# Concluzii Drumuri minime de sursă unică algoritmi

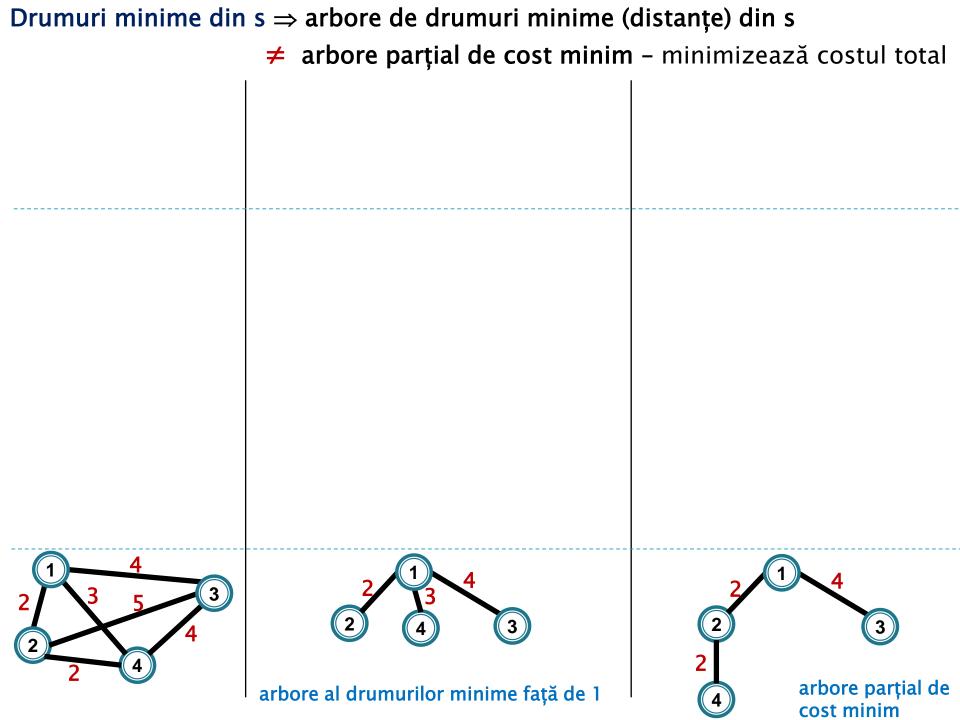
G – neponderat Parcurgere lățime BF	G - ponderat, ponderi >0 Algoritmul lui Dijkstra	G – ponderat fără circuite
BF(s)	Dijkstra(s)	DAGS(s)
coada $C \leftarrow \emptyset$ ; adauga(s, C)	<pre>(min-heap) Q ← V {se putea incepe doar cu Q ← {s} +vector viz; v∈Q ⇔ v nevizitat}</pre>	SortTop ← sortare_topologica(G)

```
G - neponderat
                                 G – ponderat, ponderi >0
                                                                              G - ponderat fără circuite
                                Algoritmul lui Dijkstra
Parcurgere lățime BF
                                 Dijkstra(s)
BF(s)
                                                                              DAGS(s)
coada C \leftarrow \emptyset;
                                 (min-heap) Q \leftarrow V
                                                                              SortTop ← sortare topologica(G)
adauga(s, C)
                                 {se putea incepe doar cu Q \leftarrow \{s\}
                                 +vector viz; \mathbf{v} \in \mathbb{Q} \Leftrightarrow \mathbf{v} \text{ nevizitat}
                                 pentru fiecare u∈V
pentru fiecare u∈V
                                                                              pentru fiecare u∈V
                                      d[u] = \infty; tata[u]=0
                                                                                   d[u] = \infty; tata[u]=0
  d[u]=\infty; tata[u]=viz[u]=0
                                 d[s] = 0
                                                                              d[s] = 0
viz[s] \leftarrow 1; d[s] \leftarrow 0
```

```
G - neponderat
                                G – ponderat, ponderi >0
                                                                            G - ponderat fără circuite
                                Algoritmul lui Dijkstra
Parcurgere lățime BF
                                Dijkstra(s)
BF(s)
                                                                            DAGS(s)
coada C \leftarrow \emptyset;
                                 (min-heap) Q \leftarrow V
                                                                             SortTop ← sortare topologica(G)
adauga (s, C)
                                 {se putea incepe doar cu Q \leftarrow \{s\}
                                +vector viz; \mathbf{v} \in \mathbb{Q} \iff \mathbf{v} \text{ nevizitat}
                                pentru fiecare u∈V
pentru fiecare u∈V
                                                                            pentru fiecare u∈V
                                     d[u] = \infty; tata[u]=0
                                                                                 d[u] = \infty; tata[u]=0
  d[u]=\infty; tata[u]=viz[u]=0
                                d[s] = 0
                                                                            d[s] = 0
viz[s] \leftarrow 1; d[s] \leftarrow 0
                                cat timp Q \neq \emptyset
cat timp C \neq \emptyset
                                                                            pentru fiecare u ∈ SortTop
                                    u = extrage(Q) vârf cu eticheta
   u \leftarrow extrage(C);
                                          d minimă
                                    pentru fiecare uv∈E
                                                                                pentru fiecare uv∈E
   pentru fiecare uv∈E
```

```
G - neponderat
                                G - ponderat, ponderi >0
                                                                            G - ponderat fără circuite
                                Algoritmul lui Dijkstra
Parcurgere lățime BF
                                Dijkstra(s)
BF(s)
                                                                            DAGS(s)
coada C \leftarrow \emptyset;
                                 (min-heap) Q \leftarrow V
                                                                            SortTop ← sortare topologica(G)
                                {se putea incepe doar cu Q \leftarrow \{s\}
adauga (s, C)
                                +vector viz; v \in Q \Leftrightarrow v \text{ nevizitat}
                                pentru fiecare u∈V
pentru fiecare u∈V
                                                                            pentru fiecare u∈V
                                     d[u] = \infty; tata[u]=0
                                                                                 d[u] = \infty; tata[u]=0
  d[u] = \infty; tata[u] = viz[u] = 0
                                d[s] = 0
                                                                            d[s] = 0
viz[s] \leftarrow 1; d[s] \leftarrow 0
                                cat timp Q \neq \emptyset
cat timp C \neq \emptyset
                                                                            pentru fiecare u ∈ SortTop
                                    u = extrage(Q) vârf cu eticheta
   u \leftarrow extrage(C);
                                          d minimă
                                    pentru fiecare uv∈E
                                                                               pentru fiecare uv∈E
   pentru fiecare uv∈E
                                       daca v \in Q si d[u]+w(u,v) < d[v]
                                                                                     daca d[u]+w(u,v)< d[v]
        daca viz[v]=0
                                               d[v] = d[u] + w(u,v)
                                                                                           d[v] = d[u] + w(u,v)
           d[v] \leftarrow d[u]+1
                                                tata[v] = u
                                                                                           tata[v] = u
           tata[v] \leftarrow u
                                                repara(v,Q)
            adauga (v, C)
           viz[v] \leftarrow 1
                                scrie d, tata
                                                                            scrie d, tata
scrie d, tata
```

G – neponderat Parcurgere lățime BF	G - ponderat, ponderi >0 Algoritmul lui Dijkstra	G – ponderat fără circuite
BF(s)	Dijkstra(s)	DAGS(s)
coada $C \leftarrow \emptyset$ ; adauga(s, C)	<pre>(min-heap) Q ← V {se putea incepe doar cu Q ← {s} +vector viz; v∈Q ⇔ v nevizitat}</pre>	SortTop ← sortare_topologica(G)
pentru fiecare u∈V d[u]=∞;tata[u]=viz[u]=0	pentru fiecare u∈V d[u] = ∞; tata[u]=0	pentru fiecare u∈V d[u] = ∞; tata[u]=0
$viz[s] \leftarrow 1; d[s] \leftarrow 0$	d[s] = 0	d[s] = 0
cat timp $C \neq \emptyset$ $u \leftarrow extrage(C)$ ;	cat timp $Q \neq \emptyset$ u = extrage(Q) vârf cu eticheta d minimă	pentru fiecare u ∈ SortTop
<pre>pentru fiecare uv∈E   daca viz[v]=0   d[v] ← d[u]+1   tata[v] ← u   adauga(v, C)   viz[v] ← 1</pre>	<pre>pentru fiecare uv∈E    daca <del>v∈Q</del> si d[u]+w(u,v)<d[v] d[v]="d[u]+w(u,v)" pre="" repara(v,q)<="" tata[v]="u"></d[v]></pre>	<pre>pentru fiecare uv∈E</pre>
scrie d, tata	scrie d, tata	scrie d, tata
O(n+m)	O(m log(n))/ O(n <sup>2</sup> )	O(n+m)

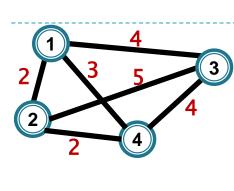


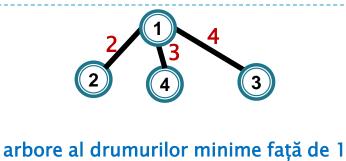
#### Drumuri minime din $s \Rightarrow$ arbore de drumuri minime (distanțe) din s

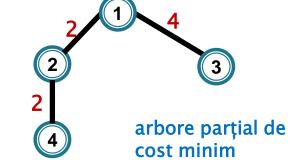
≠ arbore parțial de cost minim - minimizează costul total

G-(ne)orientat ponderat,
ponderi >0
Drumuri minime din s
Algoritmul lui Dijkstra

G- neorientat ponderat
ponderi reale
Arbore parțial de cost minim
Algoritmul lui Prim







#### Drumuri minime din $s \Rightarrow$ arbore de drumuri minime (distanțe) din s

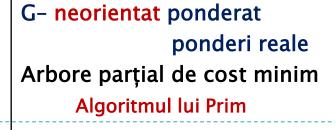
≠ arbore parțial de cost minim - minimizează costul total

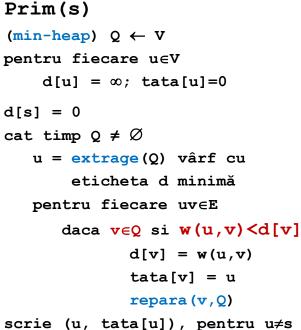
```
G-(ne)orientat ponderat,
ponderi >0
Drumuri minime din s
```

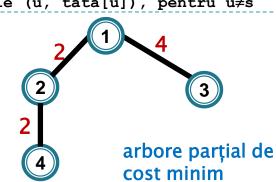
Drumun minime am s

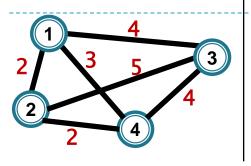
Algoritmul lui Dijkstra

arbore al drumurilor minime față de 1









#### Drumuri minime din $s \Rightarrow$ arbore de drumuri minime (distanțe) din s

≠ arbore parțial de cost minim - minimizează costul total

```
G-(ne)orientat ponderat,
                                            G- neorientat ponderat
                 ponderi >0
Drumuri minime din s
                                            Arbore parțial de cost minim
    Algoritmul lui Dijkstra
Dijkstra(s)
                                            Prim(s)
(min-heap) Q \leftarrow V
                                            (min-heap) Q \leftarrow V
pentru fiecare u∈V
                                            pentru fiecare u∈V
    d[u] = \infty; tata[u]=0
d[s] = 0
                                            d[s] = 0
cat timp Q \neq \emptyset
                                            cat timp Q \neq \emptyset
   u = extrage(Q) vârf cu eticheta
       d minimă
   pentru fiecare uv∈E
      daca v \in Q si d[u] + w(u, v) < d[v]
              d[v] = d[u] + w(u,v)
              tata[v] = u
              repara (v,Q)
scrie d, tata
```

# d[v] = w(u,v)tata[v] = urepara (v,Q) scrie (u, tata[u]), pentru u≠s arbore partial de cost minim

daca  $v \in Q$  si w(u,v) < d[v]

ponderi reale

Algoritmul lui Prim

 $d[u] = \infty$ ; tata[u]=0

u = extrage(Q) vârf cu

pentru fiecare uv∈E

eticheta d minimă

arbore al drumurilor minime față de 1

