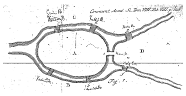
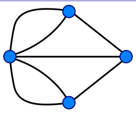




Algoritmica grafurilor
V. Arbori si paduri

Mihai Suciu

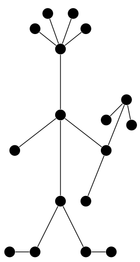
Facultatea de Matematică și Informatică (UBB)
Departamentul de Informatică

Martie, 30, 2023

Continut

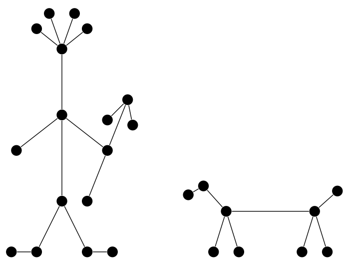
Arbori și păduri

- un arbore



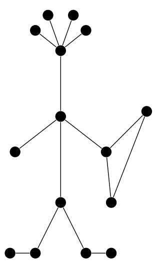
Arbori și păduri (II)

- 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 $(x_1, x_2, \ldots, x_{hk+1})$ 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,

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

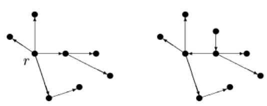
- centrul lui G este vârful pentru care

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

Arbori și păduri - definiții (VII)

Rădăcina unui arbore

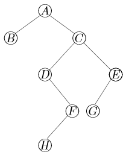
fie T un arbore și $r \in V(T)$. Un arbore cu rădăcină este perechea ordonată (T, r), vârful r se numește **rădăcina** arborelui.



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

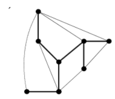
fie un graf $G = (V, E)$ simplu neorientat unde V este setul terminalelor și E este setul conexiunilor posibile între terminalele componentelor. Pentru fiecare muchie $(u, v) \in E$ avem o pondere $w(u, v)$ ce specifică costul legăturii (ex. cantitatea de cupru folosită). Vrem să găsim un subset aciclic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

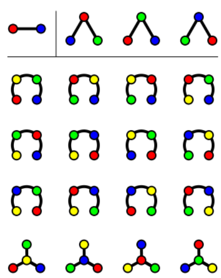
- T este conex
- T este aciclic
- T are n vârfuri
- T are $n-1$ muchii

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 = ∅
while A nu este un arbore minim de acoperire do
    găsește o muchie (u, v) sigură pentru A
    A = A ∪ {(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.

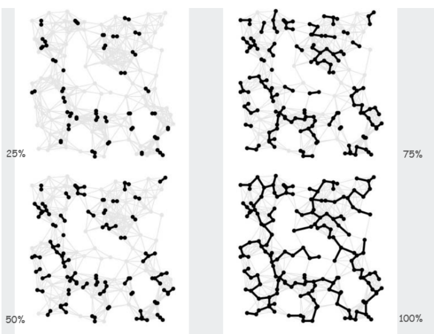
Algoritmul lui Kruskal

mst_kruskal(G,w)

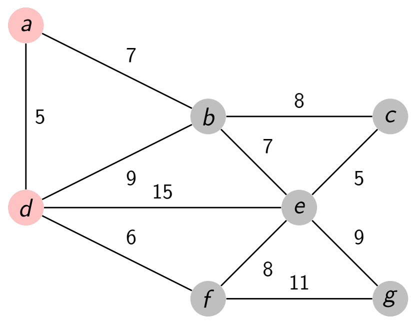
```
A = ∅
for v ∈ V do
    make_set(v)
sortare muchii crescător după ponderea w
for (u, v) ∈ E luate crescător după w do
    if find_set(u) ≠ find_set(v) then
        A = A ∪ (u, v)
        union(u,v)
return A
```

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

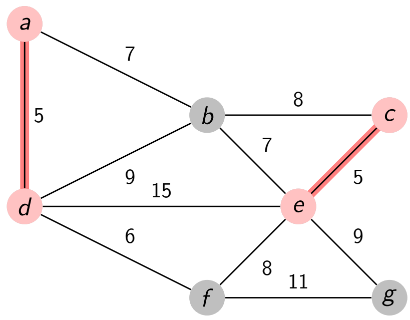
Algoritmul lui Kruskal - exemplu



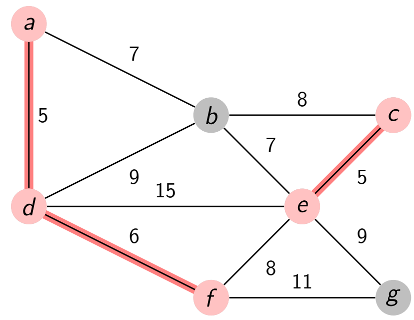
Algoritmul lui Kruskal - exemplu (II)



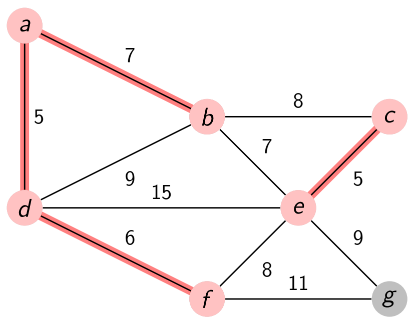
Algoritmul lui Kruskal - exemplu (II)



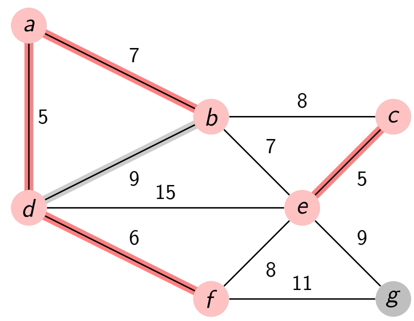
Algoritmul lui Kruskal - exemplu (II)



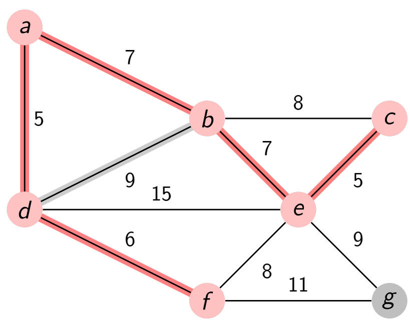
Algoritmul lui Kruskal - exemplu (II)



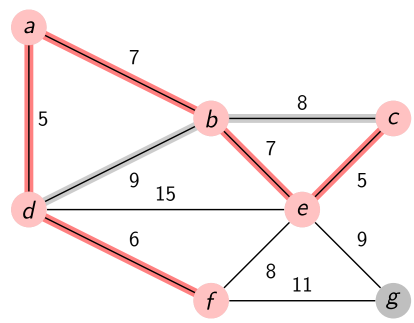
Algoritmul lui Kruskal - exemplu (II)



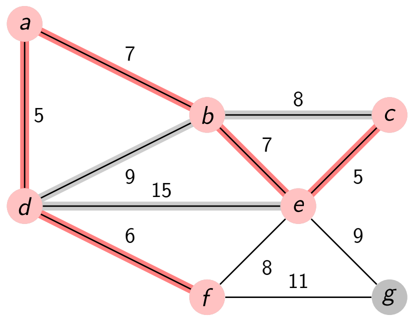
Algoritmul lui Kruskal - exemplu (II)



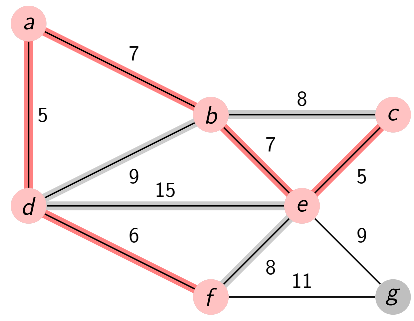
Algoritmul lui Kruskal - exemplu (II)



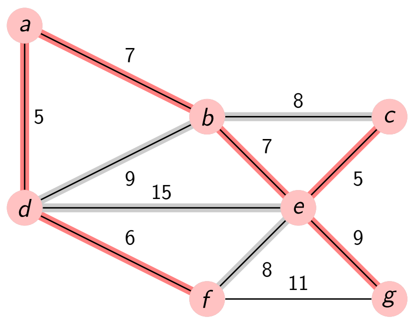
Algoritmul lui Kruskal - exemplu (II)



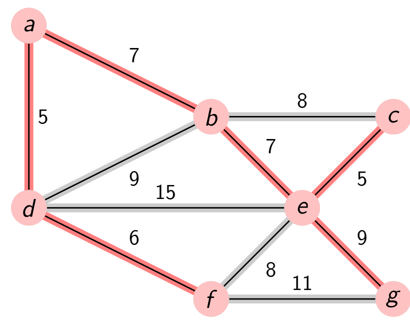
Algoritmul lui Kruskal - exemplu (II)



Algoritmul lui Kruskal - exemplu (II)



Algoritmul lui Kruskal - exemplu (II)

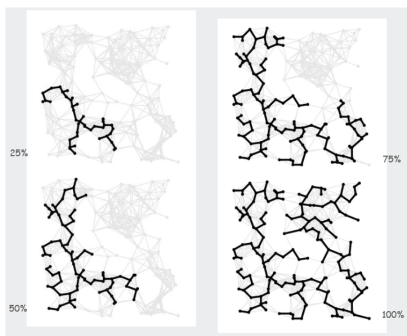


Algoritmul lui Prim

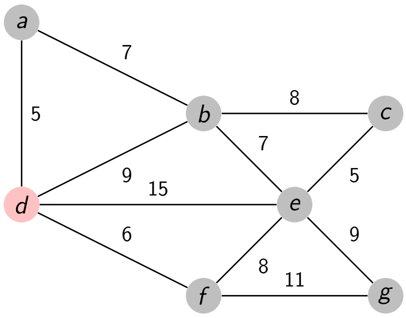
mst_prin(G,w,r)

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

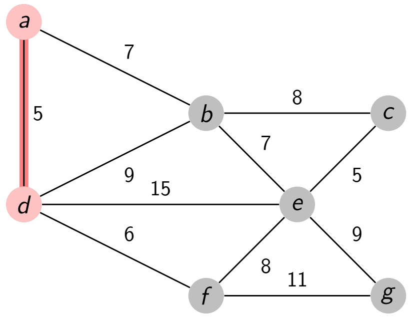
Algoritmul lui Prim - exemplu



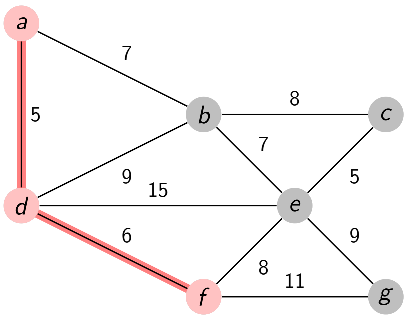
Algoritmul lui Prim - exemplu (II)



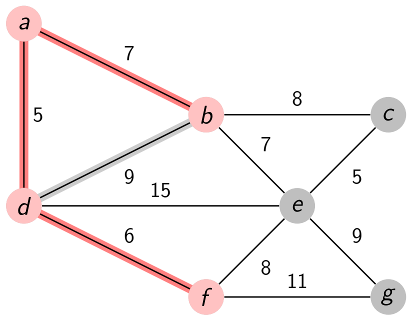
Algoritmul lui Prim - exemplu (II)



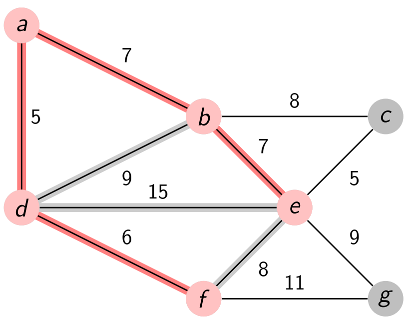
Algoritmul lui Prim - exemplu (II)



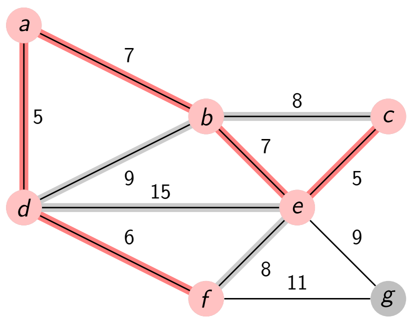
Algoritmul lui Prim - exemplu (II)



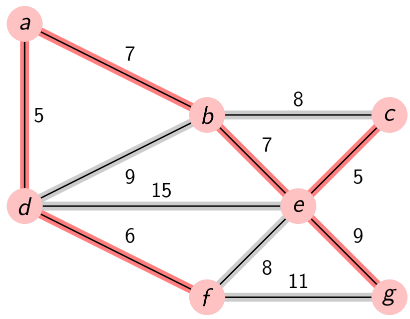
Algoritmul lui Prim - exemplu (II)



Algoritmul lui Prim - exemplu (II)



Algoritmul lui Prim - exemplu (II)



Codare Prüfer

CODARE_PRUFER(F)

$K = \varnothing$

while T conține și alte vârfuri decât rădăcina **do**

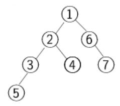
  fie v frunza minimă din T

  $K \leftarrow$ predecesor(v)

  $T = T \setminus \{v\}$

return K

exemplu:



2,3,2,1,6,1

Decodare Prüfer

DECODARE_PRUFER(K, n)

$T = \varnothing$

for $i = 1, 2, \ldots, n-1$ **do**

  x primul element din K

  y cel mai mic număr natural care nu se găsește în K

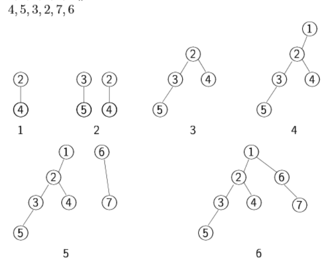
  $(x, y) \in E(T)$, x părintele lui y în T

  șterg x din K, adaugă y în K

return T

Decodare Prüfer - exemplu

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



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)