Gradele unui graf
- graf
  - mor: $2 \cdot |E|$; unde E = muchiile
  - or: gr. int = gr. ext = $|E|$
- multigraf — mor: $|\{e \in E \mid e$ nu e buclă, u extremitate a lui $e\}$ $+ 2 \cdot |\{e \in E \mid e$ este buclă, —"— $\}| = d_G(u)$ gradul unui singur nod

$u, v \in V$ (mult. vf) sunt adiacente $\Leftrightarrow [u; v] \in E$

$N_G(u)$ = mulțimea vecinilor lui u

e (muchie) incidentă cu u (vf) $\Leftrightarrow u \in e$, adică u extremitate a lui e

$e, f \in E$ adiacente $\Leftrightarrow \exists\, u \in e$ și $u \in f$

Drum/Lanț
- simplu = nu are o muchie de mai multe ori
- elementar = nu are un vf —"—

P = drum/lanț $\Rightarrow$ lungimea $P = |E(P)|$ nr muchii/arce

$P = v_1 - v_k$ drum $\Rightarrow v_1$ și v_k = extremitățile drumului

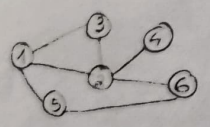
$$\begin{cases} V(P) = \{v_1, v_2 \dots v_k\}; \; e_i = (v_i, v_{i+1}); \; E(P) = \{e_1, e_2 \dots e_{k-1}\} \\ \text{pt } i \le j; \; [v_i \overset{P}{-} v_j] \text{ subdrumul lui } P \end{cases}$$

distanță, drum minim (sau $d(u,v)$)

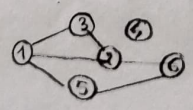
$$\delta_G(u,v) = \begin{cases} 0, & u = v \\ \infty, & \nexists \text{ drum } u - v \\ \min\{l(P) \mid P = \text{drum } v - u\}, & \text{altfel} \end{cases}$$

C = Circuit / ciclu (or / mor) = drum / lanț (or / mor) simplu cu capete identice (lungime ≥ 1) $[v_1 \dots v_k, v_1]$

- elementar = drum elementar —"— $[v_1 \dots v_k]$

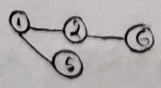


graf parțial →



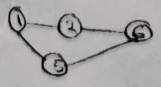
$G_1 \subseteq G \Rightarrow V_1 = V$, $E_1 \subseteq E$

subgraf →



$G_1 \prec G \Rightarrow V_1 \subseteq V$, $E_1 \subseteq E$

subgraf indus de $\{1,2,5,6\} = V_1$ →

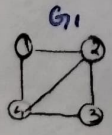


$G_1 = G[V_1] \Rightarrow V_1 \subseteq V$
$\Rightarrow E_1 = \{e \mid e \in E(G), u$ și $v \in V_1\}$

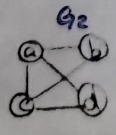
Graf conex = $\exists$ drum între oricare 2 vf dif

Componentă conexă = subgraf indus conex maximal (nu e inclus în alt subgraf conex)

Graf tare conex = gf. orientat $\forall$ drum și dus și întors între $\forall$ 2 vf



$\sim$



$G_1 \sim G_2$ (izomorfe) cu f: bij
$2 \to a$
$4 \to c$
$1 \to b$
$3 \to d$

$uv \in E_1 \Leftrightarrow f(u) f(v) \in E_2$

Graf bipartit neor. => $G(E,V)$ unde $\begin{cases} V_1 \cup V_2 = V \text{ și } V_1 \cap V_2 = \emptyset \\ |e \cap V_1| = |e \cap V_2| = 1 \quad (e=(u,v);\ u \in V_1 \text{ și } v \in V_2) \end{cases}$

K_n = graf complet de ordin n (vf)

$K_{p,q}$ = —— " —— bipartit

Arbore = conex + aciclic (neor)

Un arbore cu n vf. are $(n-1)$ muchii.

T = graf neor. cu $n-1$ vf => T arbore <=> conex muchii-minimal <=> aciclic muchii-maximal
<=> $\forall$ 2 vf $\exists$ unic lanț elementar

Orice graf neor. conex conține un arbore parțial

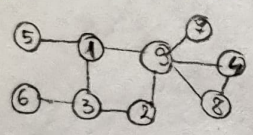
$\begin{cases} \text{Matrice ad} \Rightarrow O(n^2) \\ \text{Listă vecini} \Rightarrow O(n+m) \\ \text{Listă muchii} \Rightarrow O(m) \end{cases}$ // $m \le n(n-1)/2$

Nivelul unui nod = niv[v] = dist de la v la răd.

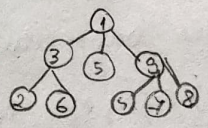
tată => niv[tată] = niv[fii] + 1

ascendent => niv[asc] < niv[desc] pe același lanț elementar

frunză => nod fără fii



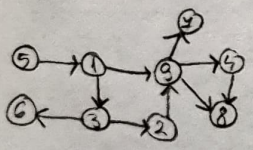
=> BFS:



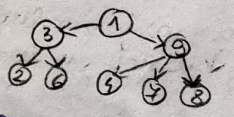
BFS: 1,3,5,9,2,6,4,7,8

tată vector: 1 2 3 4 5 6 7 8 9
0,3,1,9,1,3,9,9,1

(arbore parțial al componentei conexe a lui s(1))



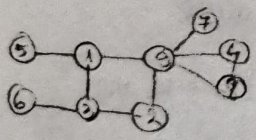
=> BFS:



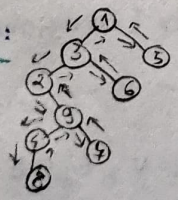
tată: 1 2 3 4 5 6 7 8 9
0, 3, 1, 9, 0, 3, 9, 9, 1

d: 0, 2, 1, 2, ∞, 2, 2, 2, 1

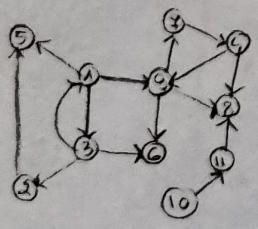
BFS: 1,3,9,2,6,4,7,8
d: 0 1 1 2 2 2 2 2



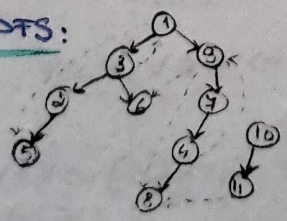
=> DFS:



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



=> DFS:



DFS: 1,3,2,5,6,9,7,4,8
10,11 (separat)

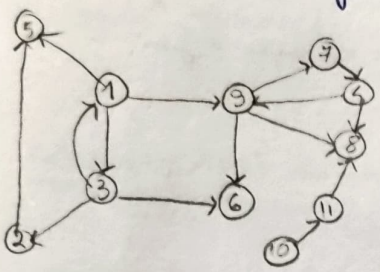
În orice căutare în adâncime a unui graf (ot/neor) $G(V,E)$ pt $\forall u,v$ vf. cu ~~desc[u]~~ desc[u] < desc[v], exact una din urm. condiții e adev:

1) [desc[u], ~~fin[u]~~] și [desc[v], fin[v]] sunt disj. & u,v nu sunt descend. unul celuilalt

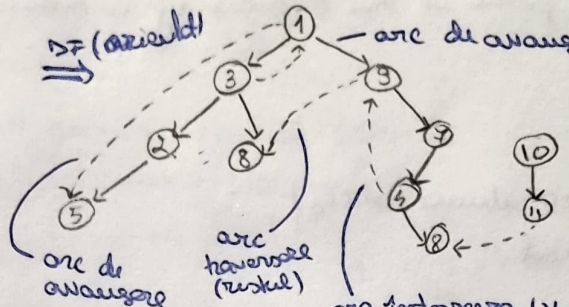
2) [desc[v], fin[v]] ⊂ [desc[u], fin[u]] => v este descendent al lui u

DF: { nevizitat - alb ; explorare - gri ; finalizat - negru
viz[x] = 0 <=> culoare[x] = alb

Dacă pe parc. alg. menținem o var. de timp care se modif atunci când se schimbă starea unui vf. (descoperim sau finalizăm), putem asoc. unui vf 2 val: { disc[v] = mom. la care vf a fost descoperit (a devenit gri)
fin[v] = — " — a fost finalizat (— " — negru)



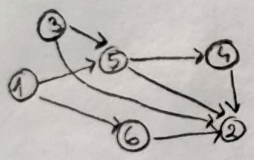
$\xrightarrow{DF \text{ (arborele)}}$



În DF neorientat avem doar: muchii de arbore ; muchii întoarcere { (→ arc ≠ tată), (Închide ciclu)
~~(Închide ciclu)~~

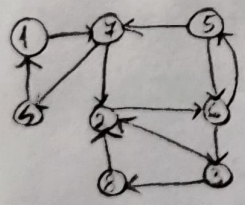
Sortare topologică (doar or.) = ordonare a vf a.î. $uv \in E$, deci u este înaintea lui v în ordonare
↳ nu e neapărat unică

G = aciclic => o sortare topologică
=> cel puțin un vf cu gradul intern $d^-(v) = 0$

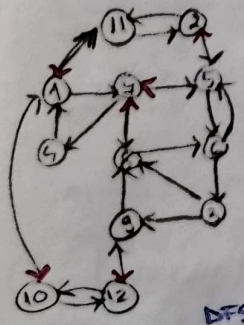


c: 1,3,6,5,4,2 => alg:

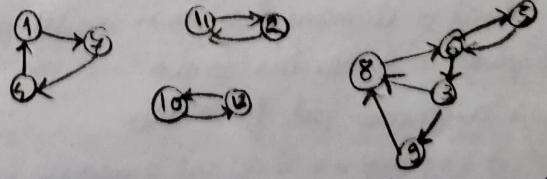
coada C = ∅
add în C toate vf. v cu $d^-[v] = 0$
cât timp C ≠ ∅ execută:
  i ← extrage (C);
  add i în sortare
  pt $ij \in E$ execută
    $d^-[j] = d^-[j] - 1$
    dacă $d^-[j] == 0$ atunci add (j, C)



Graf tare conex
(1 mtr ∀ 2 vf ∃ un drum)



$\xrightarrow{G, G^T}$ componentele tare conexe



Comp. tare conexă a vf 1 = mulț. vf accesibile din 1 în G (vizitate în DFS(1,G)) ∩
mulț. vf. — " — în G^T (— " — DFS(1, G^T))

DFS(1,G) = {1,7,4,5,6,3,9,8,11,2}
DFS(1,G^T) = {1,4,7,10,12}

=> Componenta(1) = {1,4,7}

ce este pus cu roșu => este în locul pt a forma G^T

Componente tare conexe din G = Componente tare conex din G^T
$\forall$ vf $\in$ unei unice comp. tare-conexe
Un arc poate să nu aparțină niciunei c.t.c.

C_1, C_2 = c.t.c, a.î. $\exists$ arc $C_1 \to C_2 \Rightarrow \nexists\ C_2 \to C_1$

Alg. Kosaraju ($O(m+n)$):

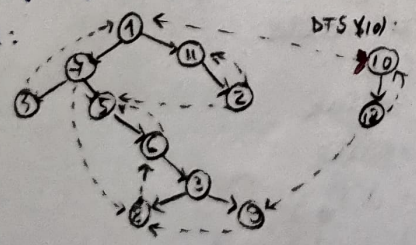
Pas 1: Parcurgem DFS graful G + punem în stiva S fiecare vf la momentul la care e fin.

```
stack S
DFS(G, i)
    viz[i] = 1
    pt ij ∈ E(G):
        dacă viz[j] == 0 atunci DFS(j)
    push(S, i) // i finalizat
pt x ∈ V execută
    dacă viz[x] == 0 atunci DFS(G, x)
```

Pas 2: Parcurgem DFS G^T considerând vf în ord. în care sunt extrase din S (descresc după timpul de fin de la pasul 1 => $f_i \neq 0$)
marcăm toate vf = nevizitate
cât timp $S \neq \emptyset$
    $x = pop(S)$
    dacă x = neviz atunci
        DFS(G^T, x)
        af c.t.c. (formată cu vf viz în DFS(G^T, x))

Exemplu pe același graf care are roșu pe el:

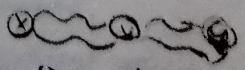
Pas 1:



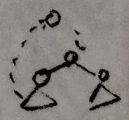
Pas 2: 1) DFS(10) => 10 → 12
4) DFS(5) => 5 → 6 → 3 → 4, 9
2) DFS(1) => 1 → 7 → 7
3) DFS(11) => 11 → 2

Ord. descresc: 10, 12, 1, 11, 2, 4, 5, 6, 3, 9, 8, 7
timp fin

Muchie critică = dacă o elimini, faci mai multe comp. conexe <=> nu e în nr-un ciclu
Graf conex fără punți = 2-muchie conex
Doar muchiile de arborescență pot fi critice
Punct critic <=> $\exists$ 2 vf $x, y \neq v$ a.î. v e oricărui x, y-drum

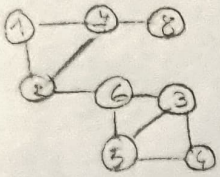


Arbore DF -> rădăcina = pct. critic <=> are cel puțin 2 fii în arbore ($\nexists$ muchii de traversare)
=> alt vf i = -"- <=> are cel puțin 1 fiu j cu $min[j] \geq min[i]$

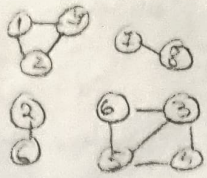


G = graf nor biconex => nu are puncte critice

Componentă biconexă (bloc) a lui G = subgraf biconex maximal



=> comp. biconexe



muchii - disjuncte, nu vf - disj

Ca să găsim comp. biconex => găsim pct. critice și despărțim

Arbore parțial de cost minim => evităm ciclurile

↳ trb să avem graf conex + fără cicluri

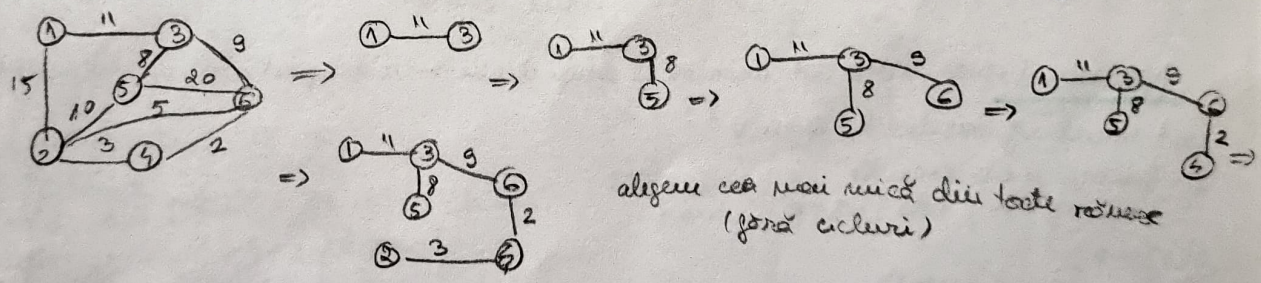
$w_{ij} = \begin{cases} 0, & i=j \\ w(i,j); & ij \in E \\ \infty, & ij \notin E \end{cases}$

Listă de ad => 1: 3/11
2: 1/15, 4/10
3: 4/8
4:

G = (V, E, w) conex ponderat => Apcm / MST = T_{min} cu $w(T_{min}) = \{w(T) \mid T$ pcm G$\}$

Kruskal => { în ord descresc. pui muchiile după cost
le adaugi una câte una (cel mai mic -> cel mai mare) în graf a.î. fără să formezi cicluri (trb să iasă arbore, logic)

Prim => pornim de la un nod oarecare și mergem pe muchiile cea mai mică, fără să formăm cicluri



alegem cea mai mică din toate rămase (fără cicluri)

Distanța de editare = nr. min de op de corectare necesar pt. transf. primei secv. în a 2-a

Dist. de edit. Levenshtein = sunt permise op. de inserare, modif, ștergere

ex: „care" → „autet" dist = 4

care → are → ane → ante → autet

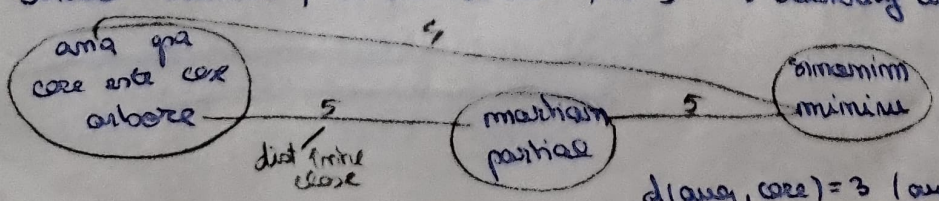
k-clustering a lui S = o partiționare a lui S în k submulț. nevide (numite clase/clustere) $\mathcal{C}(C_1, \dots C_k)$

Gradul de separare al $\mathcal{C}$ = dist min dintre 2 obiecte aflate în clase dif.

= — " — 2 clase ale lui $\mathcal{C}$

$sep(\mathcal{C}) = \min\{d(o,o') \mid o, o' \in S$, o și o' sunt clase dif — " — $\} = \min\{d(C_i, C_j) \mid i \neq j \in \{1, \dots k\}\}$

obiecte = cuvinte; de dist. de edit.; k = 3 => 3-clustering cu grad de sep = 4



d(ana, care) = 3 (ana → cana → cara → care)

Pași => det. apcm T al grafului complet G
=> consideră mult cu cele mai mari ponderi în T $\{e_{m-k+1}, \dots, e_{m-1}\}$
=> pădurea $T' = T - \{e_{m-k+1}, \dots, e_{m-1}\}$
=> def. clasele ca comp. conex ale pădurii

$$\begin{cases} s, v \in V;\ s \neq v;\ G\text{- or. pondered fără circuite de pondere neg} \\ \delta_G(s,v) = \min\{w(P) \mid P \text{ drum } s \to v\} \\ \delta_G(s,s) = 0 \\ \text{convenție } \min \emptyset = \infty \\ P \text{ drum } s \to v \Rightarrow \text{drum minim } s \to v = \text{dacă } w(P) = \delta_G(s,v) \end{cases}$$

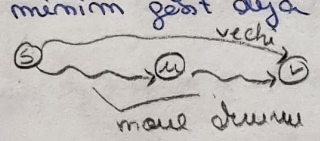
Dacă G nu e pondered => fol. BFS pt drum. min.

Arbore al dist. față de s = subgraf T al lui G care conține dist. de la s la toate vf. accesibile

$\delta_T(s,v) = \delta_G(s,v)$, $\forall v \in V$ accesibil din s

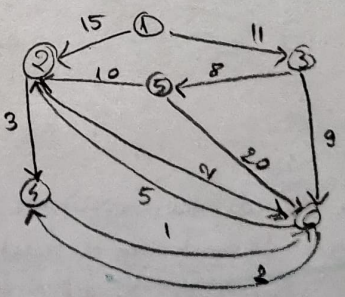
$$\begin{cases} \text{Dijkstra} = \text{alg. pt grafuri or. ponderate poz cu circuite} \\ \text{DAG (direct aciclic)} = \text{alg. pt. grafuri or. cu ponderi reale fără circuite} \\ \text{Bellman-Ford} = \text{—"— or. cu ponderi reale și circuite, care detectează existența circuitelor negative} \end{cases}$$

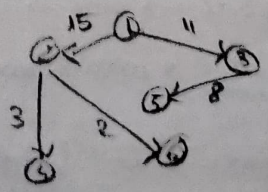
relaxarea unui arc (u,v) = verif. dacă d[v] poate fi îmbunătățit extinzând drumul minim găsit deja de la s la u cu arcul uv



Dijkstra (unde d[u] cost minim al unui drum s->u descoperit până în acel moment)

init mulț. vf. neselectate Q cu V
pt fiecare $u \in V$ execută
  d[u] = ∞; tata[u] = 0
d[s] = 0
cât timp $Q \neq \emptyset$ execută
  u = extrage vf. cu eticheta d minimă din Q
  pt. fiecare $uv \in E$ execută
    dacă d[u] + w(u,v) < d[v] atunci
      d[v] = d[u] + w(u,v)
      tata[v] = u
scrie d, tata
// scrie drum min s->t un vf t folosim tata





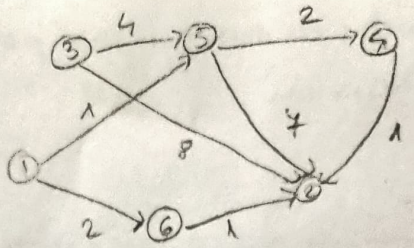
Exemplu

d/tata	1	2	3	4	5	6
	0/0	∞/0	∞/0	∞/0	∞/0	∞/0
sel 1:	—	15/1	11/1	∞/0	∞/0	∞/0
sel 3:	—	15/1	—	∞/0	19/3	20/3
sel 2:	—	—	—	18/2	19/3	14/2
sel 6:	—	—	—	18/2	19/3	—
sel 4:	—	—	—	—	19/3	—
sel 5:	—	—	—	—	—	—

soluția: [1: 0/0; 2: 15/1; 3: 11/1; 4: 18/2; 5: 19/3; 6: 14/2]

~~(dacă unui arc cost ex, adăugăm la toate cuvintele nr. ex. toate -> 0)~~

DAG ($O(n+m)$)



s - vf de start

//init dist - Dijkstra

pt. fiecare $u \in V$ execută

$d[u] = \infty$; $tata[u] = 0$;

$d[s] = 0$;

// det. o sortare top. a vf.

SortTop = sortare_top(G)

pt fiecare $u \in$ SortTop

pt fiecare $uv \in E$ execută:

dacă $d[u] + w(u,v) < d[v]$ atunci

//relaxăm uv

$d[v] = d[u] + w[u,v]$

$tata[v] = u$

scrie d, tata

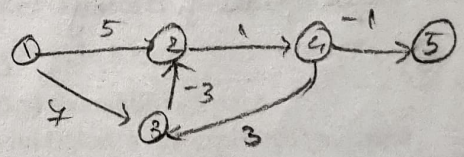
d/tata:	1	2	3	4	5	6
	∞/0	∞/0	0/0	∞/0	∞/0	∞/0
u=1	∞/0	∞/0	0/0	∞/0	∞/0	∞/0
u=3	∞/0	8/3	0/0	∞/0	4/3	∞/0
u=6	∞/0	8/3	0/0	∞/0	4/3	∞/0
u=5	∞/0	8/3	0/0	6/5	4/3	∞/0
u=4	∞/0	7/4	0/0	6/5	4/3	∞/0
u=2	∞/0	7/4	0/0	6/5	4/3	∞/0

exemplu: SortTop = 1,3,6,5,4,2

s = 3 vf start

soluție: [∞/0; 7/4; 0/0; 6/5; 4/3; ∞/0]

Bellman-Ford ($O(n \cdot m)$)



pt fiecare $u \in V$ execută

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

pt $k = 1, n-1$ execută

pt fiecare $uv \in E$ execută

dacă $d[u] + w(u,v) < d[v]$ atunci

$d[v] = d[u] + w(u,v)$

$tata[v] = u$

sol: [0, 4, 7, 5, 4]

Relaxări final => 12; 13; 24; 43; 45; 32

Floyd-Warshall - drumuri min între toate perechile de vf. ($O(n^3)$)

```
for(i=1; i<=n; i++){
    for(j=1; j<=n; j++){
        d[i][j]=w[i][j];
        if(w[i][j]==∞) p[i][j]=0;
        else p[i][j]=i;
    }
}
for(k=1; k<=n; k++){
    for(i=1; i<=n; i++){
        for(j=1; j<=n; j++){
            if(d[i][j]>d[i][k]+d[k][j]){
                d[i][j]=d[i][k]+d[k][j];
                p[i][j]=p[k][j];
            }
        }
    }
}
```

sol: mat d

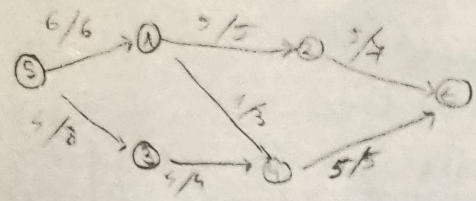
a.î. exist drum $i \to j$ cu $d[i][j] < \infty$ &

face fol. matricea p

```
void drum(int i, int j){
    if(i!=j)
        drum(i, p[i][j]);
    cout << j;
}
```

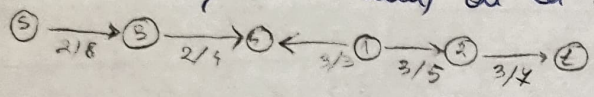
∃ circuit neg. <=> ∃ i cu $d_{ii} < 0$

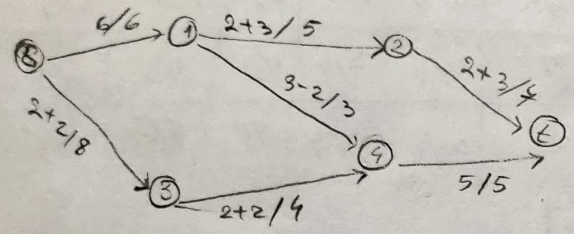
s-t tăietura (muchii cu max cap care pot împărți graful în mai multe comp)
↳ la fluxuri



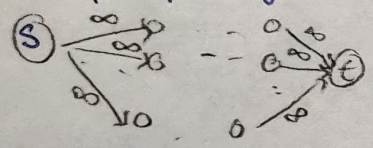
max flow = 10
tăietura = (s,1) și (3,4)

Dacă făceam greșeala și puneam pe 1→2 (3/5) și 1→4 (3/3) ⇒ condiția trb făcută pe un lanț de la s la t (nu doar arc) altfel fluxul (marfa) va rămâne blocat.
Det. un lanț (nu drum) de la s→t cu care putem modif fluxul:





Orice rețea poate fi transf într-o rețea echiv de transport adăugând un nod sursă



și un t (destinație)

Un flux într-o rețea de transport ⇒ fluxul total care intră în v = fluxul total care iese din v
$f^+(v) = f^-(v)$

$\bar{X} = V - X$ pt $X \subseteq V$

$f(X,Y) \Rightarrow f(\{s,1,3\},\{2,4,t\}) = f(1,2) + f(1,4) + f(3,4) = 10$
$= \sum f(uv)$ $uv \in E$ și $u \in X, v \in Y$

$f^+(X)$ = fluxul care iese din X $= f(X, V-X) = f(X, \bar{X}) = \sum f(uv)$ $u \in X, v \notin X$
$f^-(X)$ = — " — intră în X $= f(\bar{X}, X) = \sum f(v,u)$ $u \in X, v \notin X$
$val(f) = f^+(s) = \sum f(su) = f^-(t)$

flux maxim în N ⇒ dacă $val(f^*) = \max\{val(f) \mid f = \text{flux în } N\}$
fluxul vid ⇒ $f(e) = 0, \forall e \in E$ (∀ rețea echivalentă cu puțin un flux)

Alg Ford-Fulkerson - det. flux maxim + tăietura min