

Algoritmica grafurilor

V. Arbori si paduri

Mihai Suci

Facultatea de Matematică și Informatică (UBB)
Departamentul de Informatică

Martie, 25, 2024

1 / 47

Continut



- 1 Arbori si paduri
 - Definitii
 - Arbori de acoperire
 - Algoritmul lui Kruskal
 - algoritmul lui Prim
 - Prufer
 - codare Huffman

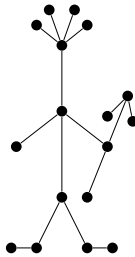
2 / 47

Arbori si paduri

Arbori și păduri



- un arbore



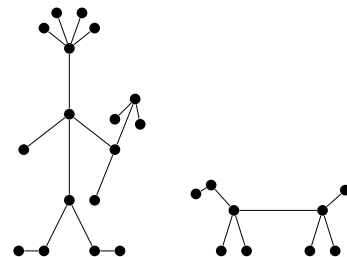
3 / 47

Arbori si paduri

Arbori și păduri (II)



- o pădure



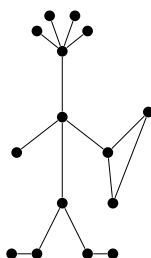
4 / 47

Arbori si paduri

Arbori și păduri



- un graf care nu este arbore sau pădure



5 / 47

Arbori si paduri Definitii

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.

6 / 47



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.

7 / 47



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 \rightarrow 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 $T \rightarrow T$ 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

9 / 47



Arbori și păduri - definiții (V)

Teorema Erdős-Szekeres

dacă $(x_1, x_2, \dots, x_{h+k+1})$ 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.

10 / 47



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

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

11 / 47



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.



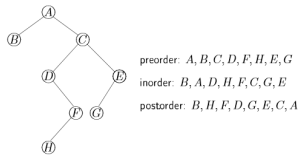
12 / 47

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.



13 / 47

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

14 / 47

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.

15 / 47

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



16 / 47

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.

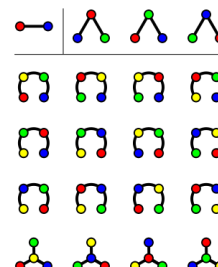
17 / 47

Arbori de acoperire - formula lui Cayley



Cayley

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



18 / 47

Arbore de acoperire minimă - metoda generică



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

- generic, abordarea folosită este surprinsă de procedura

generic_mst(G)

- 1: $A = \emptyset$
 - 2: **while** A nu este un arbore minim de acoperire **do**
 - 3: găsește o muchie (u, v) **sigură** pentru A
 - 4: $A = A \cup \{(u, v)\}$
 - 5: **return** A
- arborele minim de acoperire crește muchie cu muchie

19 / 47

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

20 / 47

Arbore de acoperire minimă - metoda generică (III)



Teorema

fie $G = (V, E)$ un graf simplu neorientat ponderat cu funcția de pondere $w : E \rightarrow \mathbb{R}$. 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 $w : E \rightarrow \mathbb{R}$. 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 .

21 / 47

Algoritmul lui Kruskal

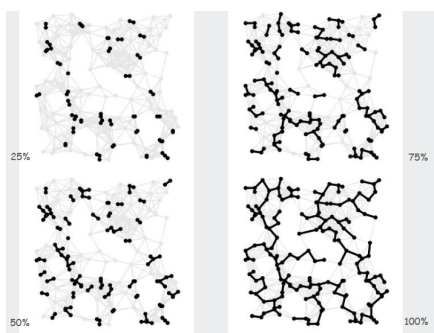


mst_kruskal(G, w)

- 1: $A = \emptyset$
 - 2: **for** $v \in V$ **do**
 - 3: make_set(v)
 - 4: sortare muchii crescător după ponderea w
 - 5: **for** $(u, v) \in E$ luate crescător după w **do**
 - 6: **if** find_set(u) \neq find_set(v) **then**
 - 7: $A = A \cup (u, v)$
 - 8: union(u, v)
 - 9: **return** A
- implementarea folosește o structură de date de tipul *disjoint-set* (union-find, merge-find)

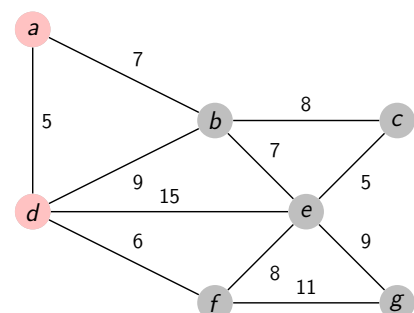
22 / 47

Algoritmul lui Kruskal - exemplu



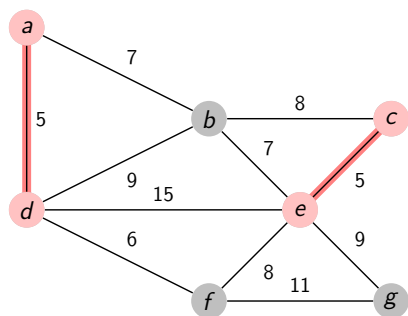
23 / 47

Algoritmul lui Kruskal - exemplu (II)





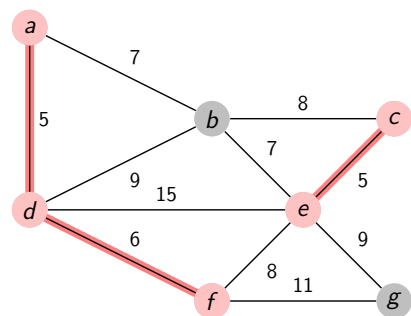
Algoritmul lui Kruskal - exemplu (II)



25 / 47



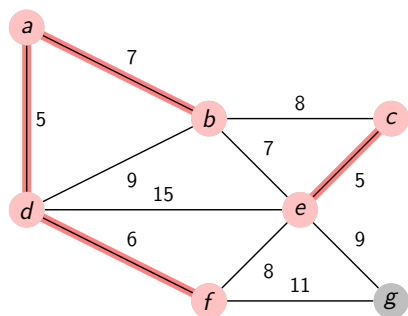
Algoritmul lui Kruskal - exemplu (II)



26 / 47



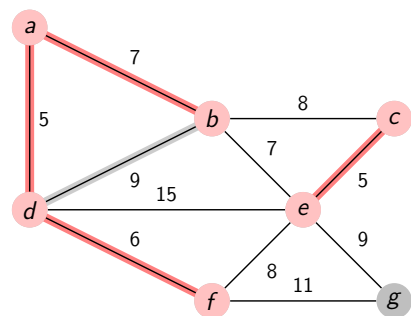
Algoritmul lui Kruskal - exemplu (II)



27 / 47



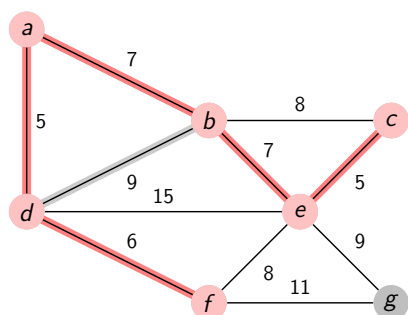
Algoritmul lui Kruskal - exemplu (II)



28 / 47



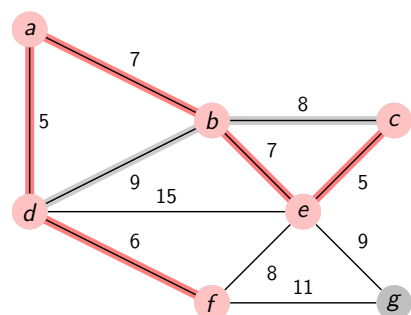
Algoritmul lui Kruskal - exemplu (II)



29 / 47

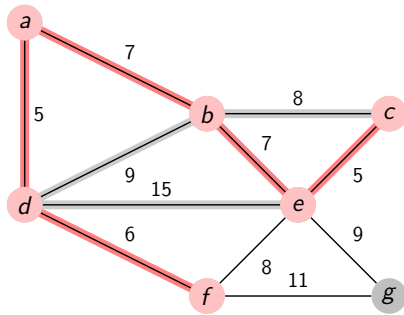


Algoritmul lui Kruskal - exemplu (II)



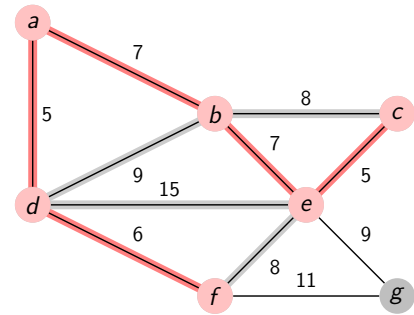
30 / 47

Algoritmul lui Kruskal - exemplu (II)



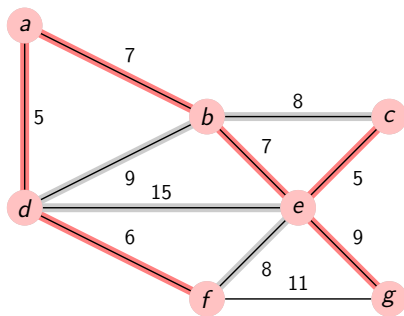
31 / 47

Algoritmul lui Kruskal - exemplu (II)



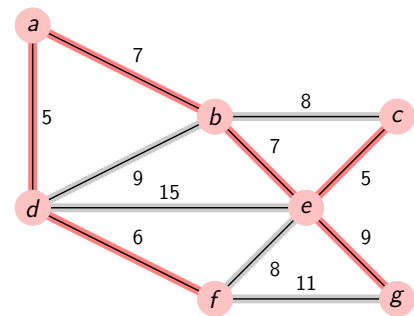
32 / 47

Algoritmul lui Kruskal - exemplu (II)



33 / 47

Algoritmul lui Kruskal - exemplu (II)



34 / 47

Algoritmul lui Prim



mst_prin(G,w,r)

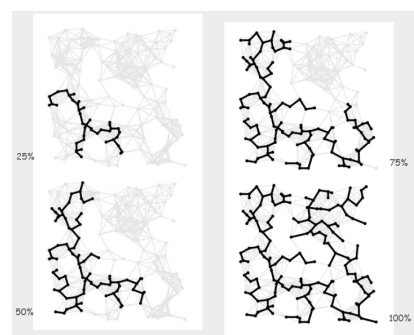
```

1: for  $u \in V$  do
2:    $u.key = \infty$ 
3:    $u.\pi = NIL$ 
4:  $r.key = 0$ 
5:  $Q = V$ 
6: while  $Q \neq \emptyset$  do
7:    $u = \text{extract\_min}(Q)$ 
8:   for  $v \in \text{Adj}[u]$  do
9:     if  $v \in Q$  și  $w(u, v) < v.key$  then
10:       $v.\pi = u$ 
11:       $v.key = w(u, v)$ 

```

35 / 47

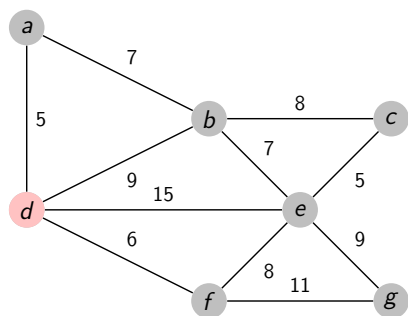
Algoritmul lui Prim - exemplu



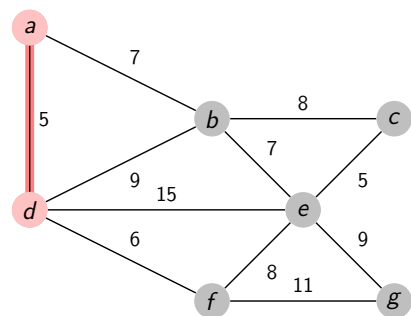
36 / 47



Algoritmul lui Prim - exemplu (II)



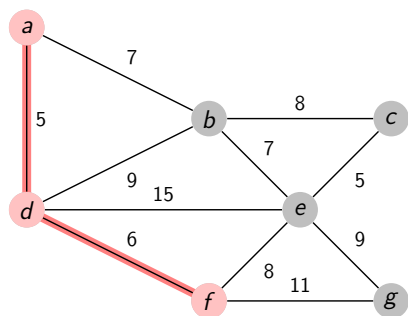
Algoritmul lui Prim - exemplu (II)



38 / 47



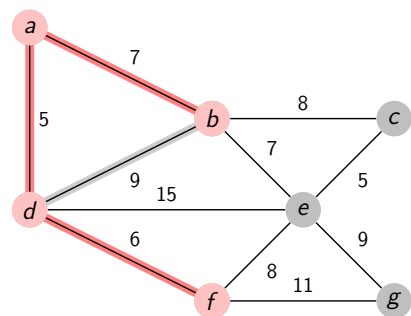
Algoritmul lui Prim - exemplu (II)



39 / 47



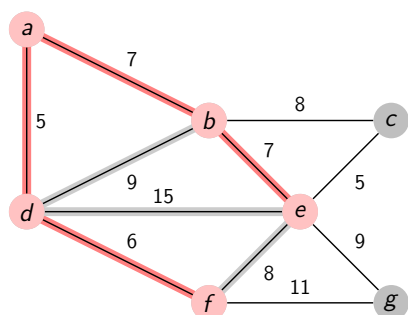
Algoritmul lui Prim - exemplu (II)



40 / 47



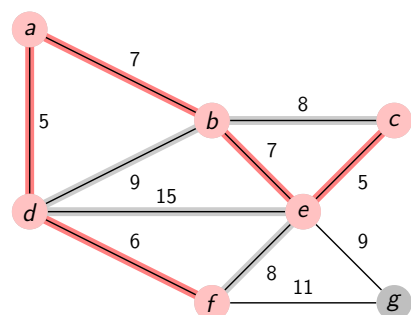
Algoritmul lui Prim - exemplu (II)



41 / 47

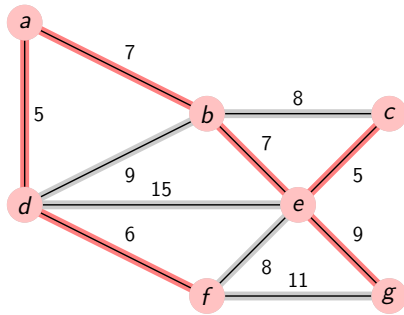


Algoritmul lui Prim - exemplu (II)



42 / 47

Algoritmul lui Prim - exemplu (II)



43 / 47

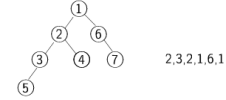
Codare Prüfer



CODARE_PRUFER(F)

1. $K = \emptyset$
2. **while** T conține și alte vârfuri decât rădăcina **do**
3. fie v frunza minimă din T
4. $K \leftarrow \text{predecesor}(v)$
5. $T = T \setminus \{v\}$
6. **return** K

exemplu:



44 / 47

Decodare Prüfer



DECODARE_PRUFER(K, n)

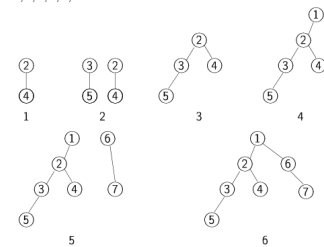
1. $T = \emptyset$
2. **for** $i = 1, 2, \dots, n-1$ **do**
3. x primul element din K
4. y cel mai mic număr natural care nu se găsește în K
5. $(x, y) \in E(T)$, x părintele lui y în T
6. șterg x din K , adaugă y în K
7. **return** T

45 / 47

Decodare Prüfer - exemplu



2, 3, 2, 1, 6, 1 || 4
3, 2, 1, 6, 1, 4 || 5
2, 1, 6, 1, 4, 5 || 3
1, 6, 1, 4, 5, 3 || 2
6, 1, 4, 5, 3, 2 || 7
1, 4, 5, 3, 2, 7 || 6
4, 5, 3, 2, 7, 6



46 / 47

Codare Huffman



HUFFMAN(C)

- 1: $n = |C|$
- 2: $Q = C$
- 3: **for** $1 \leq i \leq n-1$ **do**
- 4: alocă un nou vârf z
- 5: $z.stang = x = \text{EXTRACT_MIN}(Q)$
- 6: $z.drept = y = \text{EXTRACT_MIN}(Q)$
- 7: $z.fr = x.fr + y.fr$
- 8: INSERT(Q, z)
- 9: **return** EXTRACT_MIN(Q)

47 / 47