Aplicație
Flux maxim → Cuplaj maxim în grafuri bipartite

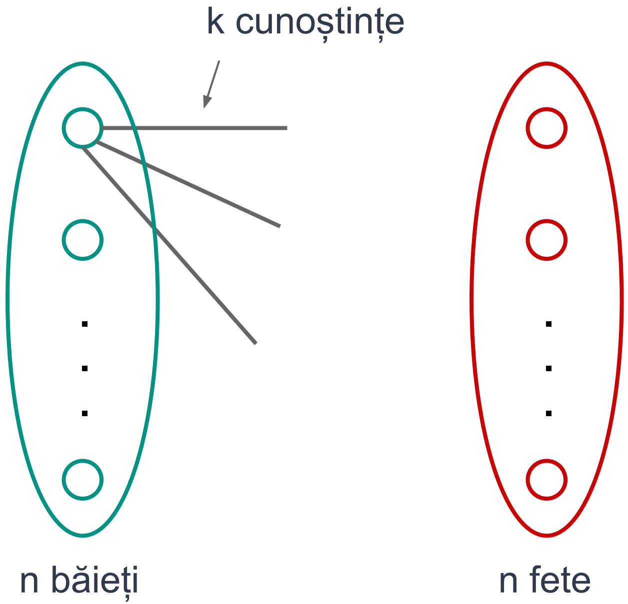
Cuplaje

Repere istorice. Aplicații

- **Problema seratei (perechilor)** - sec XIX
  - n fete, n băieți
  - un băiat cunoaște exact k fete
  - o fată cunoaște exact k băieți

Repere istorice. Aplicații

- **Problema seratei (perechilor)**



Repere istorice. Aplicații

- **Problema seratei (perechilor)** - sec XIX
    - Se poate organiza o repriză de dans astfel încât fiecare participant să danseze cu o cunoștință a sa?

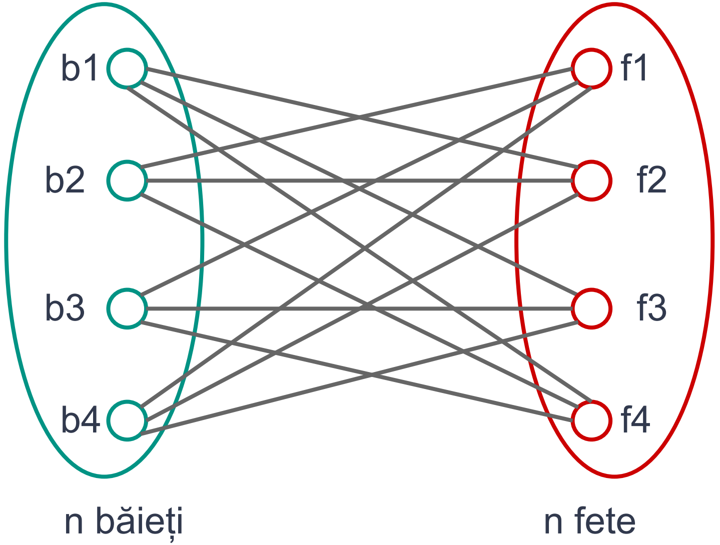
Repere istorice. Aplicații

- **Problema seratei (perechilor)** - sec XIX
  - Se poate organiza o repriză de dans astfel încât fiecare participant să danseze cu o cunoștință a sa?
  - Se pot organiza **k** reprize de dans în care fiecare participant să danseze câte un dans cu fiecare cunoștință a sa?

Repere istorice. Aplicații

- **Problema seratei (perechilor)** - sec XIX

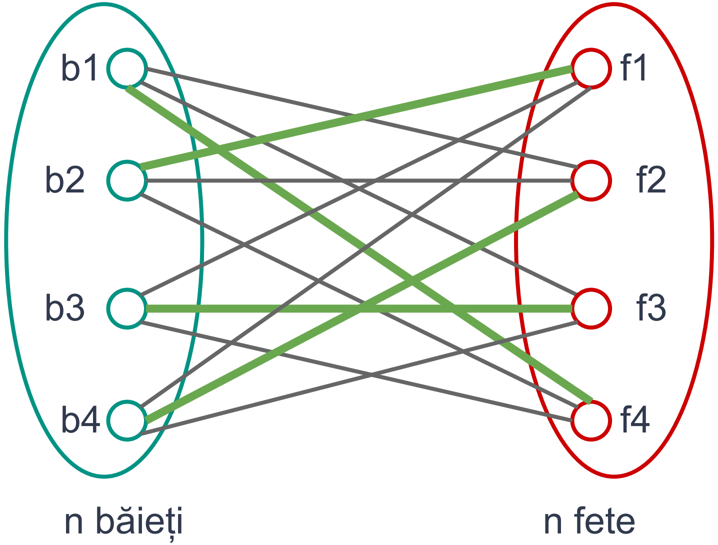
$n = 4$
$k = 3$



Repere istorice. Aplicații

- **O repriză de dans**

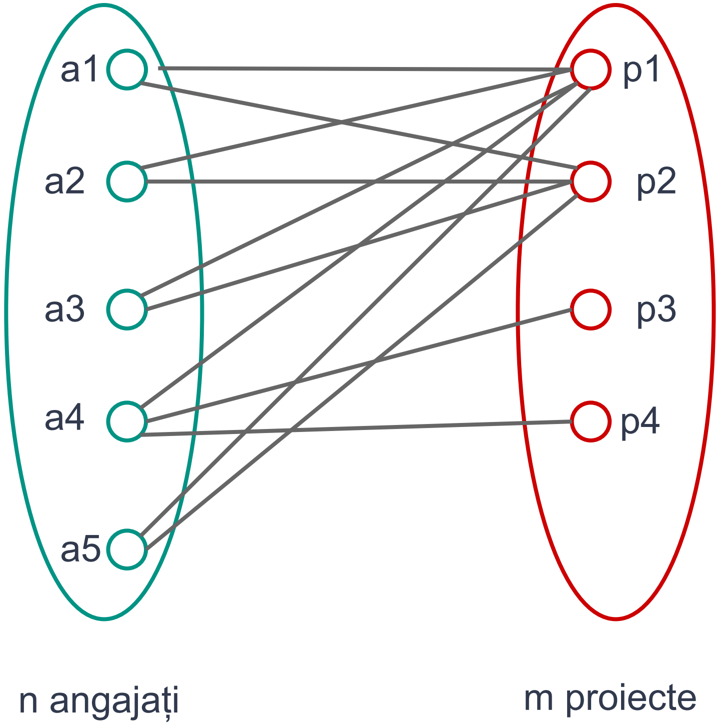
b1, f4
b2, f1
b3, f3
b4, f2



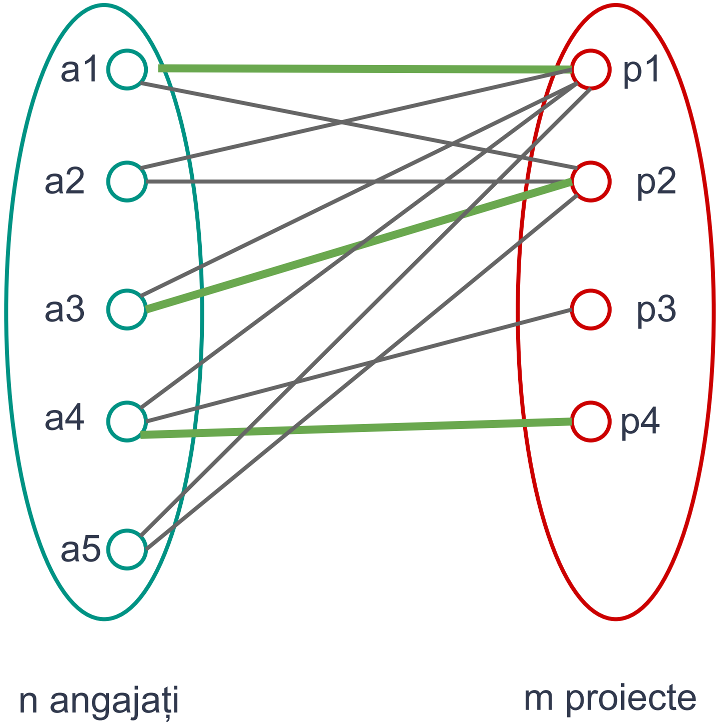
Repere istorice. Aplicații

- **Problema seratei**
- **Organizare de competiții**
- **Probleme de repartiție**
  - lucrători - locuri de muncă
  - profesori - examene / conferințe
  - **Problema orarului**

Alte aplicații



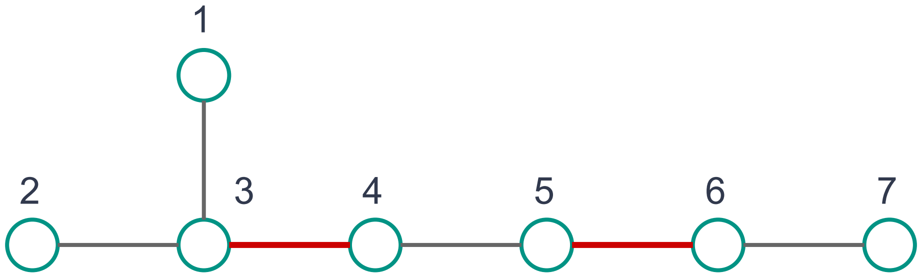
Alte aplicații



Cuplaje

Fie G = (V, E) un graf și **M ⊆ E**.

- **M** se numește **cuplaj** dacă orice două muchii din M sunt neadiacente.

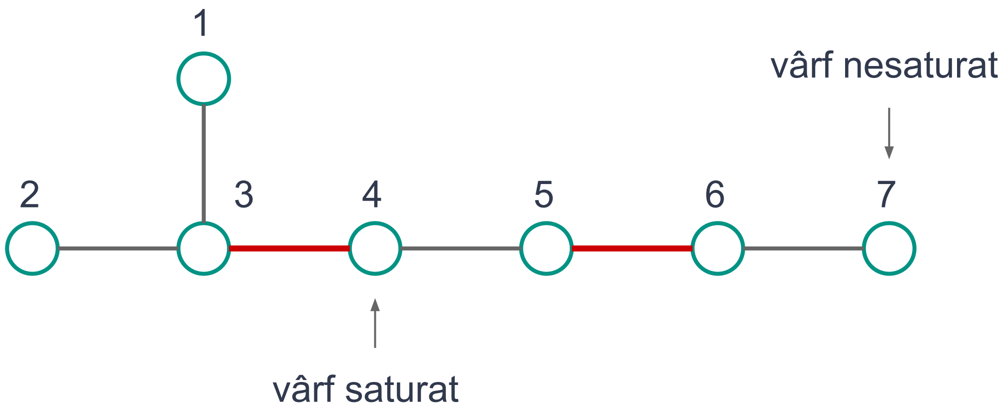


cuplaj M = { {3, 4}, {5, 6} }

Cuplaje

Fie G = (V, E) un graf și **M ⊆ E**.

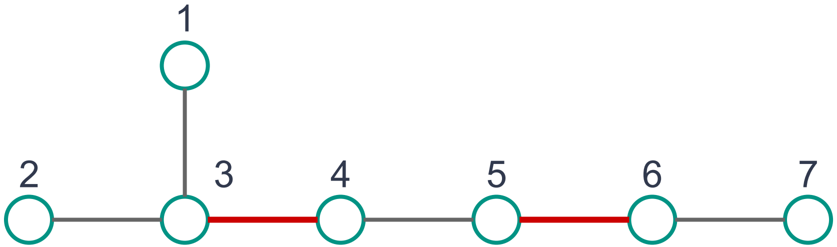
- **M** se numește **cuplaj** dacă orice două muchii din M sunt neadiacente.
- V(M) = mulțimea vârfurilor **M-saturate**
- V(G) - V(M) = mulțimea vârfurilor **M-nesaturate**



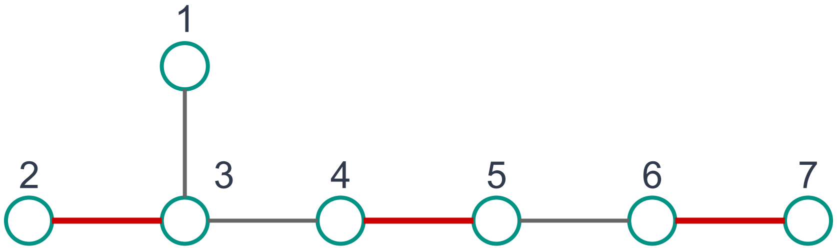
Cuplaje

- Un cuplaj M* se numește **cuplaj de cardinal maxim (cuplaj maxim)**:

  $|M^*| \geq |M|, \;\; \forall\; M \subseteq E$ cuplaj

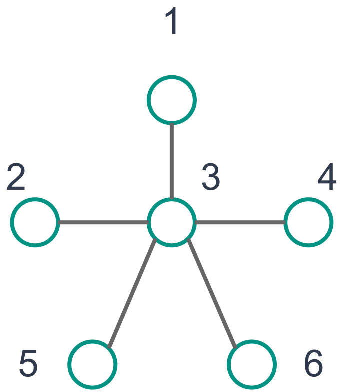


cuplaj



cuplaj de cardinal maxim

Cuplaje



cuplaj de cardinal maxim?

Grafuri bipartite

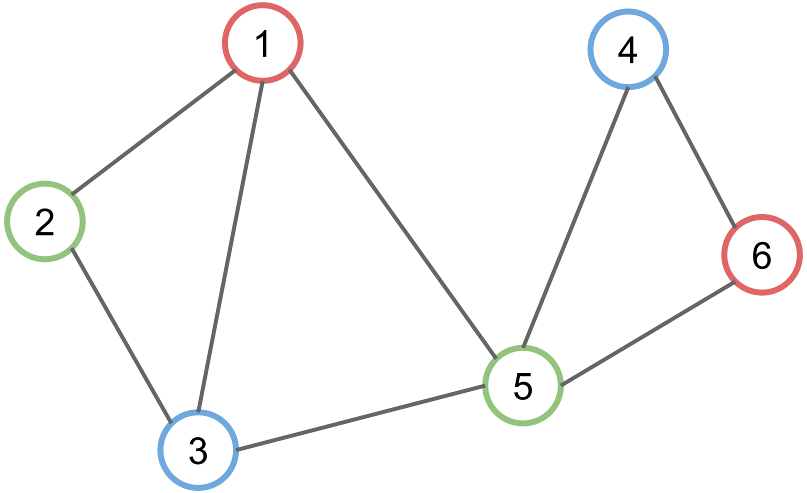
Colorări ale grafurilor

Fie $G = (V, E)$ graf neorientat

- $c : V \to \{ 1, 2, \dots, p \}$ s. n. **p-colorare** a lui G
- $c : V \to \{ 1, 2, \dots, p \}$ cu $c(x) \neq c(y)\ \forall\ xy \in E$ s. n. **p-colorare proprie** a lui G

G s. n. **p-colorabil** dacă admite o p-colorare proprie.

Colorări ale grafurilor



3-colorabil, dar nu și 2-colorabil (!)

Graf bipartit

$G = (V, E)$ graf neorientat s. n. **bipartit** $\Leftrightarrow$ există o partiție a lui V în două submulțimi V_1 și V_2 (**bipartiție**) cu:

$$V = V_1 \cup V_2$$

$$V_1 \cap V_2 = \varnothing$$

astfel încât orice muchie $e \in E$ are o extremitate în V_1 și cealaltă în V_2.

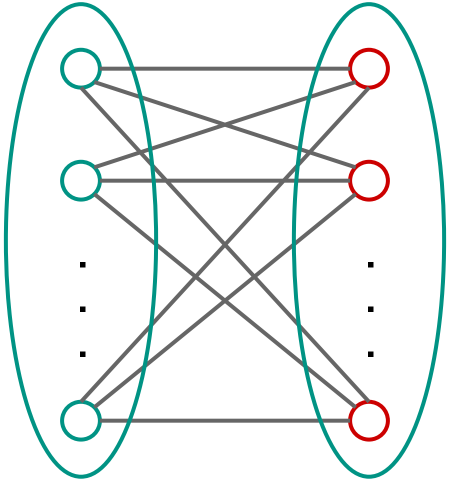
Notăm $G = (V_1 \dot\cup V_2, E)$

Graf bipartit

$G = (V, E)$ graf neorientat s. n. **bipartit complet** $\Leftrightarrow$

este bipartit și $E = \{ xy \mid x \in V_1, y \in V_2 \}$

Notăm $K_{p,\,q}$ dacă $p = |V_1|$ și $q = |V_2|$



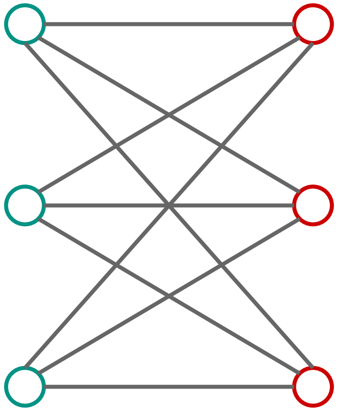
Graf bipartit

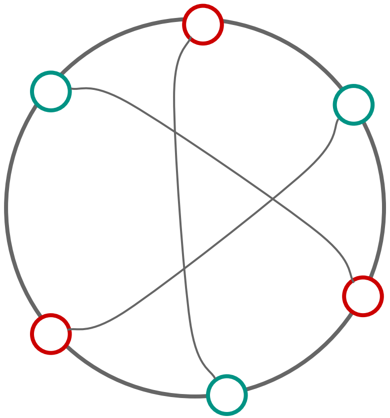
G = (V, E) graf neorientat s. n. **bipartit complet** ⇔

este bipartit și $E = \{ xy \mid x \in V_1, y \in V_2 \}$

Notăm $K_{p,\,q}$ dacă $p = |V_1|$ și $q = |V_2|$

- $K_{3,\,3}$





Graf bipartit

Observație

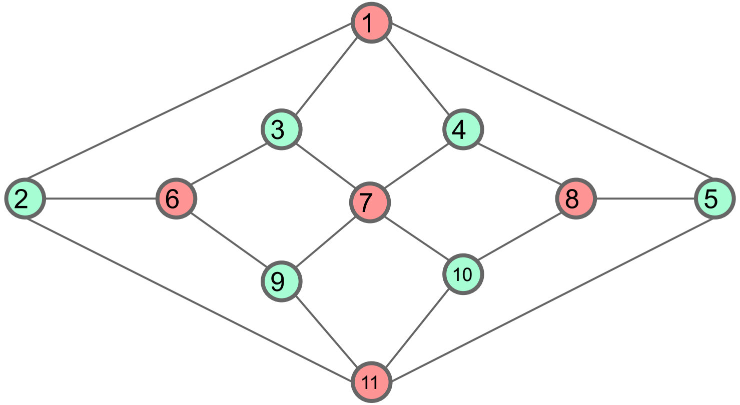
$G = (V, E)$ **bipartit** $\Leftrightarrow$

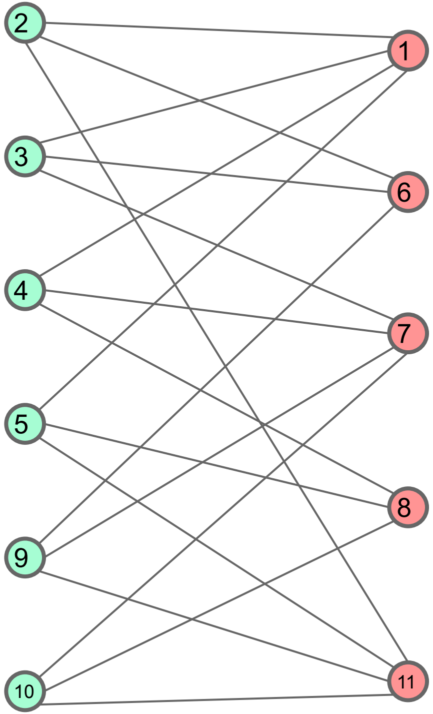
existǎ o 2-colorare proprie a vârfurilor (**bicolorare**):

$$c : V \to \{ 1, 2 \}$$

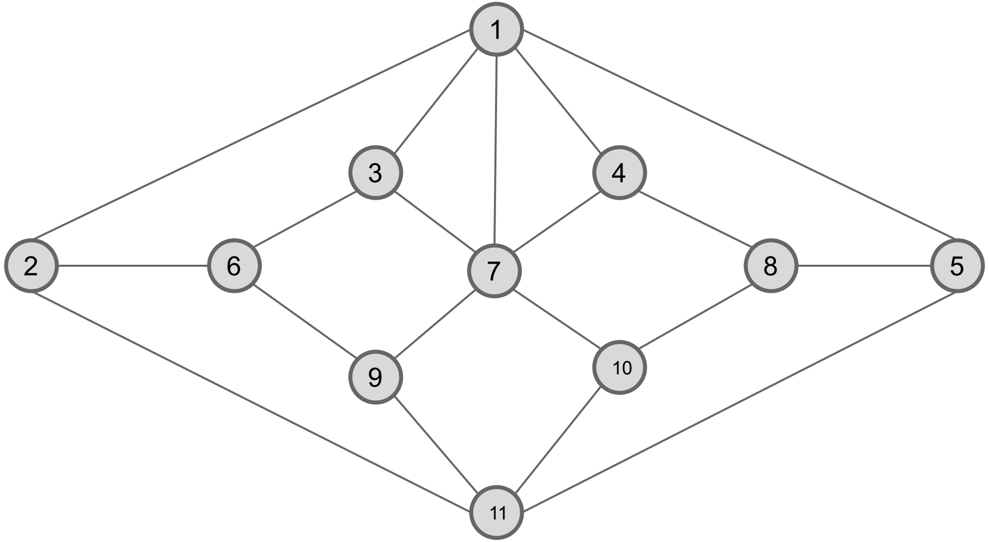
(i. e. astfel încât, pentru orice muchie $e = xy \in E$ avem $(x) \neq c(y)$)

Graf bipartit



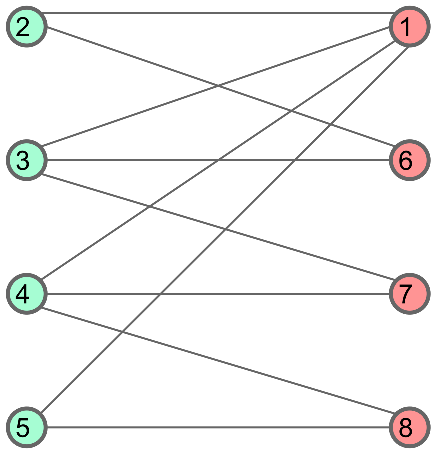


Graf bipartit



nu este bipartit

Modelare



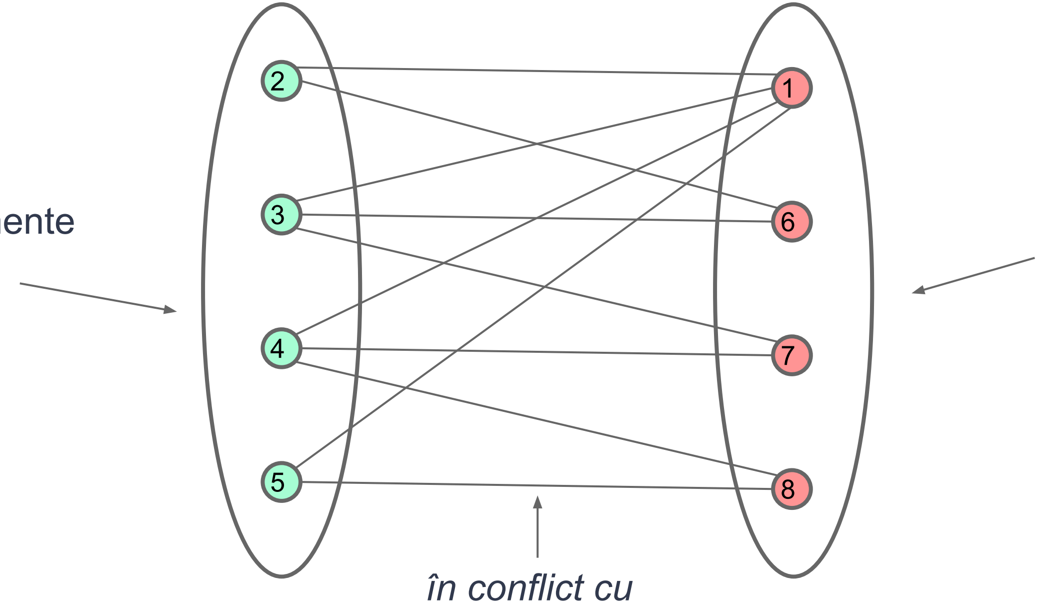
Profesori *predau la* **Cursuri**

Candidați *depun CV la* **Joburi**

Aplicații

Graf de conflicte (exemplu - substanțe care interacționează, activități incompatibile, relații în rețele de socializare)

submulțimi fără conflicte (evenimente compatibile)



submulțimi fără conflicte (evenimente compatibile)

Cuplaje, rețele …

Aplicații p-colorări

Exemplu - De câte săli este nevoie minim pentru programarea, într-o zi, a n conferințe, cu intervale de desfășurare date?

Conf. 1: interval (1, 4)

Conf. 2: interval (2, 3)

Conf. 3: interval (2, 5)

Conf. 4: interval (6, 8)

Conf. 5: interval (3, 8)

Conf. 6: interval (6, 7)

Aplicații p-colorări

Exemplu - De câte săli este nevoie minim pentru programarea, într-o zi, a n conferințe, cu intervale de desfășurare date?

Conf. 1: interval (1, 4)

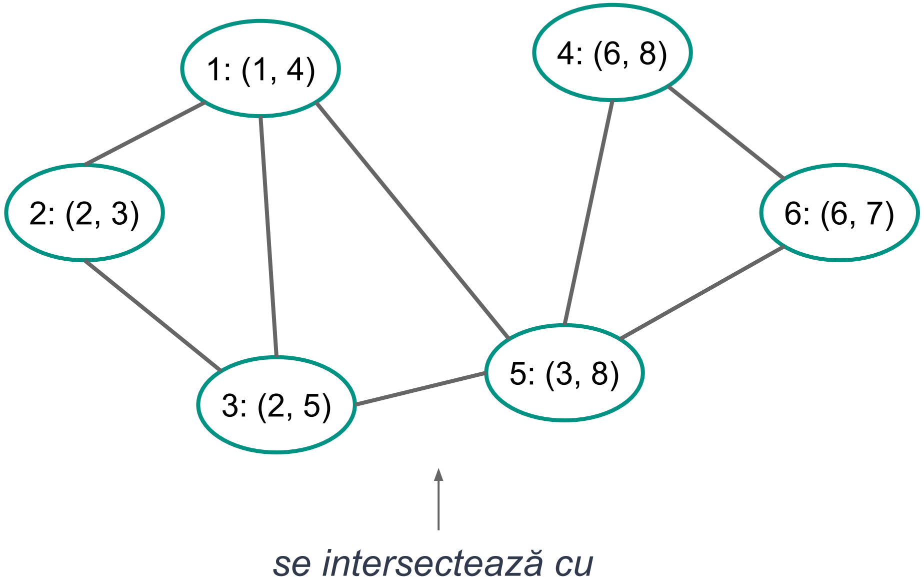
Conf. 2: interval (2, 3)

Conf. 3: interval (2, 5)

Conf. 4: interval (6, 8)

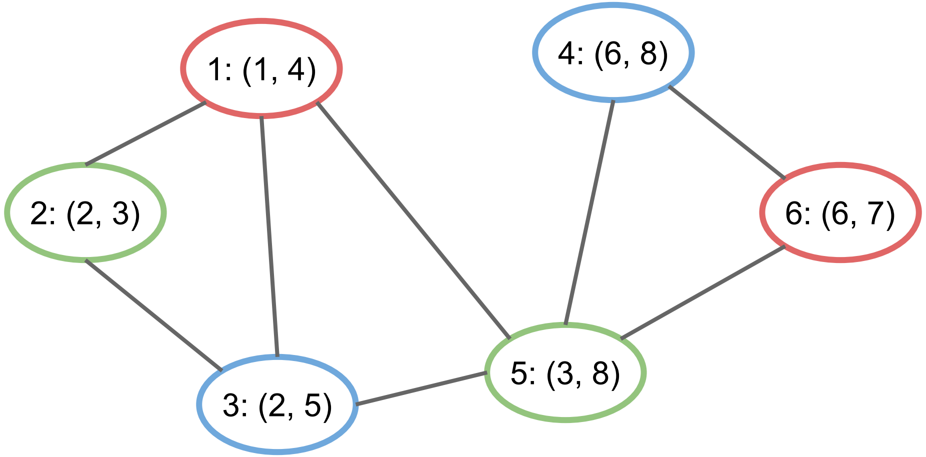
Conf. 5: interval (3, 8)

Conf. 6: interval (6, 7)



Aplicații p-colorări

Graful intersecției intervalelor este 3-colorabil



Sunt necesare minim 3 săli (corespunzătoare celor 3 culori):

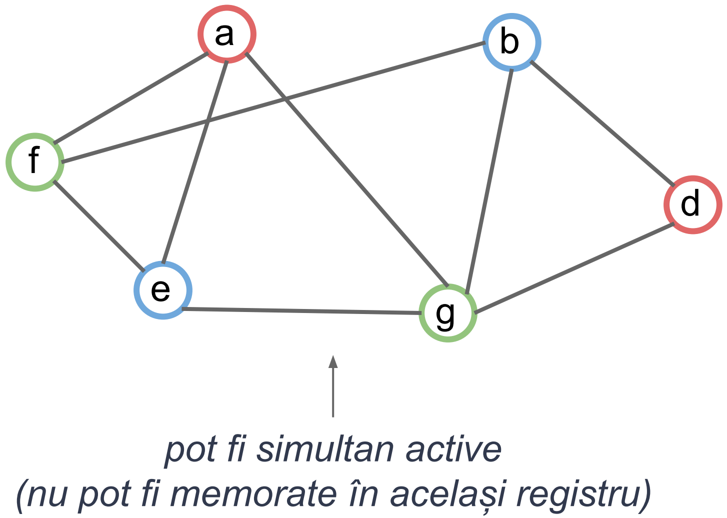
Sala 1: (1, 4), (6, 7)

Sala 2: (2, 3), (3, 8)

Sala 3: (2, 5), (6, 8)

Aplicații p-colorări

Alocare de regiștri (Register allocation problem)

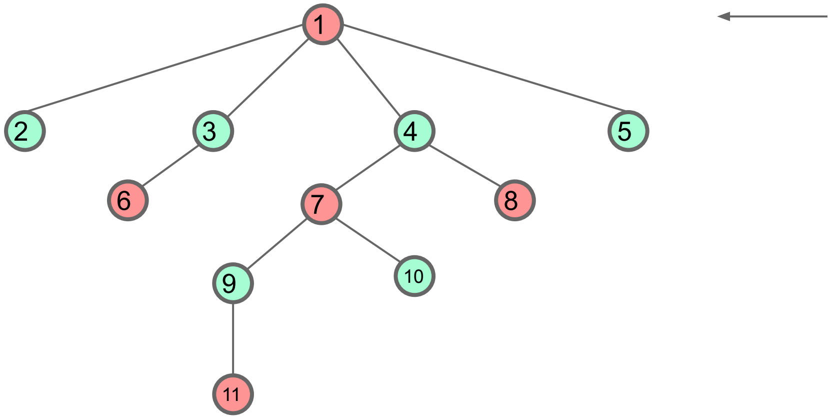


- Numărul de culori = numărul de regiștri
- Vârfuri de aceeași culoare = pot fi memorate în același registru

Graf bipartit

Propoziție

Un arbore este graf bipartit.



- Fixăm o rădăcină
- Colorăm alternativ nivelurile

Graf bipartit

Teorema König - Caracterizarea grafurilor bipartite

Fie $G = (V, E)$ un graf cu $n \geq 2$ vârfuri.

Avem

$$G \text{ este bipartit} \quad \Leftrightarrow \quad \text{toate ciclurile elementare din } G \text{ sunt pare}$$

Graf bipartit

Teorema König - Caracterizarea grafurilor bipartite

Demonstrație ”⇒”:

Evident, deoarece un ciclu impar nu poate fi colorat propriu cu două culori.

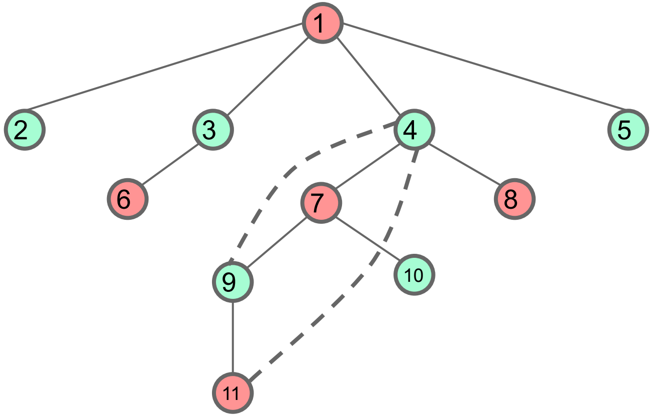
Graf bipartit

Teorema König - Caracterizarea grafurilor bipartite

Demonstrație ”⇐”: Presupunem G conex.

Colorăm propriu cu 2 culor un arbore parțial T al său.

Orice altă muchie uv din graf are extremitățile colorate diferit, deoarece



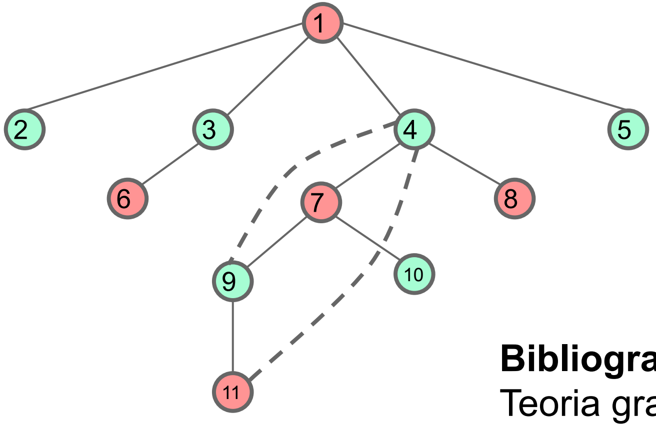
Graf bipartit

Teorema König - Caracterizarea grafurilor bipartite

Demonstrație ”⇐”: Presupunem G conex.

Colorăm propriu cu 2 culor un arbore parțial T al său.

Orice altă muchie uv din graf are extremitățile colorate diferit, deoarece formează un ciclu elementar cu lanțul de la u la v din arbore, iar acest ciclu are lungime pară, deci u și v se află pe niveluri de paritate diferită în T.



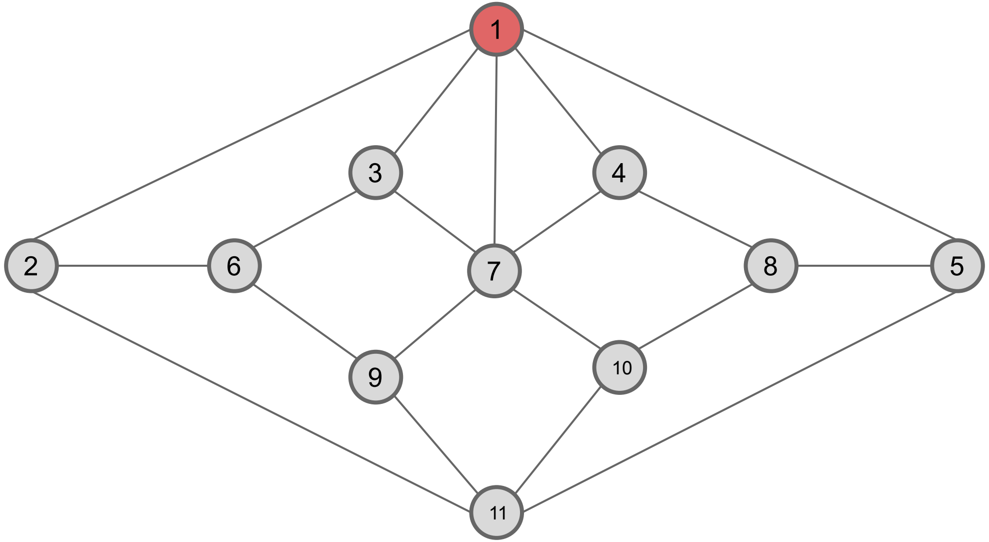
Bibliografie: DR Popescu Combinatorică și Teoria grafurilor (Teorema 4.18)

Graf bipartit

Teorema König ⇒ Algoritm pentru a testa dacă un graf este bipartit

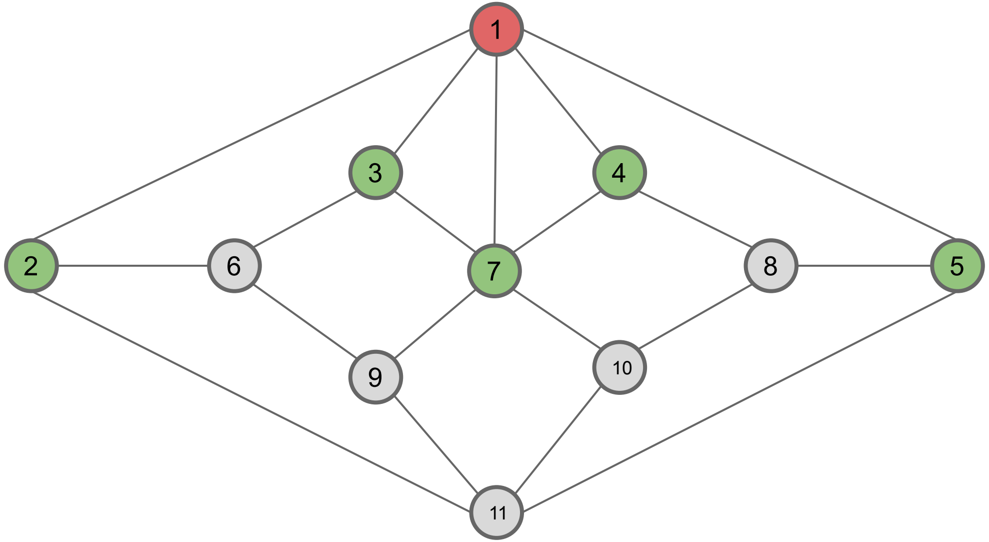
- Colorăm (propriu) cu 2 culori un arbore parțial al său, printr-o **parcurgere** (colorăm orice vecin j nevizitat al vârfului curent i cu o culoare diferită de cea a lui i)
- Testăm dacă celelalte muchii - de la i la **vecini j deja vizitați** (colorați) au extremitățile i și j colorate diferit

Exemplu de test – graf bipartit BF

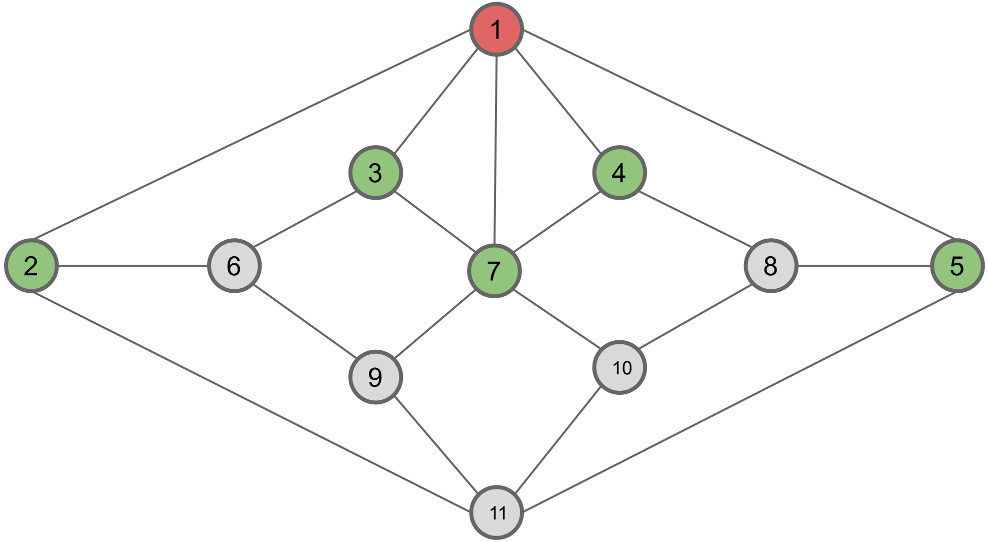


i = 1

Exemplu de test – graf bipartit BF

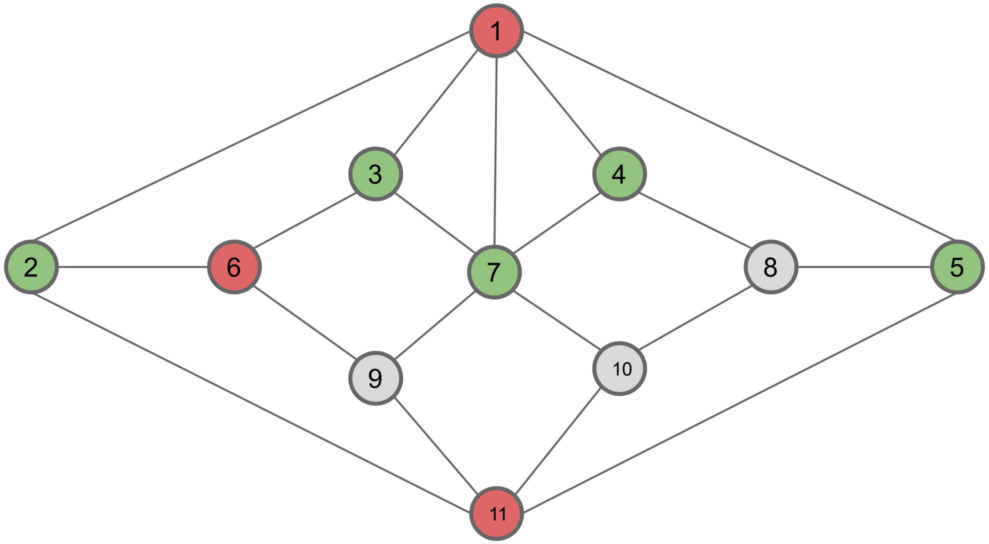


Exemplu de test – graf bipartit BF

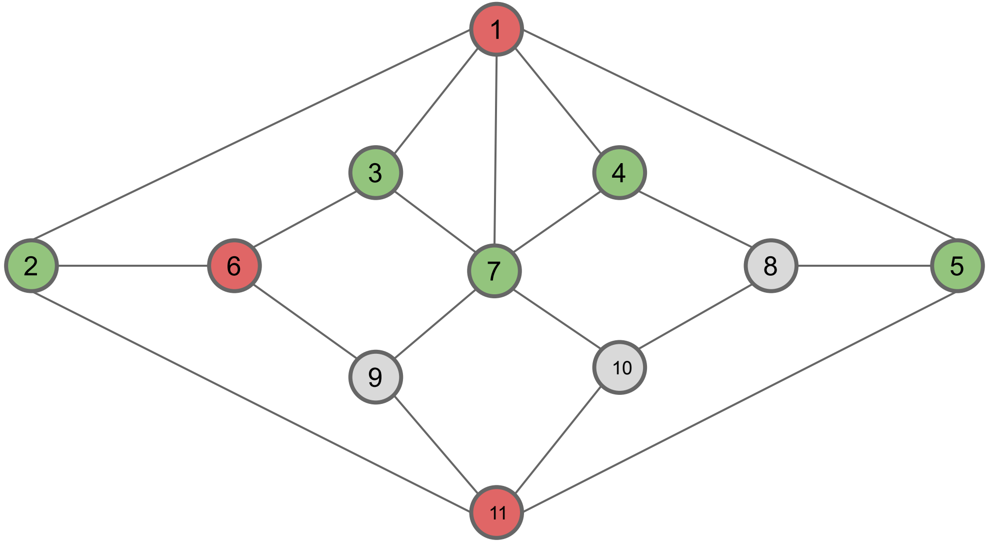


i = 2

Exemplu de test – graf bipartit BF

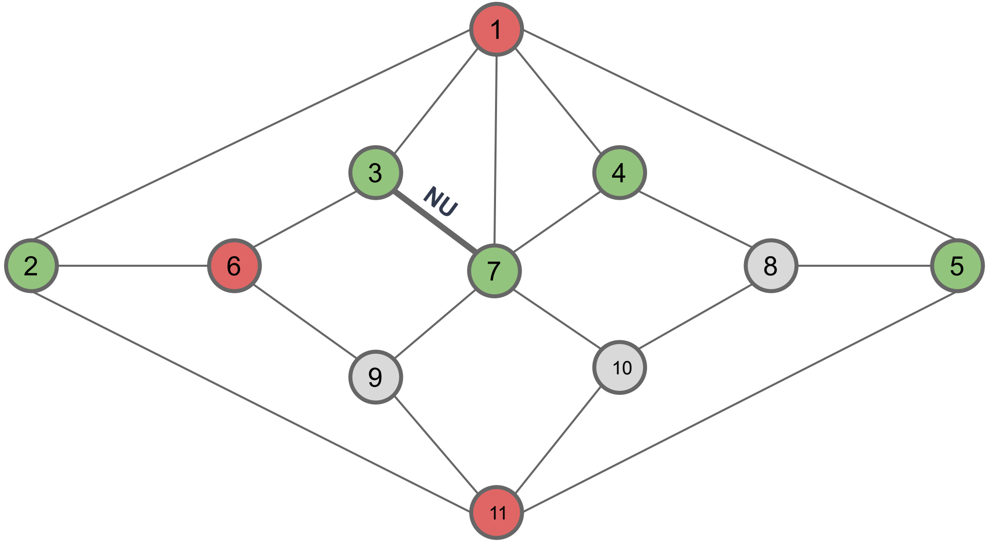


Exemplu de test – graf bipartit BF



i = 3

Exemplu de test – graf bipartit BF

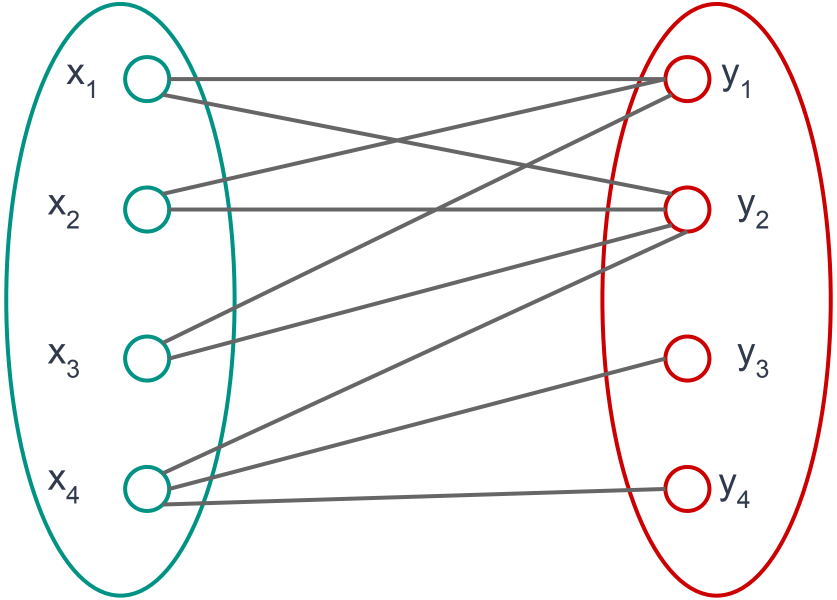


Algoritm
Flux maxim → Cuplaj maxim în grafuri bipartite

Algoritm de determinare a unui cuplaj maxim

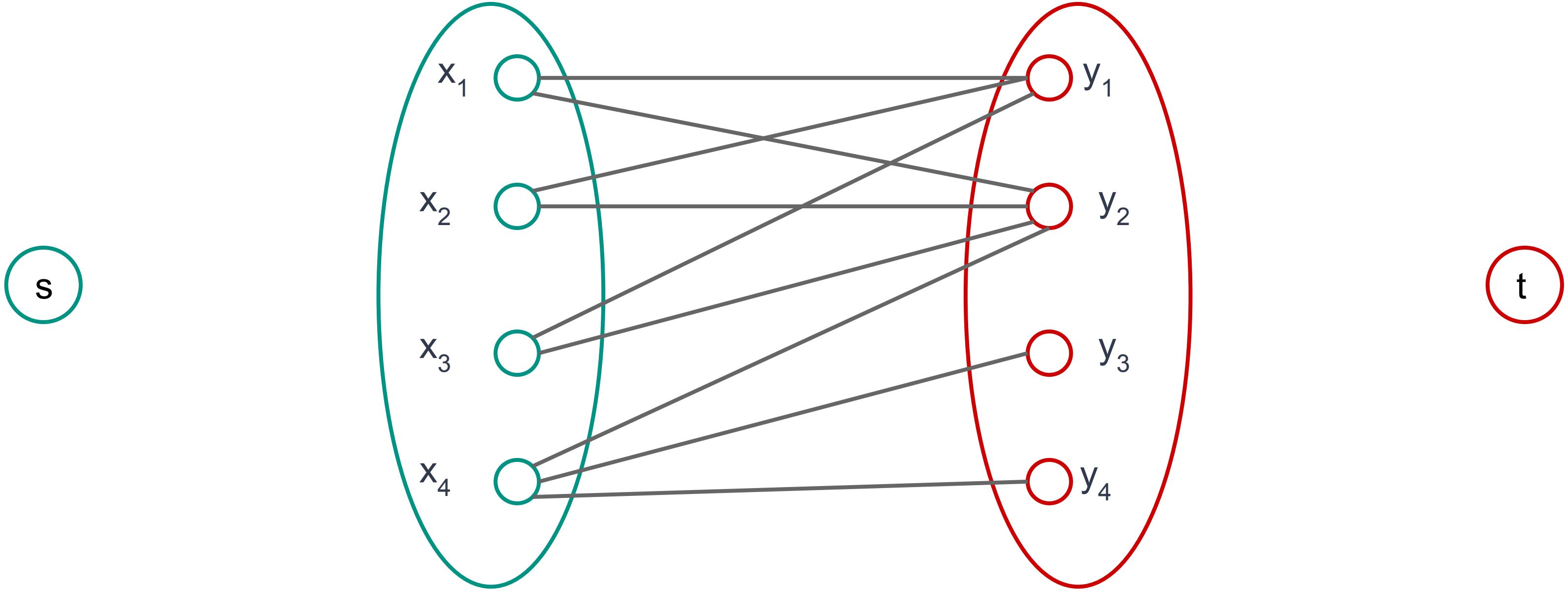
- Reducem problema determinării unui cuplaj maxim într-un graf bipartit G la determinarea unui flux maxim într-o rețea de transport asociată lui G
- Construim rețeaua de transport N_G asociată lui G astfel:

Algoritm de determinare a unui cuplaj maxim



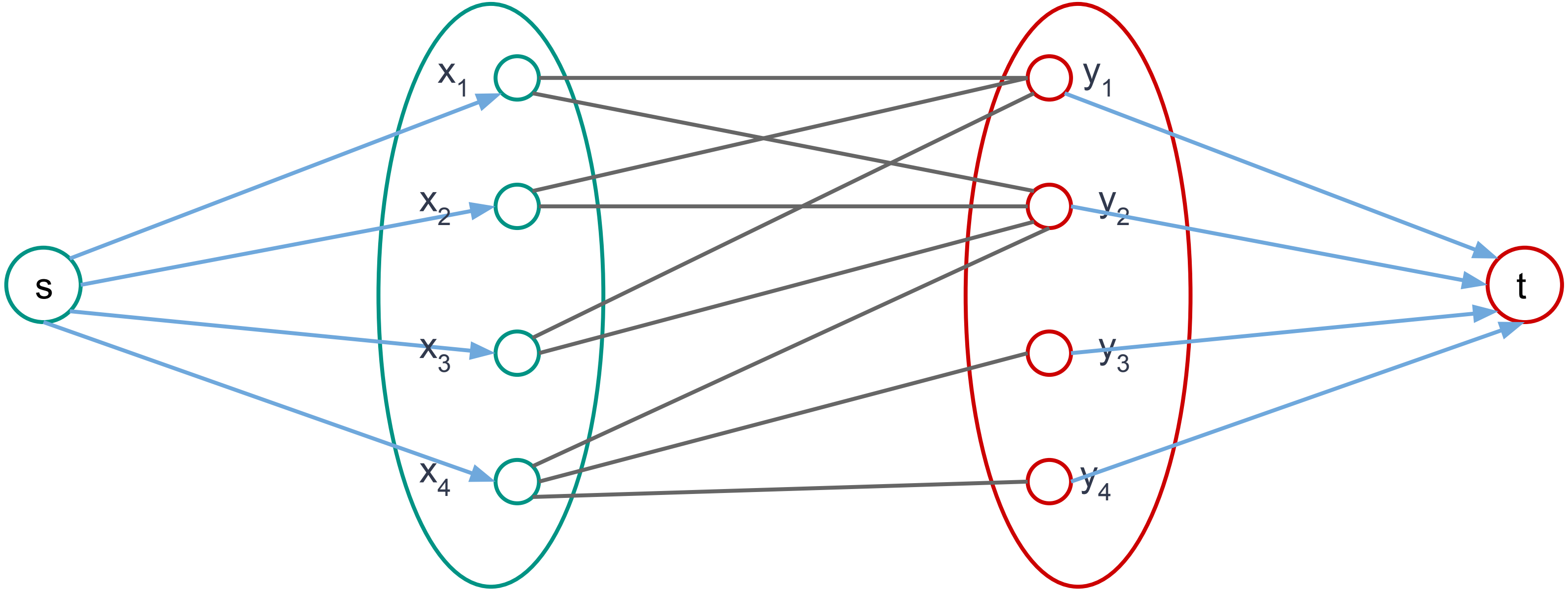
Algoritm de determinare a unui cuplaj maxim

Adăugăm două noduri noi s și t



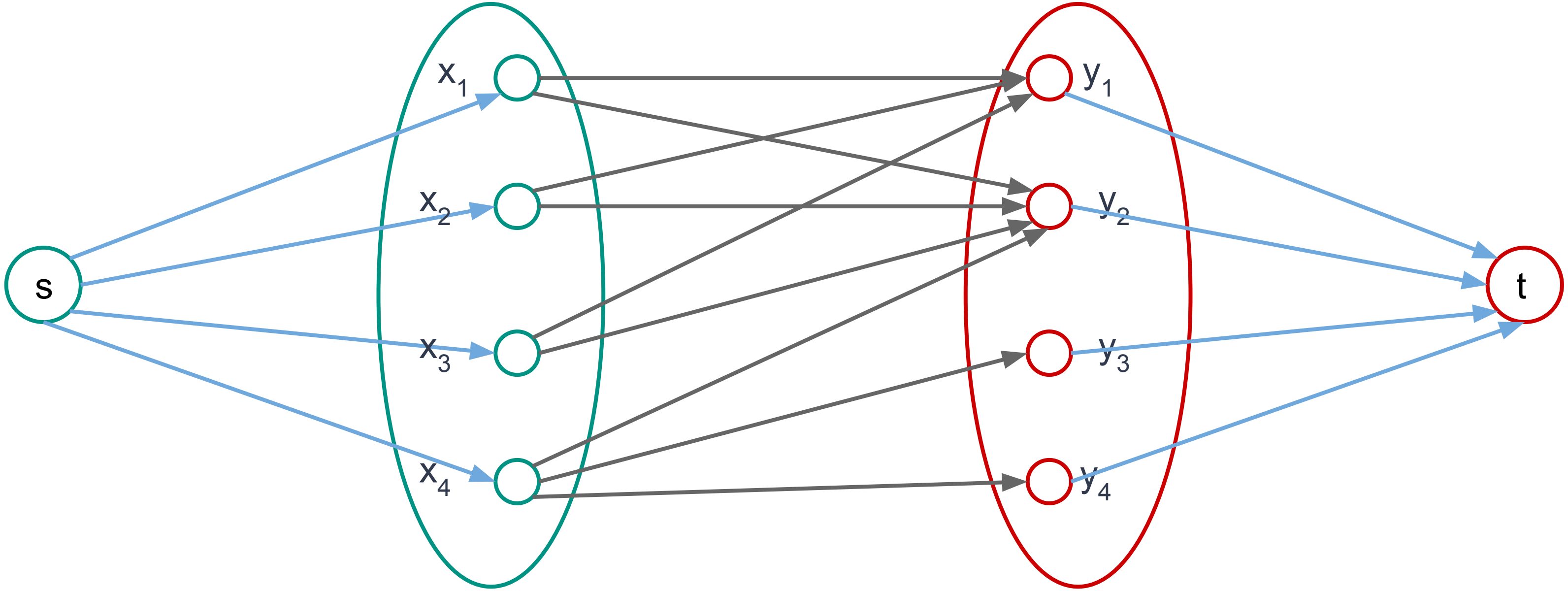
Algoritm de determinare a unui cuplaj maxim

Adăugăm arce (s, x_i), pentru $x_i \in X$ și (y_j, t) pentru $y_j \in Y$



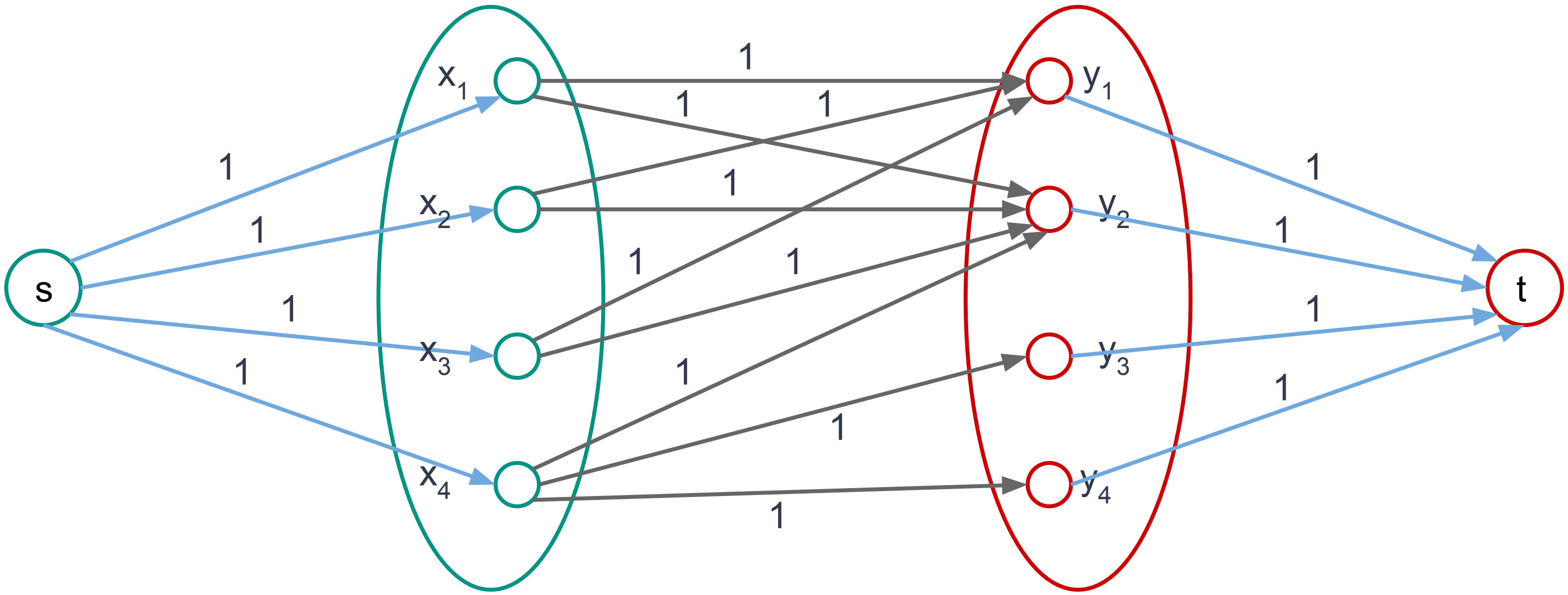
Algoritm de determinare a unui cuplaj maxim

Transformăm muchiile x_iy_j în arce (de la X la Y)



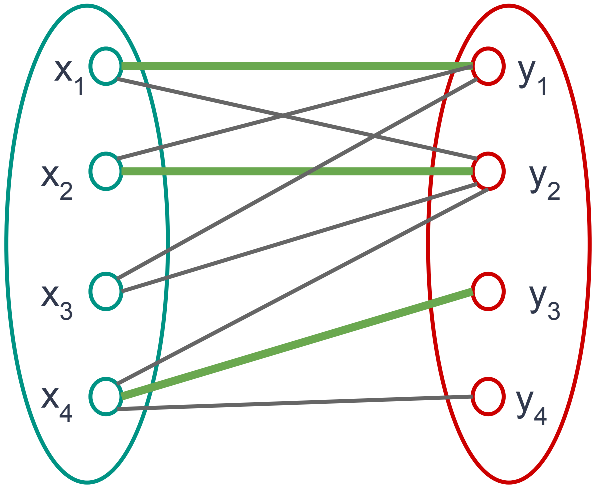
Algoritm de determinare a unui cuplaj maxim

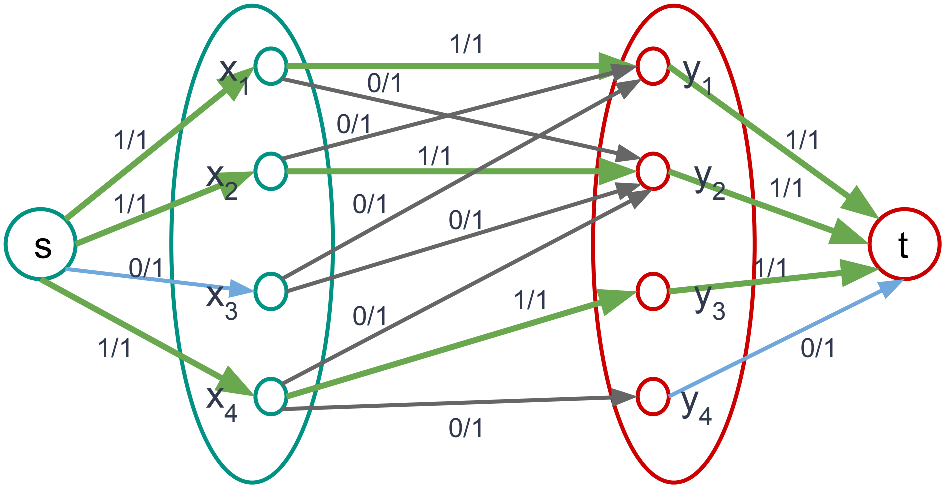
Asociem fiecărui arc capacitatea 1



Algoritm de determinare a unui cuplaj maxim

- Cuplaj M în G ⇔ flux f în rețea
- |M| = val(f)





Algoritm de determinare a unui cuplaj maxim

Proprietatea 1.

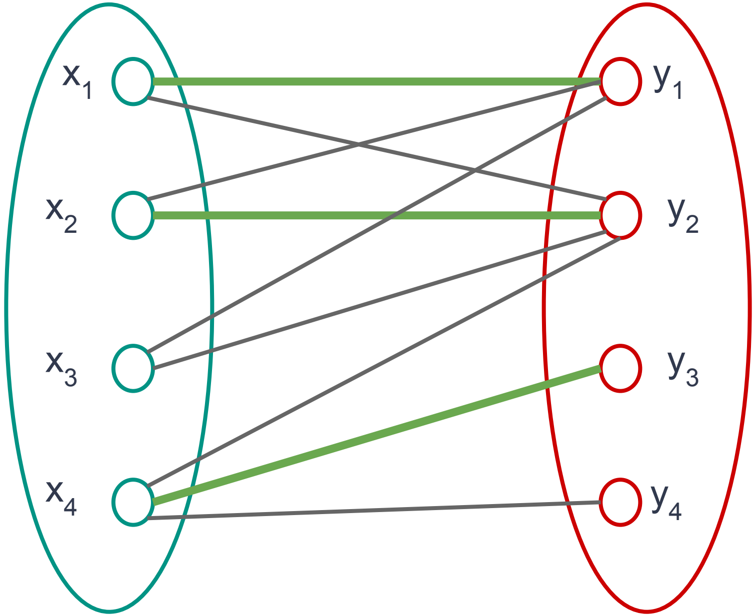
Fie $G = (X \cup Y, E)$ un graf bipartit și M un cuplaj în G.

Atunci există un flux f în rețeaua de transport asociată N_G cu **$val(f) = |M|$**.

Justificare: Dat un cuplaj M în G, se poate construi un flux f în N_G cu $val(f) = |M|$ astfel:

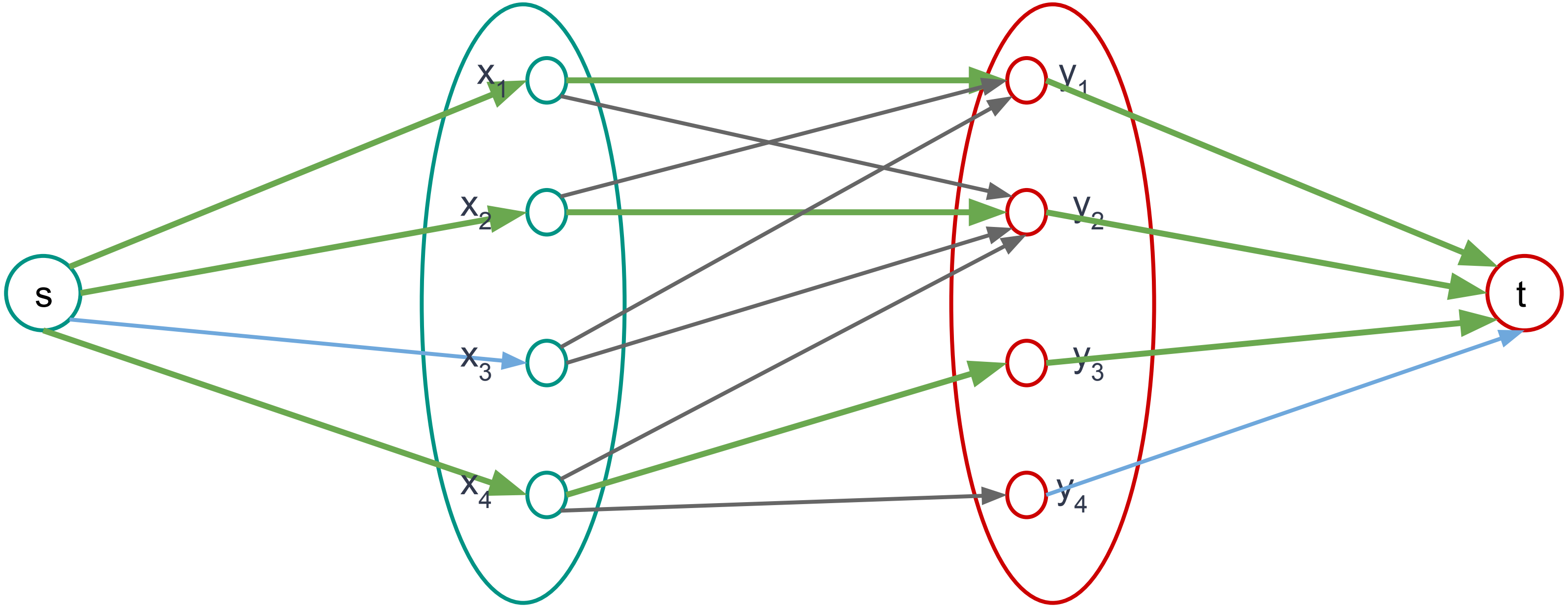
Algoritm de determinare a unui cuplaj maxim

Dat cuplaj în graf ⇒ definim un flux în rețea



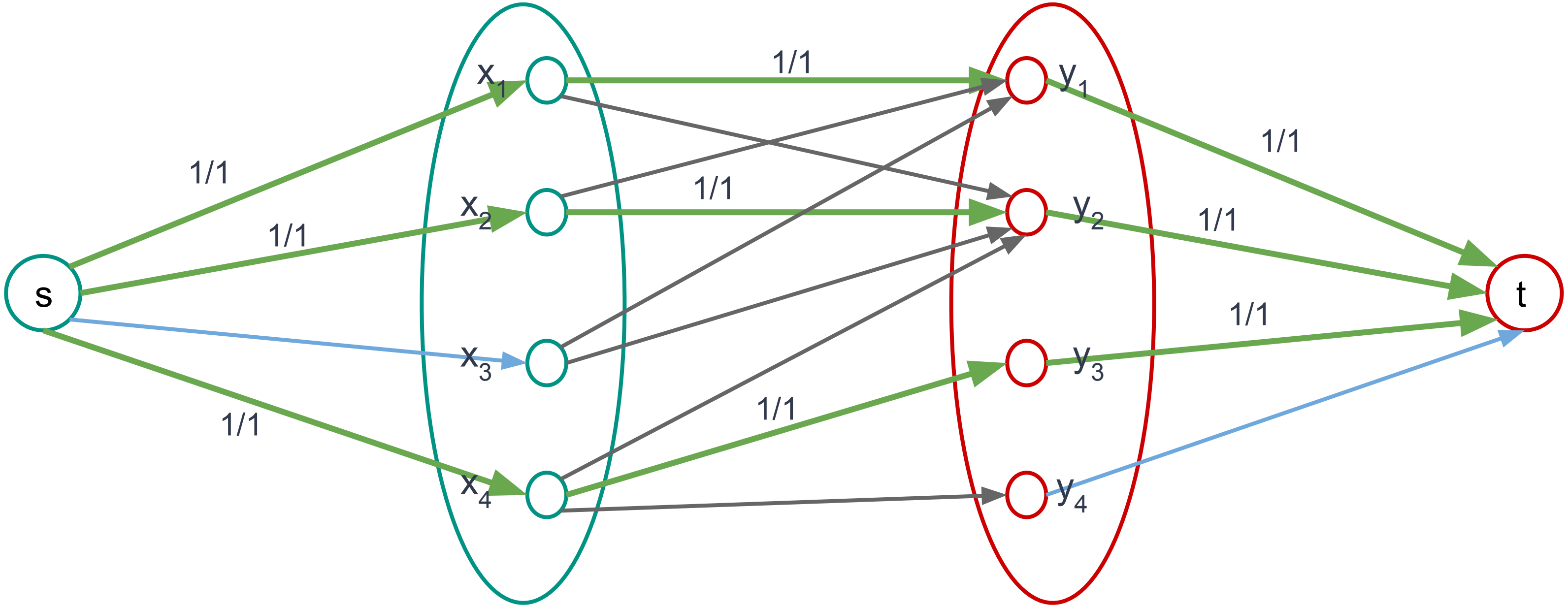
Algoritm de determinare a unui cuplaj maxim

Dat cuplaj în graf ⇒ definim un flux în rețea



Algoritm de determinare a unui cuplaj maxim

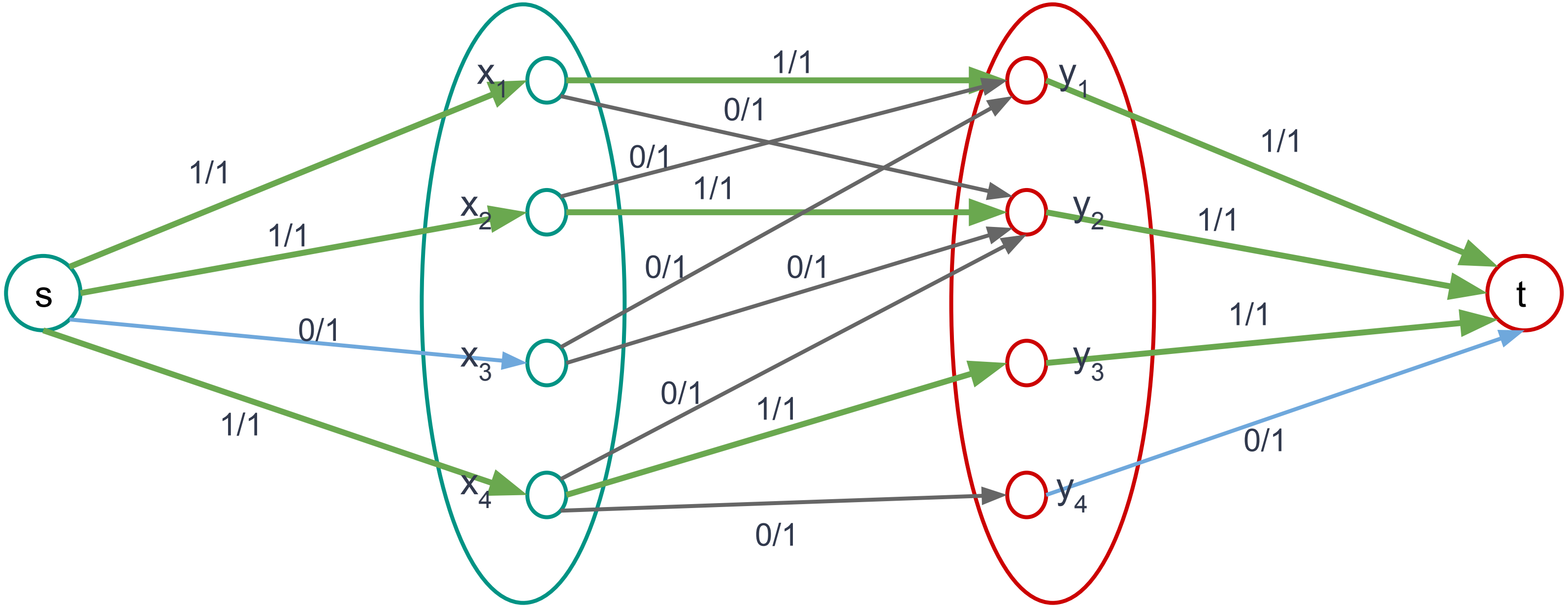
Dat cuplaj în graf ⇒ definim un flux în rețea



Celelalte arce au flux 0 și capacitate 1

Algoritm de determinare a unui cuplaj maxim

Dat cuplaj în graf ⇒ definim un flux în rețea



Algoritm de determinare a unui cuplaj maxim

Proprietatea 2.

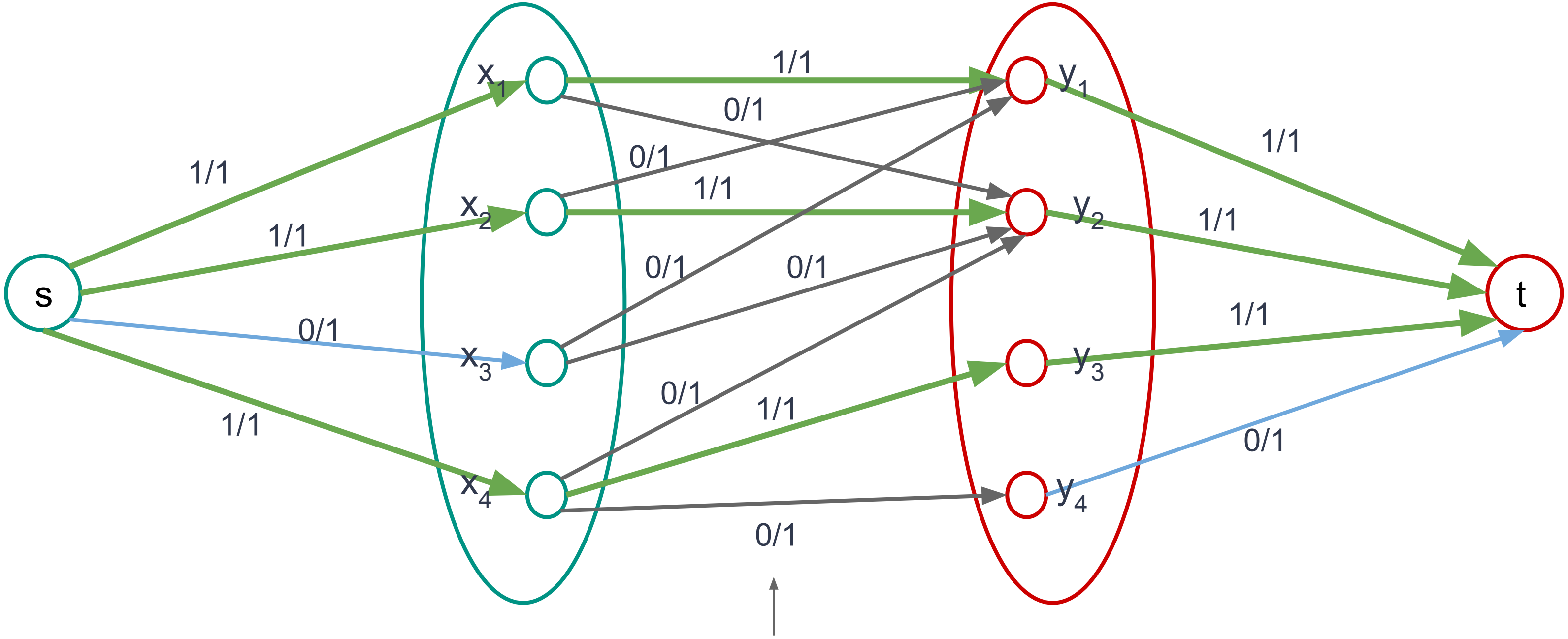
Fie $G = (X \cup Y, E)$ un graf bipartit și f un flux în rețeaua de transport N_G asociată.

Atunci există M un cuplaj în G cu **val(f) = |M|**.

Justificare: Dat un flux f în N, se poate construi un cuplaj M în G cu val(f) = |M| astfel:

Algoritm de determinare a unui cuplaj maxim

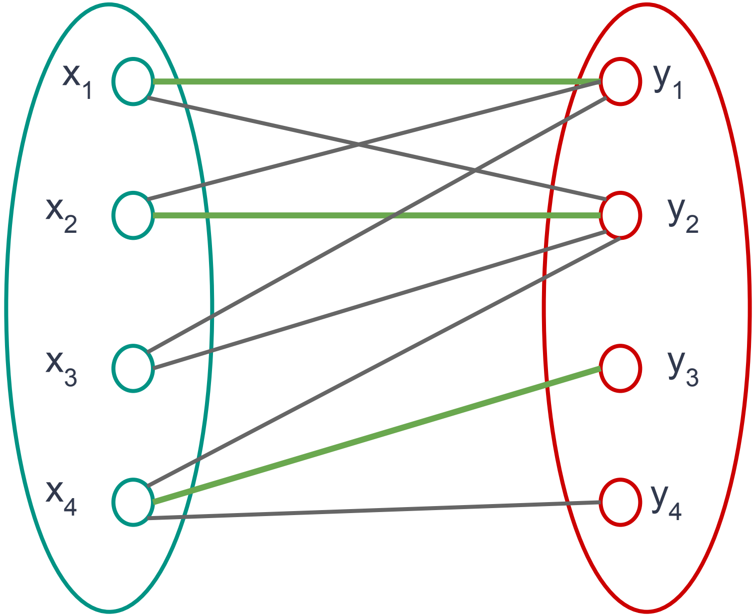
Dat cuplaj în graf ⇒ definim un flux în rețea



arcele care au flux pozitiv dau muchiile din M

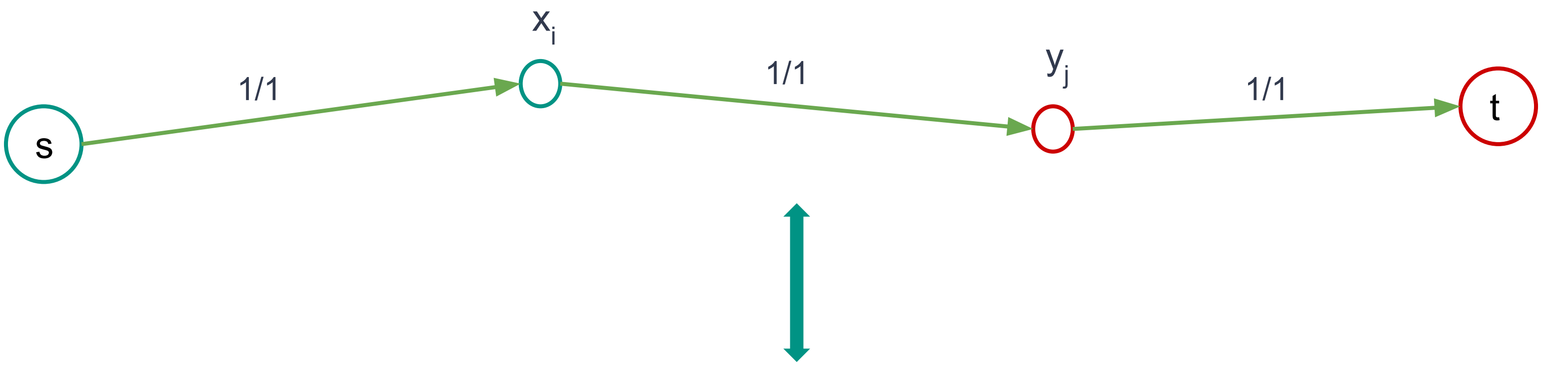
Algoritm de determinare a unui cuplaj maxim

Dat cuplaj în graf ⇒ definim un flux în rețea



Algoritm de determinare a unui cuplaj maxim

Concluzie: **Flux în rețea ⇔ cuplaj în graf**



Muchie în cuplaj

Algoritm de determinare a unui cuplaj maxim

Consecință

f^* flux maxim în N ⇒ cuplajul corespunzător M^* este cuplaj maxim în G

A determina un **cuplaj maxim** într-un graf bipartit ⇔
A determina un **flux maxim** în rețeaua asociată

Algoritm de determinare a unui cuplaj maxim

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N
3. Considerăm $M = \{ xy \mid f^*(xy)=1,\ x \in X,\ y \in Y,\ xy \in N \}$

   (pentru fiecare arc xy cu flux nenul din N, care nu este incident în s sau în t, muchia xy corespunzătoare din G se adaucă la M)
4. return M

Algoritm de determinare a unui cuplaj maxim

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N
3. Considerăm M = { xy | f*(xy)=1, x∈X, y∈Y, xy∈N }

   (pentru fiecare arc xy cu flux nenul din N, care nu este incident în s sau în t, muchia xy corespunzătoare din G se adaucă la M)
4. return M

Complexitate?

Algoritm de determinare a unui cuplaj maxim

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N
3. Considerăm M = { xy | f*(xy)=1, x∈X, y∈Y, xy∈N }

   (pentru fiecare arc xy cu flux nenul din N, care nu este incident în s sau în t, muchia xy corespunzătoare din G se adaucă la M)
4. return M

Complexitate? C = 1 (sau L ≤ $c^+(s)$ ≤ n) ⇒ O(mn)

Aplicație
Construcția unui graf orientat din secvențele de grade

Aplicație

Se dau secvențele

- $s_0^+ = \{d_1^+, \ldots, d_n^+\}$
- $s_0^- = \{d_1^-, \ldots, d_n^-\}$

cu $d_1^+ + \ldots + d_n^+ = d_1^- + \ldots + d_n^-$

Să se construiască, **dacă se poate**, un grad orientat G, cu $s^+(G) = s_0^+$ și $s^-(G) = s_0^-$

Exemplu:

- $s_0^+ = \{1, 0, 2\}$
- $s_0^- = \{1, 1, 1\}$

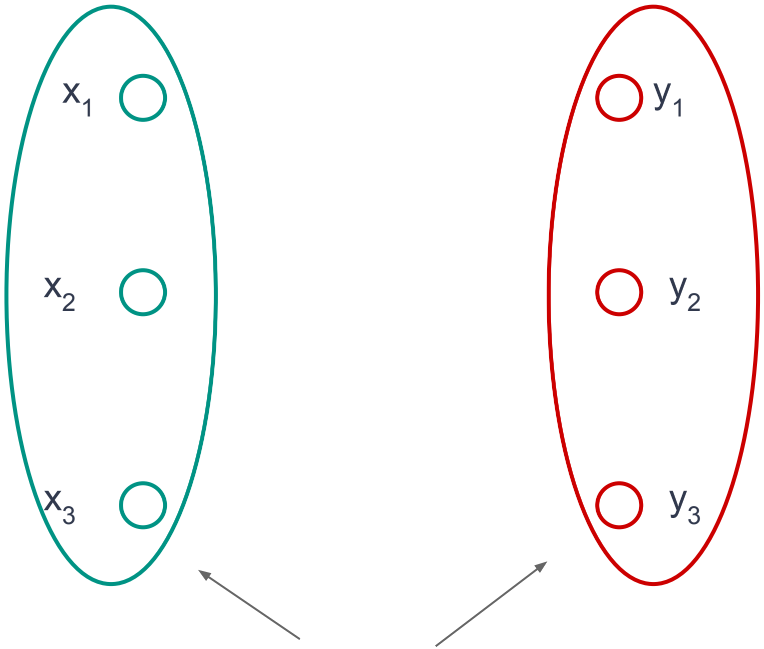
Aplicație

Exemplu:

- $s_0^+ = \{1, 0, 2\}$
- $s_0^- = \{1, 1, 1\}$

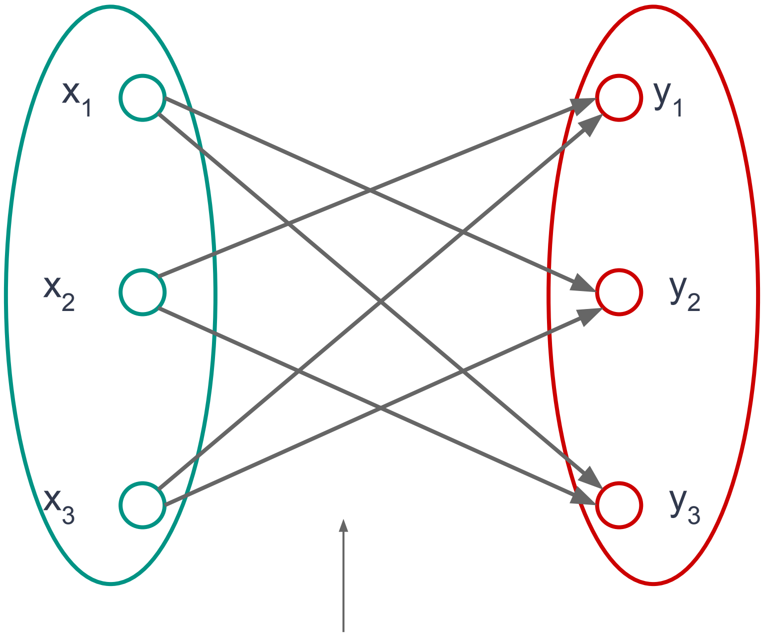
Construim o rețea asociată celor două secvențe a.î. din fluxul maxim în rețea să putem **deduce** dacă G se poate construi + arcele grafului G (în caz afirmativ).

Aplicație



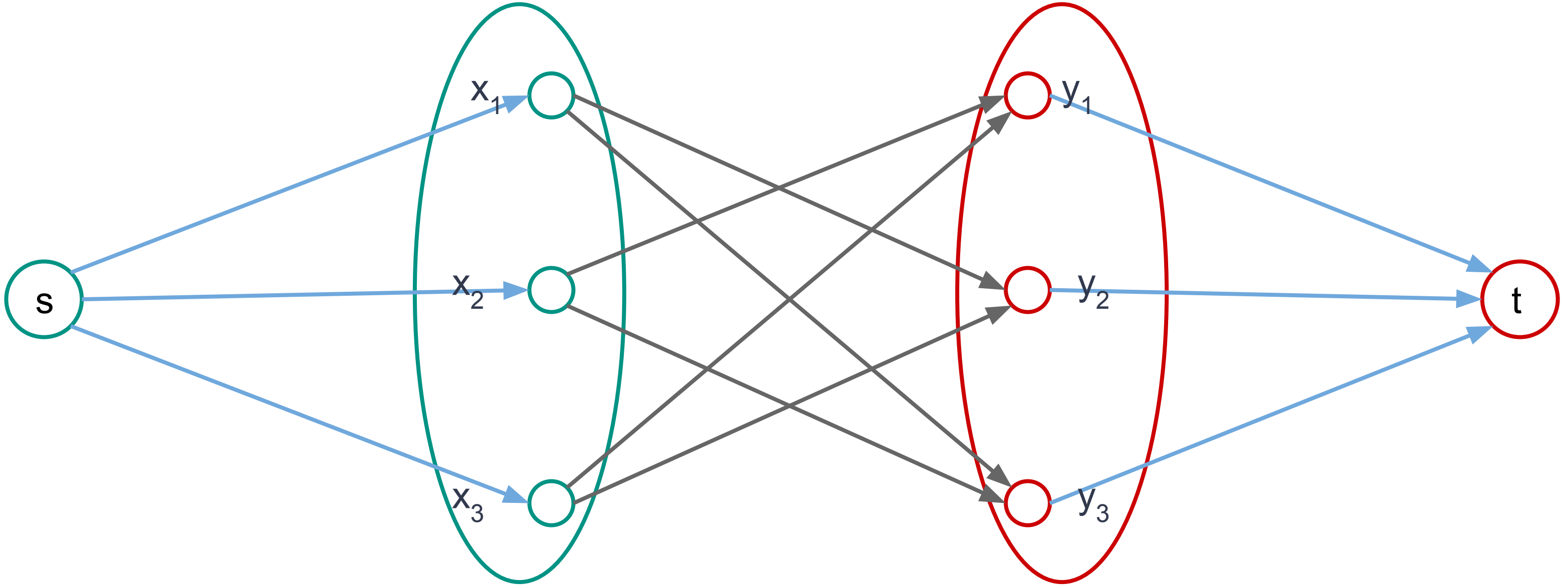
Vârfurile 1, 2, …, n se pun în ambele clase ale bipartiției (câte o copie)

Aplicație

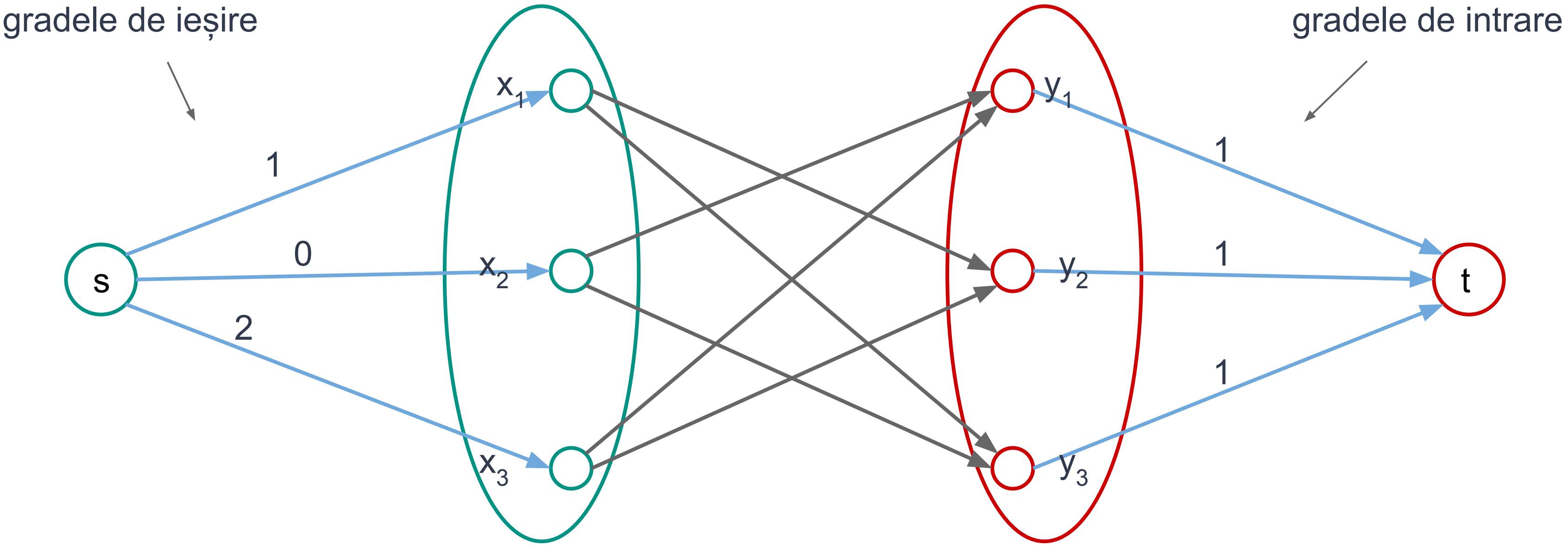


arce x_iy_j cu $i \neq j$
(fluxul pe arcul x_iy_j va fi nenul $\Leftrightarrow$ $ij \in E(G)$)

Aplicație

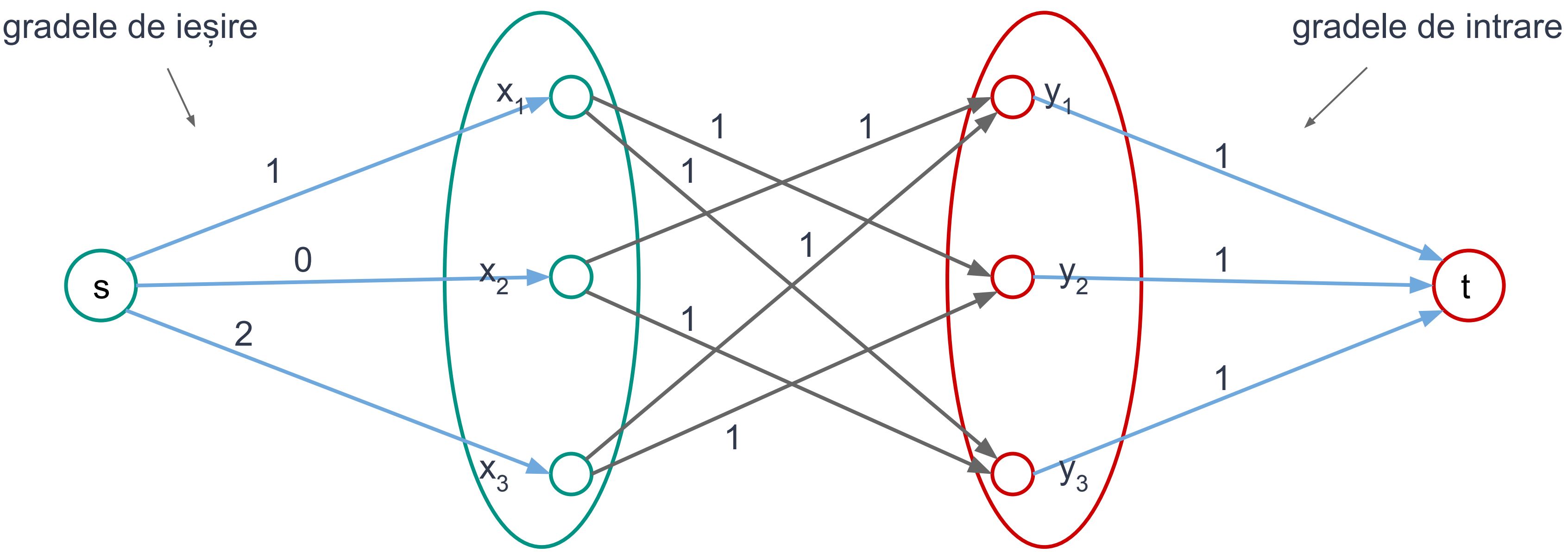


Aplicație



gradele de ieșire
gradele de intrare
1
0
2
1
1
1
s
t
x_1
x_2
x_3
y_1
y_2
y_3

Aplicație



gradele de ieșire
gradele de intrare
s
t
x1
x2
x3
y1
y2
y3
1
0
2
1
1
1
1
1
1
1
1
1

Aplicație

Proprietate

Există graf cu secvențele date ⇔ în rețeaua asociată, **fluxul de valoare maximă are**

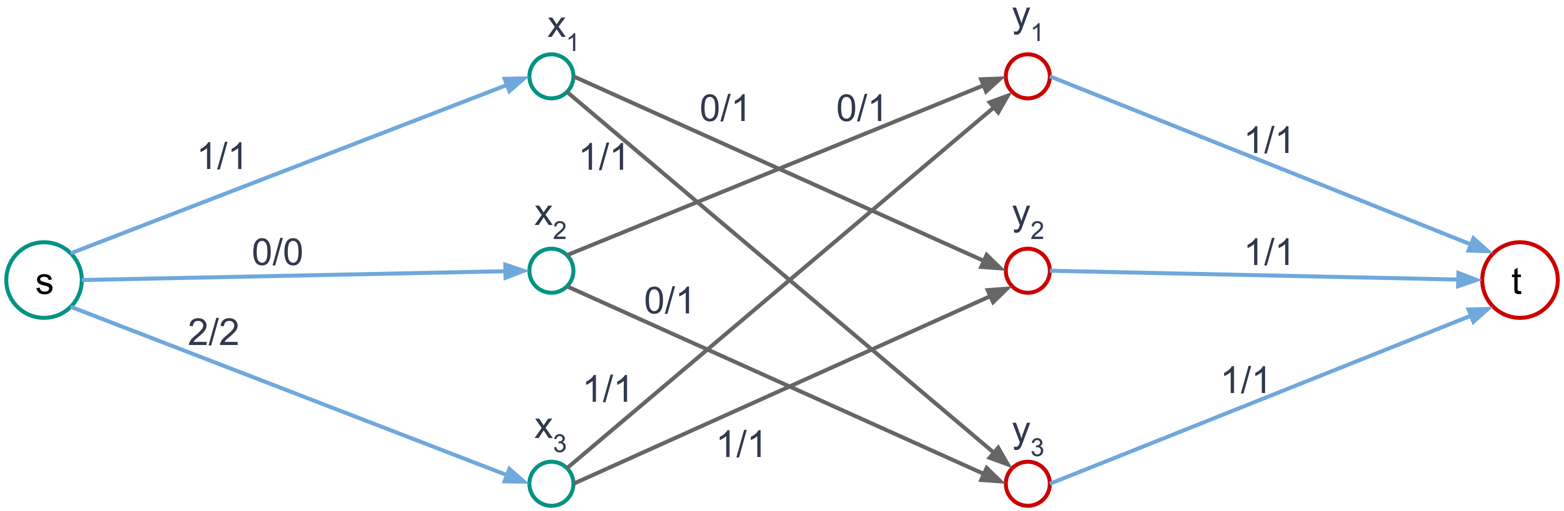
$$val(f) = d_1^+ + \ldots + d_n^+ = d_1^- + \ldots + d_n^-$$

(saturează toate arcele care ies din s și toate arcele care intră în t)

Tăieturile ({s}, V-{s}) , (V-{t}, t) sunt minime

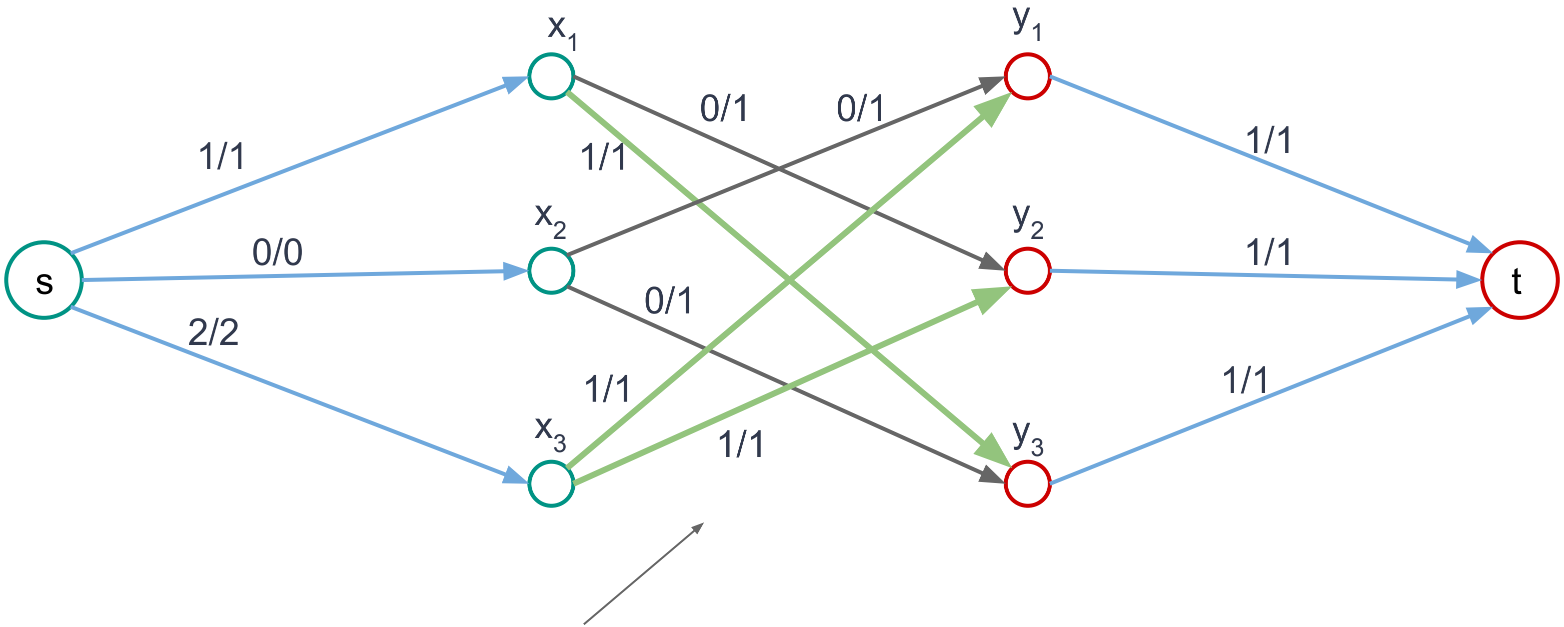
Aplicație

Flux în rețea care saturează arcele din s și t $\Rightarrow$ G



Aplicație

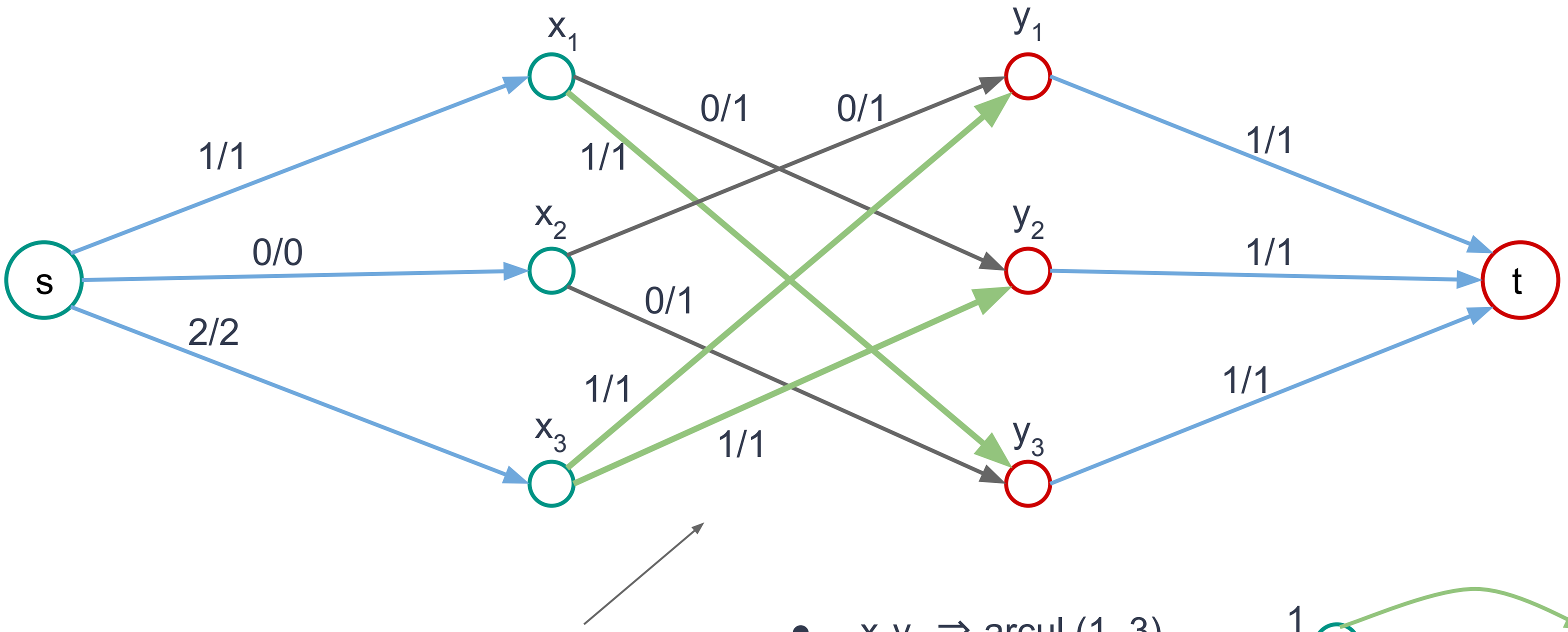
Flux în rețea care saturează arcele din s și t ⇒ G



arcele care au flux pozitiv dau arcele lui G

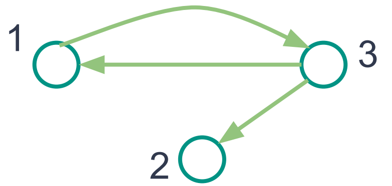
Aplicație

Flux în rețea care saturează arcele din s și t $\Rightarrow$ G



arcele care au flux pozitiv dau arcele lui G

- $x_1y_3 \Rightarrow$ arcul (1, 3)
- $x_3y_1 \Rightarrow$ arcul (3, 1)
- $x_3y_2 \Rightarrow$ arcul (3, 2)



Aplicație

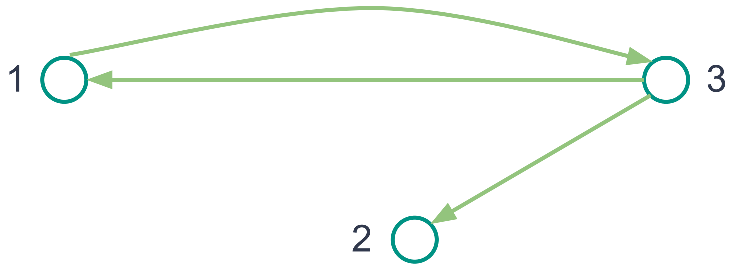
Reciproc

Flux în rețea care saturează arcele din s și t $\Leftarrow$ G

Aplicație

Reciproc

Flux în rețea care saturează arcele din s și t $\Leftarrow$ G



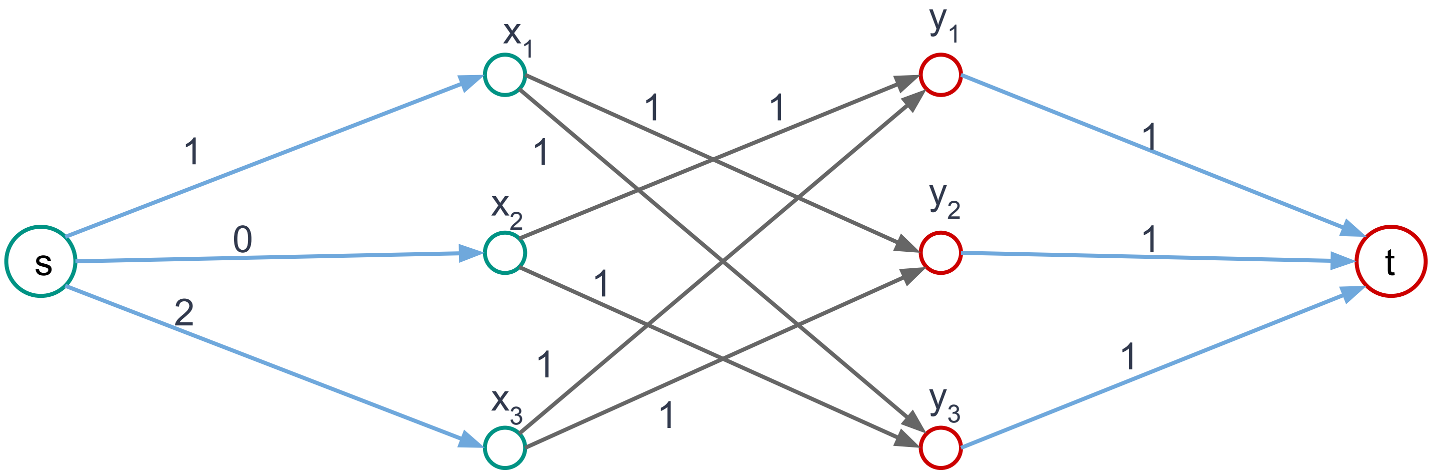
G

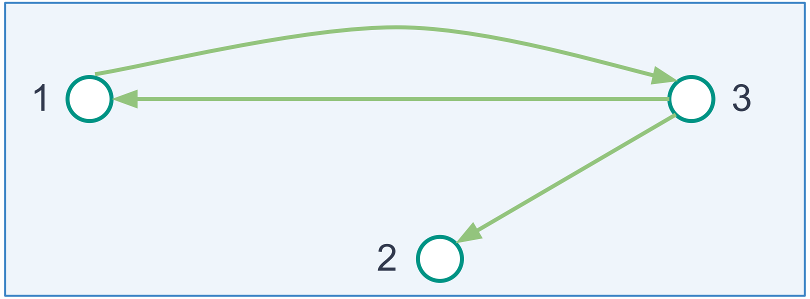
$s_0^+ = \{1, 0, 2\}$

$s_0^- = \{1, 1, 1\}$

Aplicație

Flux în rețea care saturează arcele din s și t $\Leftarrow$ G

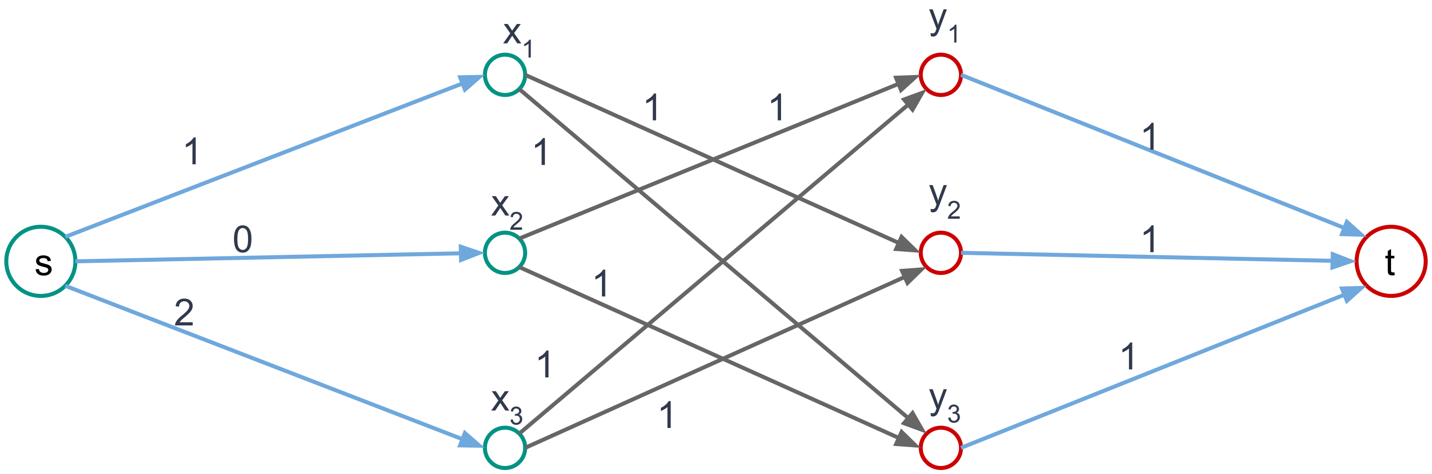




G

Aplicație

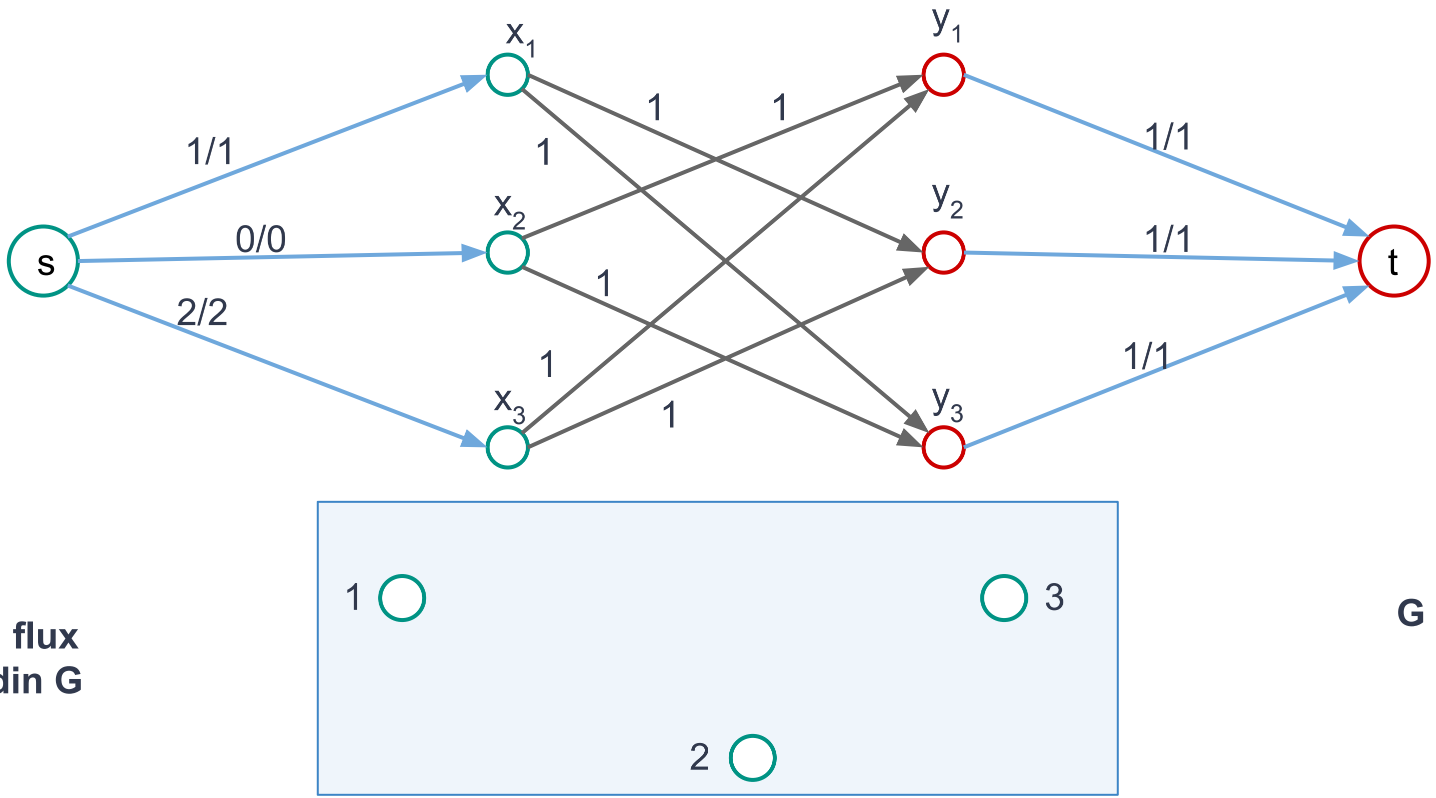
Flux în rețea care saturează arcele din s și t ⇐ G



saturăm arcele din s și t

Aplicație

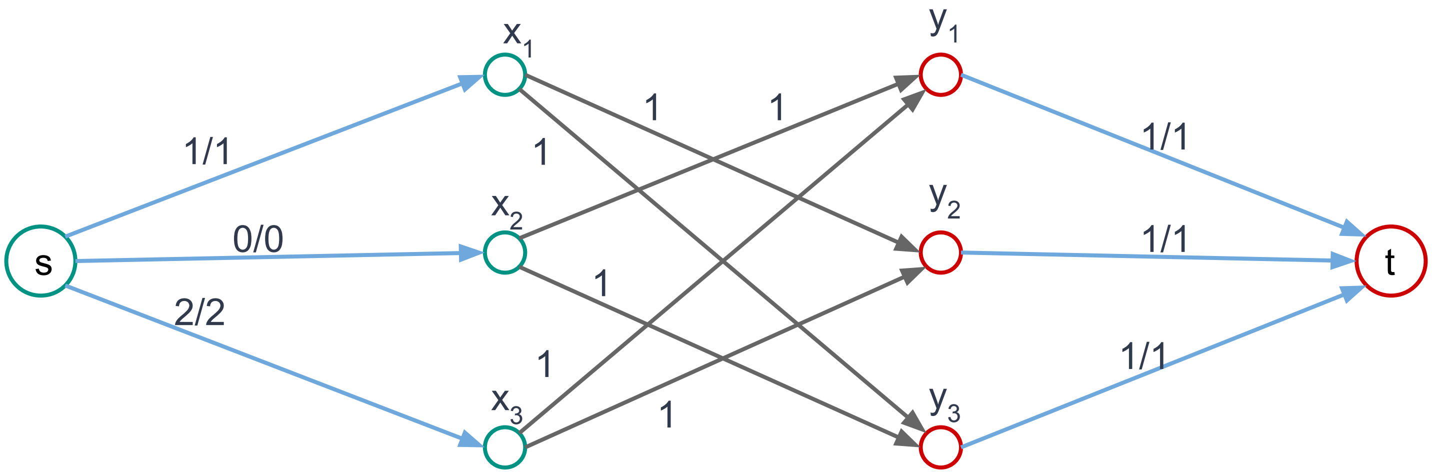
Flux în rețea care saturează arcele din s și t ⇐ G



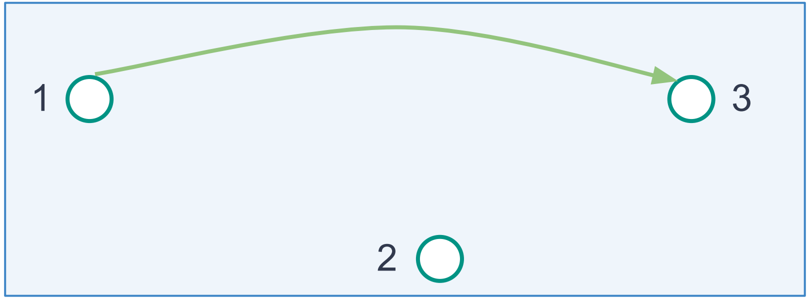
asociem flux arcelor din G

Aplicație

Flux în rețea care saturează arcele din s și t $\Leftarrow$ G



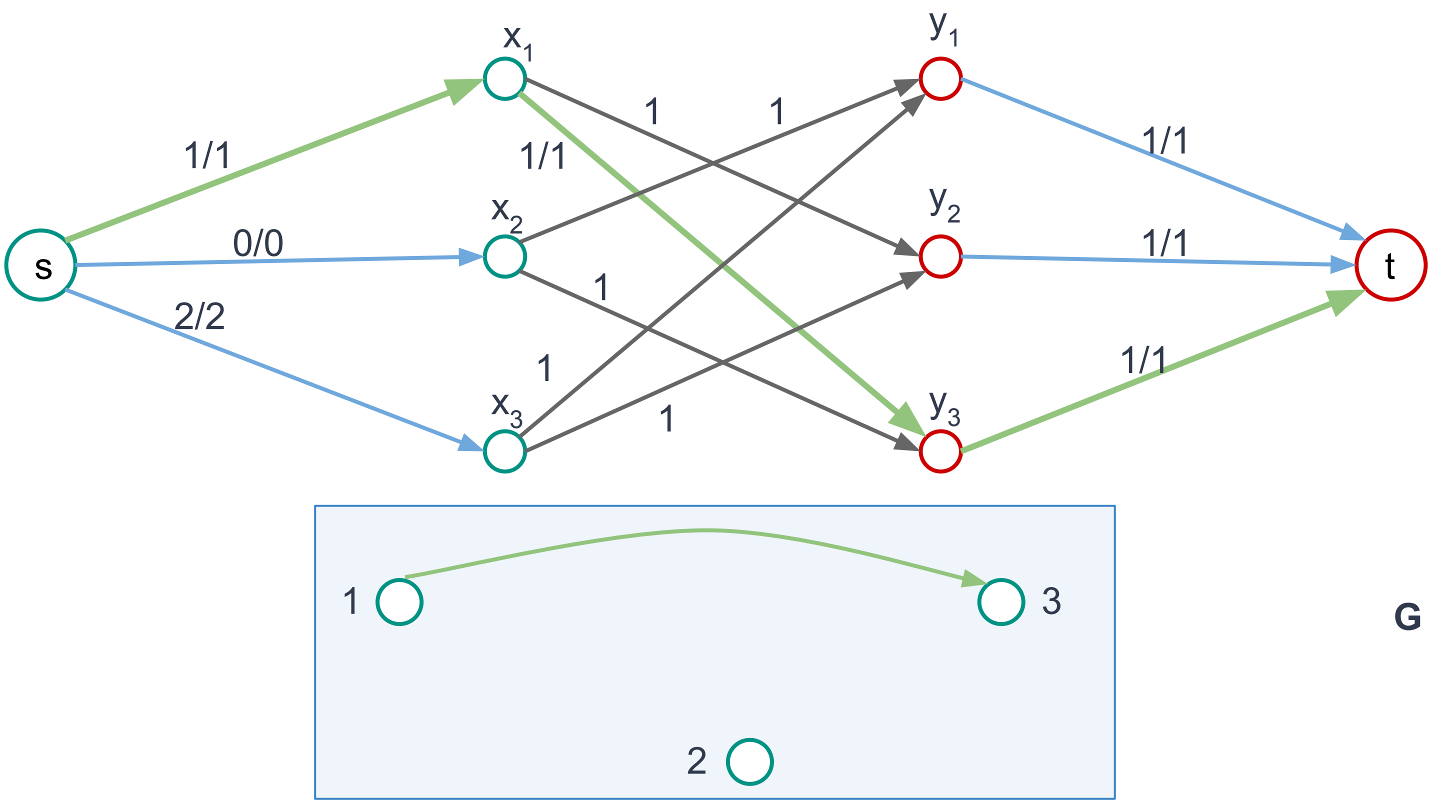
Arc (1, 3) $\Rightarrow$ flux pe arcul x_1y_3



G

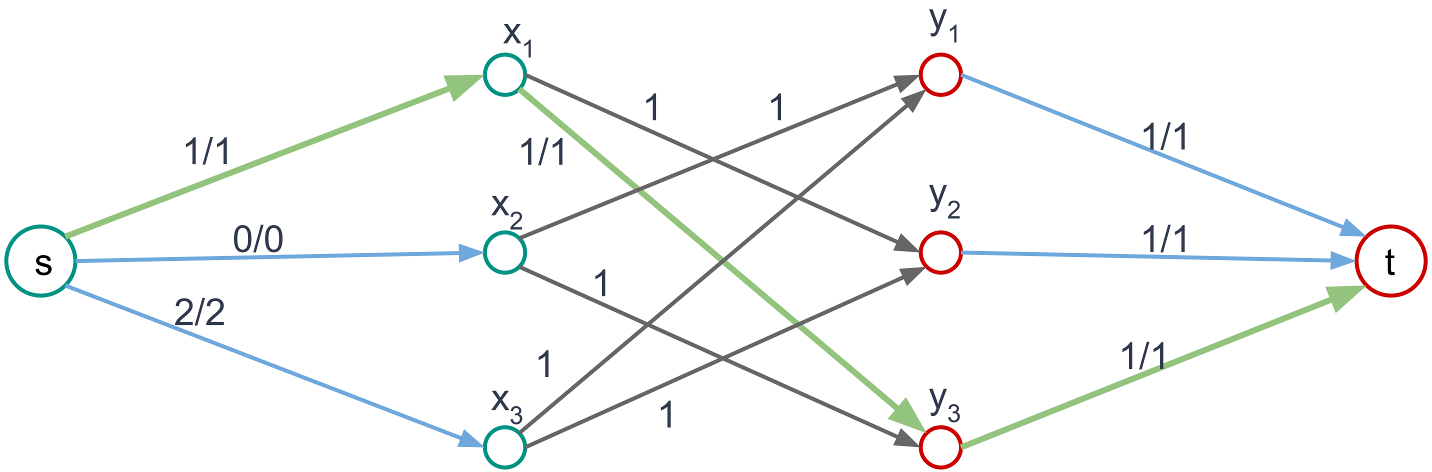
Aplicație

Flux în rețea care saturează arcele din s și t ⇐ G

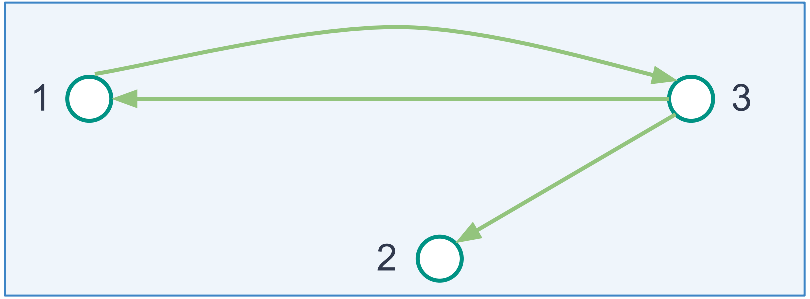


Aplicație

Flux în rețea care saturează arcele din s și t $\Leftarrow$ G



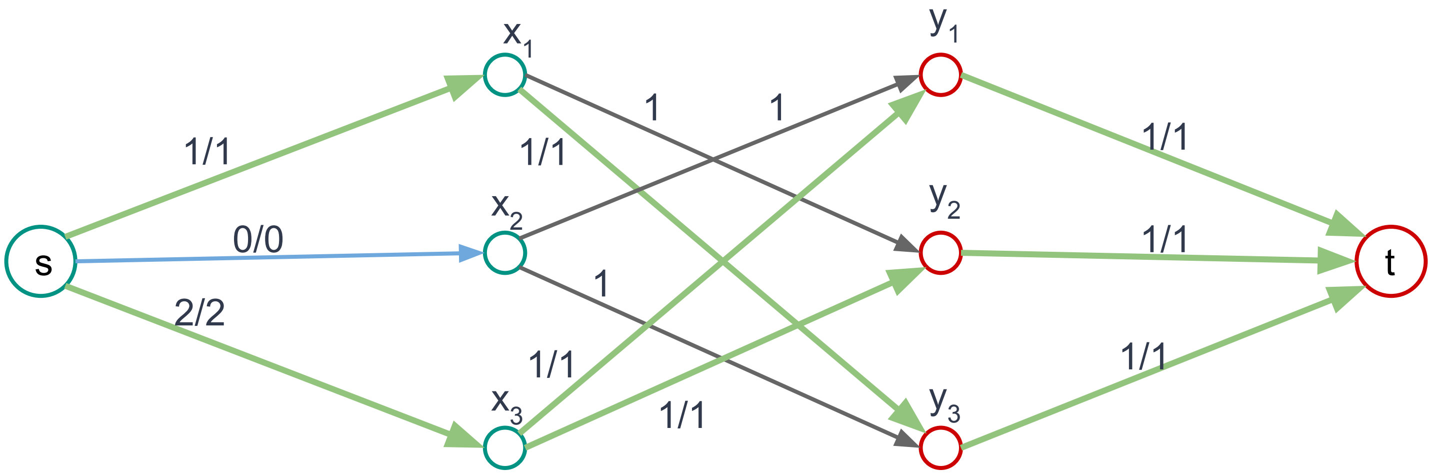
Arce (3, 1), (3, 2)
$\Rightarrow$?



G

Aplicație

Flux în rețea care saturează arcele din s și t ⇐ G



Restul arcelor au fluxul 0

Algoritm

Algoritm de determinare a unui graf orientat G cu $s^+(G) = s_0^+$ și $s^-(G) = s_0^-$

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N

Algoritm

Algoritm de determinare a unui graf orientat G cu $s^+(G) = s_0^+$ și $s^-(G) = s_0^-$

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N
3. Dacă val(f*) < $d_1^+ + \dots + d_n^+$ atunci

   Nu există G. **STOP**

Algoritm

Algoritm de determinare a unui graf orientat G cu $s^+(G) = s_0^+$ și $s^-(G) = s_0^-$

1. Construim N rețeaua de transport asociată
2. Determinăm f* flux maxim în N
3. Dacă $val(f^*) < d_1^+ + \dots + d_n^+$ atunci

   Nu există G. **STOP**
4. $V(G) = \{1, \dots, n\}$

   $E(G) = \{ij \mid x_i y_j \in N \text{ cu } f^*(x_i y_j) = 1\}$

Complexitate: $L \le c^+(s) = d_1^+ + \dots + d_n^+ = m \Rightarrow O(m^2)$

Aplicații
Alte probleme de asociere

Probleme de asociere (temă)

Se dau 2 mulțimi de obiecte, spre exemplu produse (aflate în fabrici) și clienți (joburi / mașini, pagini web / servere, echipe turneu etc).

- Pentru fiecare produs x se cunoaște
  - **c(x)** = numărul de unități disponibile din produsul x
- Pentru fiecare client y se cunoaște
  - **c(y)** = numărul maxim de unități de produse pe care le poate primi (în total, din toate produsele)
- Pentru fiecare pereche produs-client (x, y) se cunoaște
  - **c(x, y)** = numărul maxim de unități din produsul x pe care le poate primi clientul y

Să se determine o modalitate de a distribui cât mai multe produse (unităţi de produse) clienților cu respectarea constrângerilor.

Probleme de asociere (temă)

Observație: Problema determinării unui cuplaj de cardinal maxim într-un graf bipartit $G=(X \cup Y, E)$ este un caz particular al acestei probleme, pentru:

- $c(x) = c(y) = 1,\quad \forall\ x \in X,\ y \in Y$
- $c(x, y) = 1,\quad$ dacă $xy \in E$
- $c(x, y) = 0,\quad$ dacă $xy \notin E$

Aplicație
Drumuri arc-disjuncte între două vârfuri. Conectivitatea unui graf

SUPLIMENTAR

Drumuri arc-disjuncte

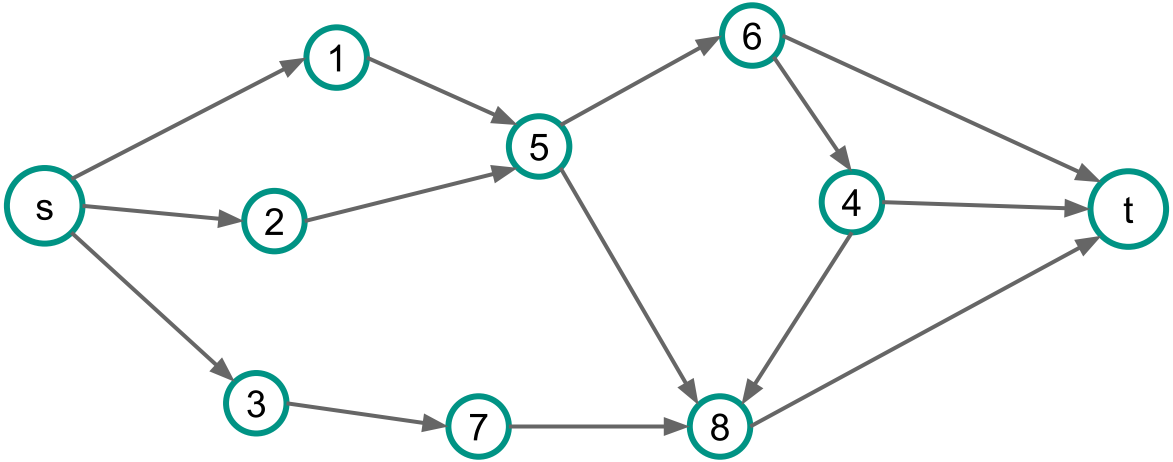
Se dau:

- $G = (V, E)$ - orientat, conex (graful neorientat suport)
- s, t - două vârfuri

Să se determine numărul maxim k de **s-t drumuri elementare arc-disjuncte (+ k astfel de drumuri)**.

Două drumuri P_1, P_2 se numesc **arc-disjuncte** dacă $E(P_1) \cap E(P_2) = \varnothing$

Drumuri arc-disjuncte



Drumuri arc-disjuncte

Se dau:

- G = (V, E) - orientat, conex (graful neorientat suport)
- s, t - două vârfuri

Să se determine numărul maxim k de **s-t drumuri elementare arc-disjuncte (+ k astfel de drumuri)**.

Aplicații:

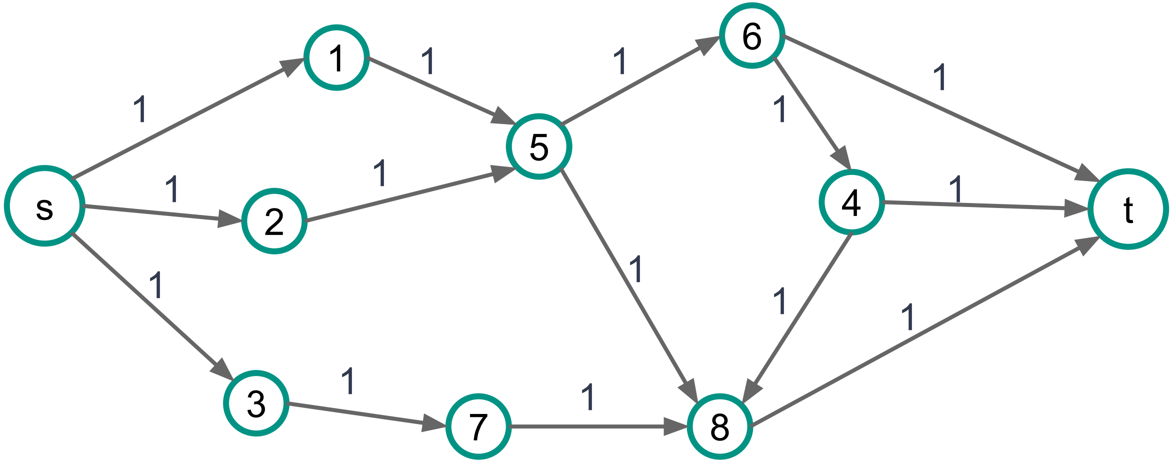
- Fiabilitatea rețelelor, conectivitate
- Probleme de strategie
- Măsuri de centralitate (a unui nod) în rețele sociale

Jon Kleinberg, Éva Tardos, **Algorithm Design**, Addison-Wesley 2005

Drumuri arc-disjuncte

Intuitiv:

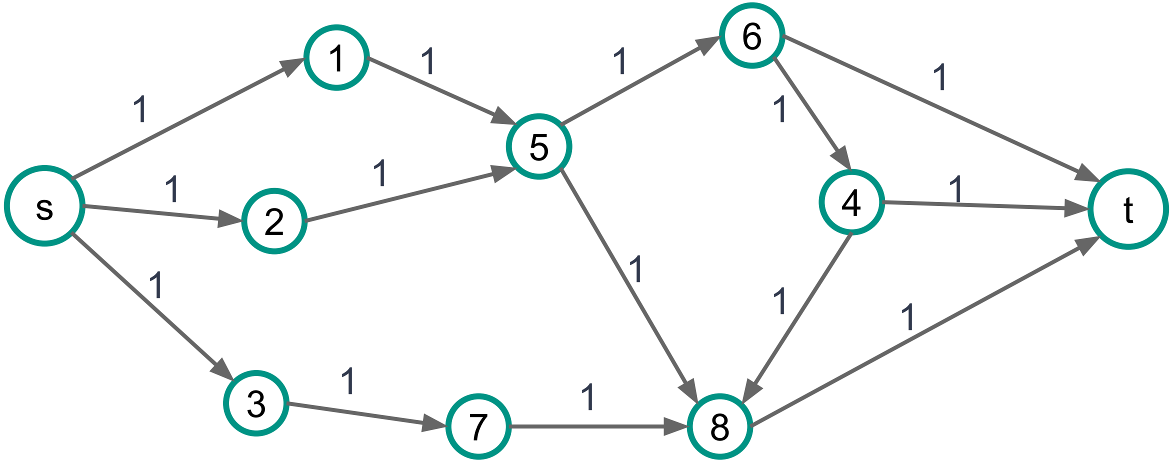
- asociem fiecărui arc capacitatea **1**
- fluxul maxim: f(e) ∈ {0, 1}



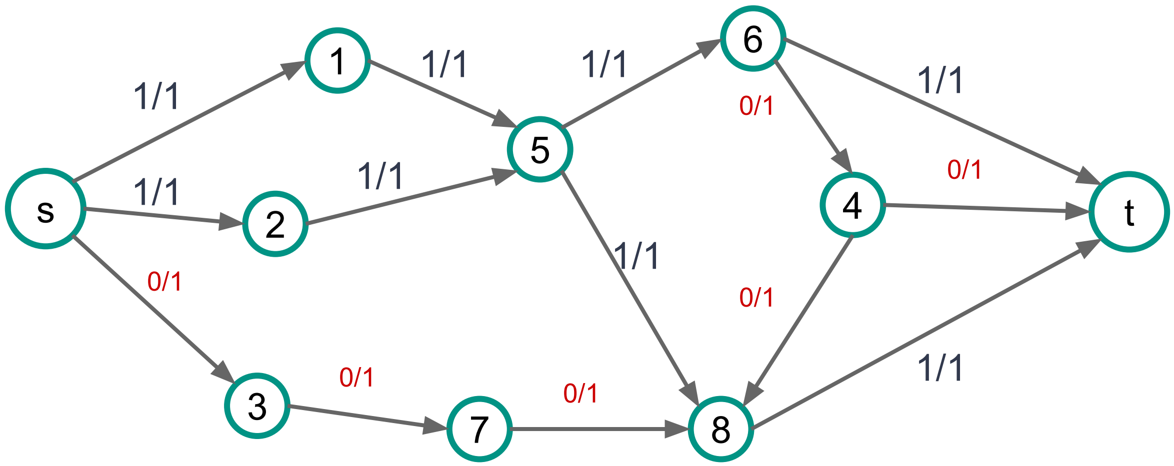
Drumuri arc-disjuncte

Intuitiv:

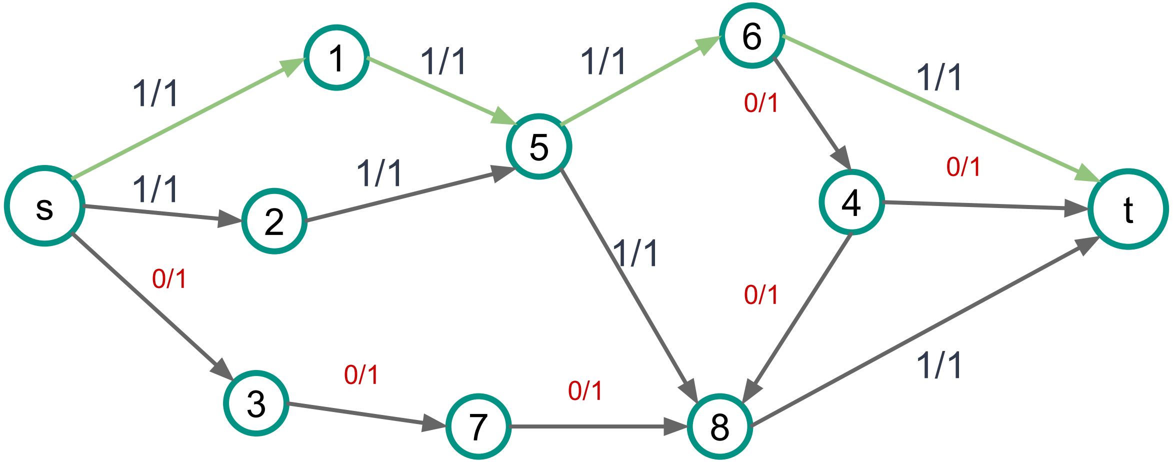
- asociem fiecărui arc capacitatea **1**
- fluxul maxim: $f(e) \in \{0, 1\}$
- un drum de la s la t = **traseul parcurs de o unitate de flux de la s la t**
- **numărul de s-t drumuri arc-disjuncte** = valoarea fluxului maxim



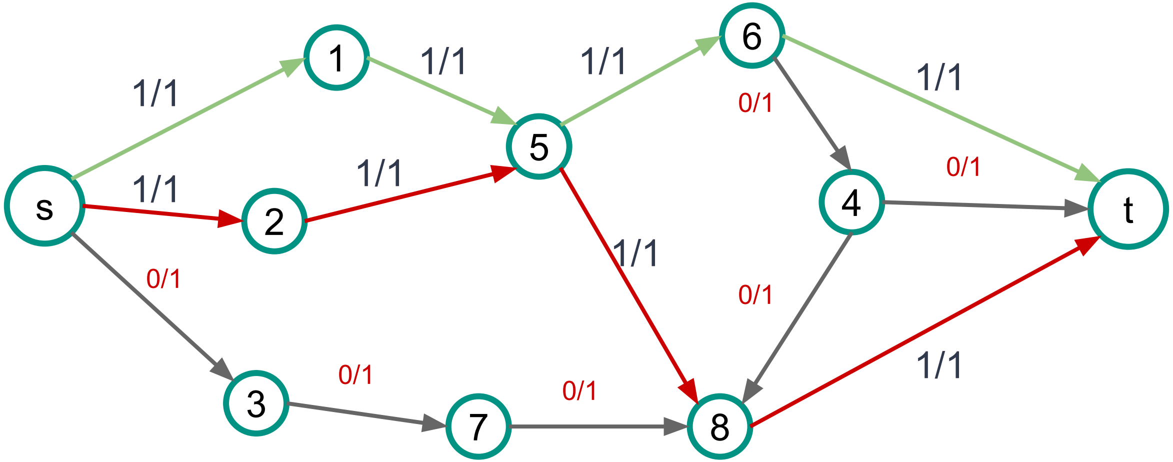
Drumuri arc-disjuncte



Drumuri arc-disjuncte



Drumuri arc-disjuncte



Drumuri arc-disjuncte

Teorema lui Menger

Fie G graf orientat, s, t două vârfuri distincte în G.

Numărul minim de arce care trebuie eliminate pentru ca s și t să nu mai fie conectate printr-un drum (să fie **separate**) =

numărul maxim de drumuri arc-disjuncte de la s la t

Drumuri arc-disjuncte

Teorema lui Menger

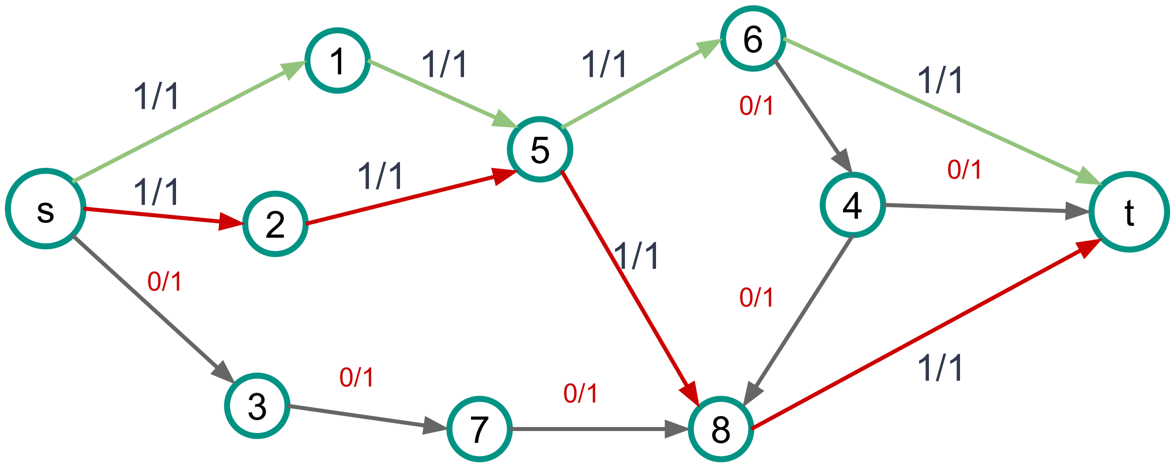
Fie G graf orientat, s, t două vârfuri distincte în G.

Numărul minim de arce care trebuie eliminate pentru ca s și t să nu mai fie conectate printr-un drum (să fie **separate**) =

numărul maxim de drumuri arc-disjuncte de la s la t

O astfel de mulțime de arce se poate determina cu algoritmul Ford-Fulkerson?

Drumuri arc-disjuncte



Drumuri arc-disjuncte

<u>Teorema lui Menger</u>

Fie G graf orientat, s, t două vârfuri distincte în G.

Numărul minim de arce care trebuie eliminate pentru ca s și t să nu mai fie conectate printr-un drum (să fie **separate**) =

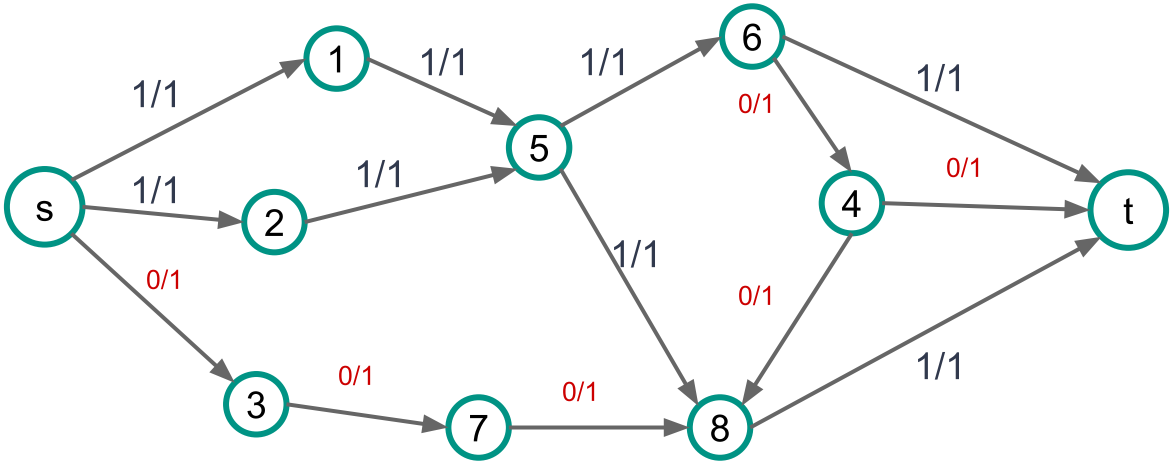
numărul maxim de drumuri arc-disjuncte de la s la t

O astfel de mulțime de arce se poate determina cu algoritmul Ford-Fulkerson?

Sunt arcele directe ale tăieturii minime

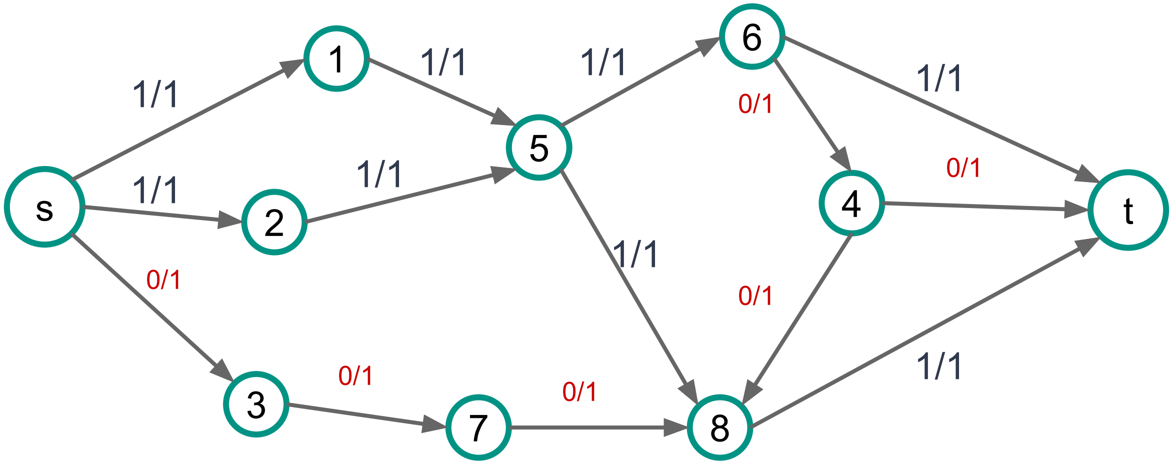
Drumuri arc-disjuncte

Cum determinăm tăietura minimă?



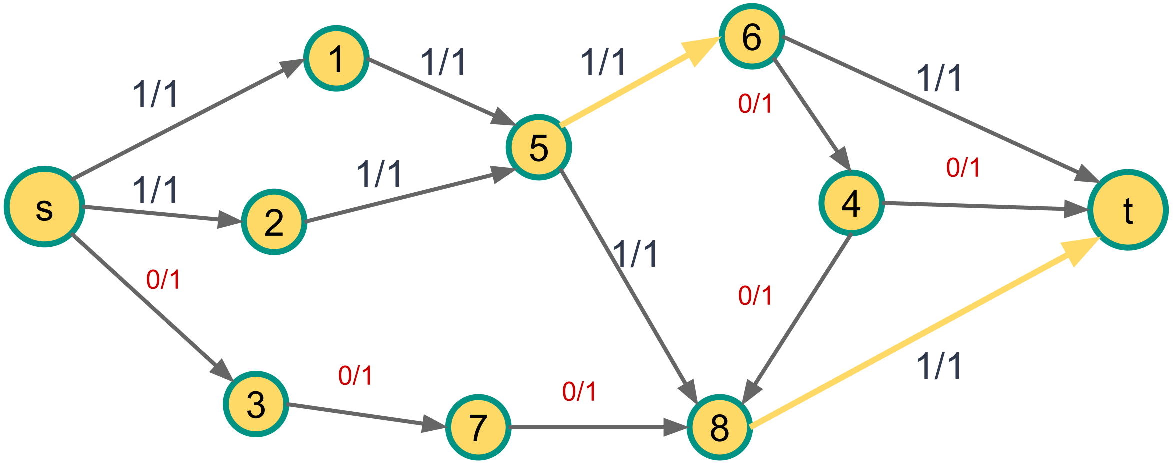
Drumuri arc-disjuncte

Mulțimea vârfurilor accesibile din s prin lanțuri f-nesaturate



Drumuri arc-disjuncte

Mulțimea vârfurilor accesibile din s prin lanțuri f-nesaturate



Drumuri arc-disjuncte

Variante

- Aceeași problemă pentru

  $G = (V, E)$ - **neorietat** conex, $|E| > 2$

- Aceeași problemă pentru **vârfuri** (s-t drumuri care nu au vârfuri interne în comun)

Drumuri arc-disjuncte

- **Muchie - conectivitatea lui G k'(G)** = cardinalul **minim** al unei mulțimi de muchii F ⊆ E cu proprietatea că

  G - F nu mai este conex

- Dacă k'(G) ≥ t, G se numește **t-muchie conex**
  - Amintim (laborator + seminar)
    - există muchie critică ⇒ G este 1-conex
    - nu există muchie critică ⇒ G este 2-conex

- **Cu ajutorul algoritmului de flux maxim, putem determina (muchie)-conectivitatea unui graf**