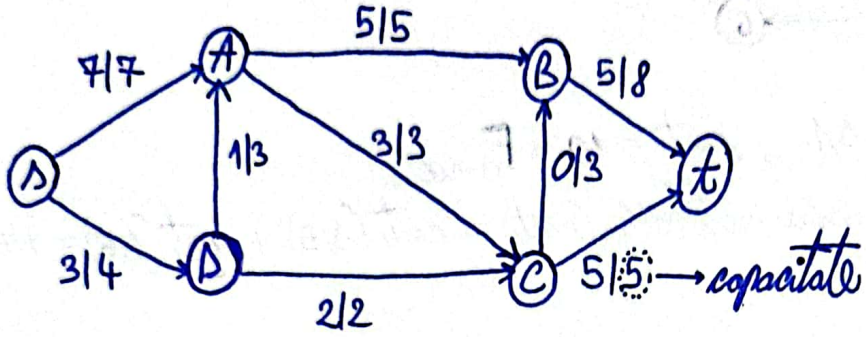


Pseudocod:

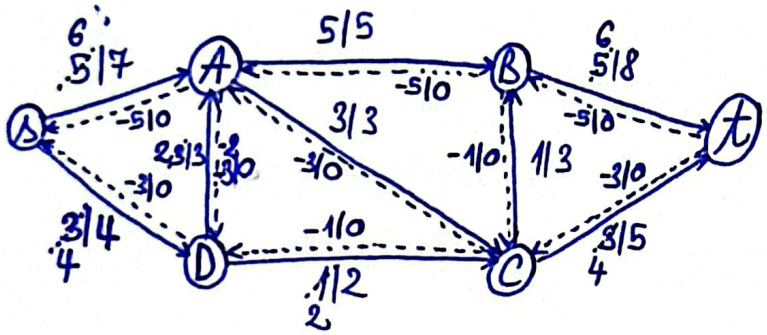
```
for(auto x : topsort)
    for(auto next : G[x])
        d[next] = min(d[next], d[x]+c[x,
```

Algoritmi Fundamentali - S5

Fluxe maxim / Tăietură maximă.
Cuplaj maxim în graf bipartit.



C. MAXIMĂ = 10

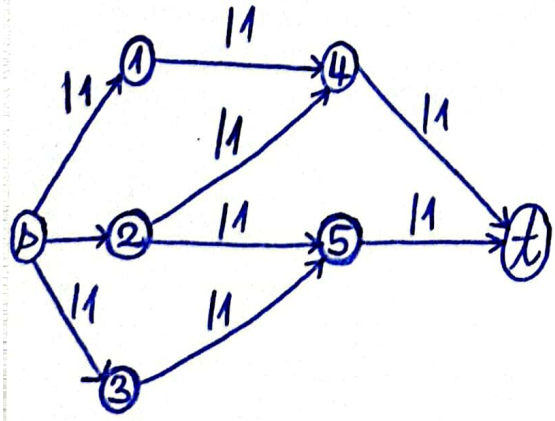


sABt → 5
sDACt → 3
sDCBt → 1
sADCt → 1

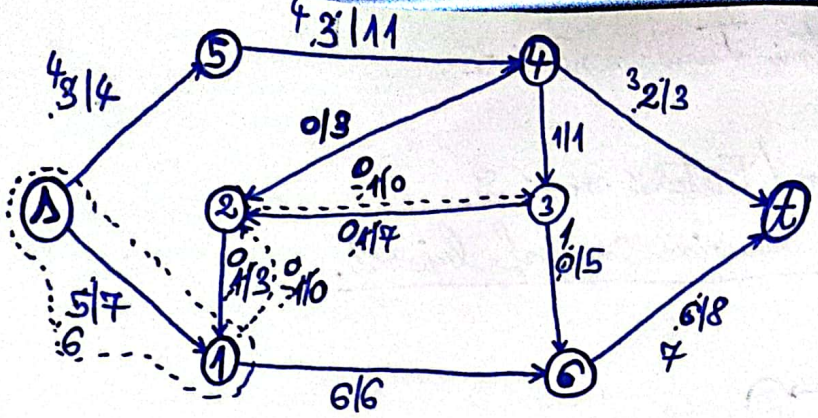
FORD-FULKERSON: $O(m F_{max})$

EDMONDS-KARP: $O(n m^2)$

Fluxul maxim = Tăietura minimă



Cuplaj maxim în graf bipartit



$s543 \longrightarrow 1$

$s1236t \longrightarrow$

Tăietura maximă: $s,1$, $cost = 10 = F_{max}$

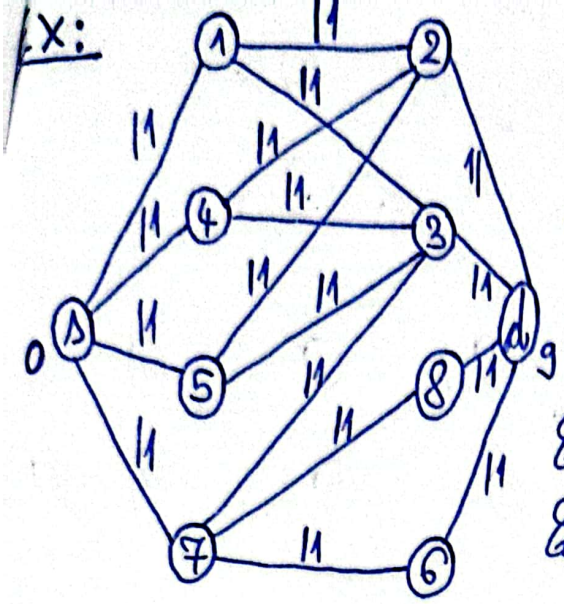
$cost = \sum$ muchiile în care ne blocăm $(cost) = cost(s,5) + cost(1,6) = 4+6=10$

EDMONDS – KARP

```
const int NMAX = 101;
int capacitate[NMAX][NMAX];
int fluxe[NMAX][NMAX];
int p[NMAX];
vector<int> G[NMAX];

void EK(int s, int d){
    queue<int> q; int viz[NMAX]={0};
    int flow = ∞;
    q.push(s);
    while(!q.empty()){
      int nod = q.front();
      q.pop();
      for(auto next: G[nod]){
          if(capacitate[nod][next] - fluxe[nod][next] > 0 && !viz[next]){
                p[next] = nod;
                flow = min(flow, capacitate[nod][next] - fluxe[nod][next]);
                q.push(next);
                viz[next] = 1;
          }
      }
    }
    return flow;
}
```

```
int maxFlow (int s, int d){
    int suma=0;
    while (true){
        int f = EK(s, d);
        suma = suma + f;
        if (f==0)
            break;
        else
            while (d!=s){
                flux[p[d]][d] += f;
                flux[d][p[d]] -= f;
                d=p[d];
            }
    }
    return suma;
}

int main(){
  int n, m; cin>>n>>m;
  int s, d; cin>>s>>d;
  for (int i=1; i<=m; i++){
      int x, y, c; cin>>x>>y>>c;
      capacitate[x][y] = c;
      G[x].push(y); G[y].push(x);
  }
  cout<<maxFlow(s, d);
}
```

Ex:



$N = 8$

$M = 9$

Să se determine cuplajul maxim.

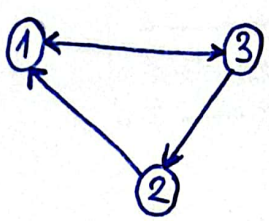
Cuplaj valid: $\{(1,2), (4,3)\}$

Cuplaj invalid: $\{(1,2), (4,2)\}$

Cuplaje maxime: $\{(1,2),(4,3),(7,8)\}$, $\{(1,2),(4,3),(7,6)\}$, $\{(1,2),(5,3),(7,6)\}$, $\{(1,2),(5,3),(7,8)\}$, $\{(1,3),(4,2),(7,8)\}$, ...

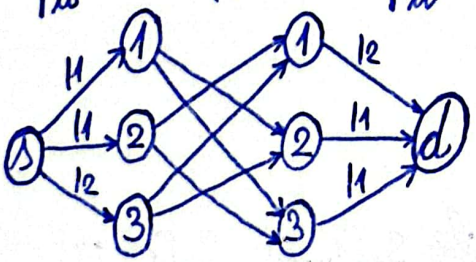
Ex: $N=3$

d^+	2	1	1
d^-	1	1	2



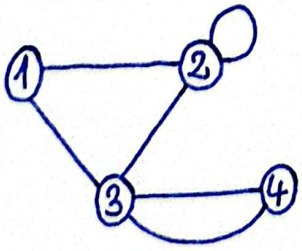
Cerința: Construiți un graf orientat cu gradele de intrare și ieșire date.

$|d^-$ 11 $|d^+$



Rulăm maxFluxe pe graf și apoi ne uităm ce noduri au fluxul 1.

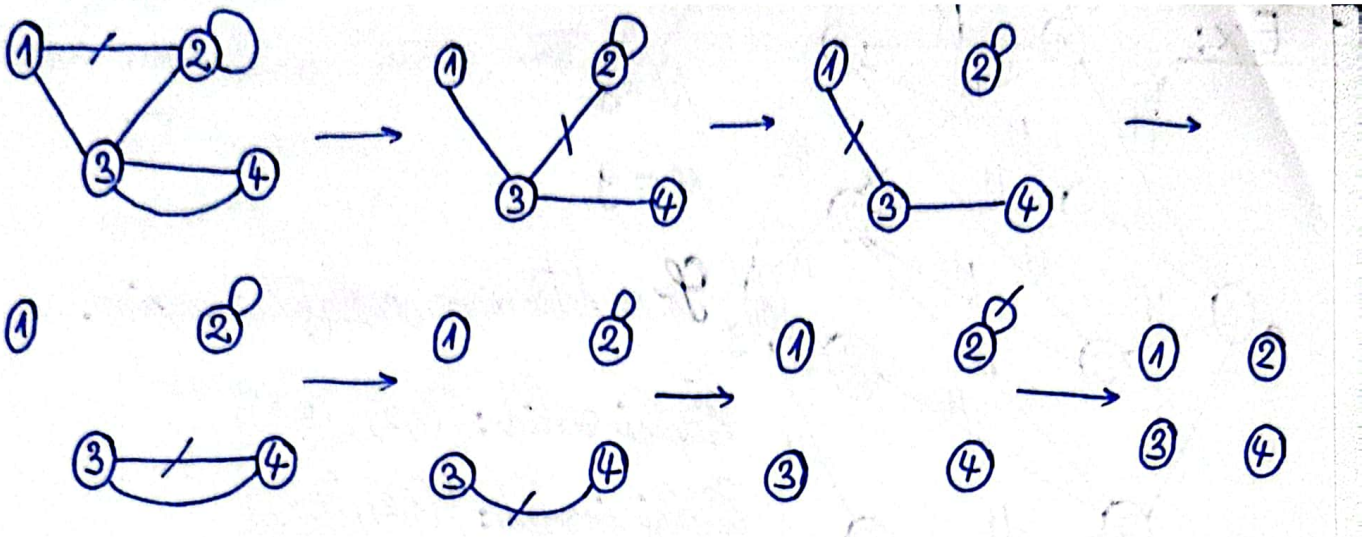
Ex: Ciclu eulerian:



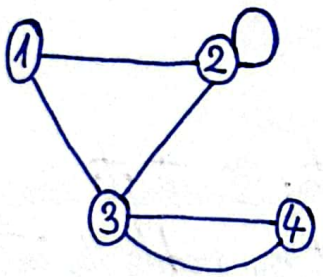
1 2 2 3 4 3 1

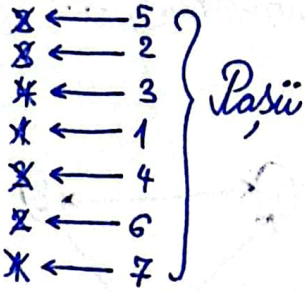
$S = \emptyset$

```
cicluEuler(v){
    while(v are vecini)
        alege w un vecin al lui v
        șterge muchia [v, w]
        cicluEuler(w)
    S = S ∪ {v}
```



1 3 4 3 2 2 1





Ex: LCS (Lowest common sequence)

$S_1 = \underline{a}\,\underline{a}\,a\,\underline{b}\,c\,\underline{d}$

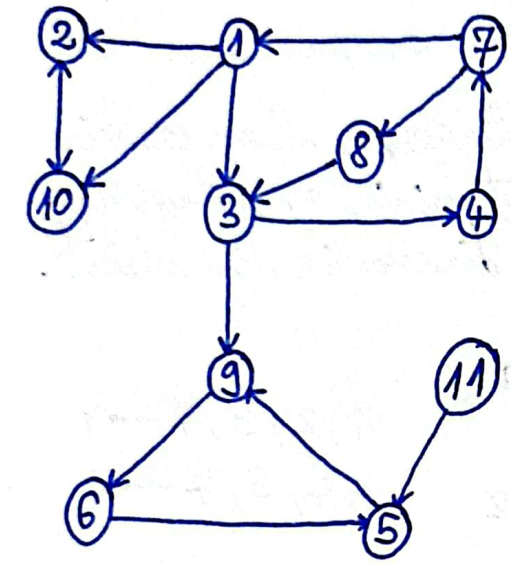
$S_2 = \underline{a}\,g\,\underline{a}\,h\,\underline{b}\,\underline{d}\,e\,r\,t$

$DP[i][j]$ = cel mai lung subșir al șirurilor $S_1[1 \ldots i]$, $S_2[1 \ldots j]$.

$$DP[i][j] = \begin{cases} 1 + DP[i-1][j-1], & S_1[i] = S_2[j] \\ \max(DP[i-1][j], DP[i][j-1] \end{cases}$$

Algoritmi Fundamentali - S7

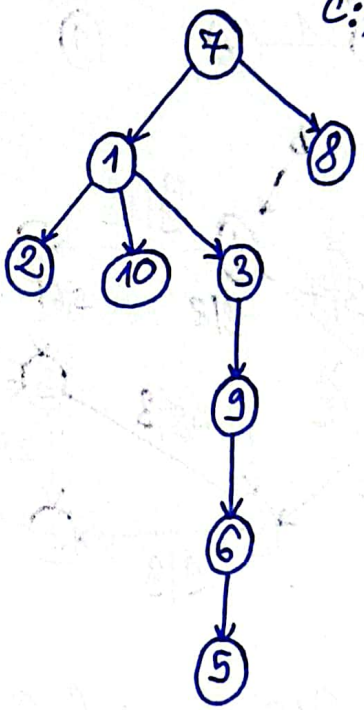
Model de examen



1) (0,5) NU are sortare topologică, are ciclu (9-6-5-9).

2) (0,5)

C: 7, 1, 8, 2, 3, 10, 9, 6, 5



3) (0,75) Componente conexe?

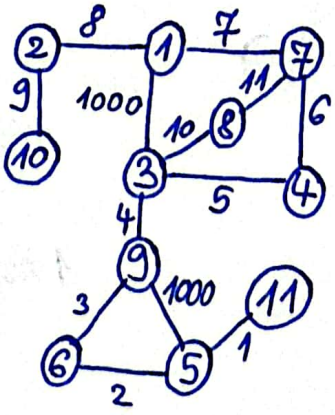
L: 10, 2, 5, 6, 9, 8, 7, 4, 3, 1, 11

C_1: 11
C_2: 1, 7, 4, 3, 8
C_3: 9, 5, 6
C_4: 2, 10

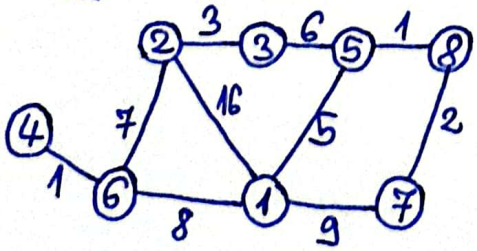
} Aici lucrăm pe GRAFUL TRANSPUS

Adăugând muchia 9-3 putem forma comp. conexă 1, 7, 4, 3, 8, 9, 5, 6 (mai mare).

4) Pt. același graf, dar neorientat, puse costuri pe muchii (și duplicate) a.î. să (∃) un singur APCM.



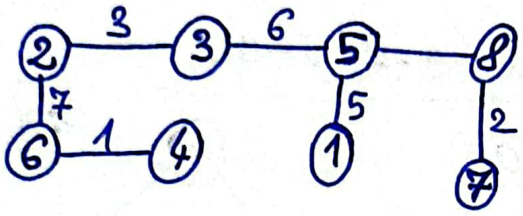
5) (0,5) x2 Lorie pași Dijkstra și Prim pt.:



Dijkstra:

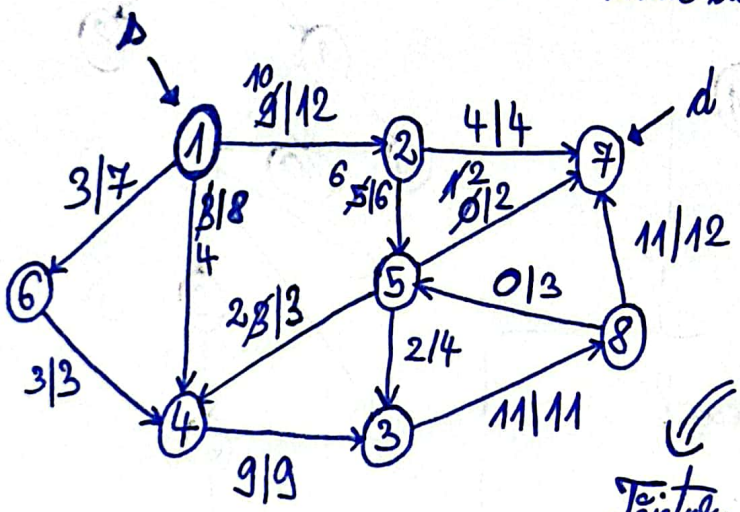
	1	2	3	4	5	6	7	8
	∞	0	∞	∞	∞	∞	∞	∞
Alegem 2	16	0	3	∞	∞	7	∞	∞
Alegem 3	16	0	3	∞	9	7	∞	∞

Prim:



La fiecare pas alegem muchia de cost minim ce se bagă de unul din nodurile deja selectate.

8) (1)

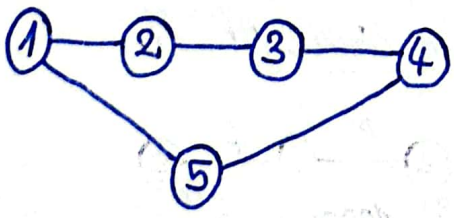


1, 2, 5, 7 → 1
1, 4, 5, 7 → 1

1, 2, 5, 3, 4, 6 → A_1
7, 8 → A_2

Tăietura minimă: 2 – 7
5 – 7
3 – 8

9) (1,5) Fie $G=(V,E)$ un graf hamiltonian și $X \subseteq V$.
Atunci $G \setminus X$ are cel mult $|X|$ comp. conexe.



În cel mai rău caz, nodurile nu sunt consecutive.

Dacă nodurile sunt consecutive, nu afectează nr. de comp. conexe.

Dacă nodurile NU sunt consecutive, crește cu 1 nr. de comp. conexe.