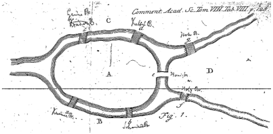
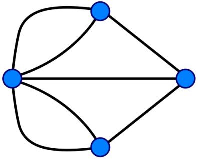




Algoritmica grafurilor

IV. Programare liniara, arbori si paduri

Mihai Suciu

Facultatea de Matematică și Informatică (UBB)
Departamentul de Informatică

Martie, 21, 2019

Continut

Programare liniară

Problema generală

fie o matrice $\boldsymbol{A}$ de dimensiune $\boldsymbol{m}$ $\boldsymbol{x}$ $\boldsymbol{n}$, un vector $\boldsymbol{b}$ de dimensiune $\boldsymbol{m}$ și un vector $\boldsymbol{c}$ de dimensiune $\boldsymbol{n}$. Trebuie găsit un vector x de n elemente care maximizează funcția obiectiv

$$\sum_{i=1}^{n} c_i x_i$$

și satisface m constrângeri date de

$$Ax \leq b.$$

- în unele cazuri nu prezintă interes funcția obiectiv, se dorește găsirea unei soluții fezabile (orice vector x ce satisface $Ax \leq b$) sau sa se arate că nu există astfel de soluții

Sistem de constrângeri

- într-un sistem de constrângeri fiecare rând din matricea A conține o valoare -1, o valoare 1 și restul valorilor sunt 0
- astfel constrângerile date de $Ax \leq b$ sunt un set de m constrângeri cu n necunoscute unde fiecare constrângere este o inecuație de forma

$$x_j - x_i \leq b_k,$$

unde $1 \leq i, j \leq n, i \neq j$ și $1 \leq k \leq m$.

Exemplu

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -0 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Exemplu (II)

- problema cere să se găsească x_1, x_2, x_3, x_4, x_5 pentru cele 8 constrângeri

$$x_1 - x_2 \leq 0,$$

$$x_1 - x_5 \leq -1,$$

$$\ldots$$

- soluția nu este unică, două posibile soluții:

$$x = (-5, -3, 0, -1, -4)$$

$$x' = (0, 2, 5, 4, 1)$$

Sistem de constrângeri (II)

Lema 4.1

fie $x = (x_1, x_2, ..., x_n)$ o soluție pentru $Ax \leq b$ și d o constantă. Atunci și $x + d = (x_1 + d, x_2 + d, ..., x_n + d)$ este o soluție pentru sistemul de constrângeri $Ax \leq b$ și d.

Demonstrație.

pentru fiecare x_i și x_j avem $(x_j + d) - (x_i + d) = x_j - x_i$. Dacă x satisface $Ax \leq b$ atunci și $x + d$ este o soluție. $\square$

Grafuri de constrângeri

Cum se poate modela problema sub forma unui graf?

Grafuri de constrângeri

- sistemul de constrângeri poate fi interpretat sub forma unui graf
- pentru un sistem $Ax \leq b$ de constrângeri, matricea A de dimensiune $m\,x\,n$ poate fi văzută ca transpusa unei matrici de incidență a unui graf cu n vârfuri și m arce
- fiecare vârf $v_i \in V, i = 1, 2, ..., n$ corespunde unei variabile x_i
- fiecare arc $(i, j) \in E$ corespunde unei inegalități

Definiție

fie un sistem $Ax \leq b$ de constrângeri, graful corespunzător acestui sistem este un graf ponderat și orientat $G = (V, E)$ unde $V = \{\mathbf{v_0}, v_1, ..., v_n\}$ și

$$\begin{aligned} E &= \{(v_i, v_j) \mid x_j - x_i \leq b_k \text{ este o constrângere}\} \\ &\cup\{(v_0, v_1), (v_0, v_2), ..., (v_0, v_n)\} \end{aligned}$$

Grafuri de constrângeri (II)

- graful conține un vârf suplimentar v_0, astfel fiecare vârf e accesibil din v_0
- $v_i \in V, i = 1, ..., n$ pentru fiecare necunoscută x_i și un vârf suplimentar v_0
- E conține un arc pentru fiecare constrângere și (v_0, v_i) pentru fiecare necunoscută x_i
- dacă $(x_j - x_i \leq b_k$ atunci $w(v_i, v_j) = b_k$
- $w(v_0, v_i) = 0, \forall i = 1, ..., n$

Grafuri de constrângeri (III)

Teorema 4.1

fie un sistem $Ax \leq b$ de constrângeri și $G = (V, E)$ graful constrângerilor. Dacă G nu conține circuite de pondere negativă, atunci

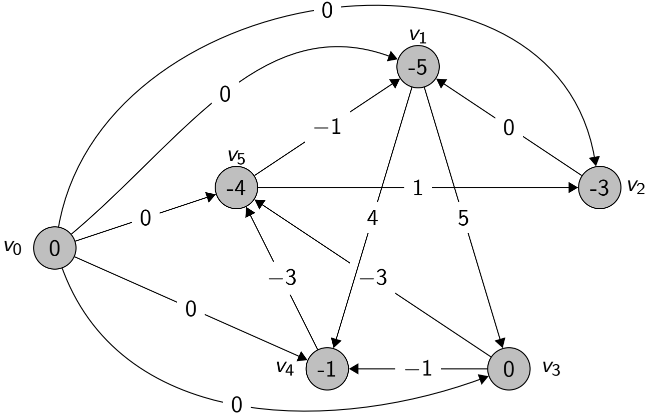
$$x = (\delta(v_0, v_1), \delta(v_0, v_2), ..., \delta(v_0, v_n))$$

este o soluție fezabilă pentru sistem. Dacă graful G conține un circuit negativ, sistemul nu are soluție.

- $\implies$ soluția unui sistem de constrângeri poate fi găsită ca și drumul de pondere minimă din graful constrângerilor

Exemplu

- valoarea $\delta(v_0, v_i)$ apare în fiecare nod
- o posibilă soluție $x = (-5, -3, 0, -1, -4)$



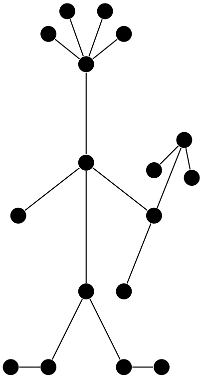
Drum de lungime minimă între toate perechile de vârfuri

Din motive de organizare această parte va fi discutată în cursul 5.

Arbori și păduri

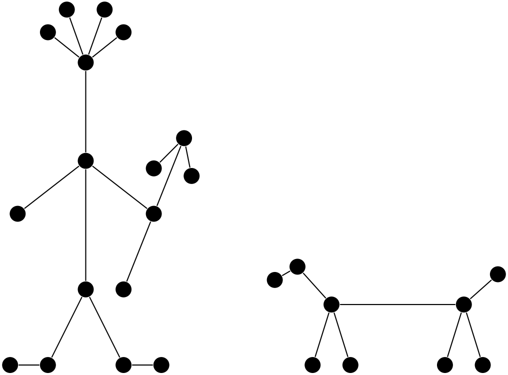
Arbori și păduri

- un arbore



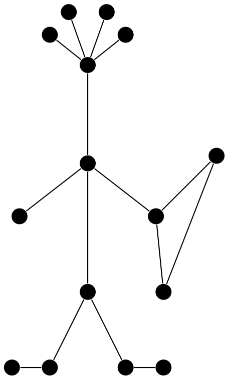
Arbori și păduri (II)

- o pădure



Arbori și păduri

- un graf care nu este arbore sau pădure



Arbori și păduri - definiții

Definiții

Un **arbore** este un graf simplu care nu are cicluri.

O **pădure** este un graf $G = (V, E)$ simplu în care fiecare componentă este un arbore.

Arbori și păduri - definiții (II)

Definiție

un vârf u al unui graf simplu $G = (V, E)$ se numește **frunză** dacă $d_G(u) = 1$. Un vârf care nu este frunză se numește **vârf intern**.

Multe proprietăți asociate arborilor pot fi derivate din următoarea teoremă

Teorema 4.2

fiecare arbore cu minim două vârfuri are cel puțin două frunze.

Arbori și păduri - definiții (III)

Demonstrație.

- fie T un arbore cu $n \geq 2$, fie p lanțul de lungime maximă din T și u, v vârfurile lui p
- se arată că u și v sunt frunze, $d(u) = d(v) = 1$, este suficient să se demonstreze pentru un singur vârf
- dacă $d(u) \geq 2 \Rightarrow \exists\, e \in E, e \notin p$, având vârfurile $u, w \in V$
- avem două cazuri:
  1. $w \notin p \Rightarrow$ lanțul compus $p' = (w, e, u)p$ este un lanț din T având lungimea lanțului p plus *1* $\longrightarrow$ contradicție (p lanțul de lungime maximă)
  2. $w \in p$, dacă p'' este lanțul de la u la v atunci avem un ciclu $c = (w, e, u)p''$ de lungime cel puțin *3* în T $\longrightarrow$ T nu este arbore
- $\Rightarrow d(u) = 1$

□

Arbori și păduri - definiții (IV)

Fie $G = (V, E)$ un graf de ordin $n \geq 2$, afirmațiile următoare sunt echivalente și caracterizează un arbore:

1. G este un arbore
2. G este fără cicluri și are $n - 1$ muchii
3. G este conex și are $n - 1$ muchii
4. G este conex și suprimând o muchie nu mai este conex
5. între oricare două vârfuri ale grafului există un singur lanț
6. G este fără cicluri și prin adăugarea unei muchii între două vârfuri neadiacente se formează un singur ciclu

Arbori și păduri - definiții (V)

Teorema Erdős-Szekeres

dacă $(x_1, x_2, ..., x_{hk+1})$ este o secvență de numere reale distincte, atunci există o subsecvență crescătoare de $h+1$ elemente sau o subsecvență descrescătoare de $k+1$ elemente.

Corolar

fiecare secvență de numere reale distincte de lungime n conține o subsecvență de lungime $\lceil \sqrt{n} \rceil$ strict crescătoare sau strict descrescătoare.

Arbori și păduri - definiții (VI

Centrul unui arbore

fie $G = (V, E)$ un graf și $u \in V$

- excentricitatea $\epsilon_G(u)$ a lui u în G este distanța de la u la vârful cel mai îndepărtat de u din G,

$$\epsilon_G(u) = \max(\delta_G(u, v) | v \in V)$$

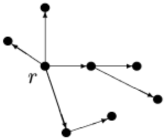
- centrul lui G este vârful pentru care

$$\min_{u \in V}(\epsilon_G(u))$$

Arbori și păduri - definiții (VII)

Rădăcina unui arbore

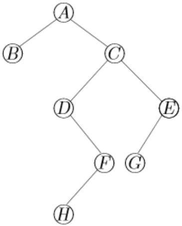
fie T un arbore și $r \in V(T)$. Un arbore cu rădăcină este perechea ordonată (T, r), vârful r se numește **rădăcina** arborelui.



Arbori și păduri - definiții (VIII)

Arbore binar

un **arbore binar** este un arbore ce are o rădăcină, este ordonat și în care fiecare vârf are cel mult doi succesori. Succesorii fiecărui vârf sunt ordonați, fiul stâng și fiul drept.



preorder: A, B, C, D, F, H, E, G

inorder: B, A, D, H, F, C, G, E

postorder: B, H, F, D, G, E, C, A

Arbori de acoperire (spanning trees)

Ex. realizarea unui circuit electronic

- terminalele mai multor componente electronice trebuie interconectate
- pentru a conecta n terminale e nevoie de $n-1$ conexiuni, fiecare conectând două terminale
- dintre toate aranjamentele cel mai dezirabil este cel care folosește cât mai puțin cupru pentru a conecta terminalele

Arbori de acoperire (II)

problema poate fi rezolvată cu ajutorul unui graf

Definire problemă

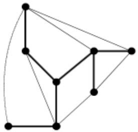
fie un graf $G = (V, E)$ simplu neorientat unde V este setul terminalelor și E este setul conexiunilor posibile între terminalele componentelor. Pentru fiecare muchie $(u, v) \in E$ avem o pondere $w(u, v)$ ce specifică costul legăturii (ex. cantitatea de cupru folosită). Vrem să găsim un subset aciclic $T \subseteq E$ care leagă toate vârfurile având costul total

$$w(t) = \sum_{(u,v) \in T} w(u, v)$$

minim.

Arbori de acoperire (III)

- deoarece T este aciclic și leagă toate vârfurile, T este un arbore numit **arbore de acoperire**
- problema cere determinarea arborelui minim de acoperire



Arbori de acoperire (IV)

Un arbore de acoperire T are următoarele proprietăți

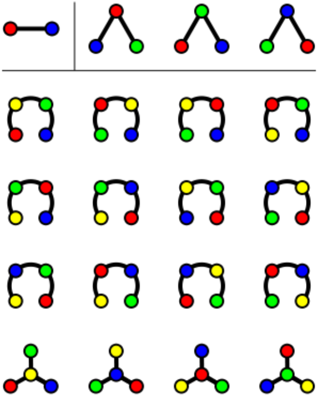
- T este conex
- T este aciclic
- T are n vârfuri
- T are $n-1$ muchii

Dacă un subgraf T al unui graf $G = (V, E)$ are oricare trei astfel de proprietăți atunci T este un arbore de acoperire.

Arbori de acoperire - formula lui Cayley

Cayley

fie un graf complet K_n, numărul arborilor etichetați este n^{n-2}



Arbore de acoperire minimă - metoda generică

Fie un graf simplu neorientat $G = (V, E)$ cu funcția de pondere $w : E \to \mathbb{R}$ și vrem să găsim arborele minim de acoperire a lui G.

- generic, abordarea folosită este surprinsă de procedura

generic_mst(G)

$A = \varnothing$
while A nu este un arbore minim de acoperire **do**
    găsește o muchie (u, v) **sigură** pentru A
    $A = A \cup \{(u, v)\}$
return A

- arborele minim de acoperire crește muchie cu muchie

Arbore de acoperire minimă - metoda generică (II)

- înainte de fiecare iterație A este un subset al unui arbore minim de acoperire
- în fiecare pas se găsește o muchie care împreună cu A formează un subset al unui arbore minim de acoperire (muchie *sigură*)
- **partea dificilă**: găsirea muchiei (u, v) astfel încât $A \subseteq T$
- o tăietură $(S, V - S)$ a unui graf neorientat $G = (V, E)$ este o partiție a lui V

Arbore de acoperire minimă - metoda generică (III)

Teorema

fie $G = (V, E)$ un graf simplu neorientat ponderat cu funcția de pondere $w : E \to \mathbb{R}$. Fie A un subset al lui E inclus într-un arbore minim de acoperire al lui G, fie $(S, V - S)$ o tăietură a lui G ce respectă A și (u, v) muchia de pondere minimă ce traversează tăietura $(S, V - S)$. În acest caz, muchia (u, v) este sigură pentru A.

Corolar

$G = (V, E)$ un graf simplu neorientat ponderat cu funcția de pondere $w : E \to \mathbb{R}$. Fie A un subset al lui E inclus într-un arbore minim de acoperire al lui G, fie $C = (V_C, E_C)$ o componentă conexă (arbore) în pădurea $G_A = (V, A)$. Dacă (u, v) este o muchie de pondere minimă ce leagă componenta C de o altă componentă din G_A, atunci (u, v) este sigură pentru A.

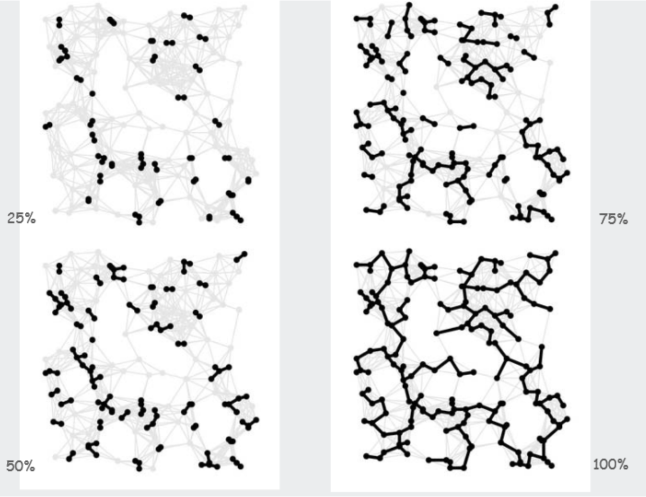
Algoritmul lui Kruskal

mst_kruskal(G,w)

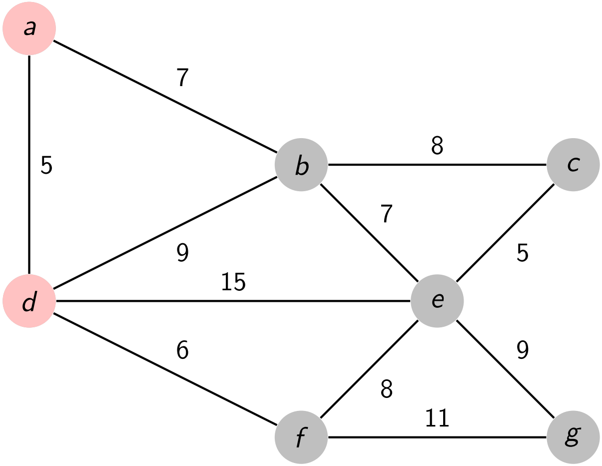
$A = \varnothing$
for $v \in V$ **do**
    make_set(v)
sortare muchii crescător după ponderea w
for $(u, v) \in E$ luate crescător după w **do**
    if find_set(u) $\neq$ find_set(v) **then**
        $A = A \cup (u, v)$
        union(u,v)
return A

- implementarea folosește o structură de date de tipul *disjoint-set (union–find, merge-find)*

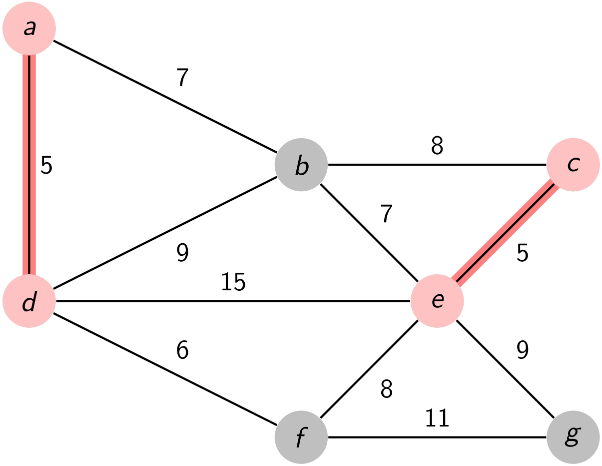
Algoritmul lui Kruskal - exemplu



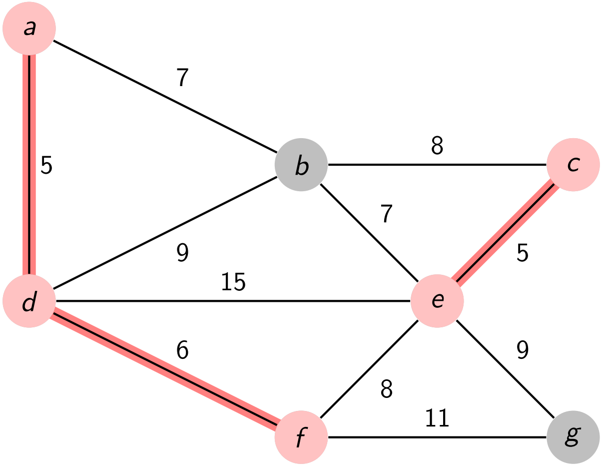
Algoritmul lui Kruskal - exemplu (II)



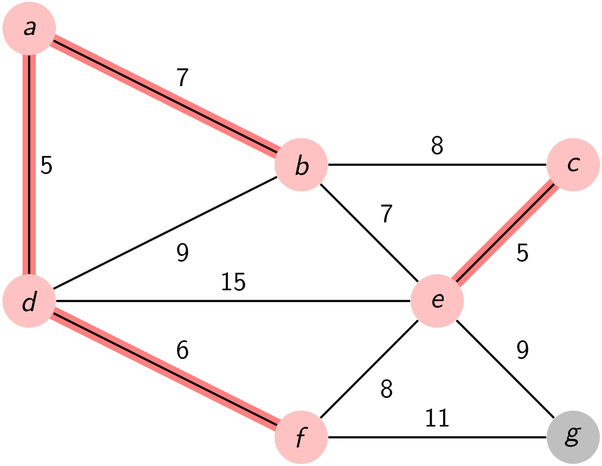
Algoritmul lui Kruskal - exemplu (II)



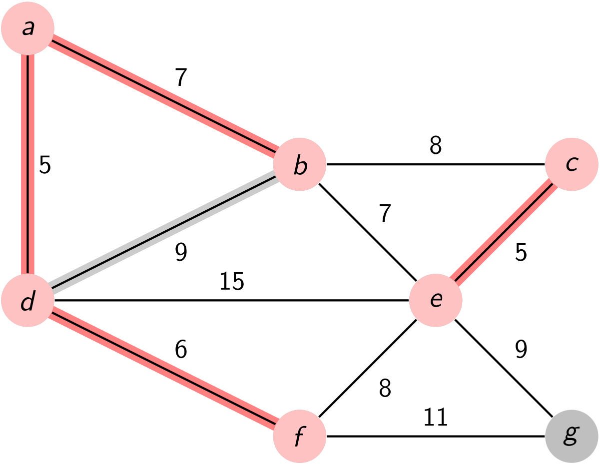
Algoritmul lui Kruskal - exemplu (II)



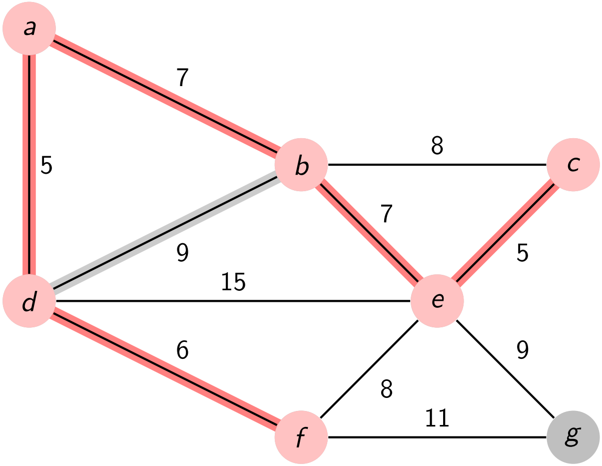
Algoritmul lui Kruskal - exemplu (II)



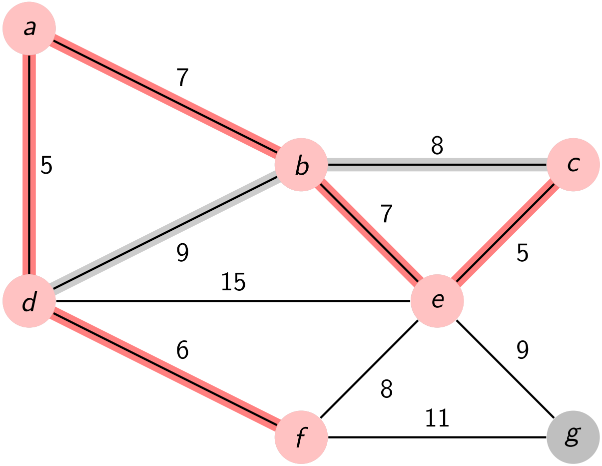
Algoritmul lui Kruskal - exemplu (II)



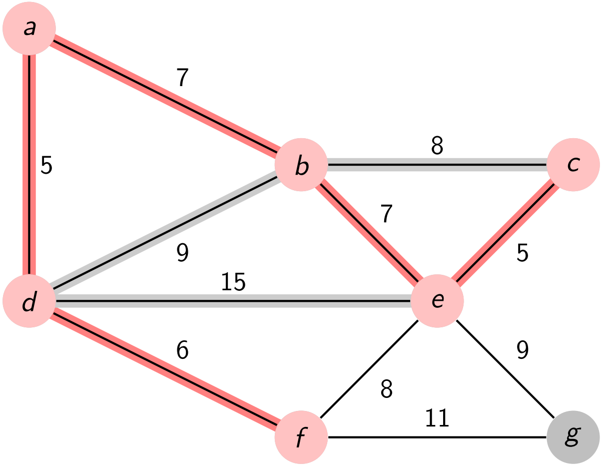
Algoritmul lui Kruskal - exemplu (II)



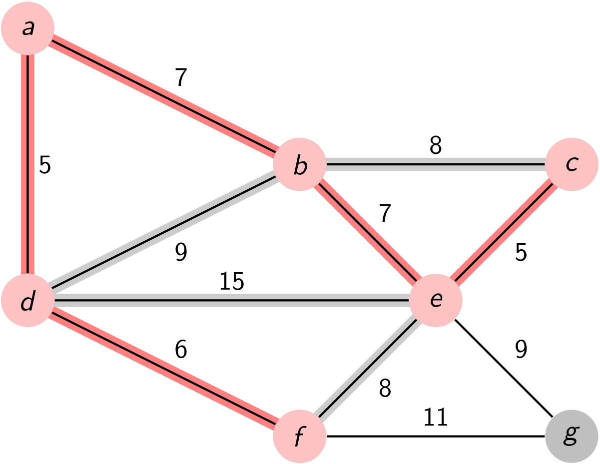
Algoritmul lui Kruskal - exemplu (II)



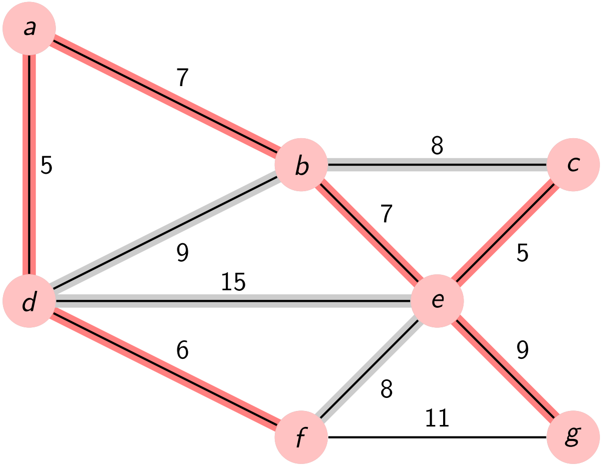
Algoritmul lui Kruskal - exemplu (II)



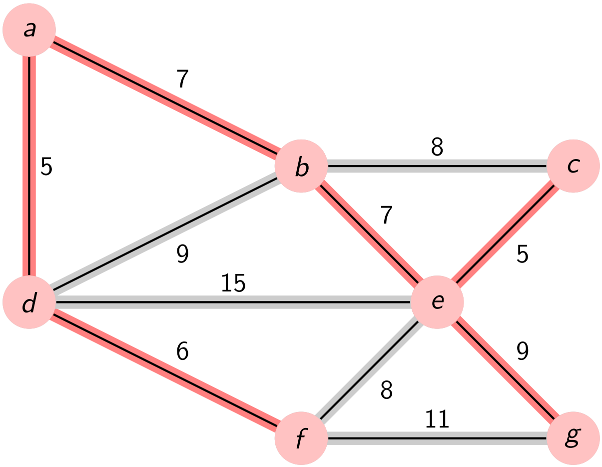
Algoritmul lui Kruskal - exemplu (II)



Algoritmul lui Kruskal - exemplu (II)



Algoritmul lui Kruskal - exemplu (II)

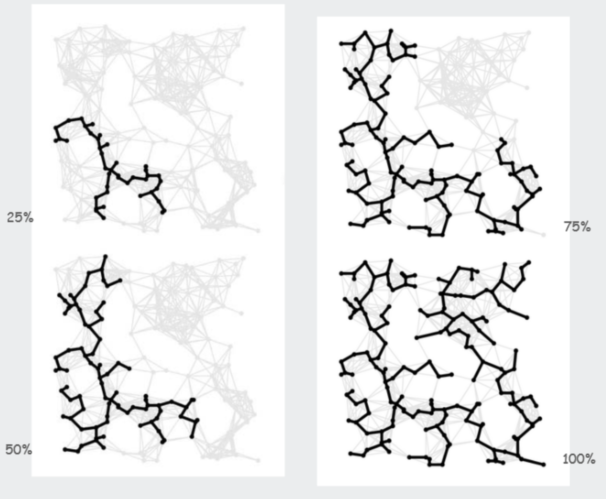


Algoritmul lui Prim

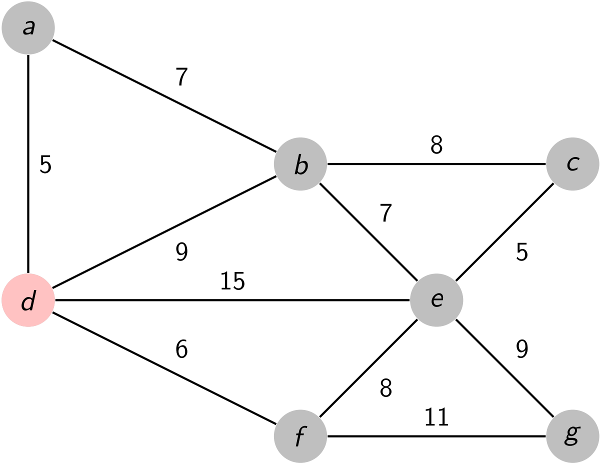
mst_prin(G,w,r)

for $u \in V$ **do**
    $u.key = \infty$
    $u.\pi = NIL$
$r.key = 0$
$Q = V$
while $Q \neq \emptyset$ **do**
    u = extract_min(Q)
    for $v \in Adj[u]$ **do**
        if $v \in Q$ și $w(u, v) < v.key$ **then**
            $v.\pi = u$
            $v.key = w(u, v)$

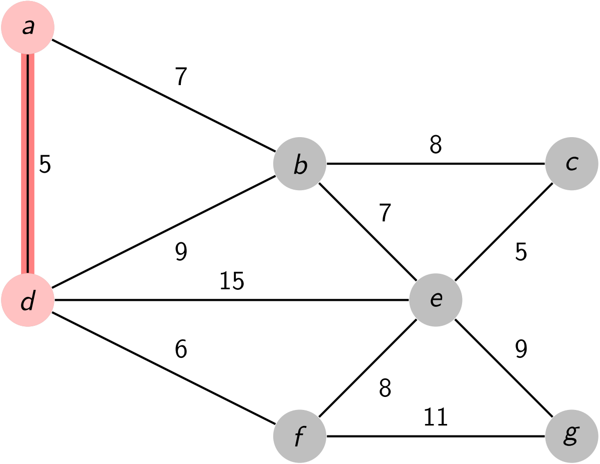
Algoritmul lui Prim - exemplu



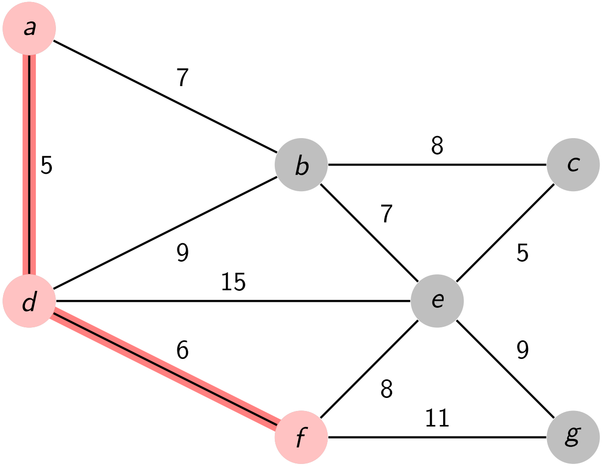
Algoritmul lui Prim - exemplu (II)



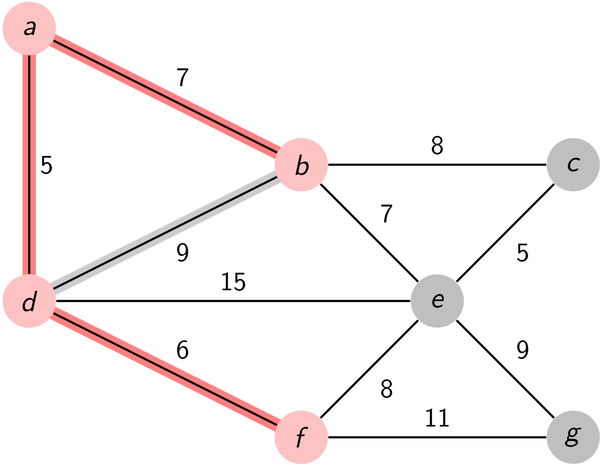
Algoritmul lui Prim - exemplu (II)



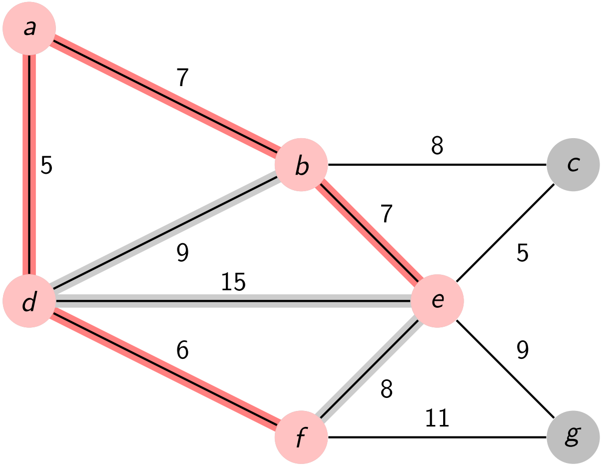
Algoritmul lui Prim - exemplu (II)



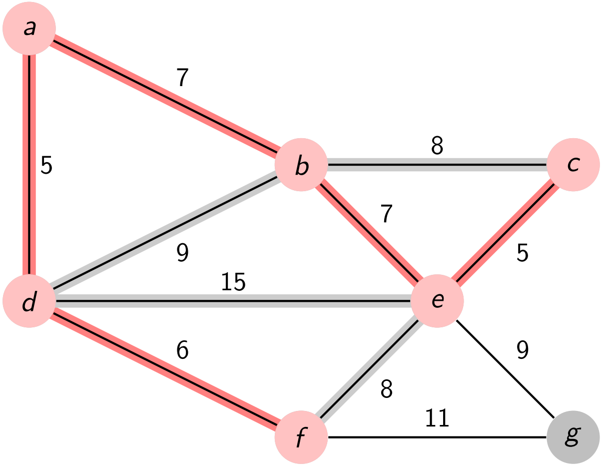
Algoritmul lui Prim - exemplu (II)



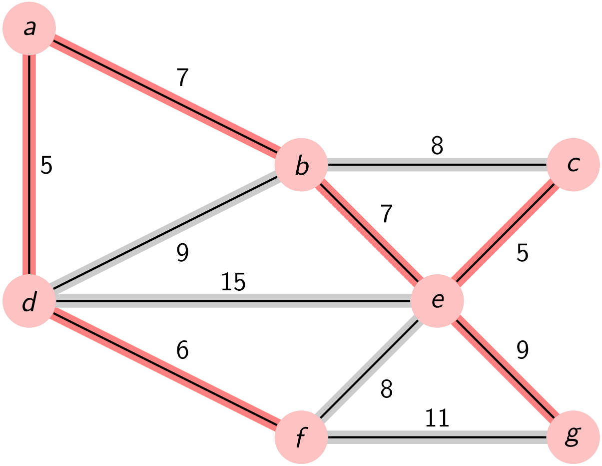
Algoritmul lui Prim - exemplu (II)



Algoritmul lui Prim - exemplu (II)



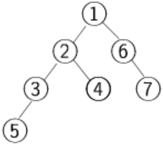
Algoritmul lui Prim - exemplu (II)



Codare Prüfer

```
CODARE_PRUFER(F)
    K = ∅
    while T conține și alte vârfuri decât rădăcina do
            fie v frunza minimă din T
            K ← predecesor(v)
            T = T \ {v}
    return K
```

exemplu:



2,3,2,1,6,1

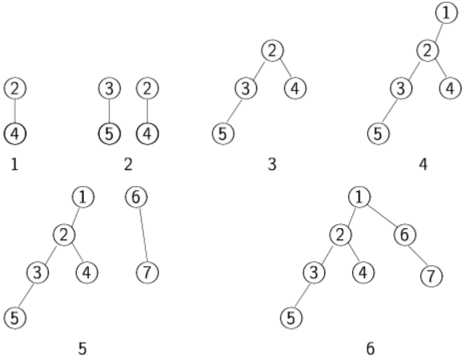
Decodare Prüfer

$\text{DECODARE\_PRUFER}(K, n)$

1. $T = \varnothing$
2. **for** $i = 1, 2, \ldots, n-1$ **do**
3. $\quad$ x primul element din K
4. $\quad$ y cel mai mic număr natural care nu se găsește în K
5. $\quad$ $(x, y) \in E(T)$, x părintele lui y în T
6. $\quad$ șterg x din K, adaugă y în K
7. **return** T

Decodare Prüfer - exemplu

$2, 3, 2, 1, 6, 1 \mid\mid 4$
$3, 2, 1, 6, 1, 4 \mid\mid 5$
$2, 1, 6, 1, 4, 5 \mid\mid 3$
$1, 6, 1, 4, 5, 3 \mid\mid 2$
$6, 1, 4, 5, 3, 2 \mid\mid 7$
$1, 4, 5, 3, 2, 7 \mid\mid 6$
$4, 5, 3, 2, 7, 6$



Codare Huffman

HUFFMAN(C)

$n = |C|$
$Q = C$
for $1 \leq i \leq n-1$ **do**
    alocă un nou vârf z
    $z.stang = x =$EXTRACT_MIN(Q)
    $z.drept = y =$EXTRACT_MIN(Q)
    $z.fr = x.fr + y.fr$
    INSERT(Q, z)
return EXTRACT_MIN(Q)