Exerciții

Arbori. Parcurgeri

Exerciții

1. Se dă un graf neorientat conex G = (V, E). Se consideră parcurgerile DFS și BFS care pornesc din nodul 1, iar vecinii unui nod sunt considerați în ordine crescătoare.

Ce proprietăți trebuie să respecte G pentru ca cele două parcurgeri să obțină același arbore parțial?

Exerciții

2. Fie G_1 si G_2 două grafuri cu proprietatea că ordinea în care sunt parcurse vârfurile în BF este aceeași pentru ambele grafuri, la fel și în DF (vecinii unui vârf sunt parcurși în ordine crescătoare).

Sunt G_1 și G_2 egale?

Exerciții

3. Fie G_1 si G_2 două grafuri cu proprietatea că arborii BF și DF ai celor două grafuri sunt egali (vecinii unui vârf sunt parcurși în ordine crescătoare).

Sunt G_1 și G_2 egale?

Exerciții

4. Se dă un graf neorientat conex $G = (V, E)$ și un vârf s. Care este numărul minim de muchii ale unui graf parțial al lui G care conservă distanțele de la s la celelalte vârfuri

Exerciții

5. Fie T un graf neorientat cu n vârfuri.

Arătați că T este arbore $\Leftrightarrow$ T este conex și are n−1 muchii

Exerciții

6. Fie G un graf conex neorientat. Care dintre următoarele afirmații sunt adevărate? Justificați.

a) Extremitățile unei muchii critice în G sunt puncte critice

b) Un punct critic în G este extremitate a unei muchii critice

Arbori parțiali de cost minim

Arbori parțiali de cost minim

Kruskal

- Inițial T= (V; ∅)
- pentru i = 1, n-1
  - alege o muchie uv cu **cost minim** a.î. u,v sunt în **componente conexe diferite (T+uv aciclic)**
  - $E(T) = E(T) \cup uv$

Prim

- s- vârful de start
- Inițial T= ({s}; ∅)
- pentru i = 1, n-1
  - alege o muchie uv cu **cost minim** a.î. $u \in V(T)$ și $v \notin V(T)$
  - $V(T) = V(T) \cup \{v\}$
  - $E(T) = E(T) \cup uv$

Kruskal

```
sorteaza(E)
for(v=1;v<=n;v++)
     Initializare(v);
nrmsel=0
for(uv ∈ E)
 if(Reprez(u)!=Reprez(v))
  {
      E(T) = E(T) ∪ {uv};
      Reuneste(u,v);
      nrmsel=nrmsel+1;
      if(nrmsel==n-1)
           STOP;
   }
```

Kruskal

Complexitate – dacă folosim arbori

- **Sortare** -> O(m log m) = O(m log n)
- **n * Initializare** -> O(n)
- **2m * Reprez** -> O(m log n)
- **(n-1) * Reuneste** -> O(n log n)

O(m log n)

Prim(G, w, s)

```
pentru fiecare u∈V executa
      d[u] = ∞; tata[u]=0
 d[s] = 0
 inițializează Q cu V
 cat timp Q ≠ ∅ executa
       u=extrage vârf cu eticheta d minimă din Q
       pentru fiecare v adiacent cu u executa
             daca v∈Q si w(u,v)<d[v] atunci
                 d[v] = w(u,v)
                 tata[v] = u
                 //actualizeaza Q - pentru Q heap
 scrie (u, tata[u]), pentru u≠ s
```

Prim

Complexitate

Varianta 1 – cu vector de vizitat

- **Inițializări** -> $O(n)$
- **n * extragere vârf minim** -> $O(n^2)$
- **actualizare etichete vecini** -> $O(m)$

$O(n^2)$

Prim

Varianta 2 – memorarea vârfurilor din într-un min-heap Q (min-ansamblu)

- **Inițializare Q** -> O(n)
- **n * extragere vârf minim** -> O(n log n)
- **actualizare etichete vecini** -> O(m log n)

O(m log n)

Algoritmi bazaţi pe eliminare de muchii

Care dintre următorii algoritmi determină corect un arbore parţial de cost minim (justificaţi)?

1.
```
T ← G
cât timp T conţine cicluri execută
      alege e o muchie de cost maxim care este
      conţinută într-un ciclu din T
      T ← T - e
```

2.
```
T ← G
cât timp T conţine cicluri execută
      alege C un ciclu oarecare din T şi fie e
      muchia de cost maxim din C
      T ← T - e
```

Arbori parțiali de cost minim

- Fie G un graf conex ponderat, C un ciclu in G și e o muchie de cost maxim în C.

Arătați că există un apcm al lui G care nu conține e

(=> corectitudine algoritmi pentru apcm prin eliminare de muchii, muchiile din G-e $\supseteq$ apcm)

Algoritmul Reverse-Delete

```
sorteaza_descrescator(E)
T = G
for(uv ∈ E)
     if T - uv este conex
     (⇔ if uv este continuta intr-un ciclu in T)
         T = T - uv
     if(|E(T)|==n-1)
         STOP;
```

Second best

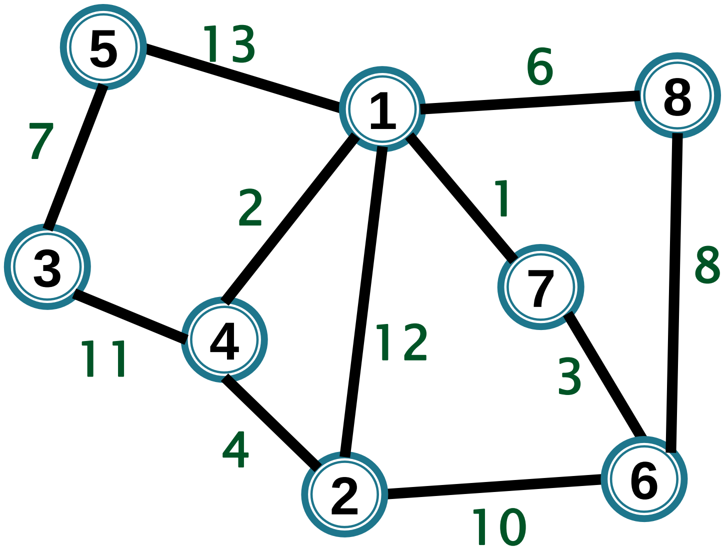
Fie G=(V,E,w) cu ponderi distincte

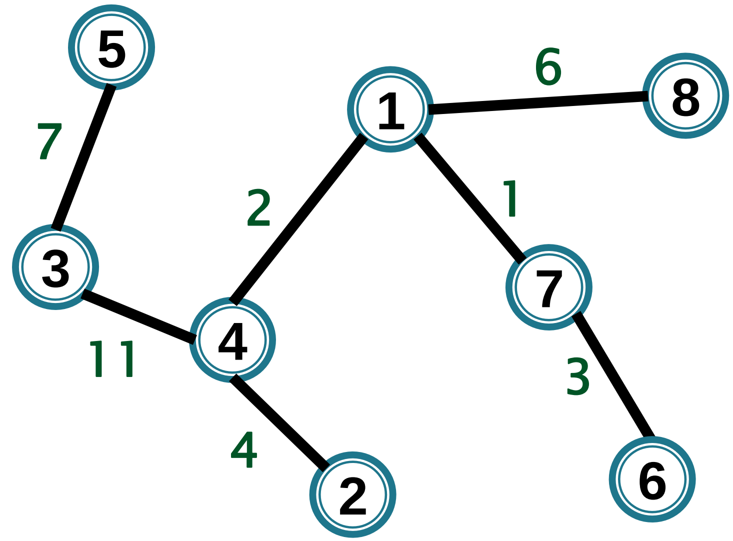
Atunci există un unic apcm T_{min} al lui G

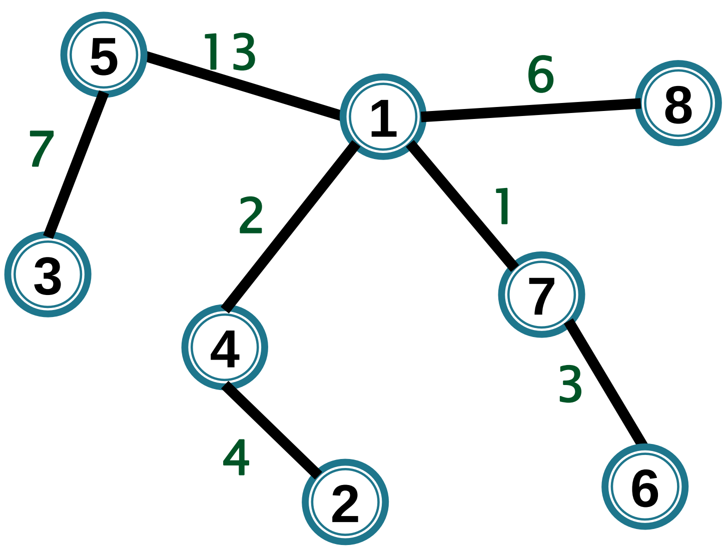
Second best apcm – al doilea apcm = arbore parțial T_s cu

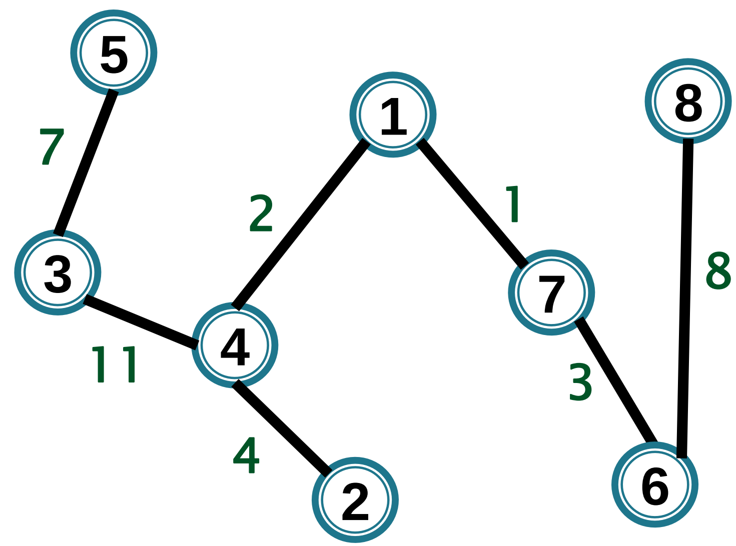
$$w(T_s) = \min\{w(T) \mid T \text{ arbore parțial în G diferit de } T_{min}\}$$

Second best – nu este neapărat unic


5
13
6
1
8
7
2
1
3
8
4
12
7
11
3
4
2
6
10


5
6
1
8
7
2
1
3
4
7
11
3
4
2
6


5
13
6
1
8
7
2
1
3
7
4
3
4
2
6


5
1
8
7
2
1
8
3
7
11
4
3
4
2
6

Second best

Cum se poate obtine second best din apcm?

Second best

Cum se poate obtine second best din apcm?

Idee:

second = apcm în care se schimbă doar o muchie

Second best

Propoziție Fie $G=(V,E,w)$ conex cu ponderi distincte

Fie T_{min} unicul apcm al lui G

Fie T_s un arbore second best

Atunci există $uv \in T_{min}$ și $xy \notin T_{min}$ astfel încât

$$T_s = T_{min} - uv + xy$$

Demonstrație – Tema

Second best

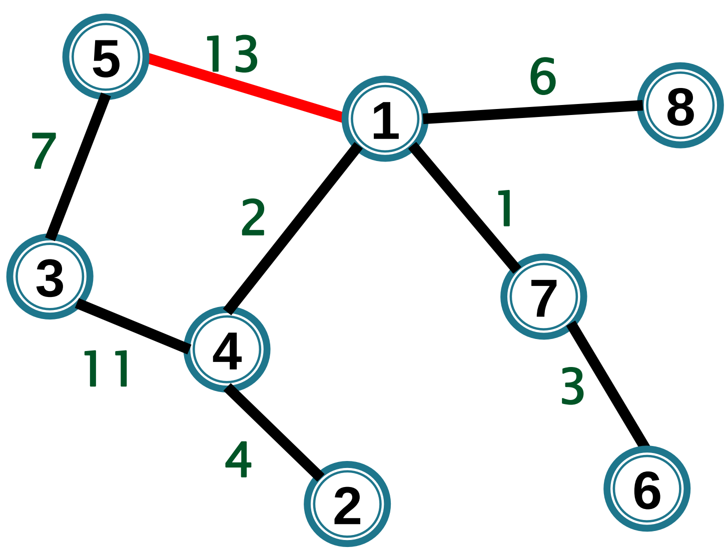
Idee algoritm:

Fie T_{min} apcm

Cum derminăm $uv \in T_{min}$ și $xy \notin T_{min}$ a.î

$T_{min} - uv + xy$ să fie arbore și

$w(xy)-w(uv)$ minim cu această proprietate?



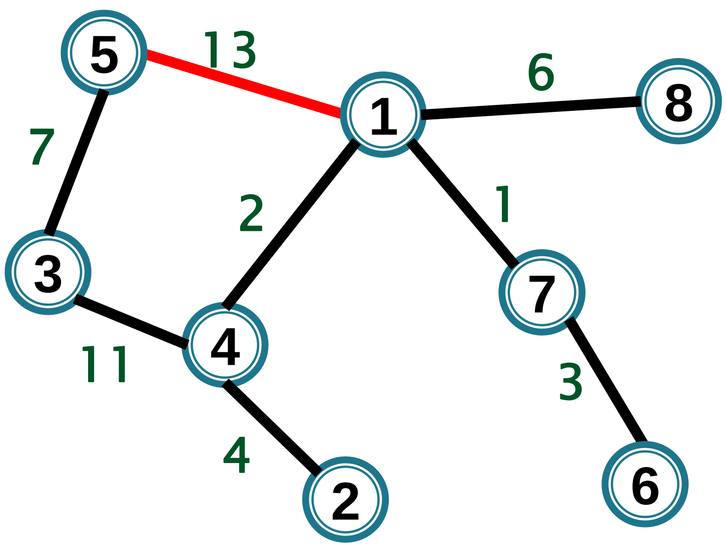
Second best

Idee algoritm:

Dacă **fixăm** un $xy \notin T_{min}$, atunci cum determinăm $uv \in T_{min}$ a.î

T_{min} - uv +xy să fie arbore și

w(xy)-w(uv) minim cu această proprietate?



Second best

Idee algoritm:

Dacă **fixăm** un $xy \notin T_{min}$, atunci cum determinăm $uv \in T_{min}$ a.î

$T_{min} - uv + xy$ să fie arbore și

$w(xy)-w(uv)$ minim cu această proprietate?

- uv este muchia de cost maxim din ciclul închis de xy în T_{min}, adică muchia de cost maxim din lanțul de la x la y din T_{min}

Second best

Idee algoritm:

Fie T unicul apcm

Se determină xy pentru care se atinge minimul:

$$\min\{\ w(xy) - w(\max[x,y]) \mid xy \notin T\}$$

unde

max[x,y] = **muchia** maximă din lanțul de la x la y din T

$T_s = T + xy - \max[x,y]$

Second best

Algoritm second best

1. Determinăm T apcm în G

2. Pentru orice $x, y \in T$ determină:

    $max[x,y]$ = **muchia** maximă din lanțul de la x la y din T

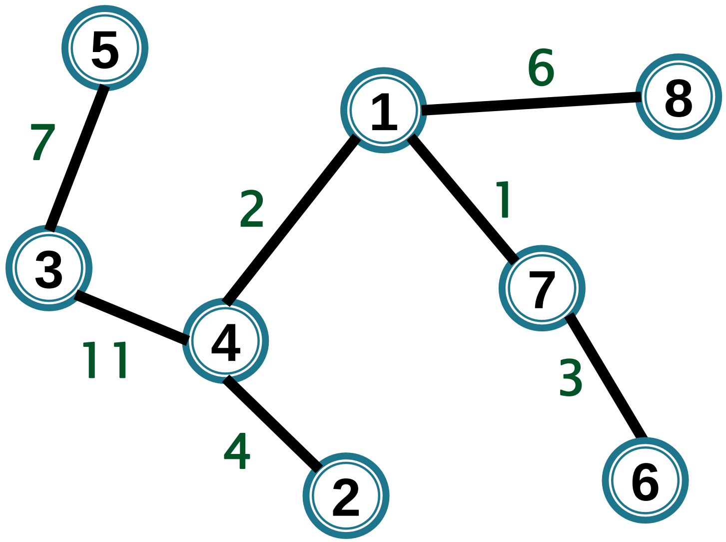
3. Determină o muchie $xy \notin T$ cu

    $w(x,y) - w(max[x,y])$ minim

4. $T_s = T + xy - max[x,y]$

Second best

Cum determinăm max[s,x] pentru orice s,x in T ?

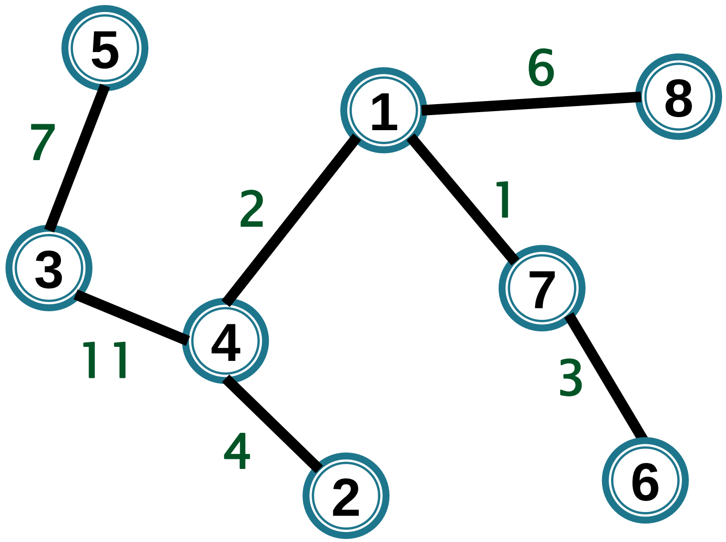


arbore parțial de cost minim

x	1	2	3	4	5	6	7	8
max[1,x]								
max[2,x]								

Second best

Cum determinăm max[s,x] pentru orice s,x in T ?



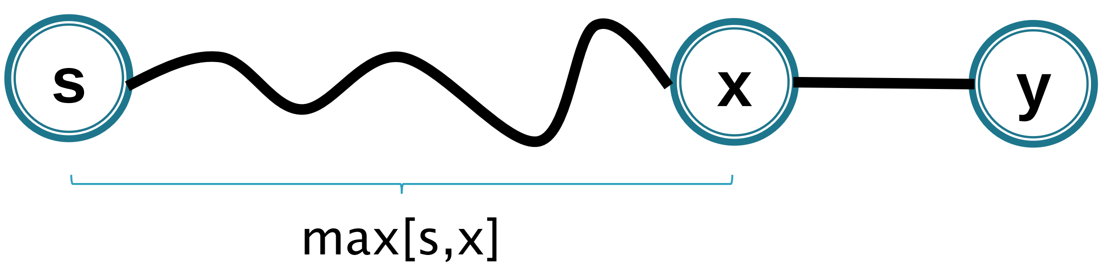
arbore parțial de cost minim

x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2) 4	(4,3) 11	(1,4) 2	(4,3) 11	(7,6) 3	(1,7) 1	(1,8) 6
max[2,x]	(2,4) 2	0	(3,4) 11	(2,4) 4	(3,4) 11	(2,4) 4	(2,4) 4	(1,8) 6

Second best

Determinare max[s,x] pentru orice s,x in T

Pentru s fixat – **parcurgere din s**

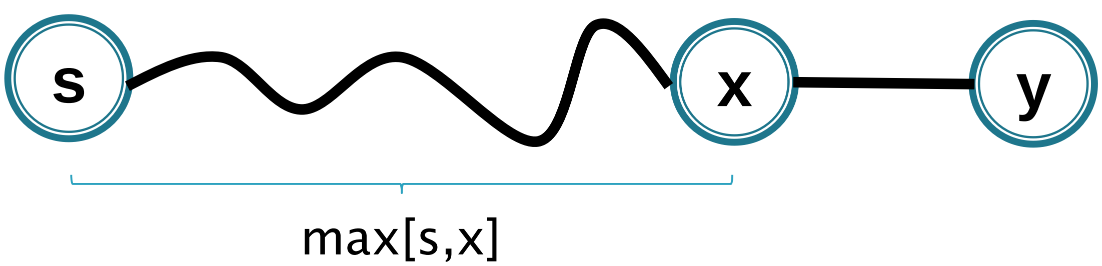


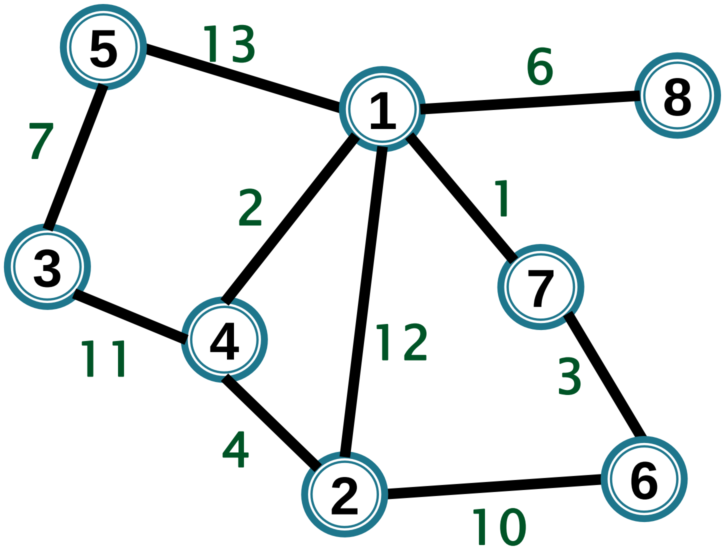
Second best

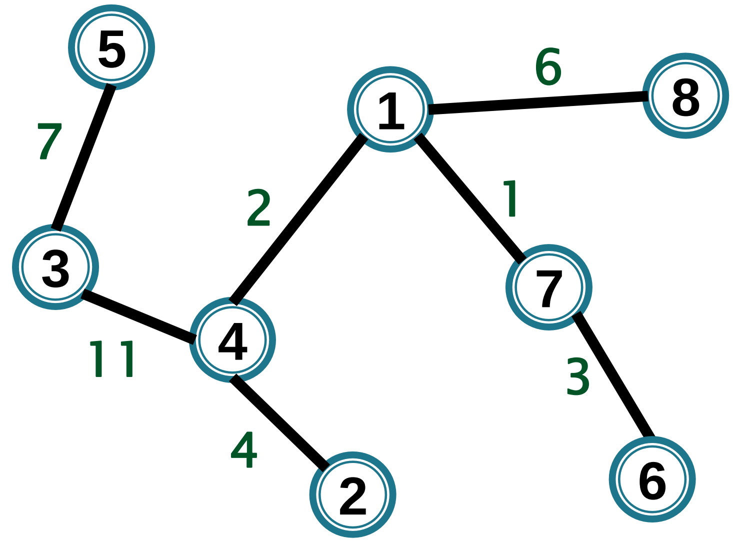
Determinare max[s,x] pentru orice s,x în T

Pentru s fixat – **parcurgere din s**, actualizând pentru un vârf y descoperit din x max[s,y] astfel:

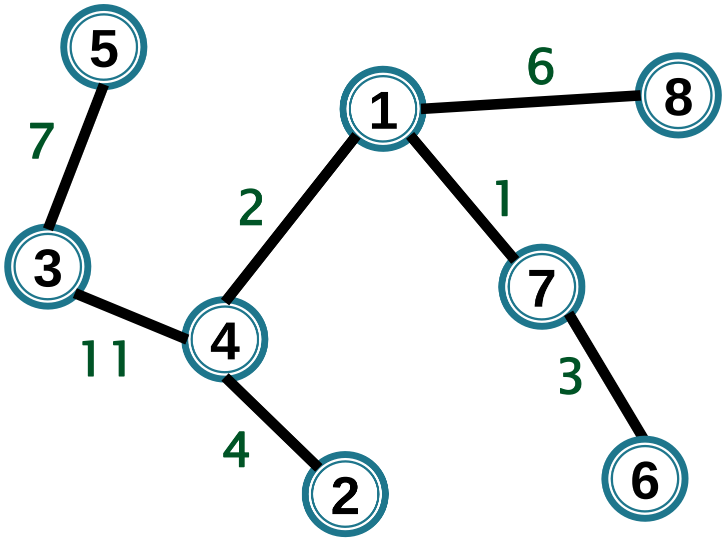
$$\text{max[s,y]} = \begin{cases} xy, & \text{dacă } w(x,y) > w(\text{max[s,x]}) \\ \text{max[s,x]}, & \text{altfel} \end{cases}$$





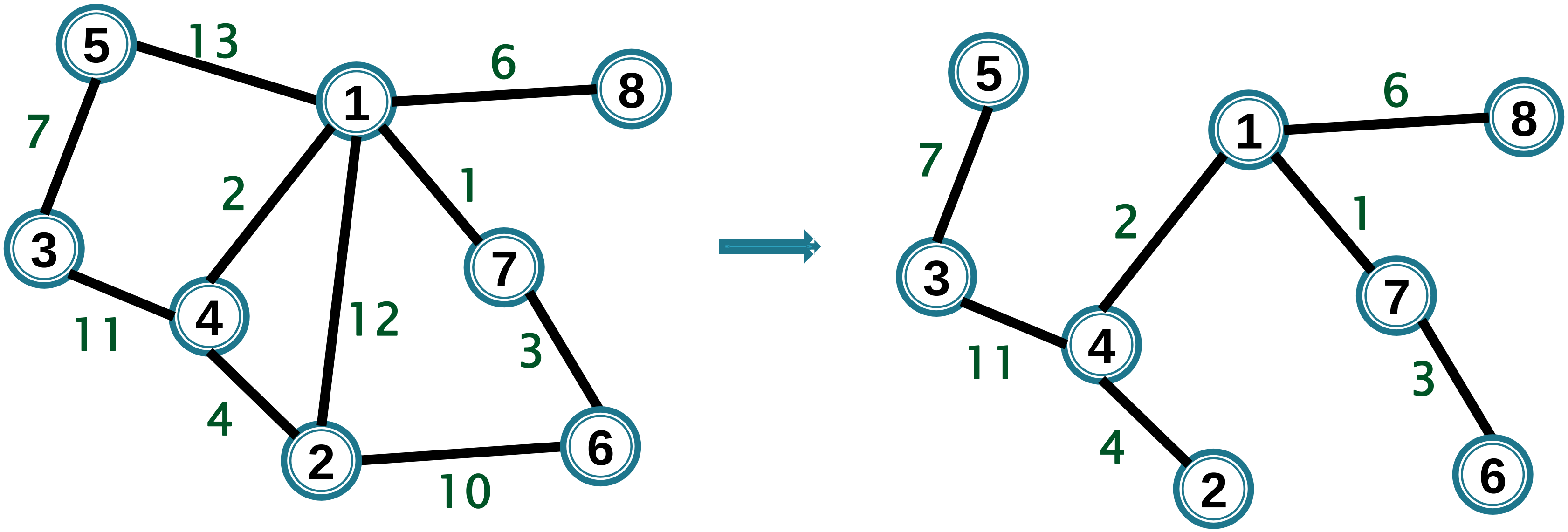


arbore parțial de cost minim

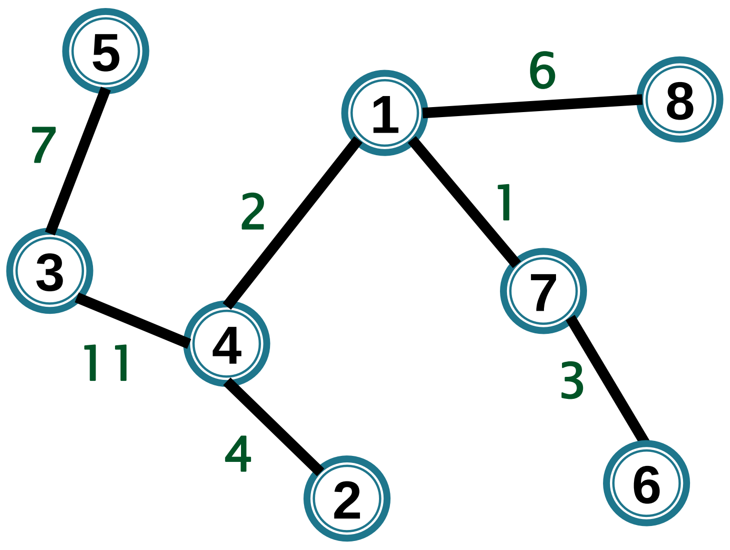


arbore parțial de cost minim

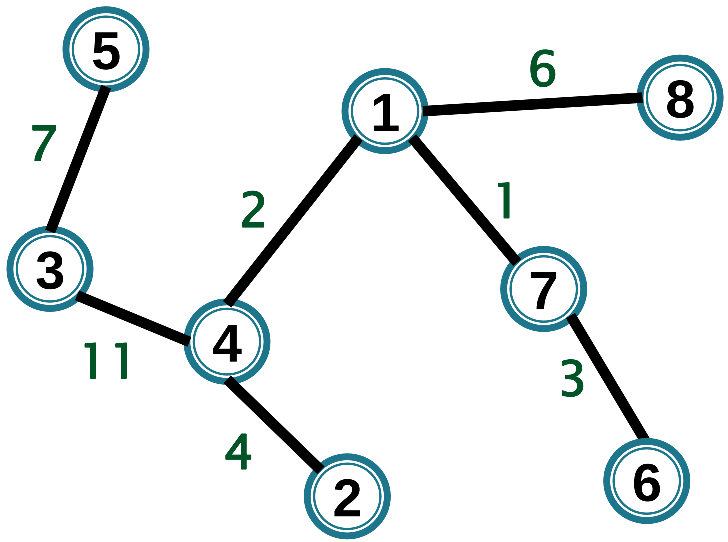
Calculăm max[1, x] folosind BF(1)

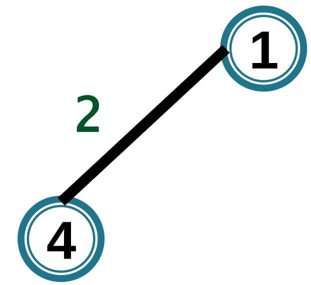


arbore parțial de cost minim

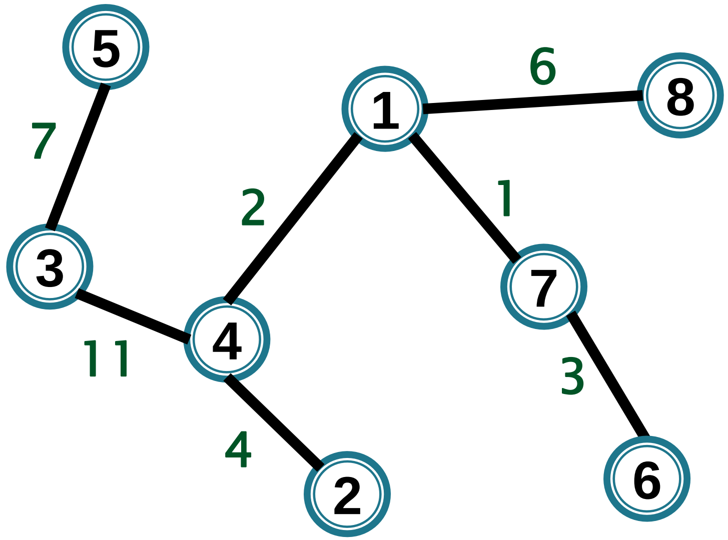


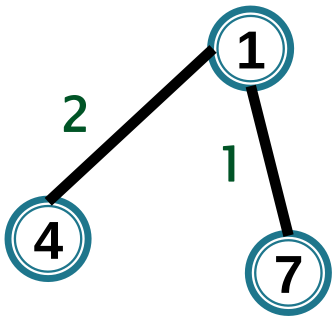
x	1	2	3	4	5	6	7	8
max[1,x] d	0							



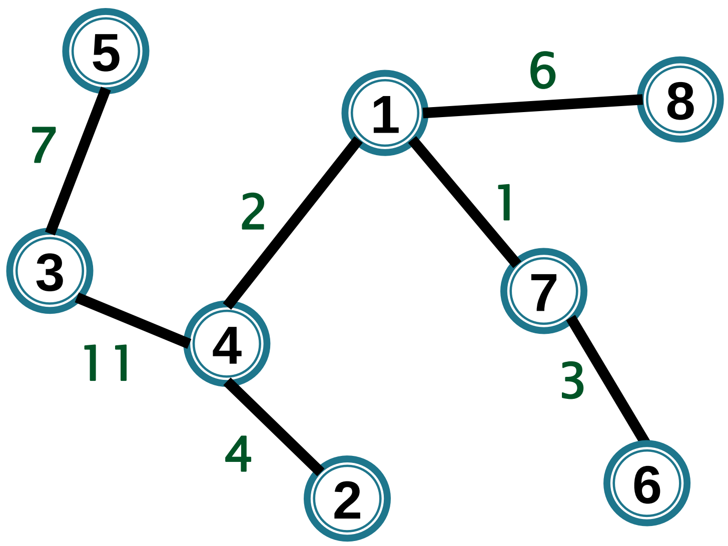


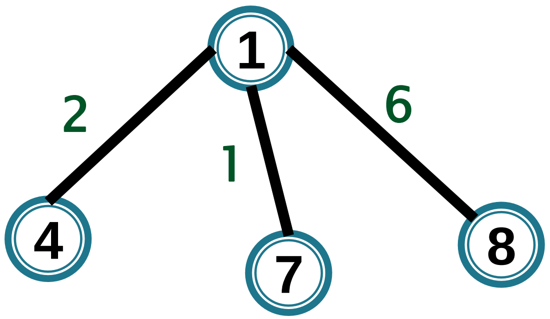
x	1	2	3	4	5	6	7	8
max[1,x] d	0			(1,4) 2				



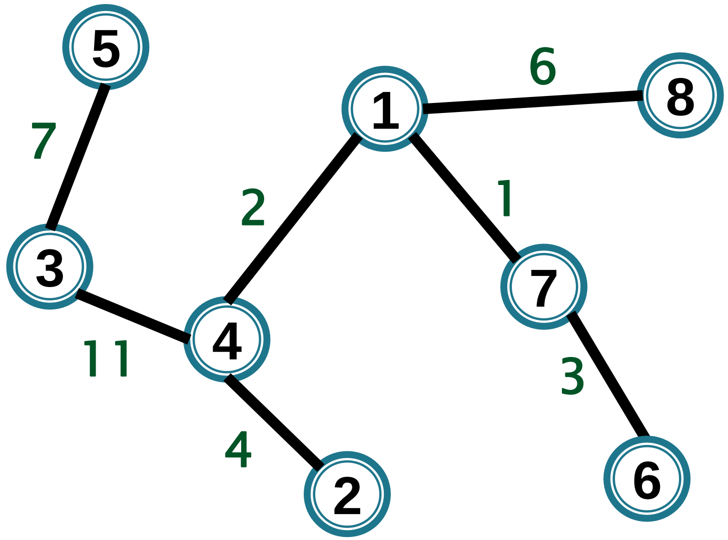


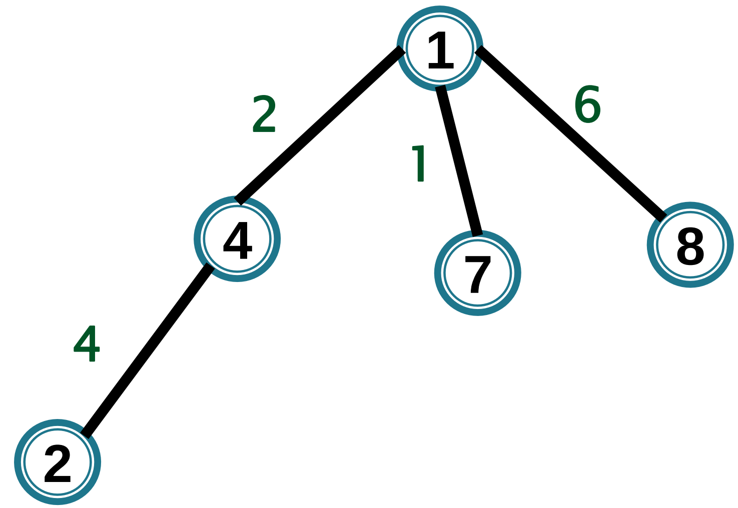
x	1	2	3	4	5	6	7	8
max[1,x]	0			(1,4) 2			**(1,7) 1**	



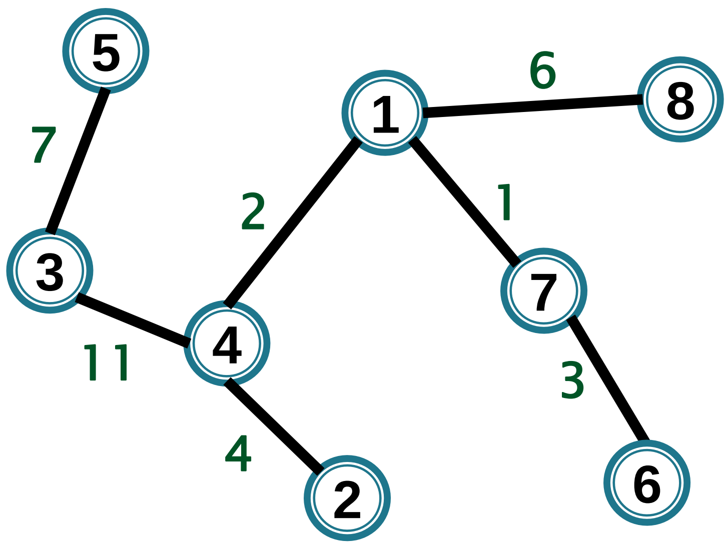


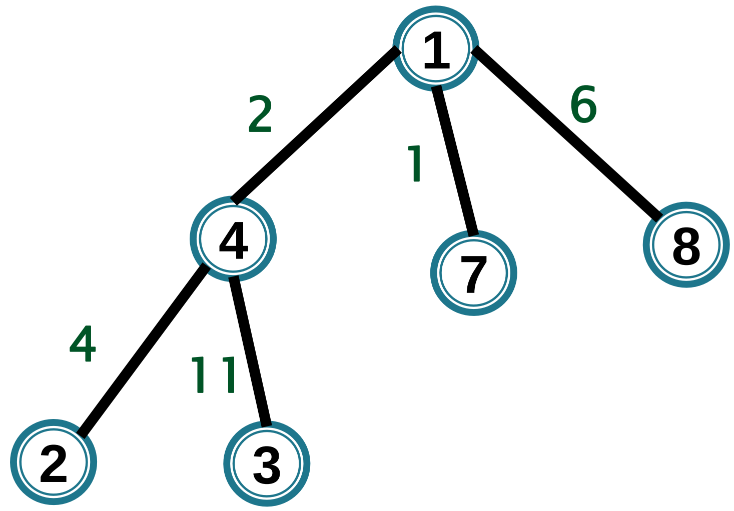
x	1	2	3	4	5	6	7	8
max[1,x]	0			(1,4) 2			(1,7) 1	**(1,8) 6**



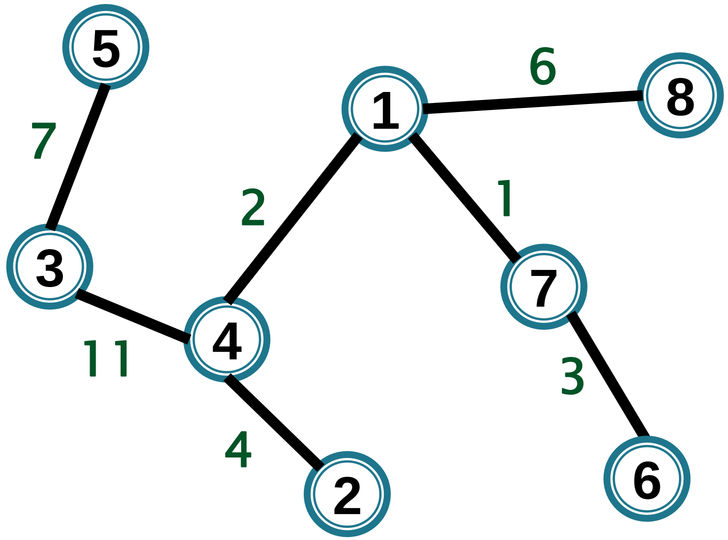


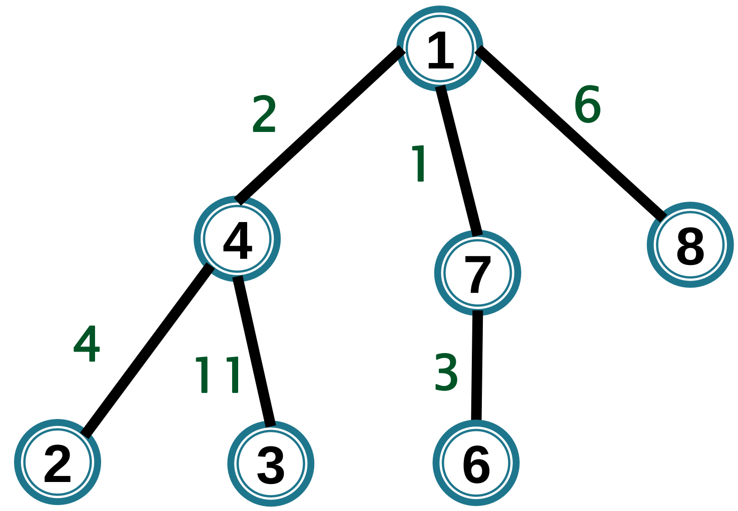
x	1	2	3	4	5	6	7	8
max[1,x]	0	**(4,2)** **4**		(1,4) 2			(1,7) 1	(1,8) 6



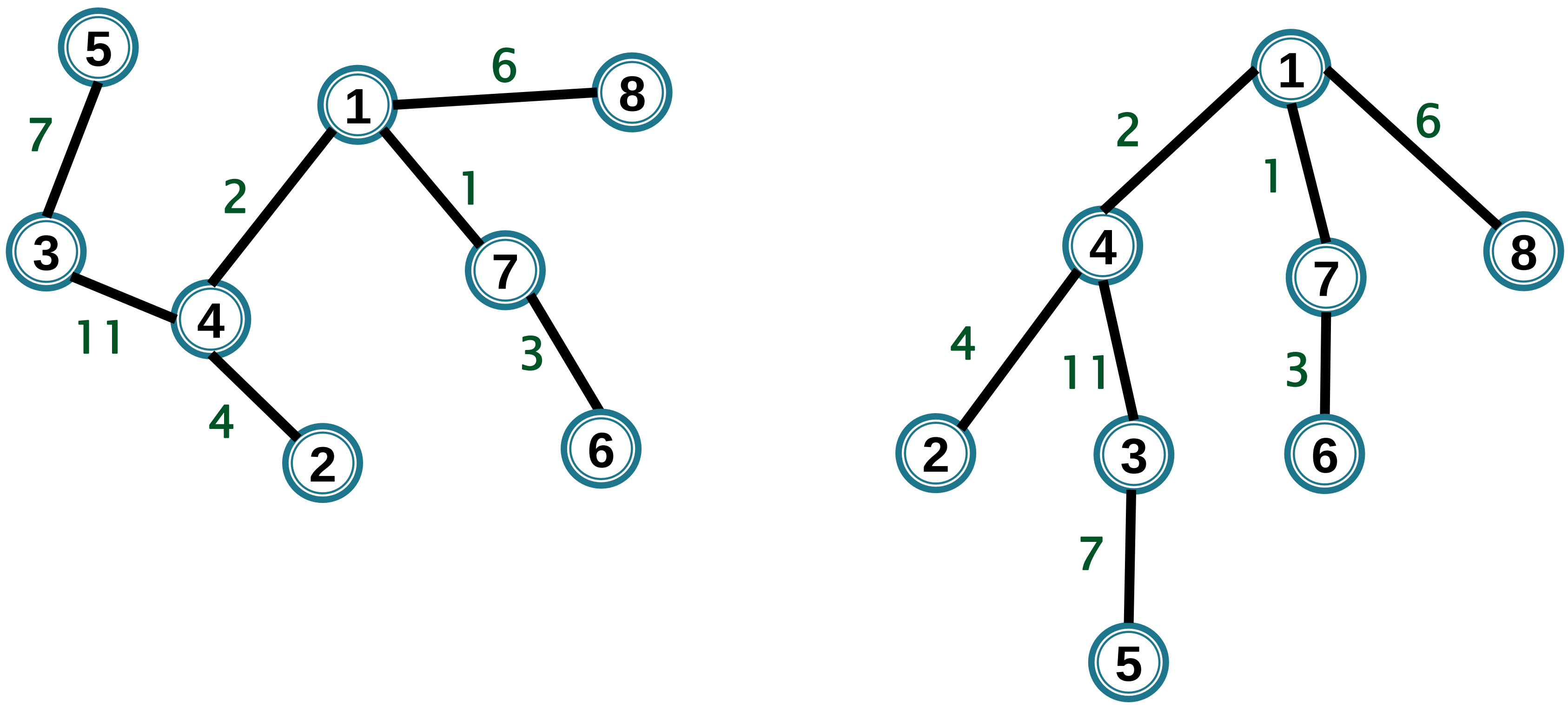


x	1	2	3	4	5	6	7	8
max[1,x] d	0	(4,2) 4	**(4,3)** **11**	(1,4) 2			(1,7) 1	(1,8) 6

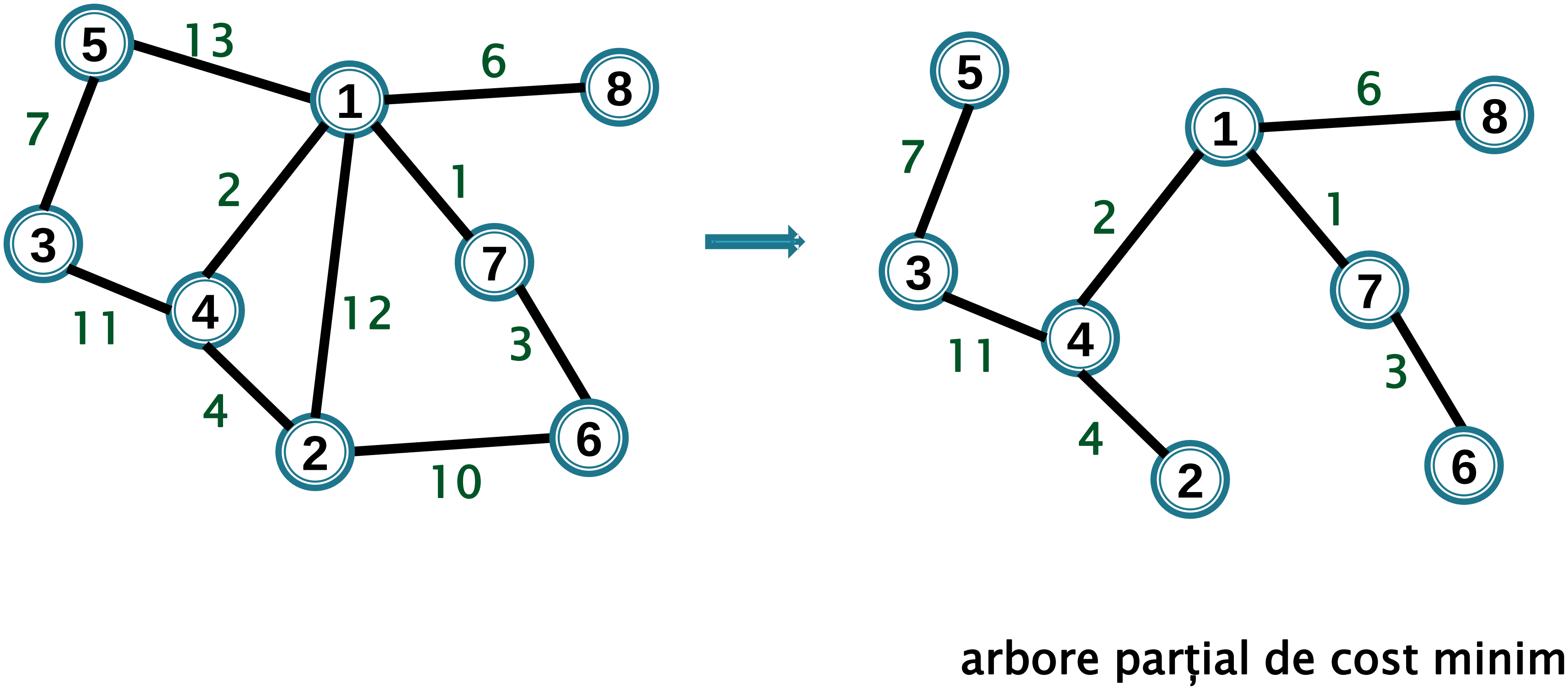




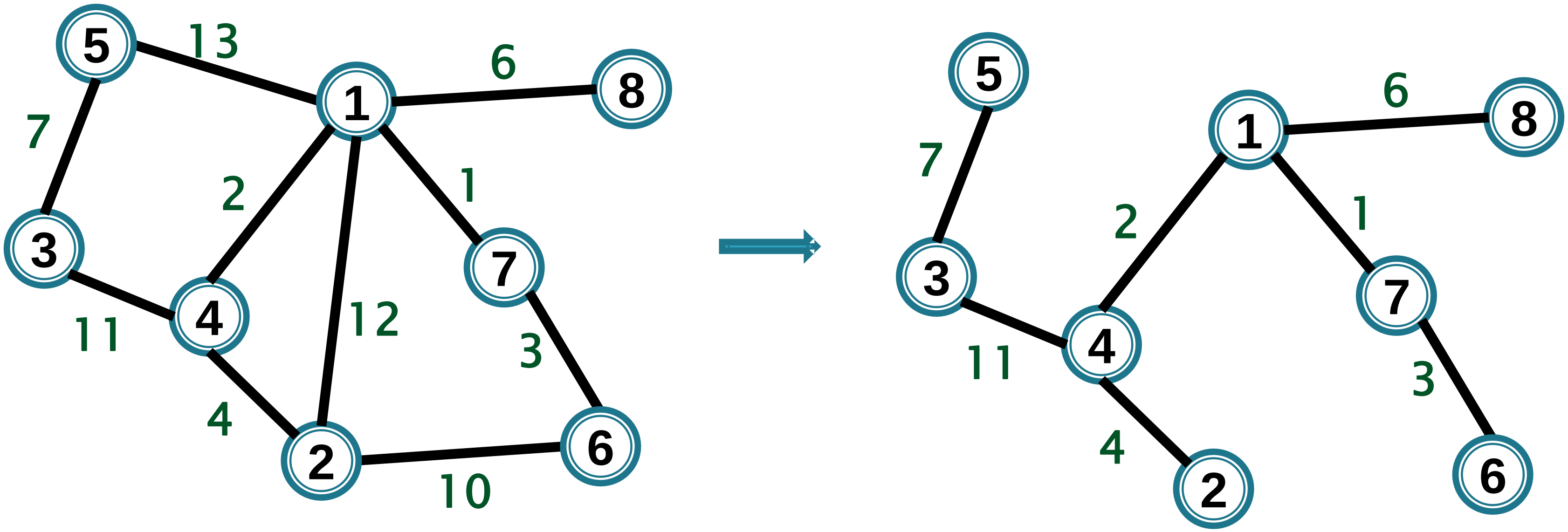
x	1	2	3	4	5	6	7	8
max[1,x] d	0	(4,2) 4	(4,3) 11	(1,4) 2		**(7,6)** **3**	(1,7) 1	(1,8) 6



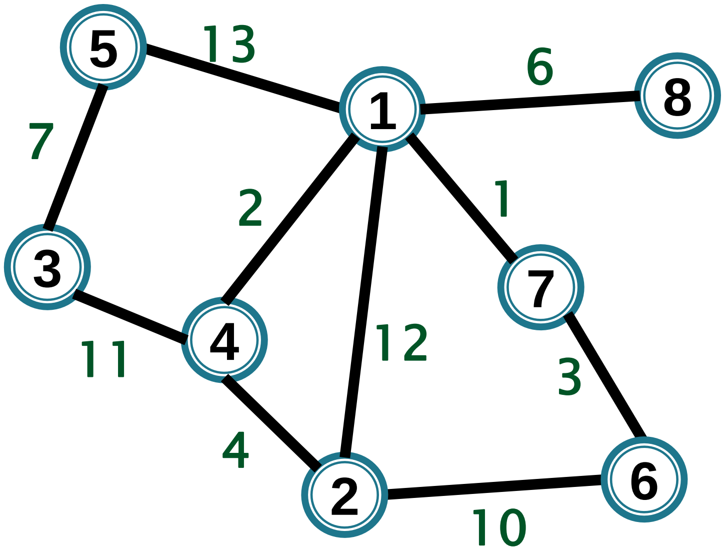
x	1	2	3	4	5	6	7	8
max[1,x] d	0	(4,2) 4	(4,3) 11	(1,4) 2	**(4,3) 11**	(7,6) 3	(1,7) 1	(1,8) 6

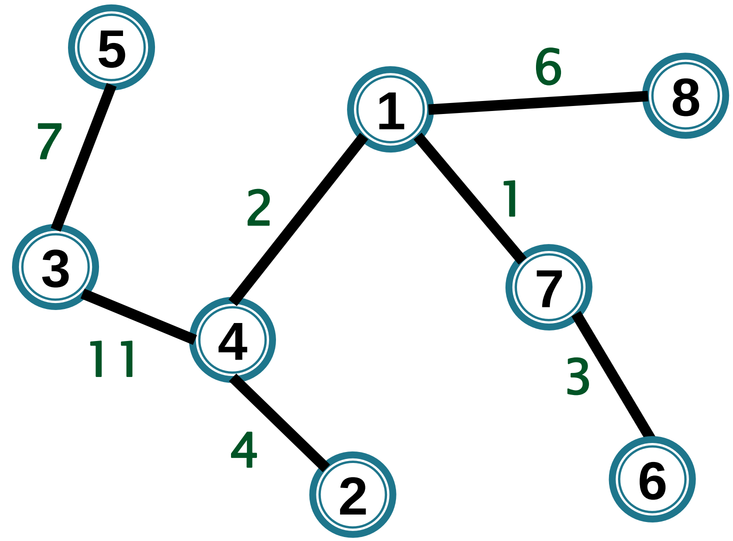


Calculăm max[2, x] folosind BF(2)



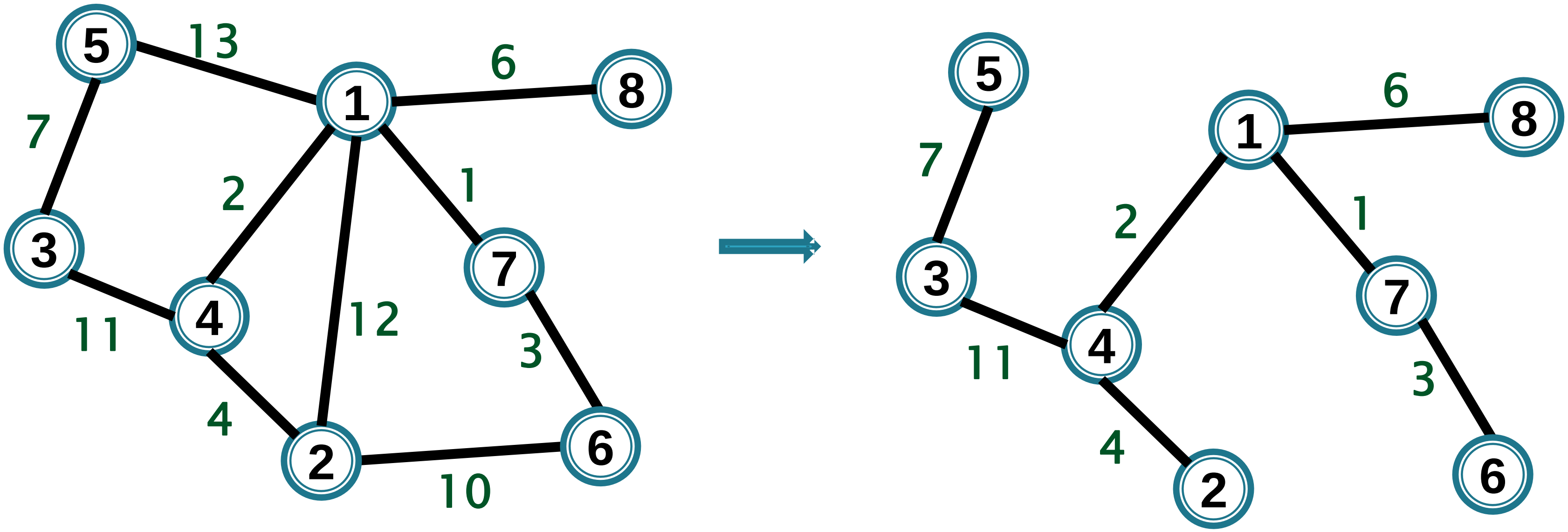
x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)





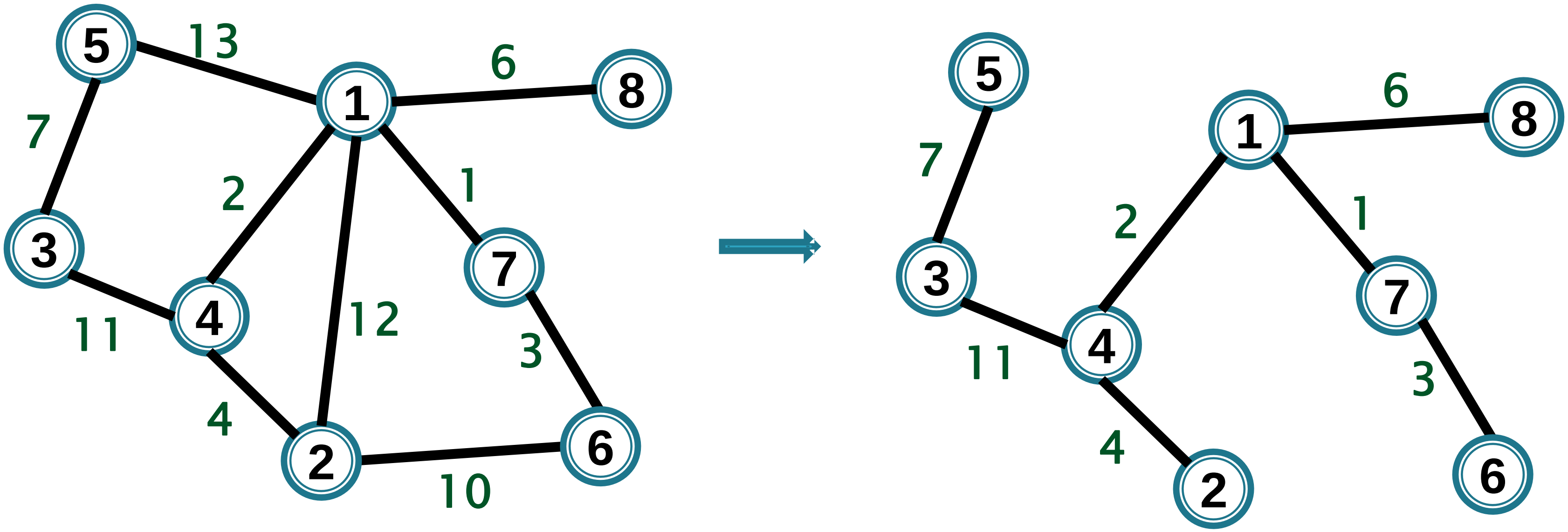
x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

Considerăm pe rând muchiile care nu sunt în apcm:



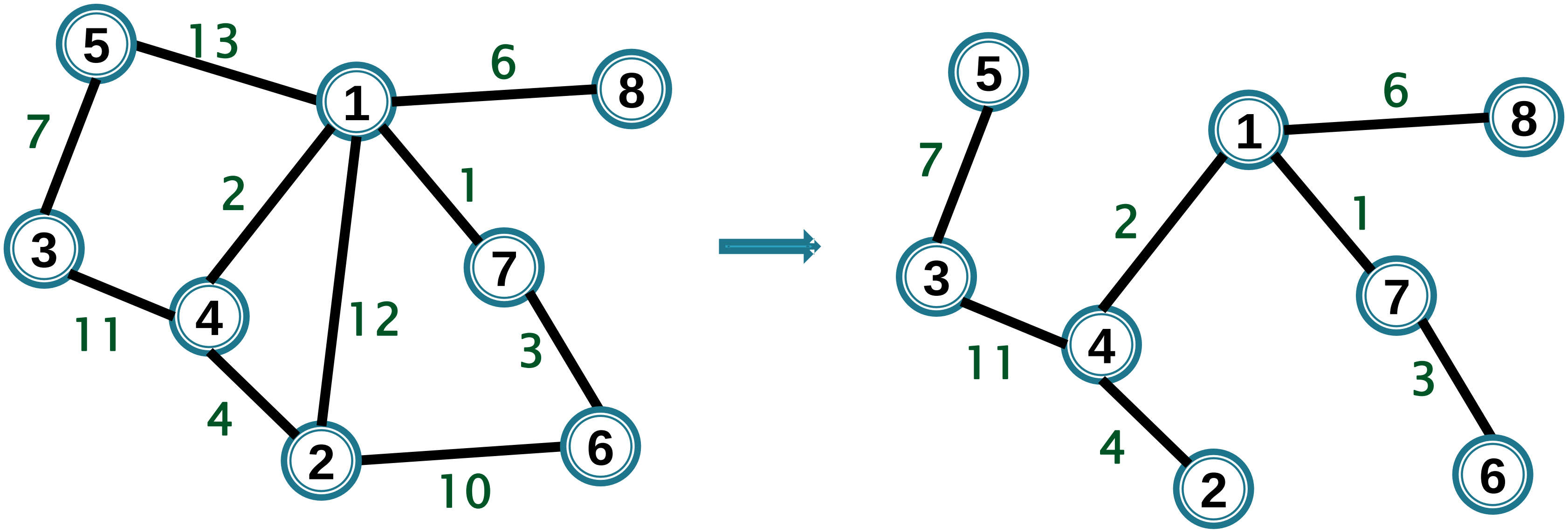
x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

(2,6):



x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

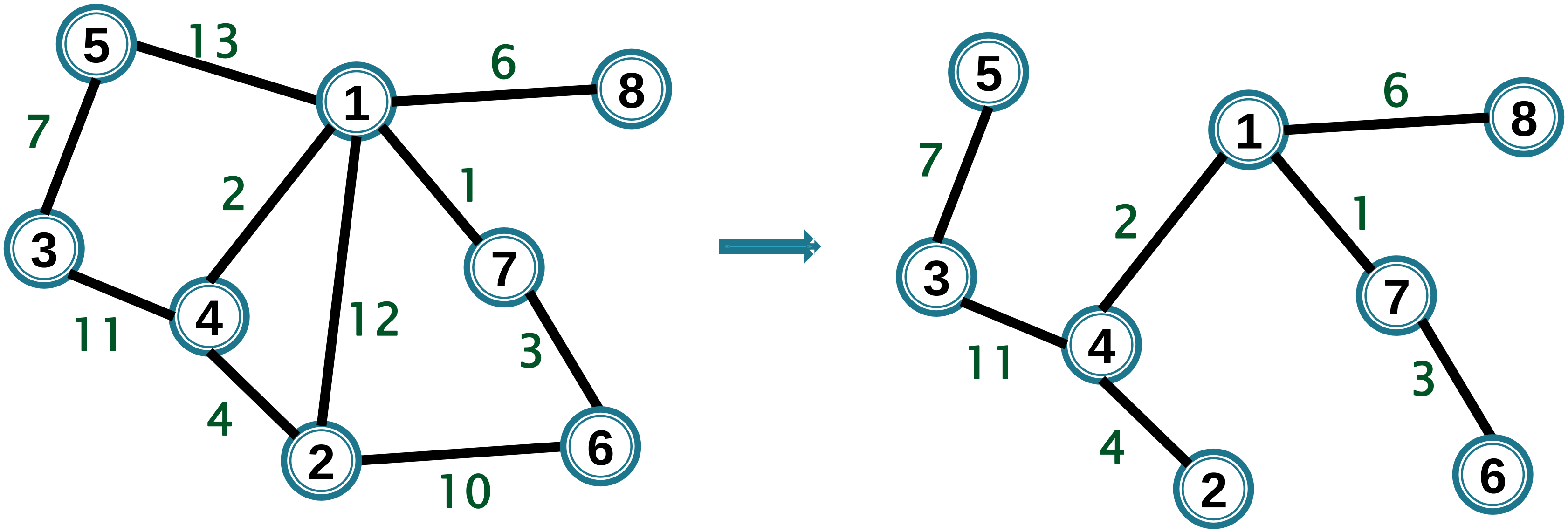
(2,6): w(2,6) – w(max[2,6]) = w(2,6) – w(2,4) = 10 – 4 = 6



x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

(2,6): $w(2,6) - w(\max[2,6]) = w(2,6) - w(2,4) = 10 - 4 = 6$

(1,2): $w(1,2) - w(\max[1,2]) = w(1,2) - w(2,4) = 12 - 4 = 8$

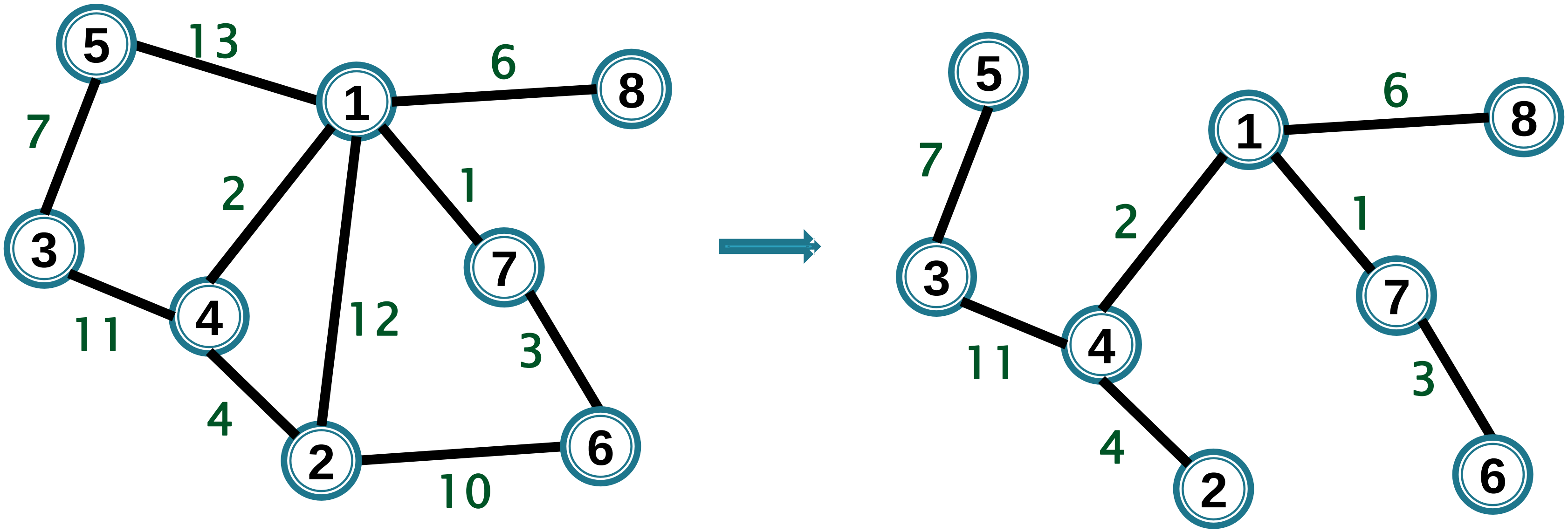


x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

(2,6): $w(2,6) - w(\max[2,6]) = w(2,6) - w(2,4) = 10 - 4 = 6$

(1,2): $w(1,2) - w(\max[1,2]) = w(1,2) - w(2,4) = 12 - 4 = 8$

(1,5): $w(1,5) - w(\max[1,5]) = w(1,5) - w(4,3) = 13 - 11 = 2$



x	1	2	3	4	5	6	7	8
max[1,x]	0	(4,2)	(4,3)	(1,4)	(4,3)	(7,6)	(1,7)	(1,8)
max[2,x]	(2,4)	0	(3,4)	(2,4)	(3,4)	(2,4)	(2,4)	(1,8)

(2,6): w(2,6) – w(max[2,6]) = w(2,6) – w(2,4) = 10 – 4 = 6

(1,2): w(1,2) – w(max[1,2]) = w(1,2) – w(2,4) = 12 – 4 = 8

(1,5): w(1,5) – w(max[1,5]) = w(1,5) – w(4,3) = 13 – 11 = 2

=> **Second best = T_{min} – max[1,5] + (1,5) = T_{min} – (3,4) + (1,5)**

Second best

Algoritm second best

1. Determinăm T apcm în G

2. Pentru orice $x, y \in T$ determină:

    $max[x,y]$ = **muchia** maximă din lanțul de la x la y din T

3. Determină o muchie $xy \notin T$ cu

    $w(x,y) - w(max[x,y])$ minim

4. $T_s = T + xy - max[x,y]$

Complexitate?

Second best

Algoritm second best

1. Determinăm T apcm în G

2. Pentru orice x, y $\in$ T determină:

   max[x,y] = **muchia** maximă din lanțul de la x la y din T

3. Determină o muchie xy $\notin$ T cu

   w(x,y) – w(max[x,y]) minim

4. T_s = T + xy – max[x,y]

Complexitate O(n^2) – cu varianta descrisă la pasul 3

Alte idei pentru pasul 3?