R2.02 Graphes

Corentin Dufourg, Régis Fleurquin et Thibault Godin

IUT de Vannes Informatique

Graphe Valué

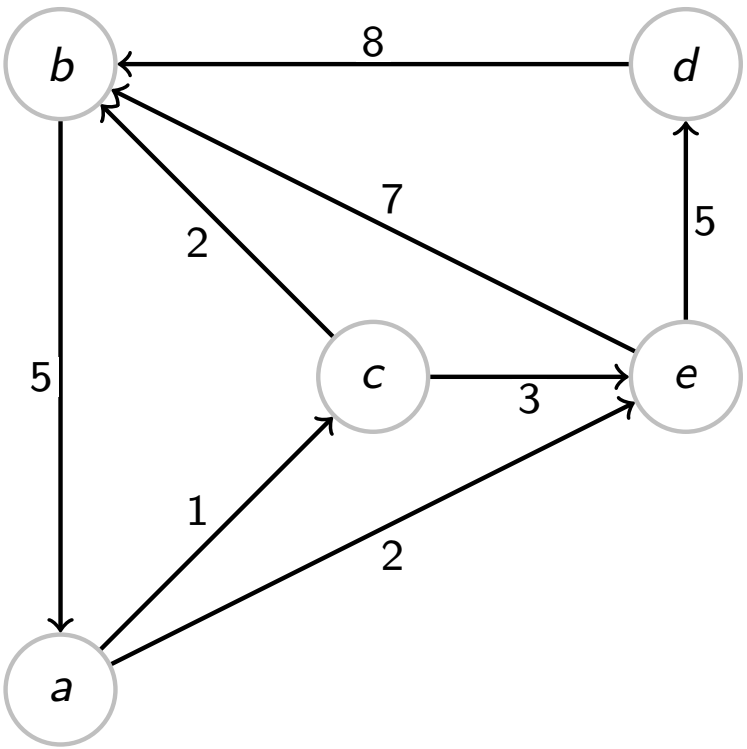
Soit $\mathcal{G} = (S, A)$ un graphe est dit **valué** (ou pondéré, en anglais weighted) s'il existe une application $w : A \to \mathbb{R}$. On utilise souvent une matrice de valuation W de taille n.

$$W_{i,j} = \begin{cases} w((i,j)) \text{ si } iAj \\ \infty \text{ sinon.} \end{cases}$$

Graphe Valué

Soit $\mathcal{G} = (S, A)$ un graphe est dit **valué** (ou pondéré, en anglais weighted) s'il existe une application $w : A \to \mathbb{R}$. On utilise souvent une matrice de valuation W de taille n.

$$W_{i,j} = \begin{cases} w((i,j)) \text{ si } iAj \\ \infty \text{ sinon.} \end{cases}$$



A pour matrice de valuation :

$$\begin{pmatrix} \infty & \infty & 1 & \infty & 2 \\ 5 & \infty & \infty & \infty & \infty \\ \infty & 2 & \infty & \infty & 3 \\ \infty & 8 & \infty & \infty & \infty \\ \infty & 7 & \infty & \infty & 5 \end{pmatrix}$$

Plus court chemin

Pour aller voir ses amis, une personne doit aller de Montpellier à Poitiers.
Pour cela, elle dispose des temps de trajets SNCF suivant :

Plus court chemin

Pour aller voir ses amis, une personne doit aller de Montpellier à Poitiers.
Pour cela, elle dispose des temps de trajets SNCF suivant :

	Montpellier	Lyon	Marseille	Lille	Paris	Bordeaux	Poitiers	Toulouse
Mpt	0	2h30	1h					3h
Lyon	2h30	0	1h30		2h	8h		
Mars	1h	1h30	0					
Lil				0	1h		5h	
Par		2h		1h	0		2h	
Bdx		8h				0	1h	2h
Poit				5h	2h	1h	0	
Tou	3h					2h		0

Plus court chemin

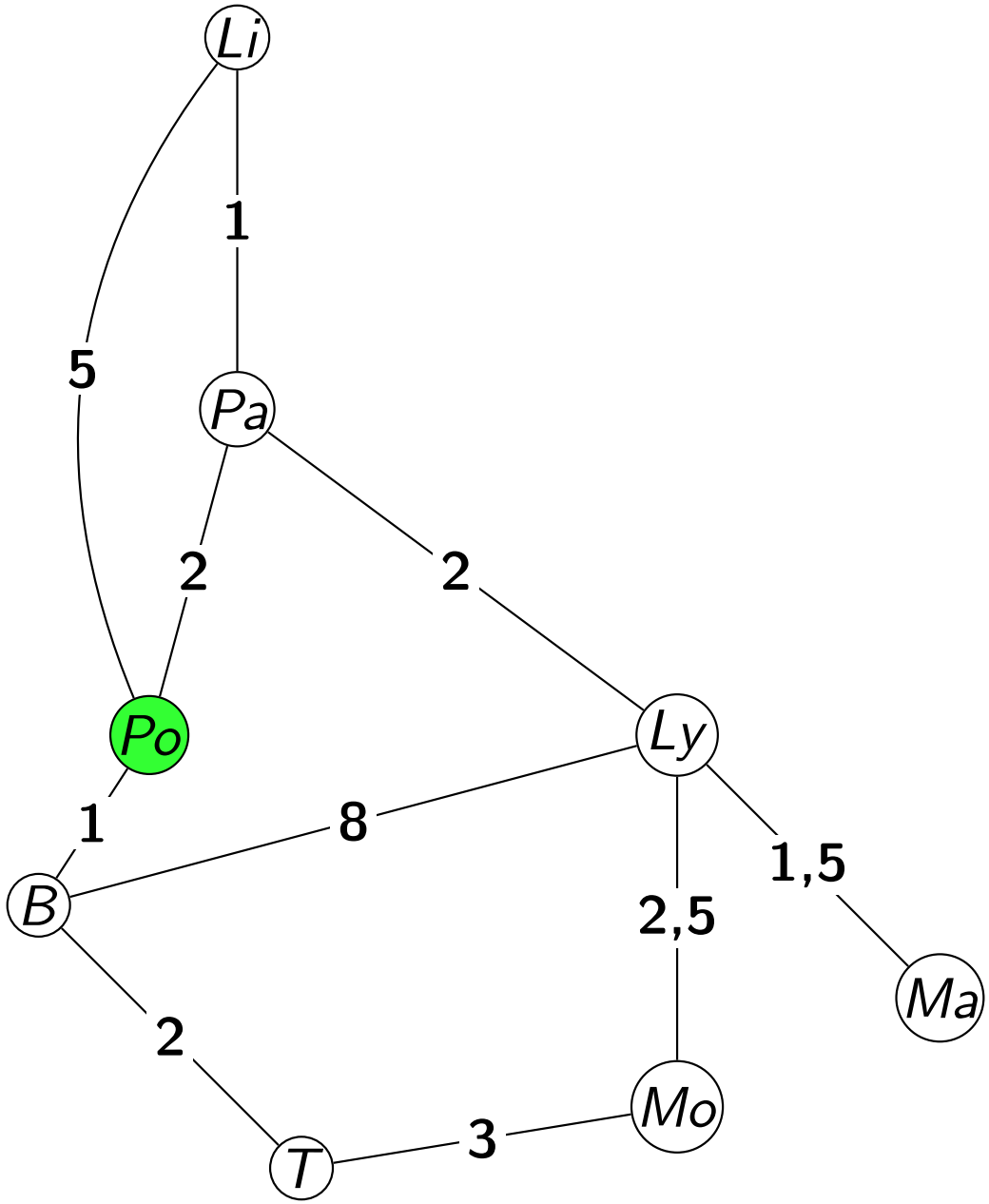
Pour aller voir ses amis, une personne doit aller de Montpellier à Poitiers.
Pour cela, elle dispose des temps de trajets SNCF suivant :

	Montpellier	Lyon	Marseille	Lille	Paris	Bordeaux	Poitiers	Toulouse
Mpt	0	2h30	1h					3h
Lyon	2h30	0	1h30		2h	8h		
Mars	1h	1h30	0					
Lil				0	1h		5h	
Par		2h		1h	0		2h	
Bdx		8h				0	1h	2h
Poit				5h	2h	1h	0	
Tou	3h					2h		0

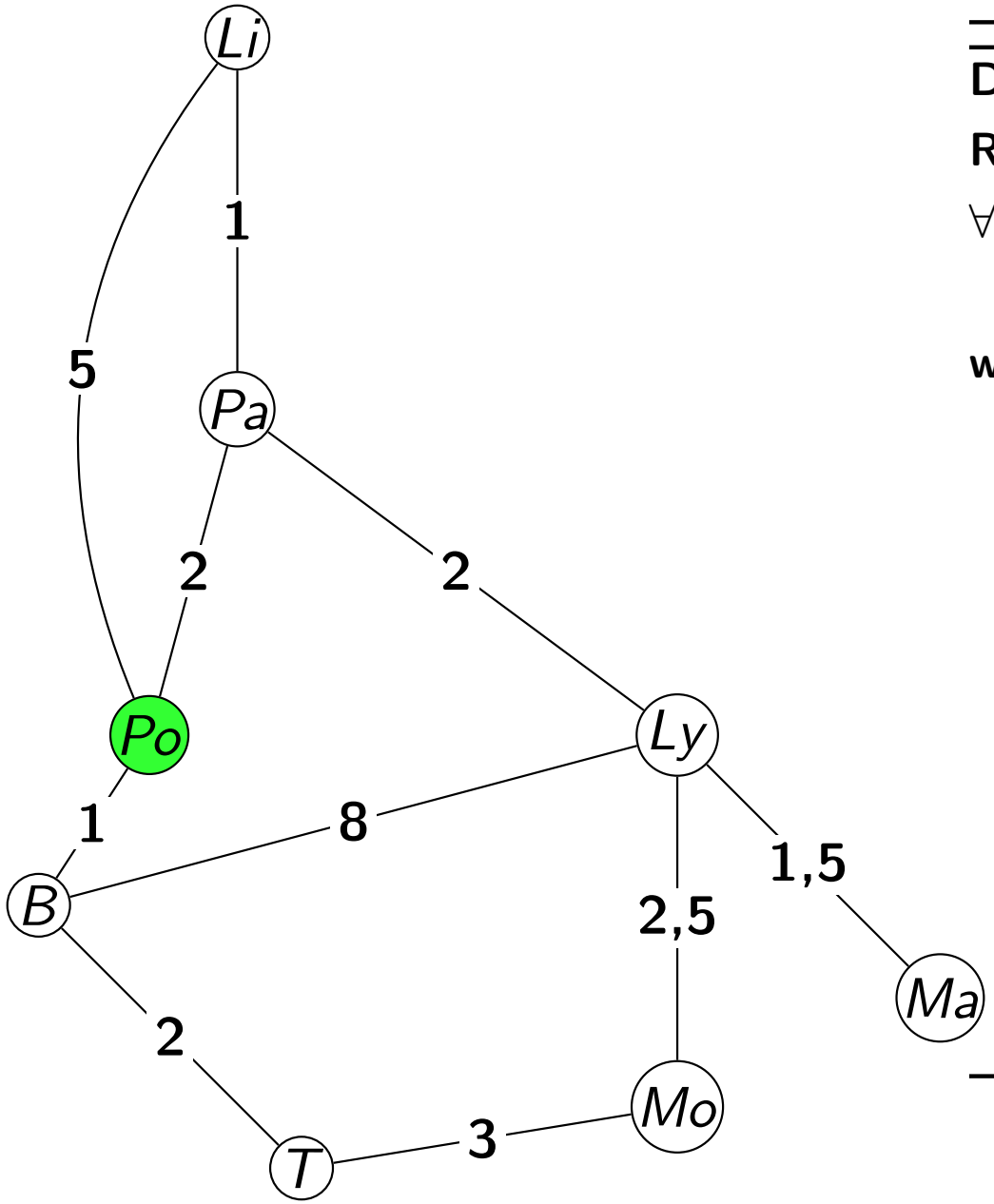
Quel est le trajet le plus indiqué ?

Plan

Dijkstra



Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

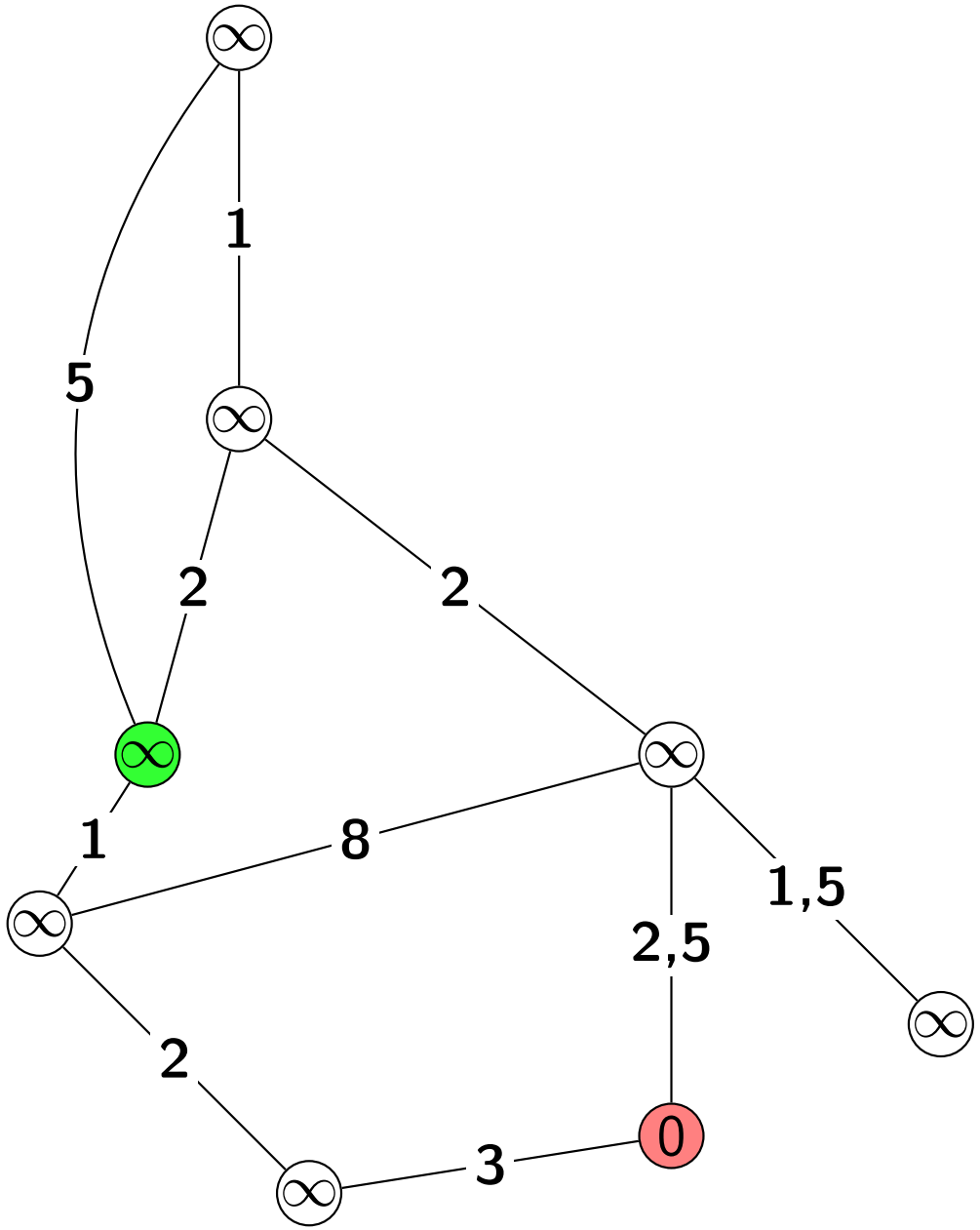
$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

- a tq $D[a]$ min dans P;
- $V \leftarrow a$;
- $P \leftarrow P \setminus \{a\}$;
- **for** b *voisin de* a **do**
  - **if**
    $D[b] > D[a] + W[a, b]$
    then
    $D[b] \leftarrow$
    $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

  a tq $D[a]$ min dans P;

  $V \leftarrow a;$

  $P \leftarrow P \setminus \{a\};$

  for b *voisin de* a **do**

    if

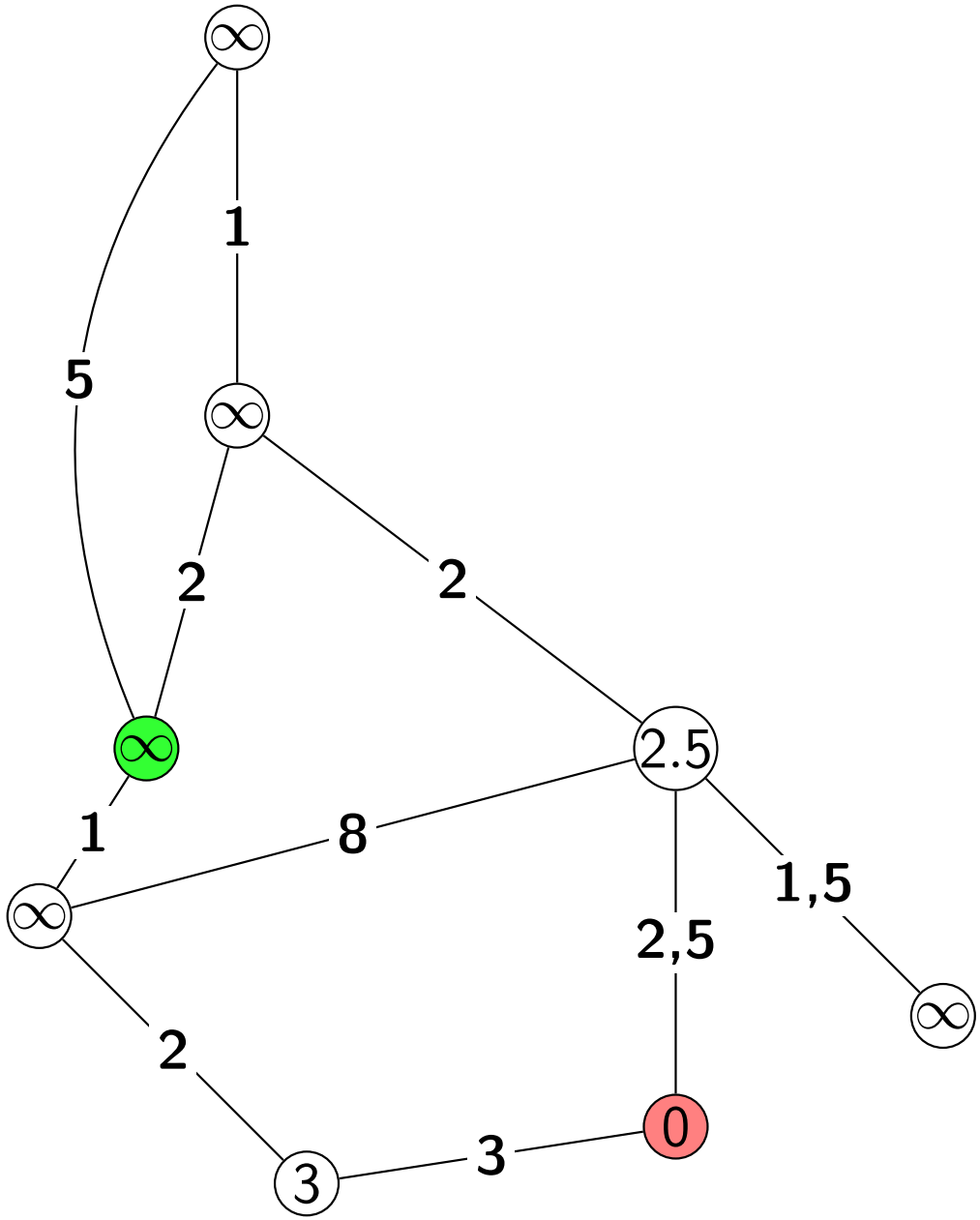
      $D[b] > D[a] + W[a, b]$

      then

        $D[b] \leftarrow$

          $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

  a tq $D[a]$ min dans P;

  $V \leftarrow a;$

  $P \leftarrow P \setminus \{a\};$

  for *b voisin de a* **do**

    if

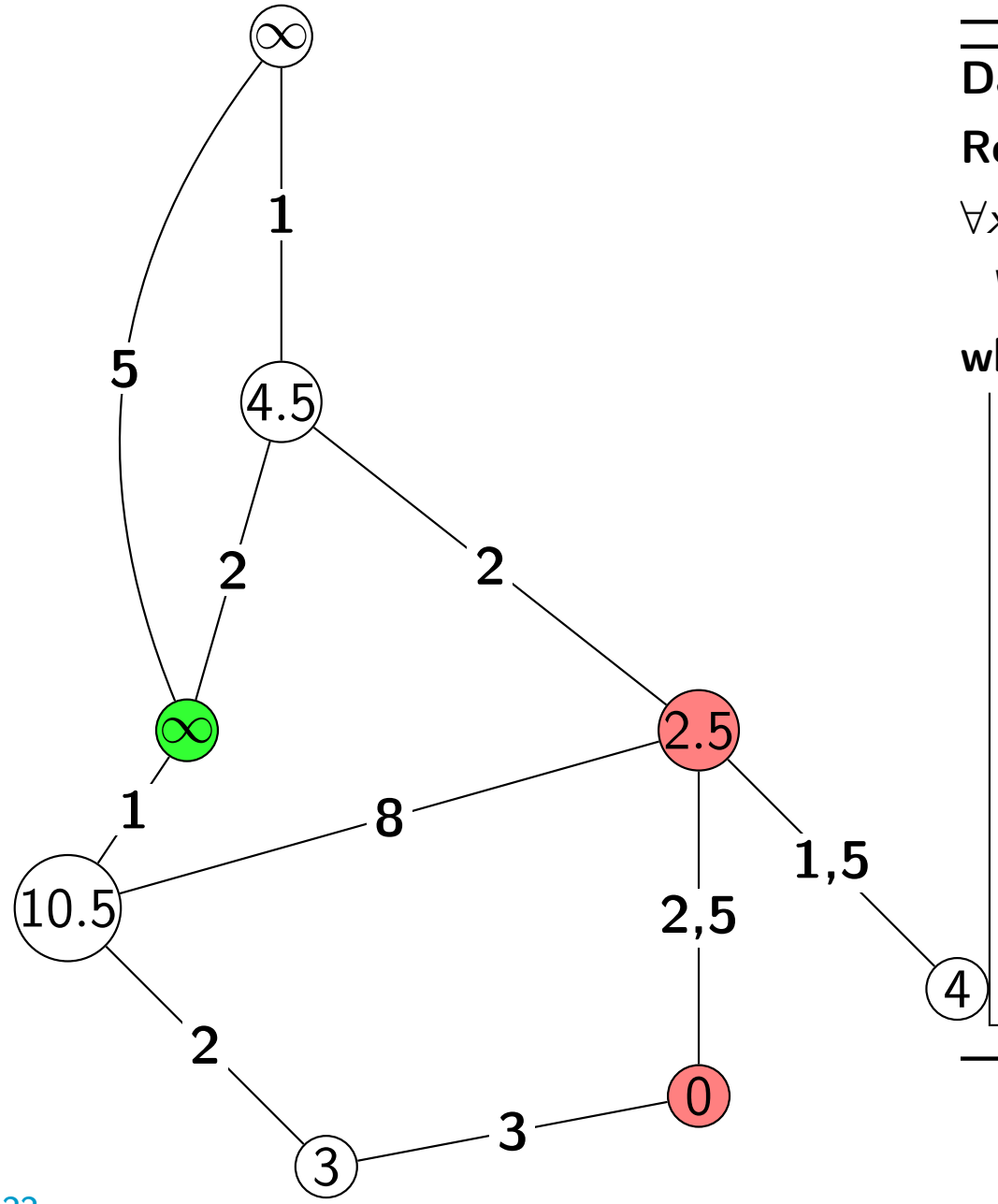
      $D[b] > D[a] + W[a, b]$

      then

        $D[b] \leftarrow$

          $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

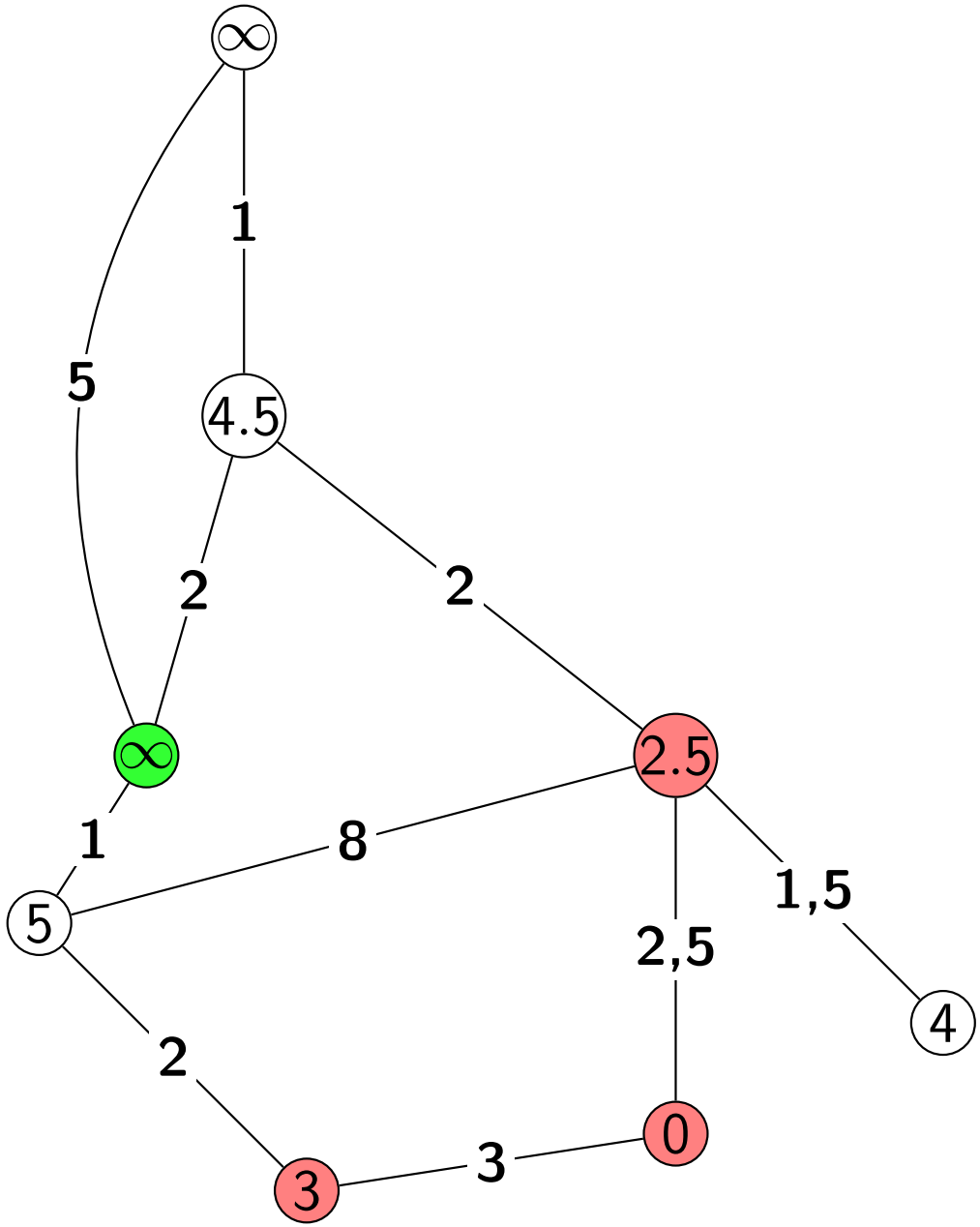
$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

- a tq $D[a]$ min dans P;
- $V \leftarrow a;$
- $P \leftarrow P \setminus \{a\};$
- **for** b *voisin de* a **do**
  - **if**
    $D[b] > D[a] + W[a, b]$
    then
    $D[b] \leftarrow$
    $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

  a tq $D[a]$ min dans P;

  $V \leftarrow a;$

  $P \leftarrow P \setminus \{a\};$

  for *b voisin de a* **do**

    if

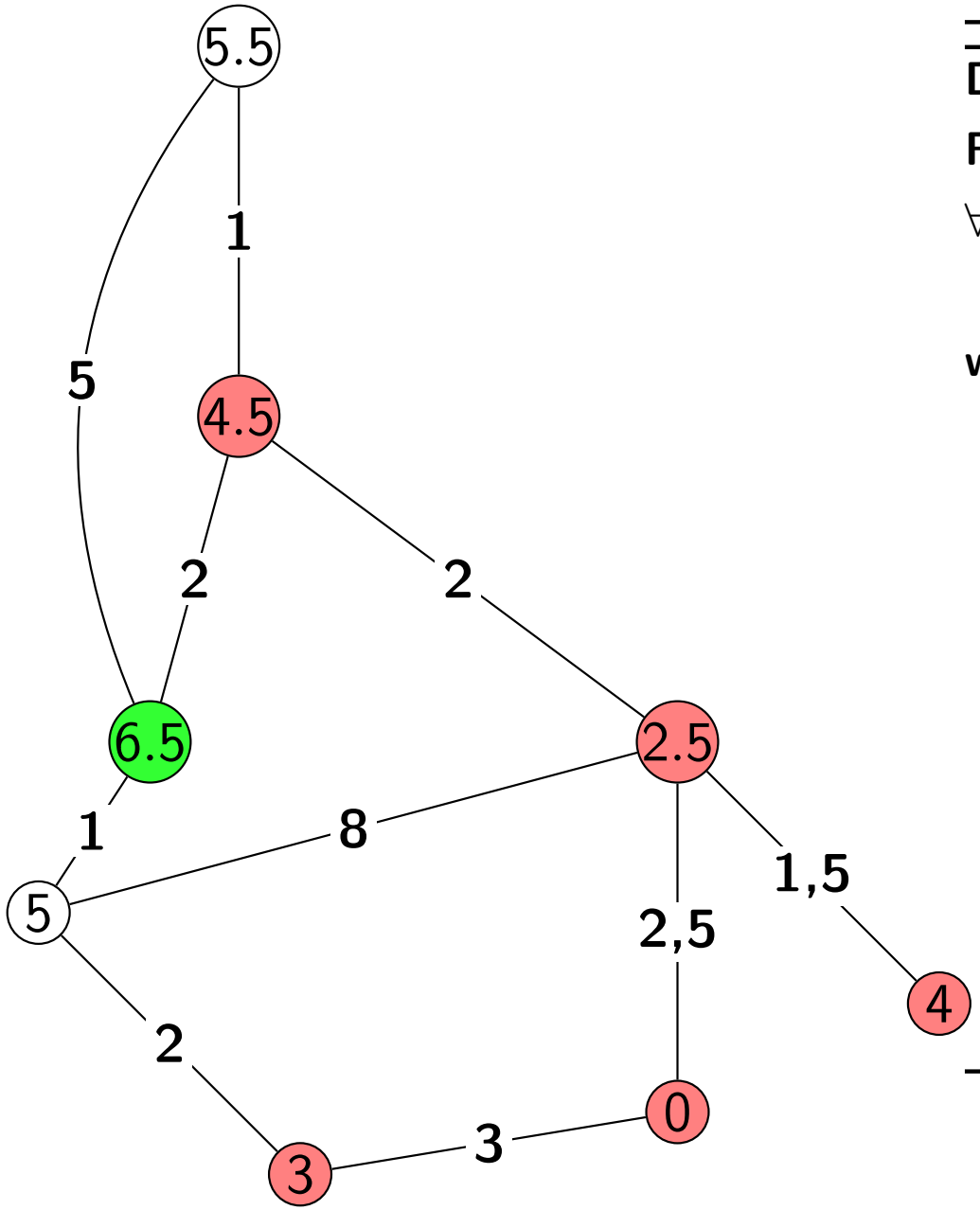
      $D[b] > D[a] + W[a, b]$

      then

        $D[b] \leftarrow$

          $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

  a tq $D[a]$ min dans P;

  $V \leftarrow a;$

  $P \leftarrow P \setminus \{a\};$

  for b *voisin de* a **do**

    if

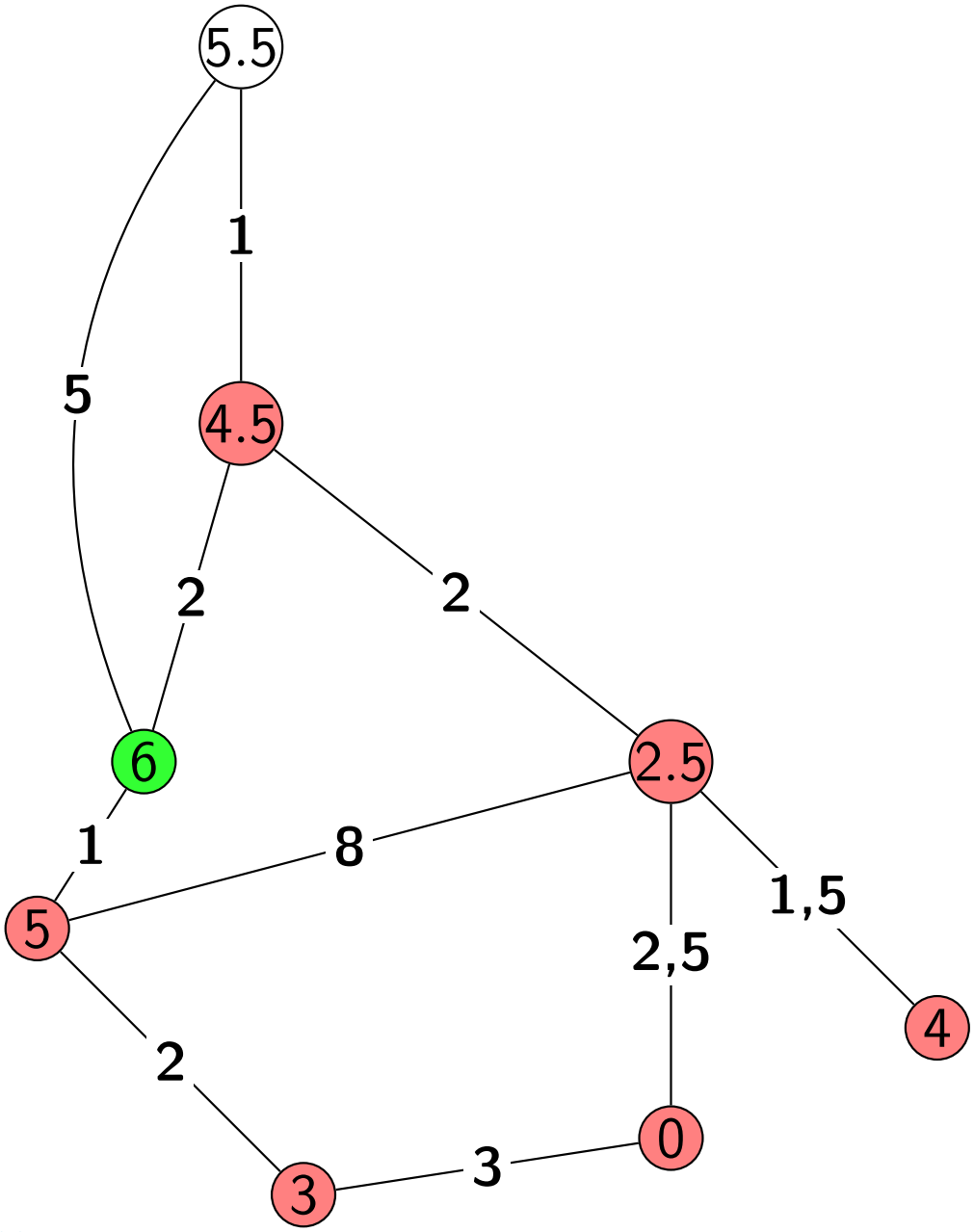
      $D[b] > D[a] + W[a, b]$

      then

        $D[b] \leftarrow$

          $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

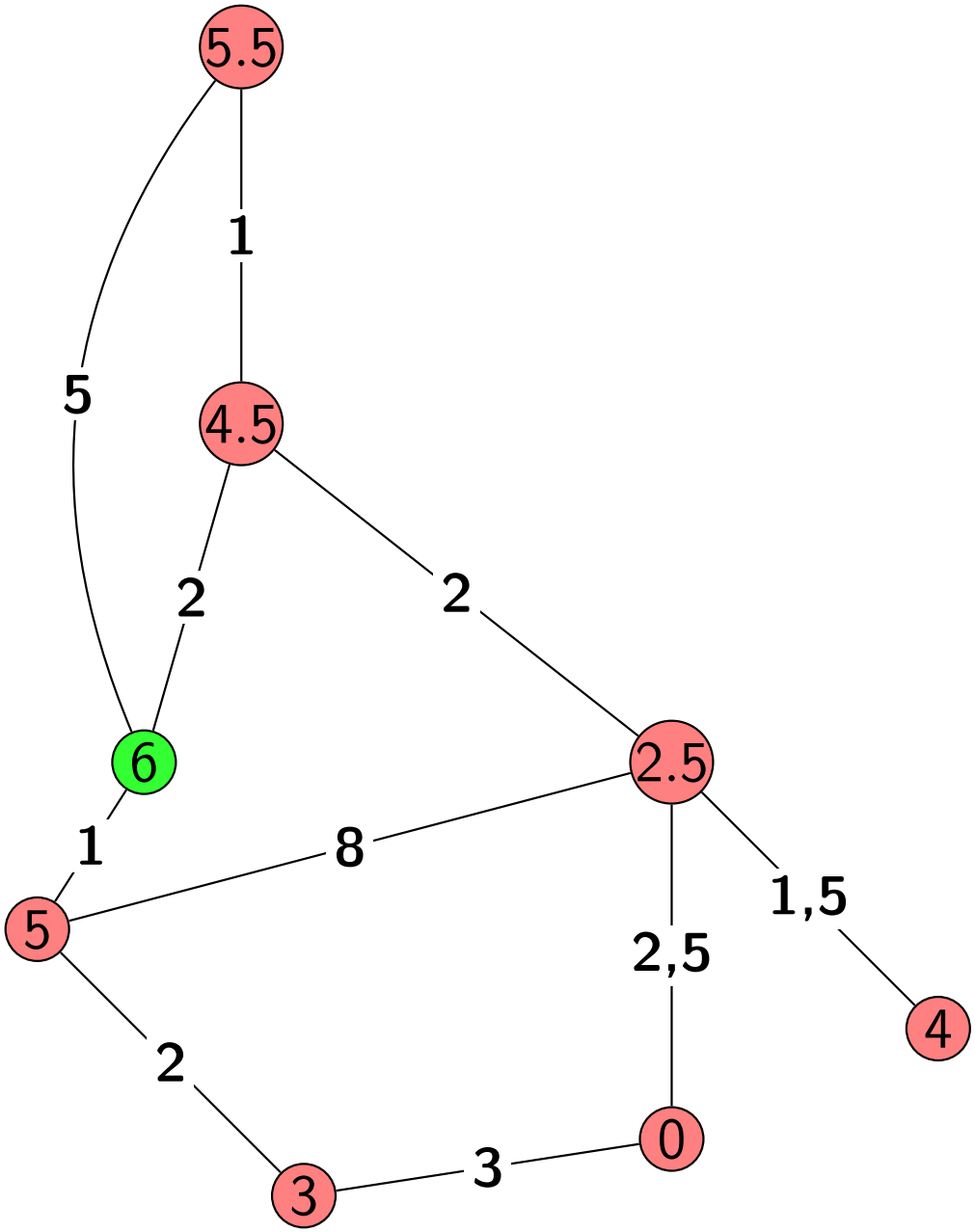
$\forall x \in S, D[x] = \infty \; D[s] = 0$

$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

- a tq $D[a]$ min dans P;
- $V \leftarrow a$;
- $P \leftarrow P \setminus \{a\}$;
- **for** b *voisin de* a **do**
  - **if**
    $D[b] > D[a] + W[a, b]$
    then
    $D[b] \leftarrow$
    $D[a] + W[a, b]$

Dijkstra



Data: graphe $\mathcal{G} = (S, A)$, W, s,t

Result: dist(s,t)

$\forall x \in S, D[x] = \infty \; D[s] = 0$

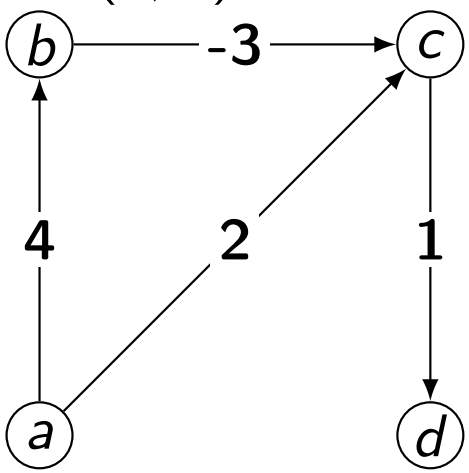
$V = \emptyset \; P = S;$

while $P \neq \emptyset$ **do**

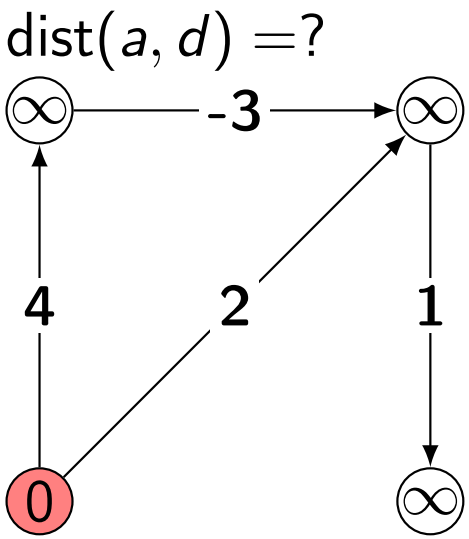
- a tq $D[a]$ min dans P;
- $V \leftarrow a$;
- $P \leftarrow P \setminus \{a\}$;
- **for** b *voisin de* a **do**
  - **if**
    $D[b] > D[a] + W[a, b]$
    then
    $D[b] \leftarrow$
    $D[a] + W[a, b]$

Dijkstra

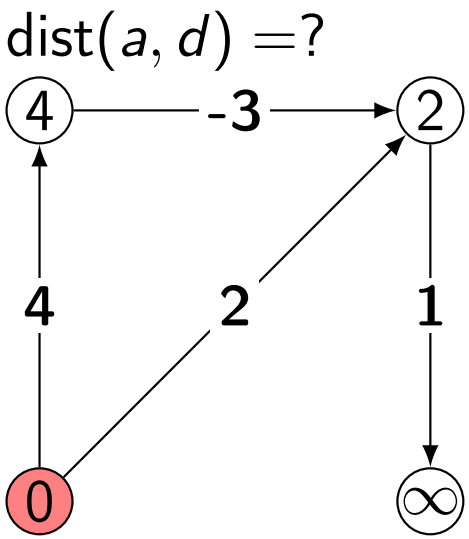
$\text{dist}(a, d) = ?$



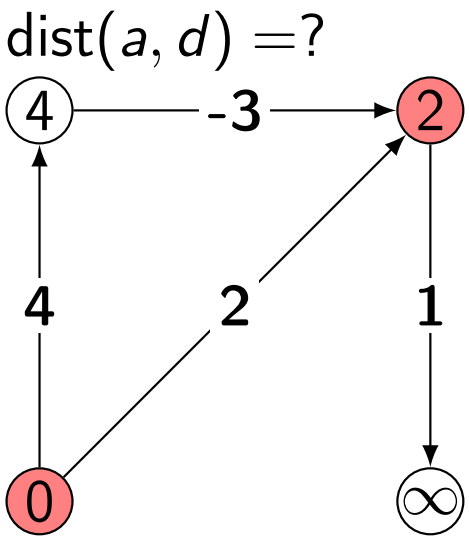
Dijkstra



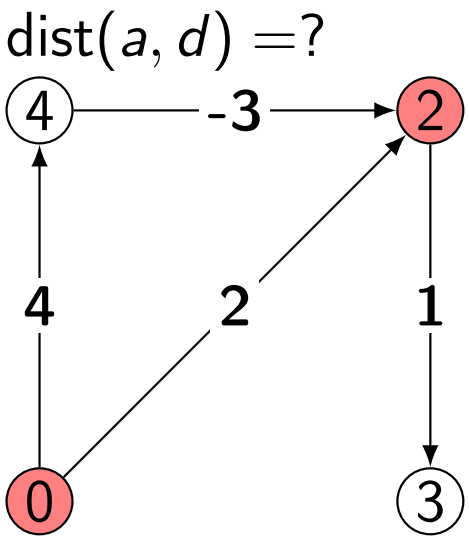
Dijkstra



Dijkstra

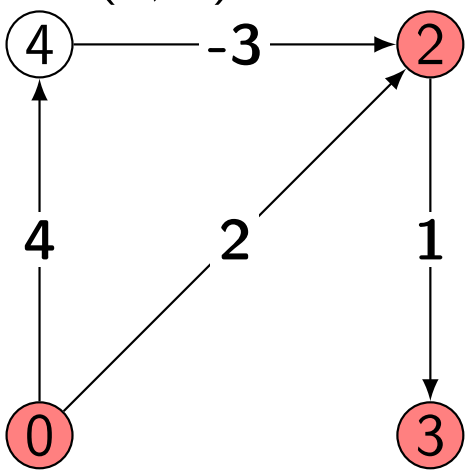


Dijkstra



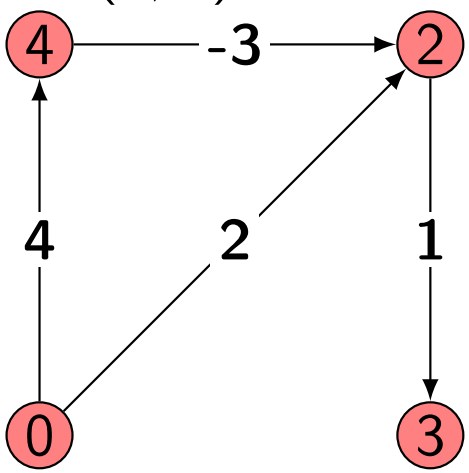
Dijkstra

$\text{dist}(a, d) = ?$



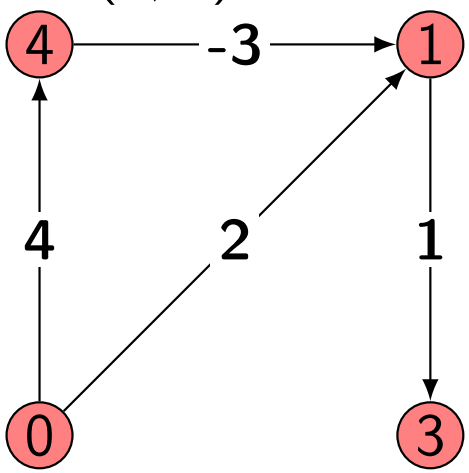
Dijkstra

$\text{dist}(a, d) = ?$



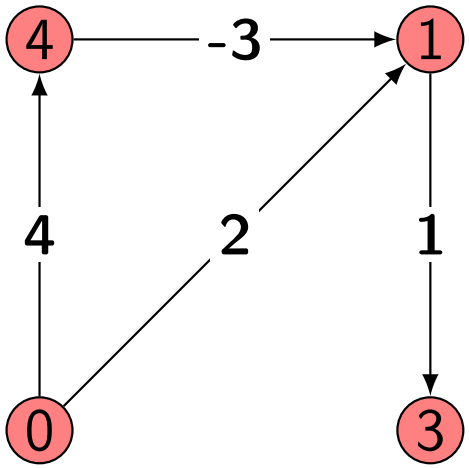
Dijkstra

$\text{dist}(a, d) = ?$



Dijkstra

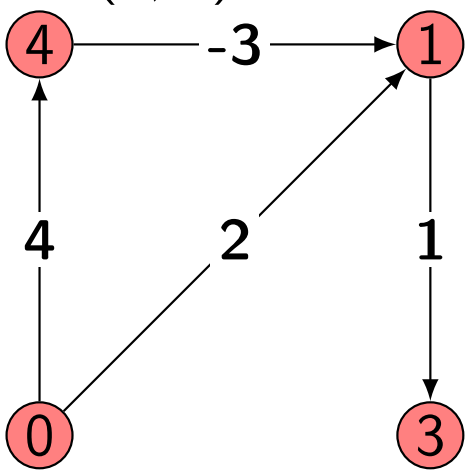
dist$(a, d) = ?$



Dijskra : dist$(a, d) = 3$

Dijkstra

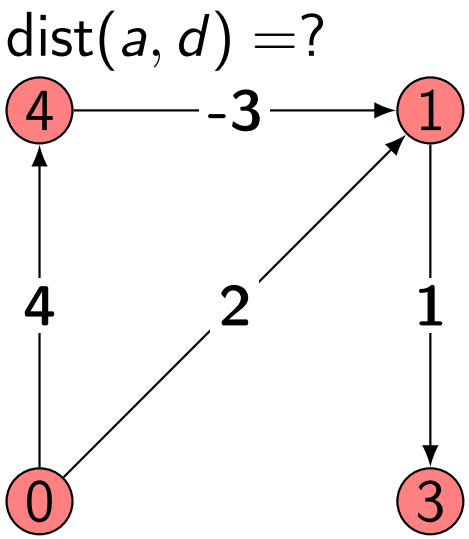
dist(a, d) =?



Dijskra : dist$(a, d) = 3$

Faux : dist$(a, d) = 2$

Dijkstra



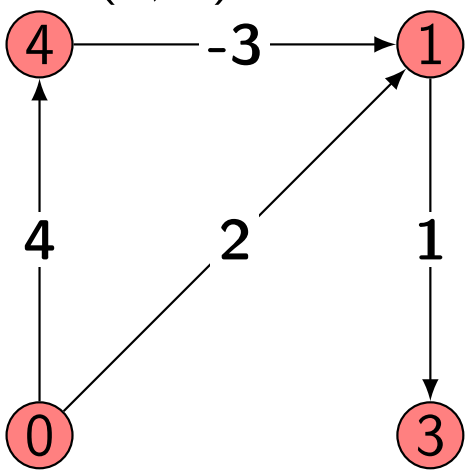
Dijskra : $\text{dist}(a, d) = 3$

Faux : $\text{dist}(a, d) = 2$

Dijskra : seulement poids positifs

Dijkstra

dist$(a, d) =?$



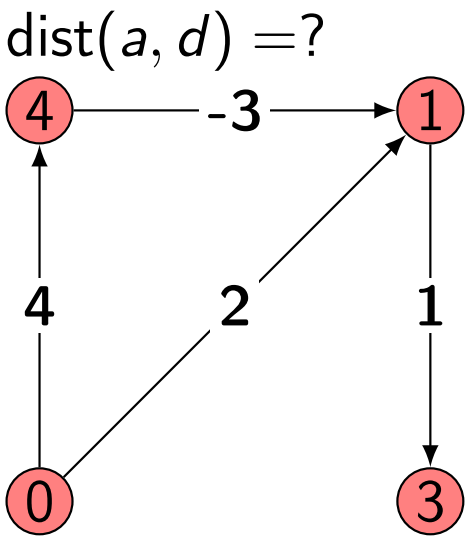
Dijskra : dist$(a, d) = 3$

Faux : dist$(a, d) = 2$

Dijskra : seulement poids positifs

Mais : bonne complexité $\approx |S|^2$

Dijkstra



Dijskra : $\text{dist}(a, d) = 3$

Faux : $\text{dist}(a, d) = 2$

Dijskra : seulement poids positifs

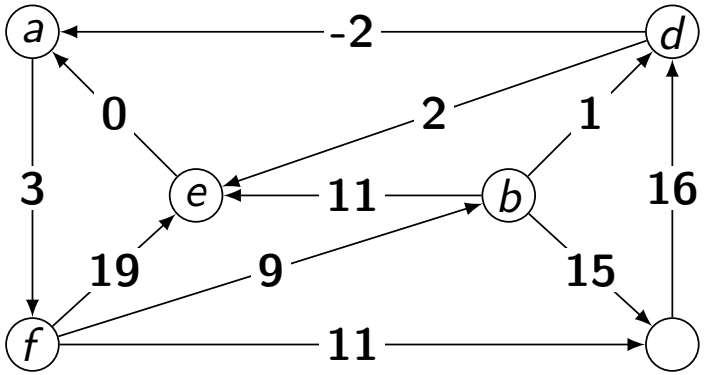
Mais : bonne complexité $\approx |S|^2$

rmq : BFS cas spécial de Disjstra

Plan

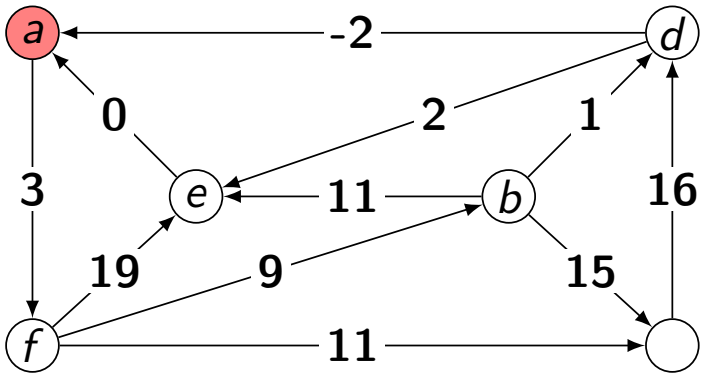
Floyd–Warshall

Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D



Floyd–Warshall

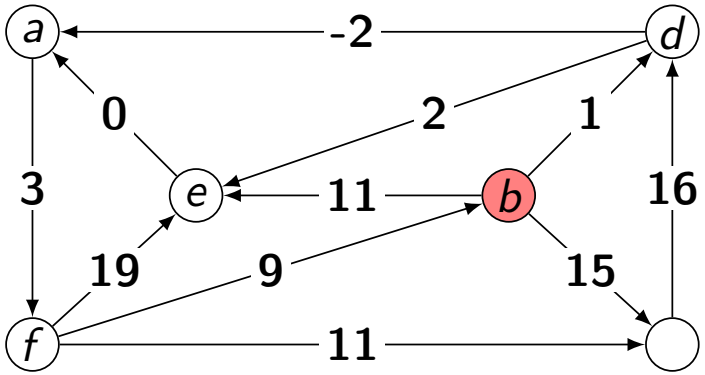
Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D



$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix}$$

Floyd–Warshall

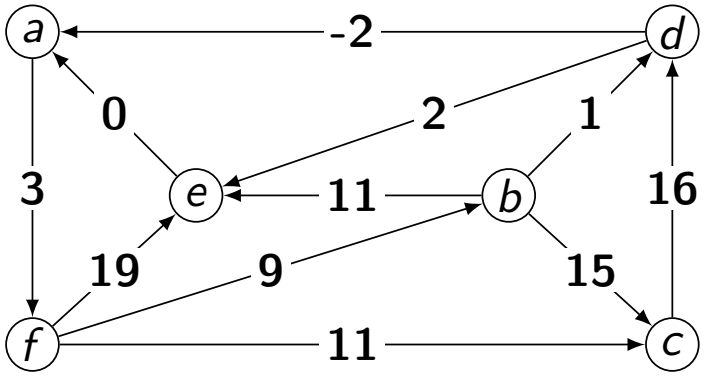
Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D



$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

Floyd–Warshall

Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D

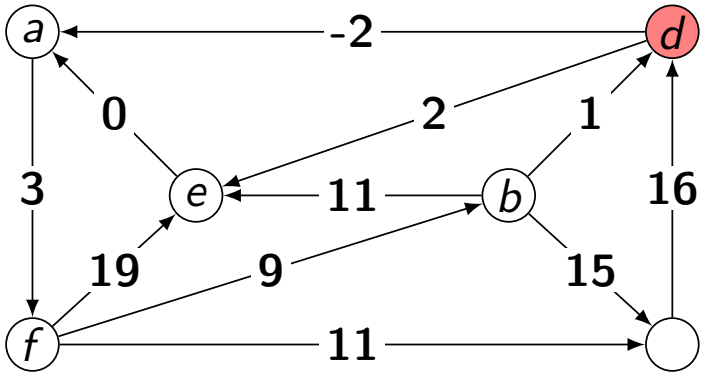


$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

Floyd–Warshall

Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D

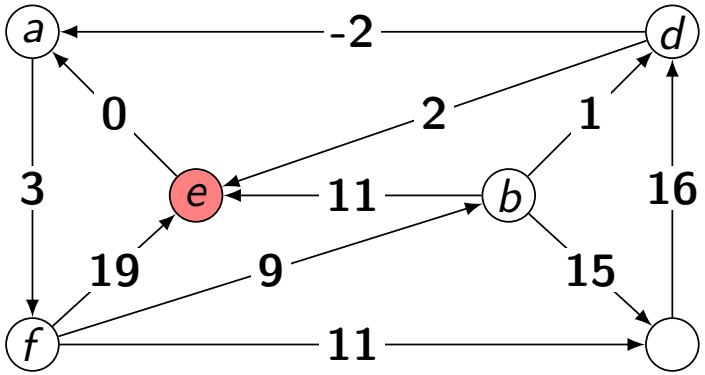


$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ -1 & 0 & 15 & 1 & 3 & 2 \\ 14 & \infty & 0 & 16 & 18 & 17 \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix}$$

Floyd–Warshall

Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D



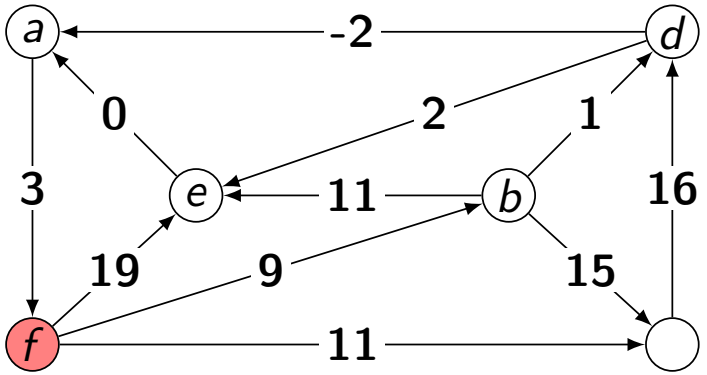
$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ -1 & 0 & 15 & 1 & 3 & 2 \\ 14 & \infty & 0 & 16 & 18 & 17 \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ -1 & 0 & 15 & 1 & 3 & 2 \\ 14 & \infty & 0 & 16 & 18 & 17 \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix}$$

Floyd–Warshall

Data: graphe $\mathcal{G} = (S, A)$, W, s
Result: dist(s,t)
$D = Copy(W)$;
for k *de* 1 *à* n **do**
  $D[k][k] = 0$
for k *de* 1 *à* n **do**
  for s *de* 1 *à* n **do**
    for t *de* 1 *à* n **do**
      $D[s][t] = \min(D[s][t], D[s][k] + D[k][t])$
return D



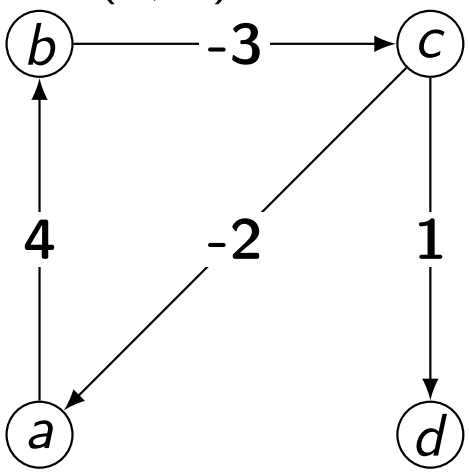
$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & \infty & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 0 & 15 & 1 & \infty & \infty \\ \infty & \infty & 0 & 16 & \infty & \infty \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ \infty & 9 & 11 & 10 & 19 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ -1 & 0 & 15 & 1 & 3 & 2 \\ 14 & \infty & 0 & 16 & 18 & 17 \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \infty & \infty & \infty & \infty & 3 \\ -1 & 0 & 15 & 1 & 3 & 2 \\ 14 & \infty & 0 & 16 & 18 & 17 \\ -2 & \infty & \infty & 0 & 2 & 1 \\ 0 & \infty & \infty & \infty & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 12 & 14 & 13 & 15 & 3 \\ -1 & 0 & 13 & 1 & 3 & 2 \\ 14 & 26 & 0 & 16 & 18 & 17 \\ -2 & 10 & 12 & 0 & 2 & 1 \\ 0 & 12 & 14 & 13 & 0 & 3 \\ 8 & 9 & 11 & 10 & 12 & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

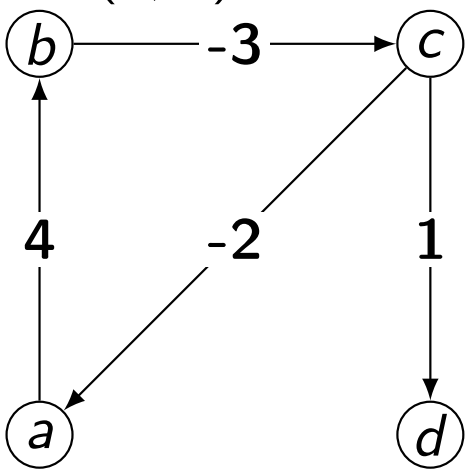


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

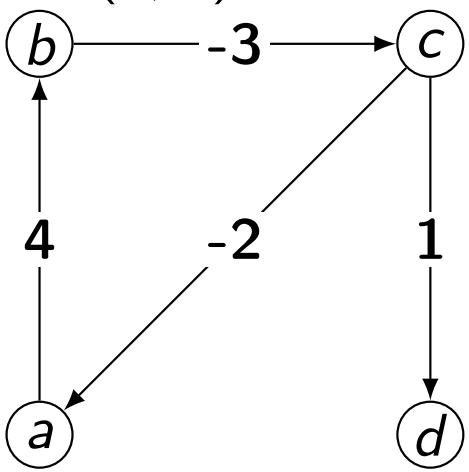


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

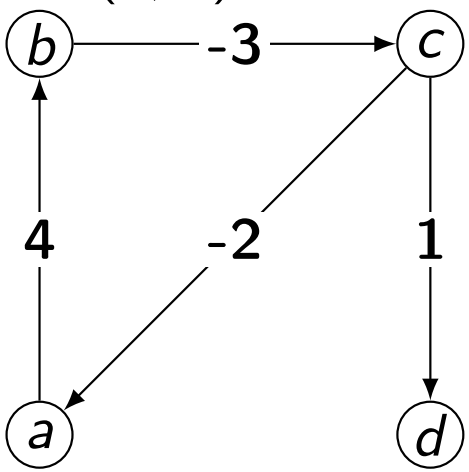


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

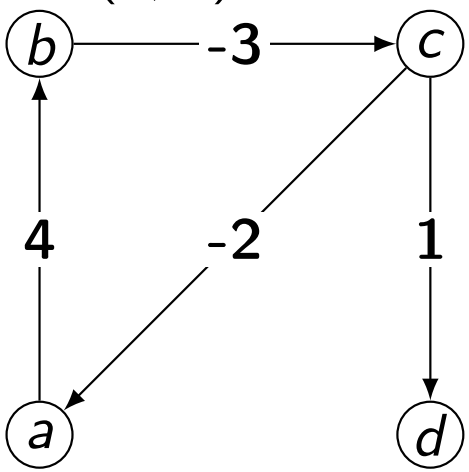


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

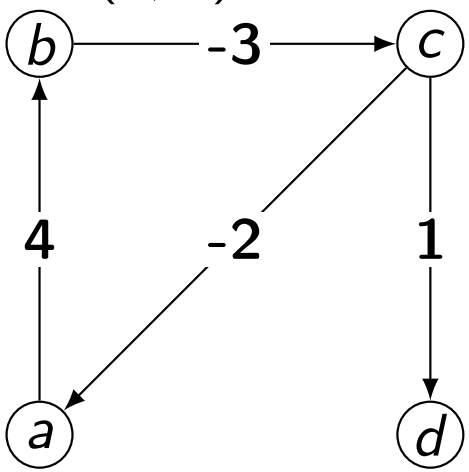


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

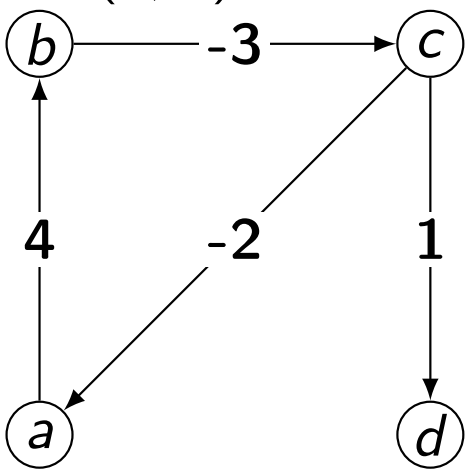


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

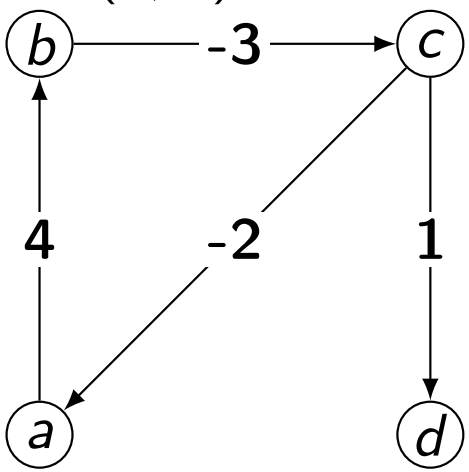


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

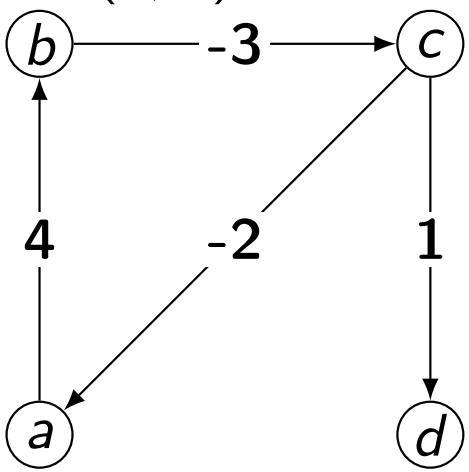


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$

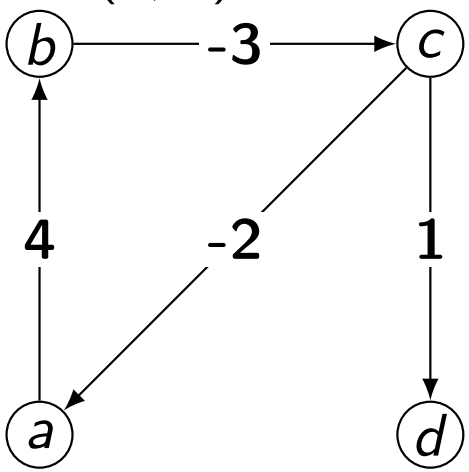


$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\mathrm{dist}(a, d) = ?$



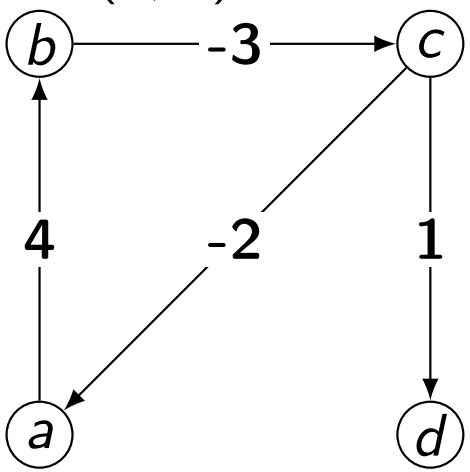
$\mathrm{dist}(a, d) \to -\infty$

$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Floyd–Warshall

$\text{dist}(a, d) = ?$



$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

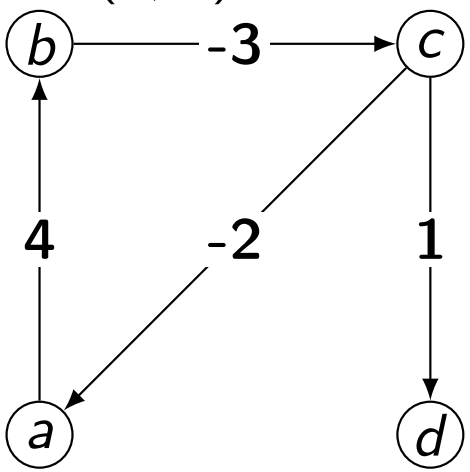
$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$\text{dist}(a, d) \to -\infty$

Il y a un circuit de poids négatif si après exécution $\exists s, \quad D[s][s] < 0$

Floyd–Warshall

$\text{dist}(a, d) = ?$



$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

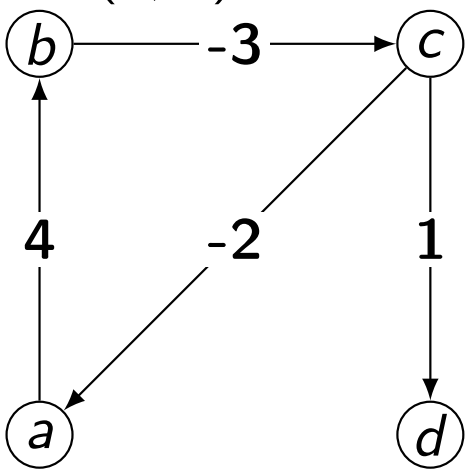
$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$\text{dist}(a, d) \to -\infty$

Il y a un circuit de poids négatif si après exécution $\exists s, \quad D[s][s] < 0$

Floyd–Warshall

$\text{dist}(a, d) = ?$



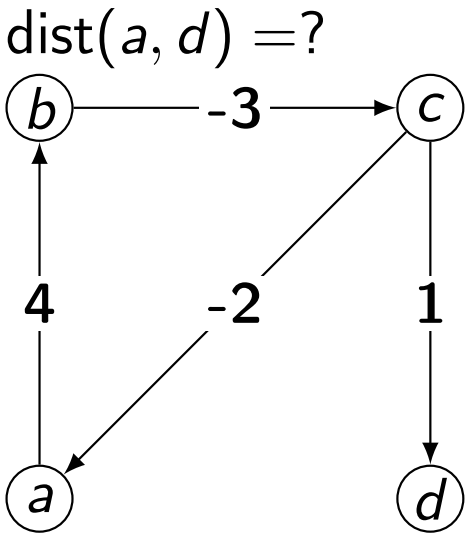
$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$\text{dist}(a, d) \to -\infty$

Il y a un circuit de poids négatif si après exécution $\exists s, \quad D[s][s] < 0$

Floyd–Warshall



$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

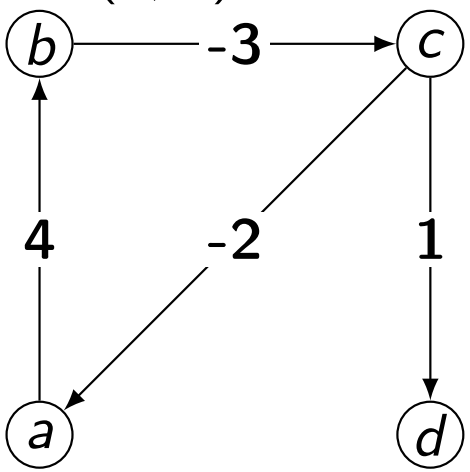
$\text{dist}(a, d) \to -\infty$

Il y a un circuit de poids négatif si après exécution $\exists s, \quad D[s][s] < 0$

Mais : moins bonne complexité $\approx |S|^3$

Floyd–Warshall

dist$(a, d) = ?$



$$\begin{pmatrix} 0 & 4 & \infty & \infty \\ \infty & 0 & -3 & 0 \\ -2 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 4 & 1 & 4 \\ \infty & 0 & -3 & 0 \\ -2 & 2 & -1 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 3 & 0 & 1 \\ -5 & -1 & -4 & -3 \\ -3 & 1 & -2 & -1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

dist$(a, d) \to -\infty$

Il y a un circuit de poids négatif si après exécution $\exists s, \quad D[s][s] < 0$

Mais : moins bonne complexité $\approx |S|^3$

Il existe un plus court chemin si et seulement ssi il n'y a pas de circuit de poids < 0

Plan

Arbres couvrants

En 1926, Otakar Borůvka doit rendre le réseau électrique de Moravie efficace : il doit relier toutes les maisons pour un coût aussi bas que possible.

Arbres couvrants

En 1926, Otakar Borůvka doit rendre le réseau électrique de Moravie efficace : il doit relier toutes les maisons pour un coût aussi bas que possible.

Soit $\mathcal{G} = (S, A)$ un graphe. Un arbre couvrant (*spanning tree*) est un graphe partiel $\mathcal{T} = (S, A')$ qui est un arbre.

Arbres couvrants

En 1926, Otakar Borůvka doit rendre le réseau électrique de Moravie efficace : il doit relier toutes les maisons pour un coût aussi bas que possible.

Soit $\mathcal{G} = (S, A)$ un graphe. Un arbre couvrant (*spanning tree*) est un graphe partiel $\mathcal{T} = (S, A')$ qui est un arbre.

Lemme

Pour tout graphe non-orienté connexe, il existe un arbre couvrant

Arbres couvrants

En 1926, Otakar Borůvka doit rendre le réseau électrique de Moravie efficace : il doit relier toutes les maisons pour un coût aussi bas que possible.

Soit $\mathcal{G} = (S, A)$ un graphe. Un arbre couvrant (*spanning tree*) est un graphe partiel $\mathcal{T} = (S, A')$ qui est un arbre.

Lemme

Pour tout graphe non-orienté connexe, il existe un arbre couvrant

preuve : on fait un parcours de graphe

Plan

Kruskal

Data: graphe G=(S,A) (non-orienté)

Result: arbre T

$n = \text{length}(S)$;

$L = sort(A)$; (*tri des arêtes par ordre croissant de leur poids*)

$E := \emptyset \qquad T := \emptyset$;

while $|E| < n - 1$ **do**

    uv=head(L);

    (*soit uv l'arête suivante dans l'ordre des poids*) ;

    if *uv n'induit pas de cycle* **then**

        $T := T \bigcup \{u, v\}$;

        $E := E \bigcup \{uv\}$;

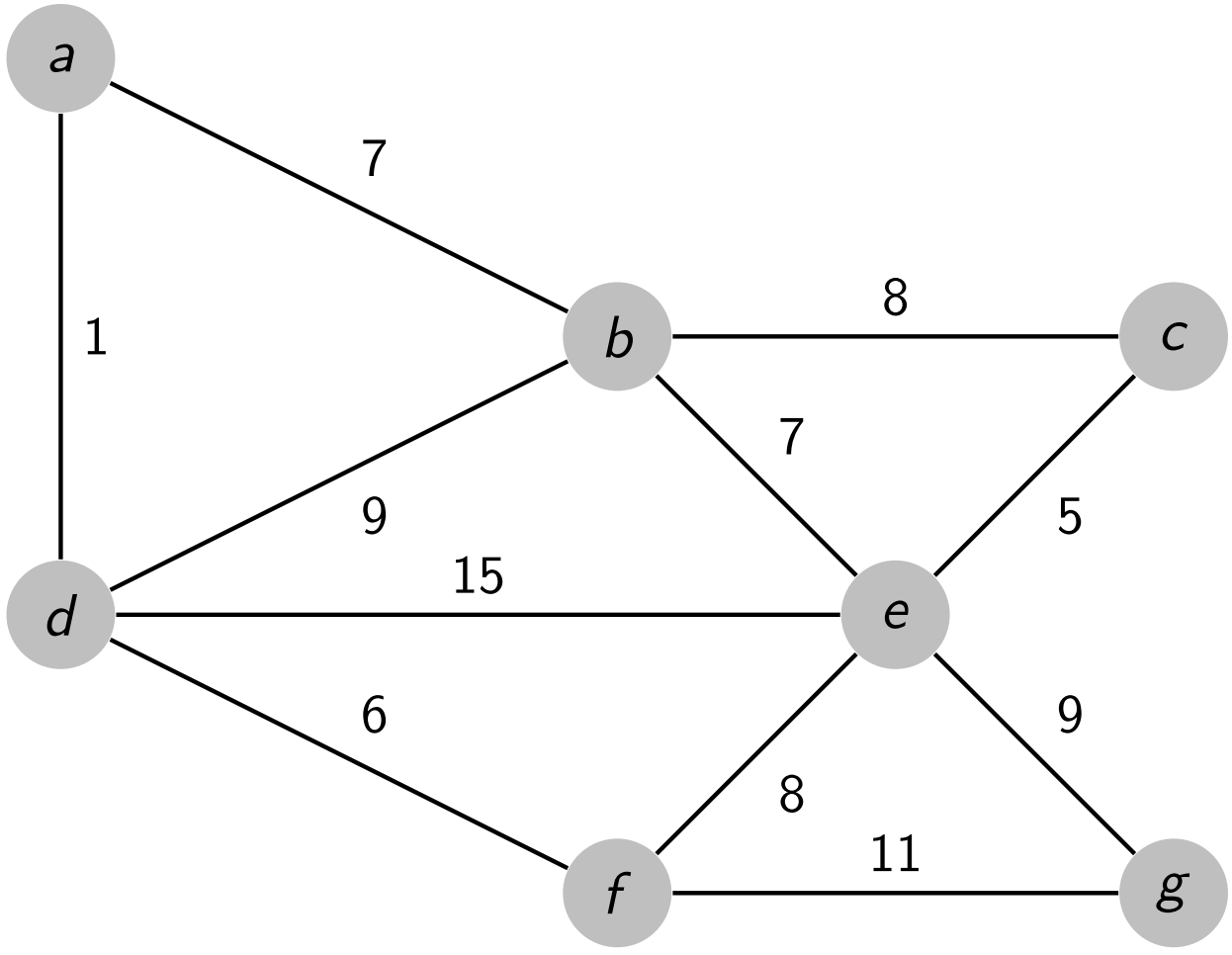
    L=tail(L) ; (* on supprime l'arête de la liste*)

Kruskal

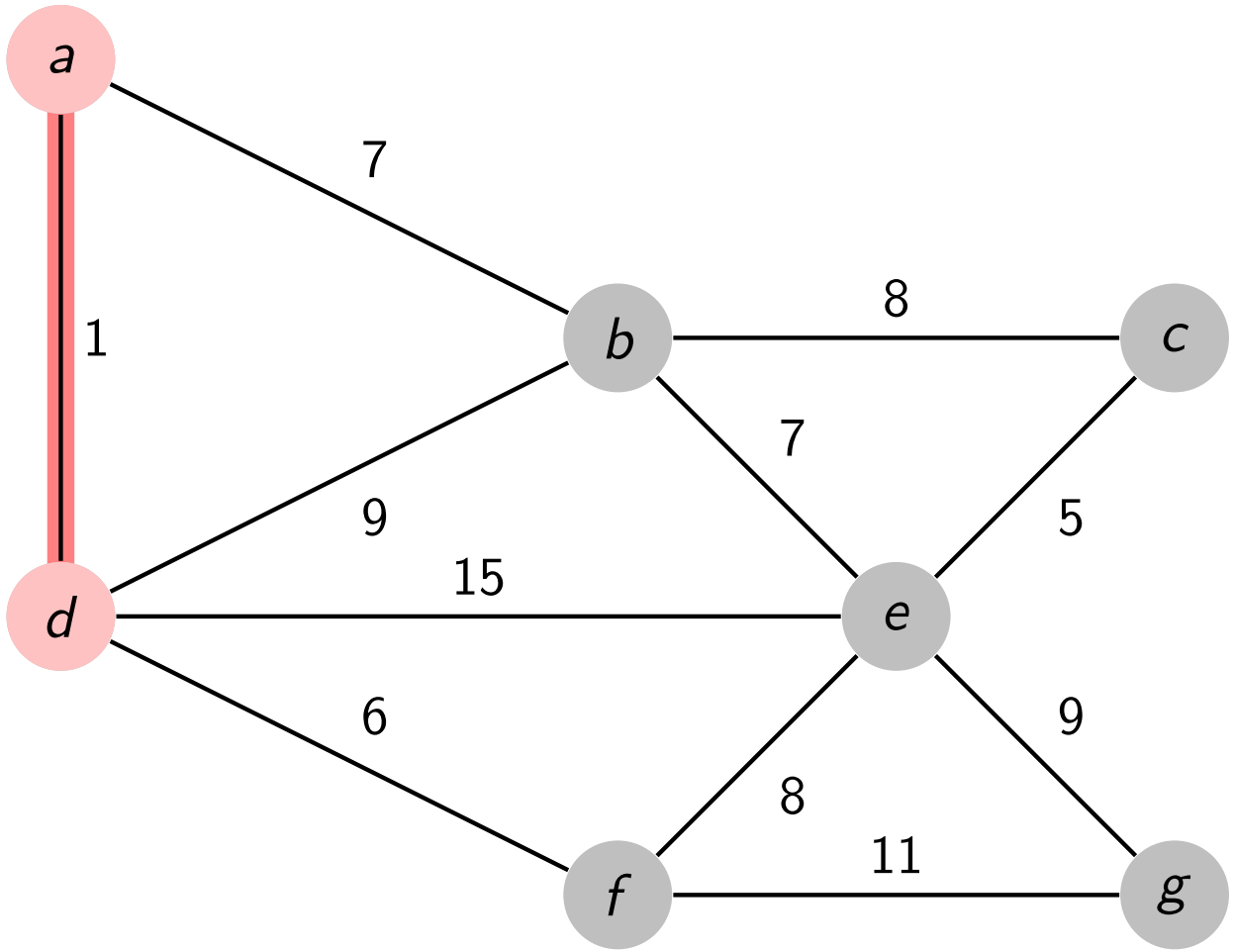
Data: graphe G=(S,A) (non-orienté)
Result: arbre T
$n = \text{length}(S)$;
$L = sort(A)$; (*tri des arêtes par ordre croissant de leur poids*)
$E := \emptyset \qquad T := \emptyset$;
while $|E| < n - 1$ **do**
    uv=head(L);
    (*soit uv l'arête suivante dans l'ordre des poids*) ;
    if *uv n'induit pas de cycle* **then**
        $T := T \bigcup \{u, v\}$;
        $E := E \bigcup \{uv\}$;
    L=tail(L) ; (* on supprime l'arête de la liste*)

Complexité $\approx |A| \log |A|$

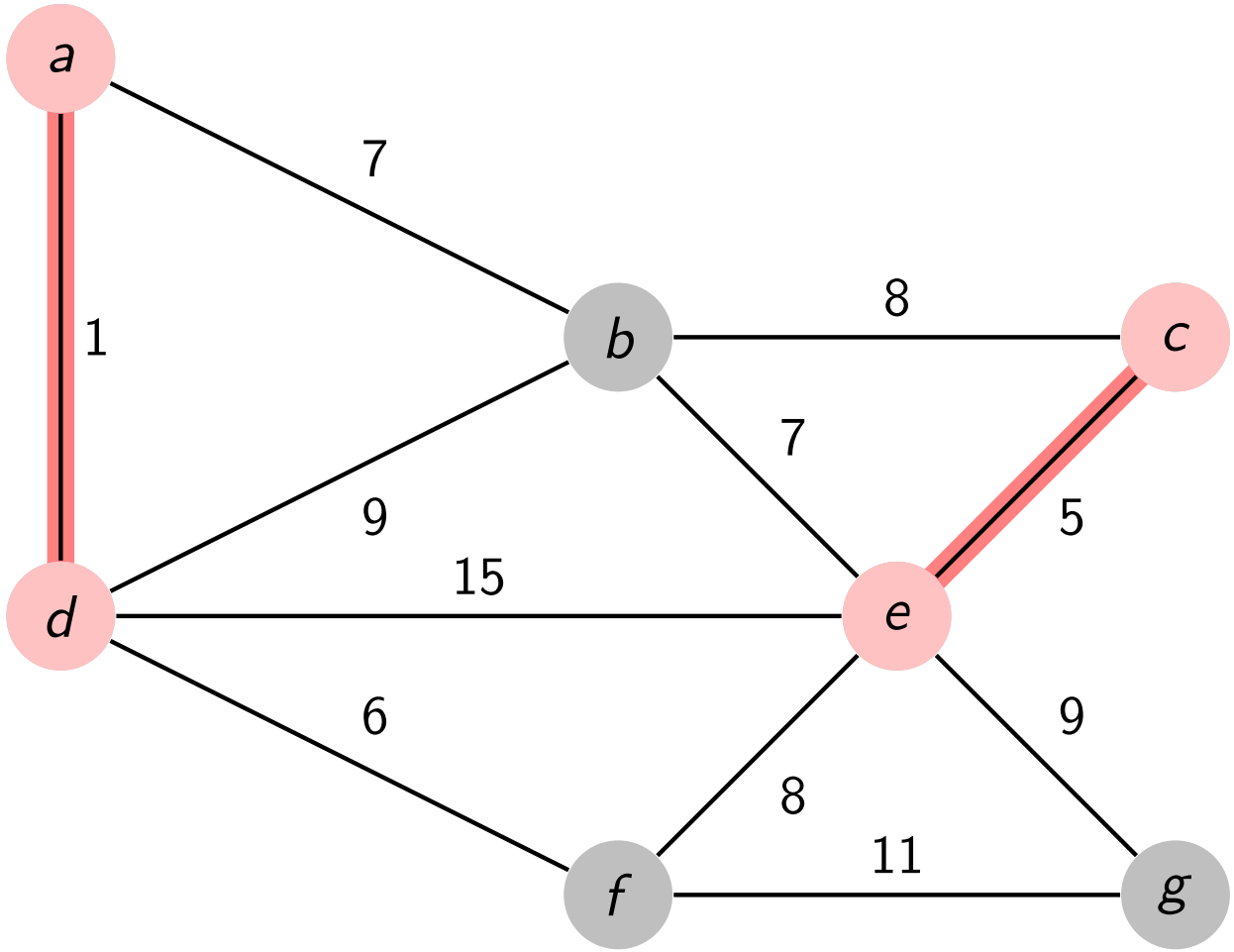
Kruskal's algorithm



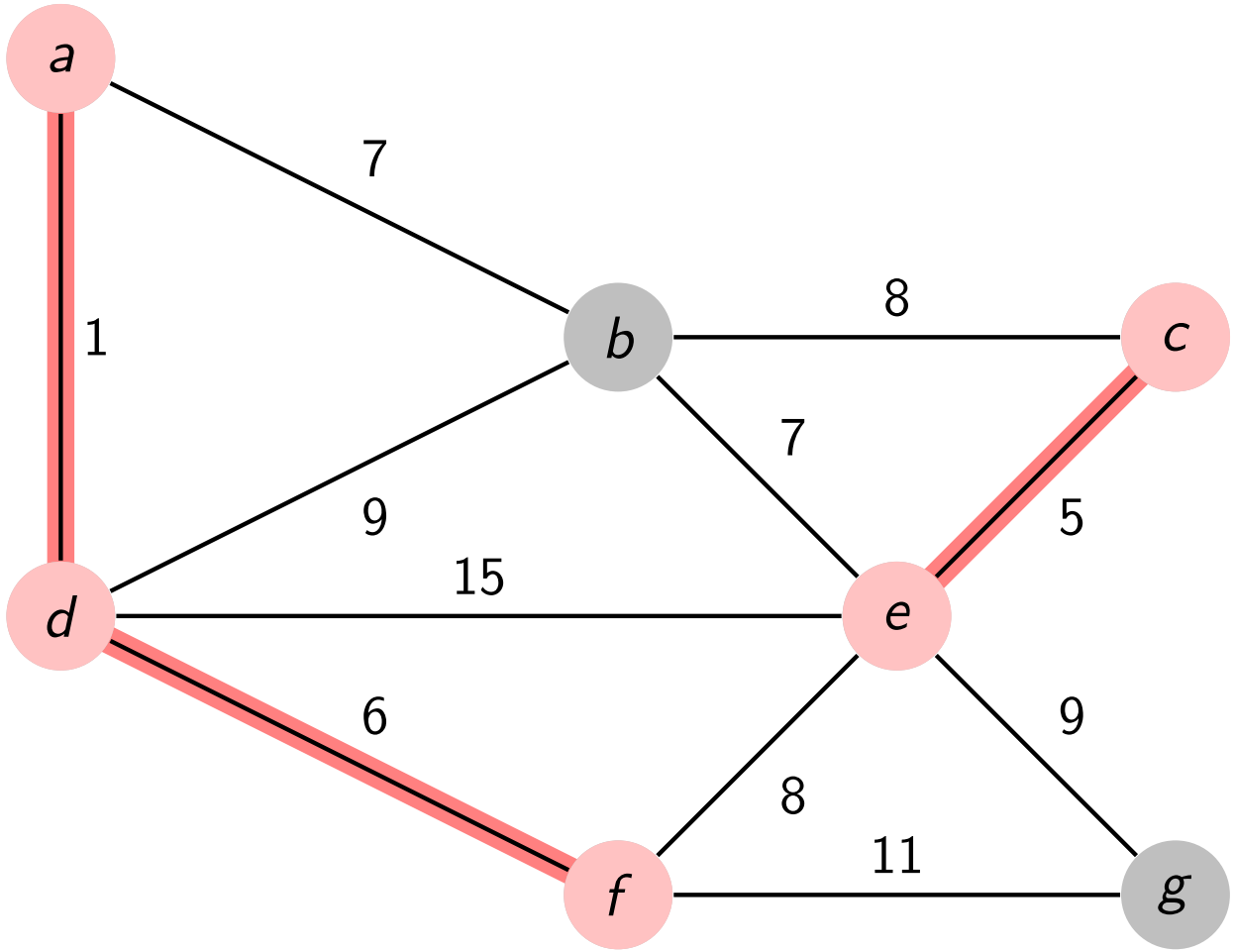
Kruskal's algorithm



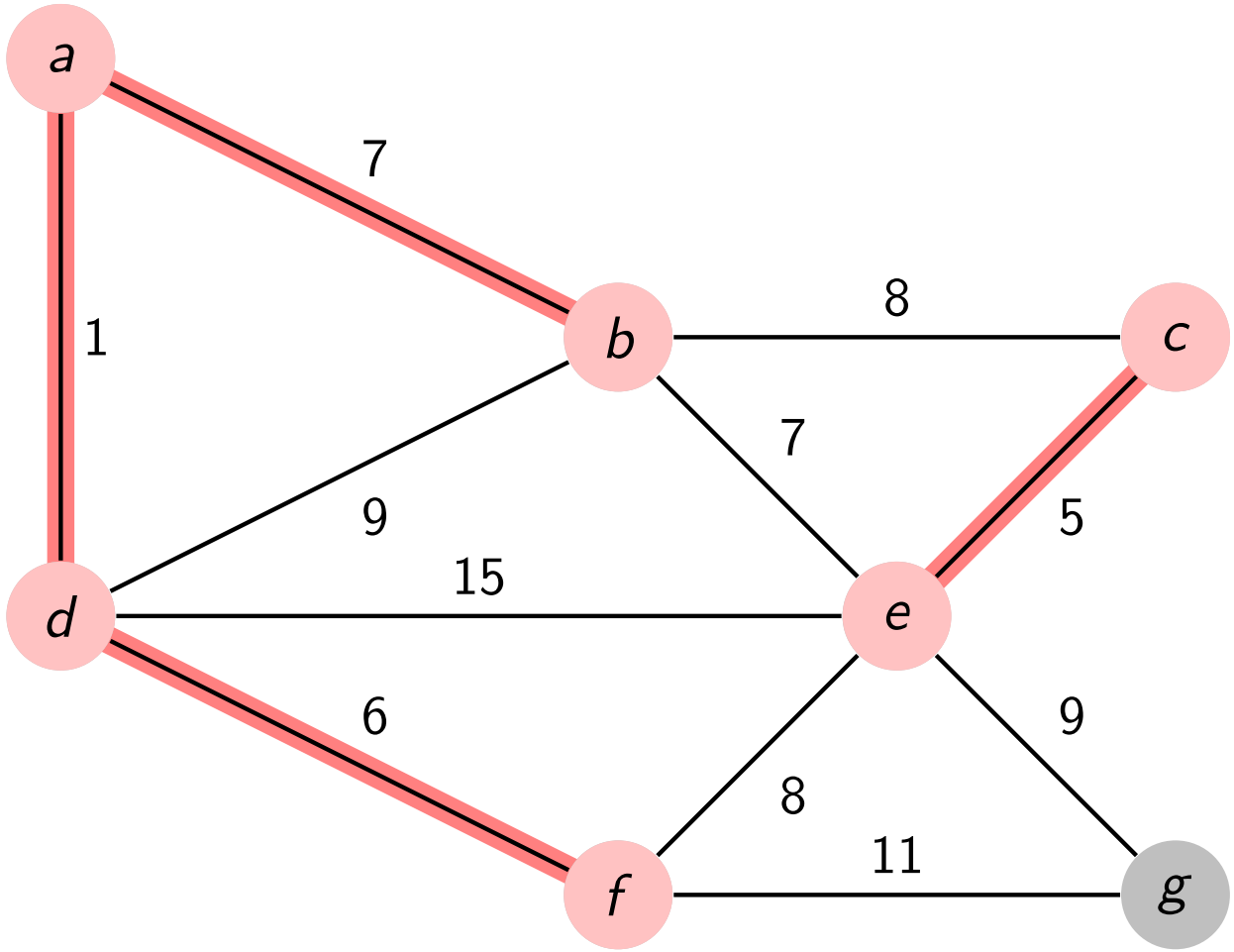
Kruskal's algorithm



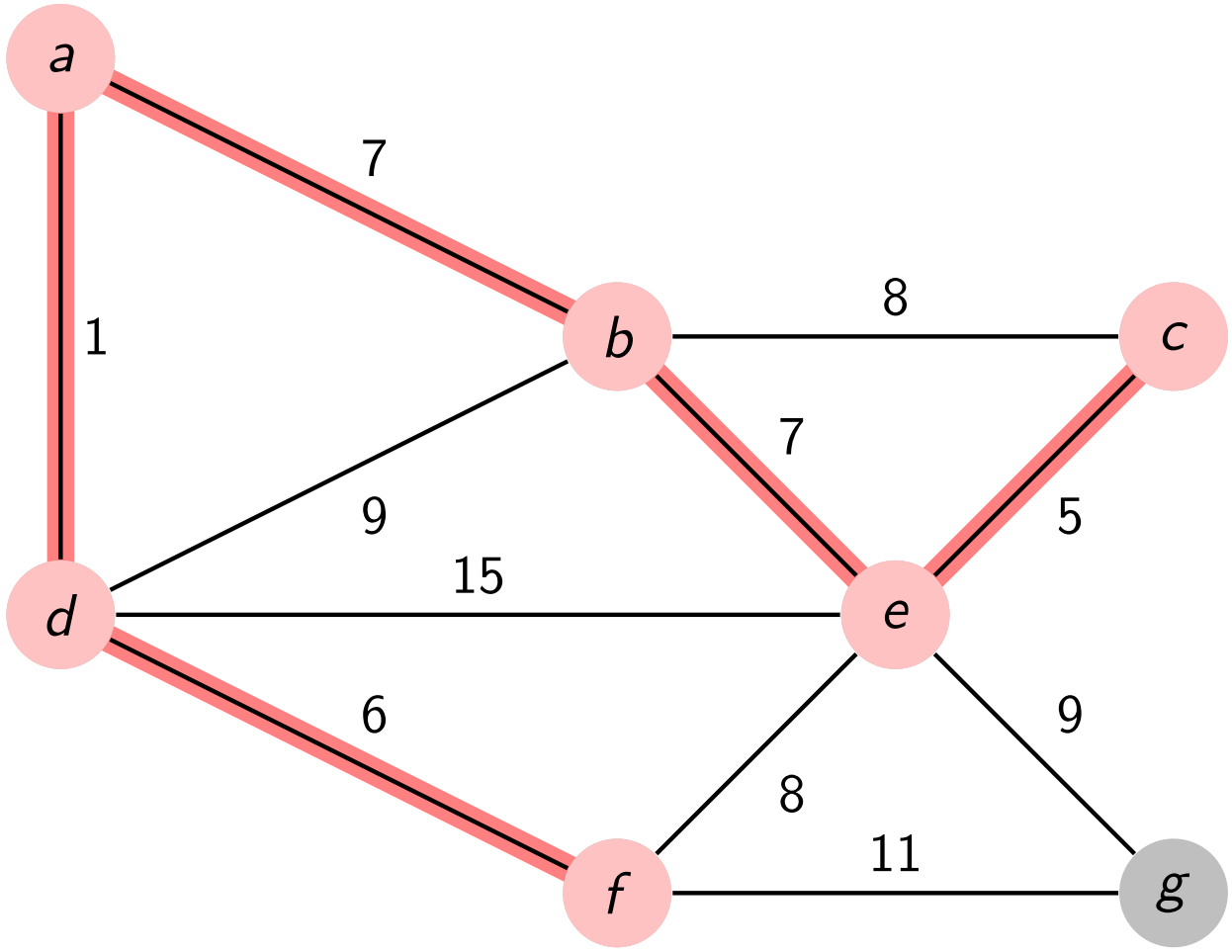
Kruskal's algorithm



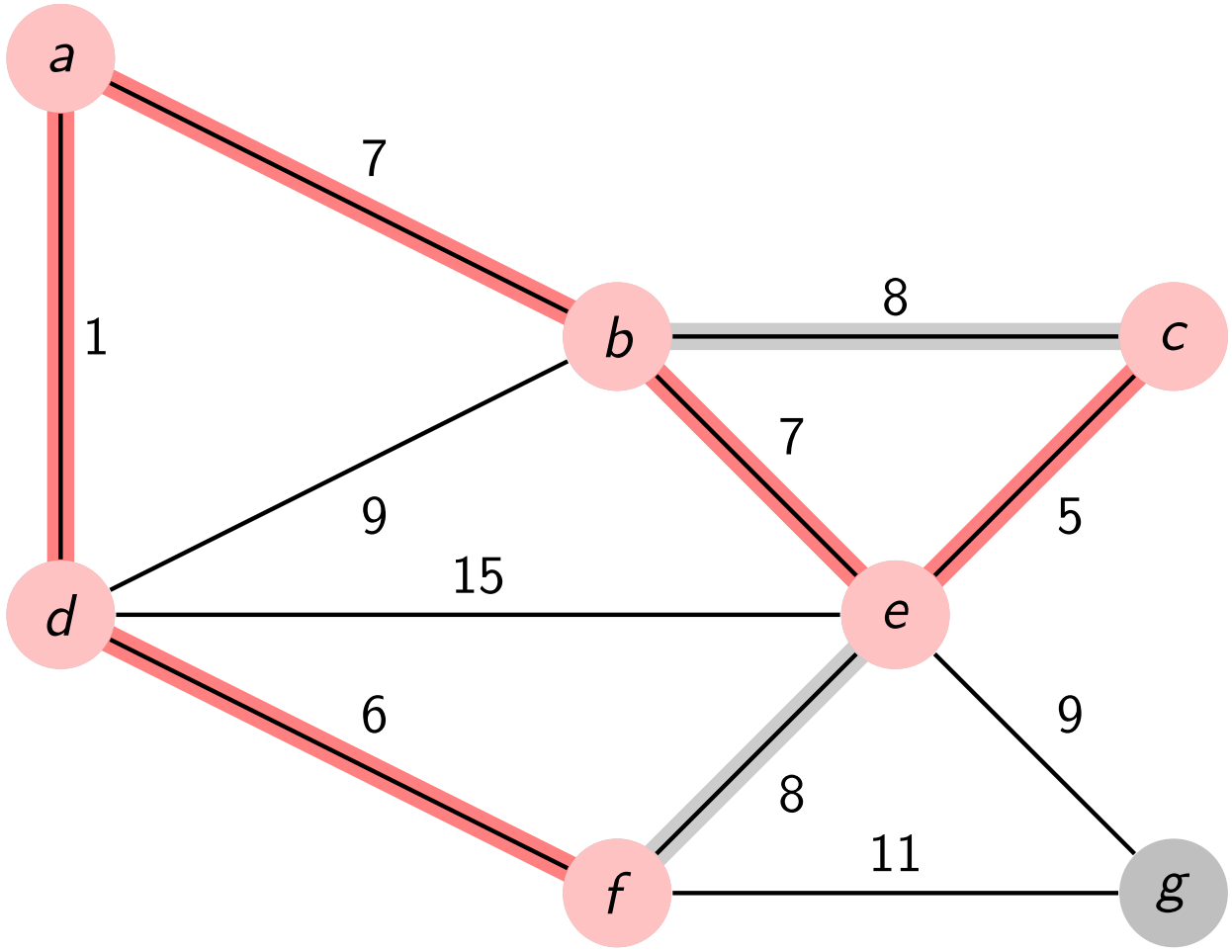
Kruskal's algorithm



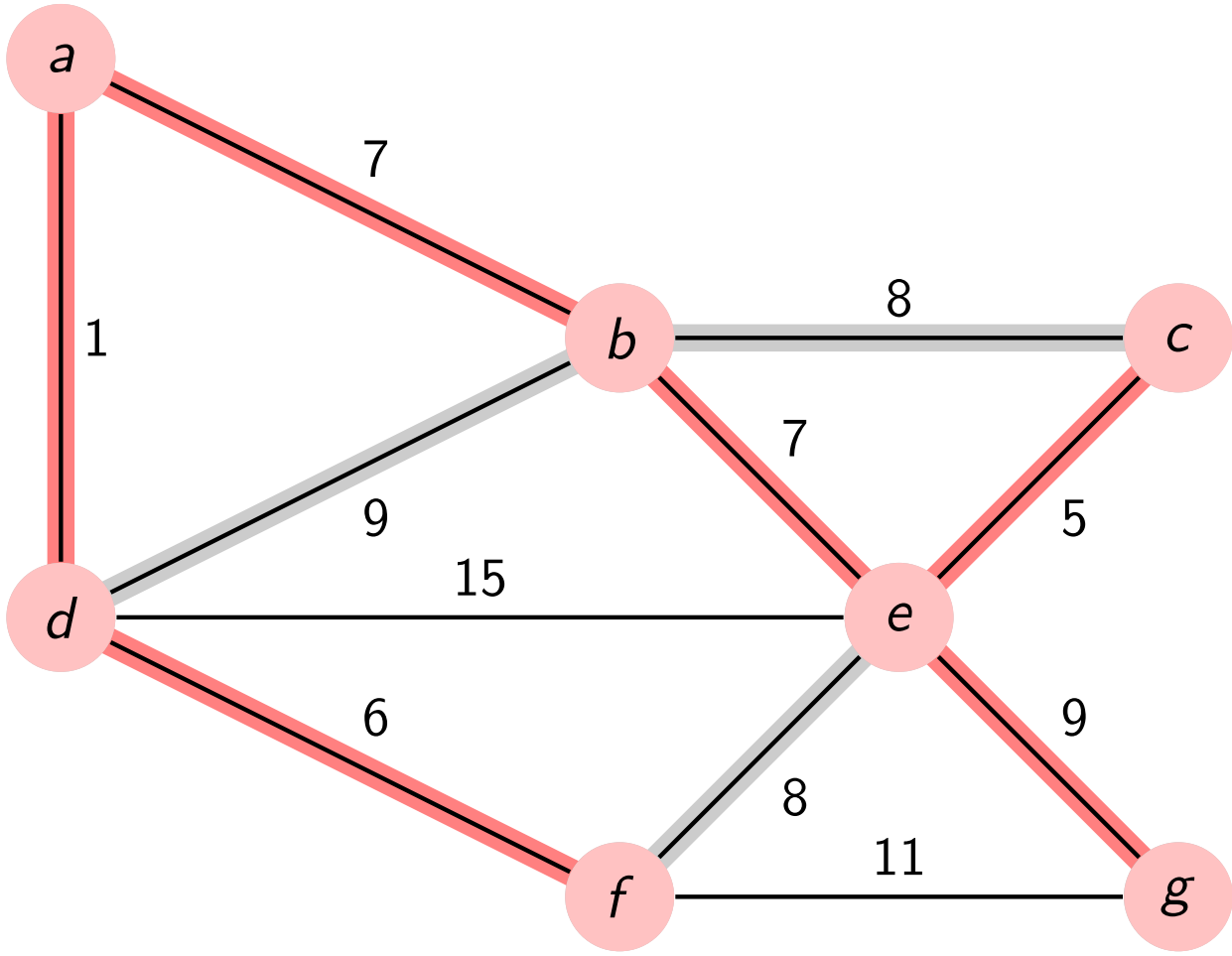
Kruskal's algorithm



Kruskal's algorithm



Kruskal's algorithm



Plan

Prim

Data: graphe $\mathcal{G} = (S, A)$
(non-orienté)

Result: arbre T

$T := \{r\}$ $E := \emptyset$

while Card $T \neq n$*(*tant qu'on a pas tous les sommets*)* **do**

choisir une arête dans A telle que $uv \in A, u \in T, v \notin T, w(uv) = \min_A\{w(ab), a \in T, b \in S \setminus T, ab \in A\}$;

$T := T \bigcup \{v\}$;

$E := E \bigcup \{uv\}$;

Prim

Data: graphe $\mathcal{G} = (S, A)$
(non-orienté)
Result: arbre T
$T := \{r\}$ $E := \emptyset$
while Card $T \neq n$*(*tant qu'on a pas tous les sommets*)* **do**
  choisir une arête dans A
    telle que $uv \in A, u \in T, v \notin T, w(uv) = \min_A\{w(ab), a \in T, b \in S \setminus T, ab \in A\}$;
  $T := T \bigcup \{v\}$;
  $E := E \bigcup \{uv\}$;

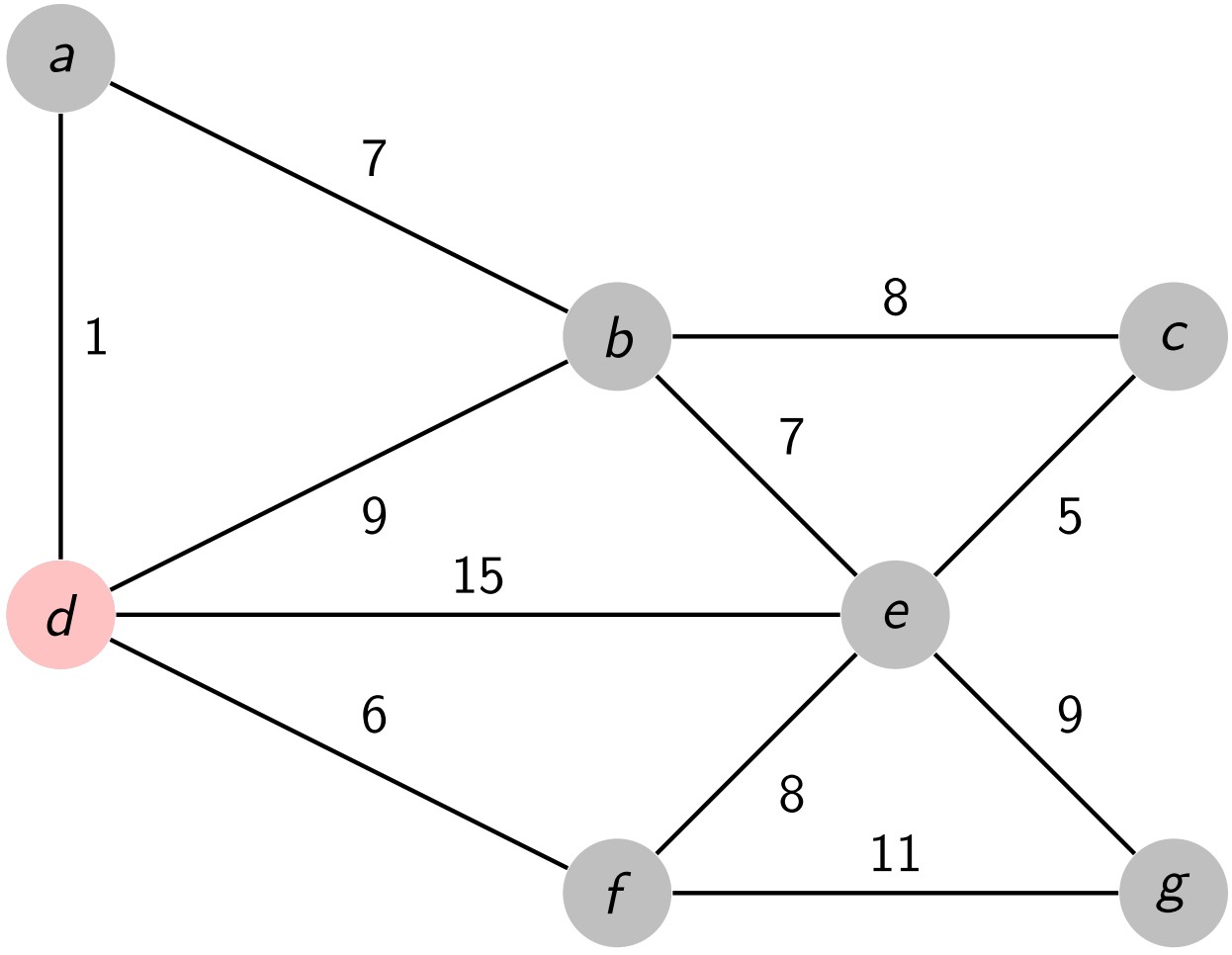
Complexité $\approx |S|^2$

Prim

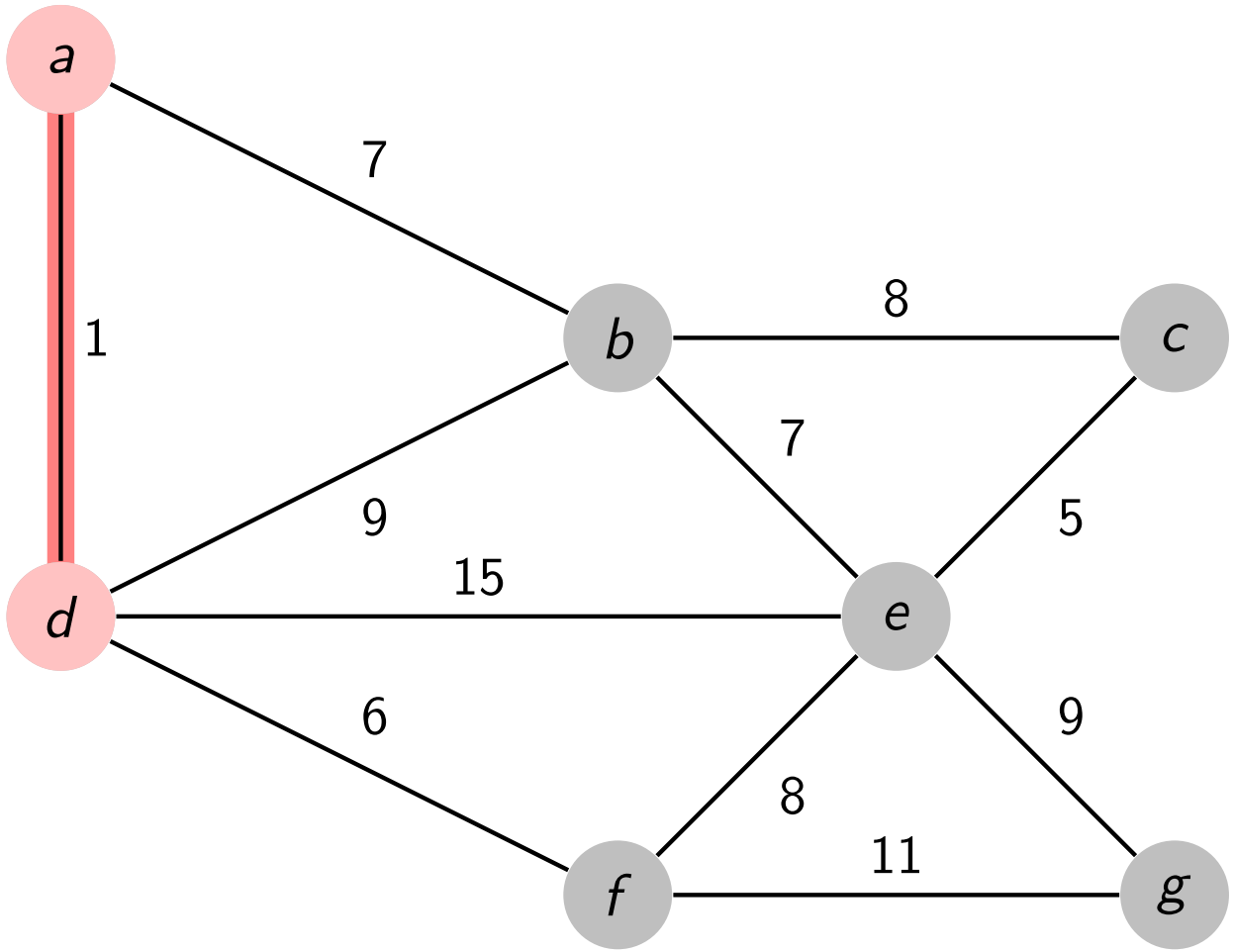
Data: graphe $\mathcal{G} = (S, A)$
(non-orienté)
Result: arbre T
$T := \{r\}$ $E := \emptyset$
while Card $T \neq n$*(*tant qu'on a pas tous les sommets*)* **do**
  choisir une arête dans A
    telle que $uv \in A, u \in T, v \notin T, w(uv) = \min_A\{w(ab), a \in T, b \in S \setminus T, ab \in A\}$;
  $T := T \bigcup \{v\}$;
  $E := E \bigcup \{uv\}$;

Complexité $\approx |S|^2$ reste connexe

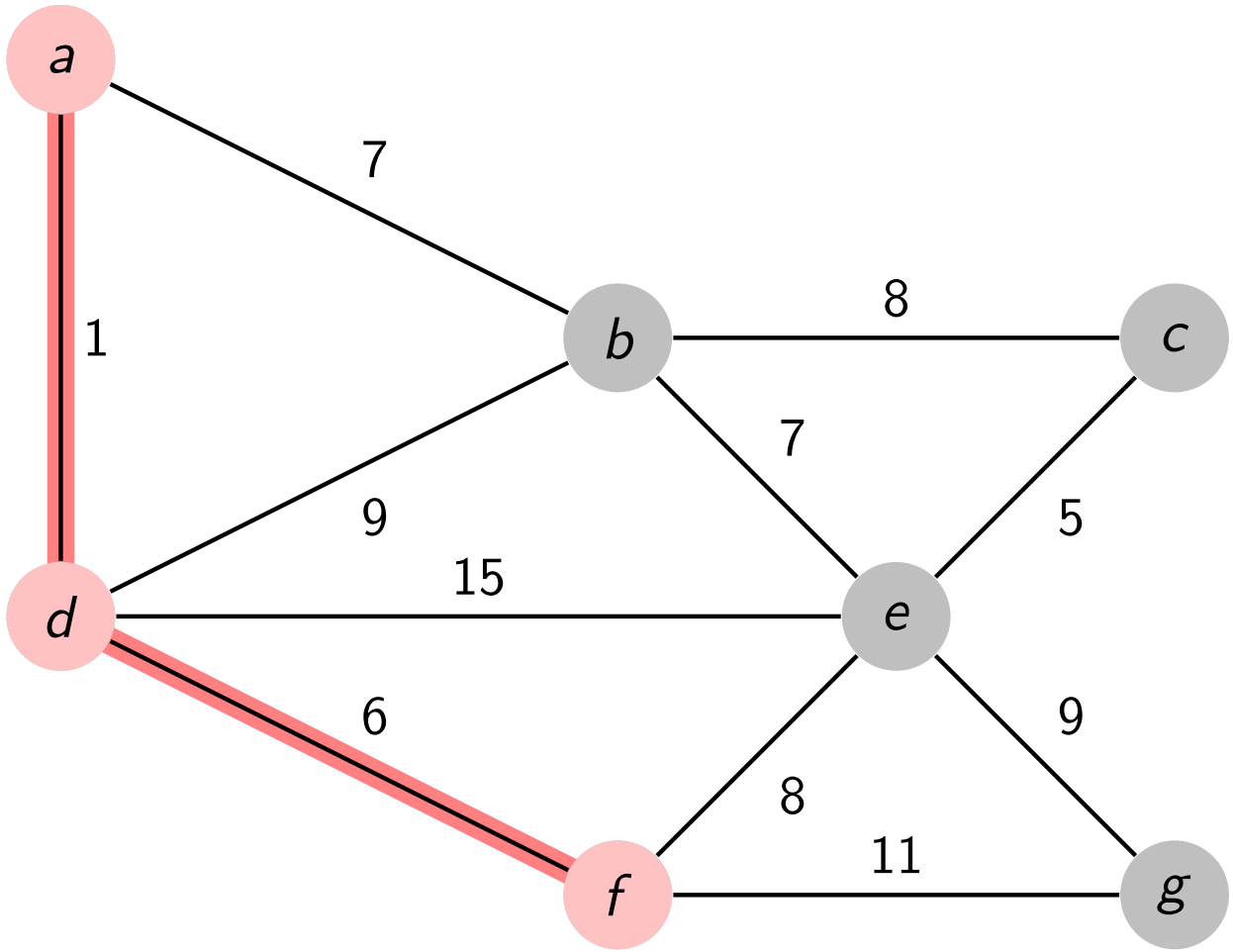
Prim's algorithm



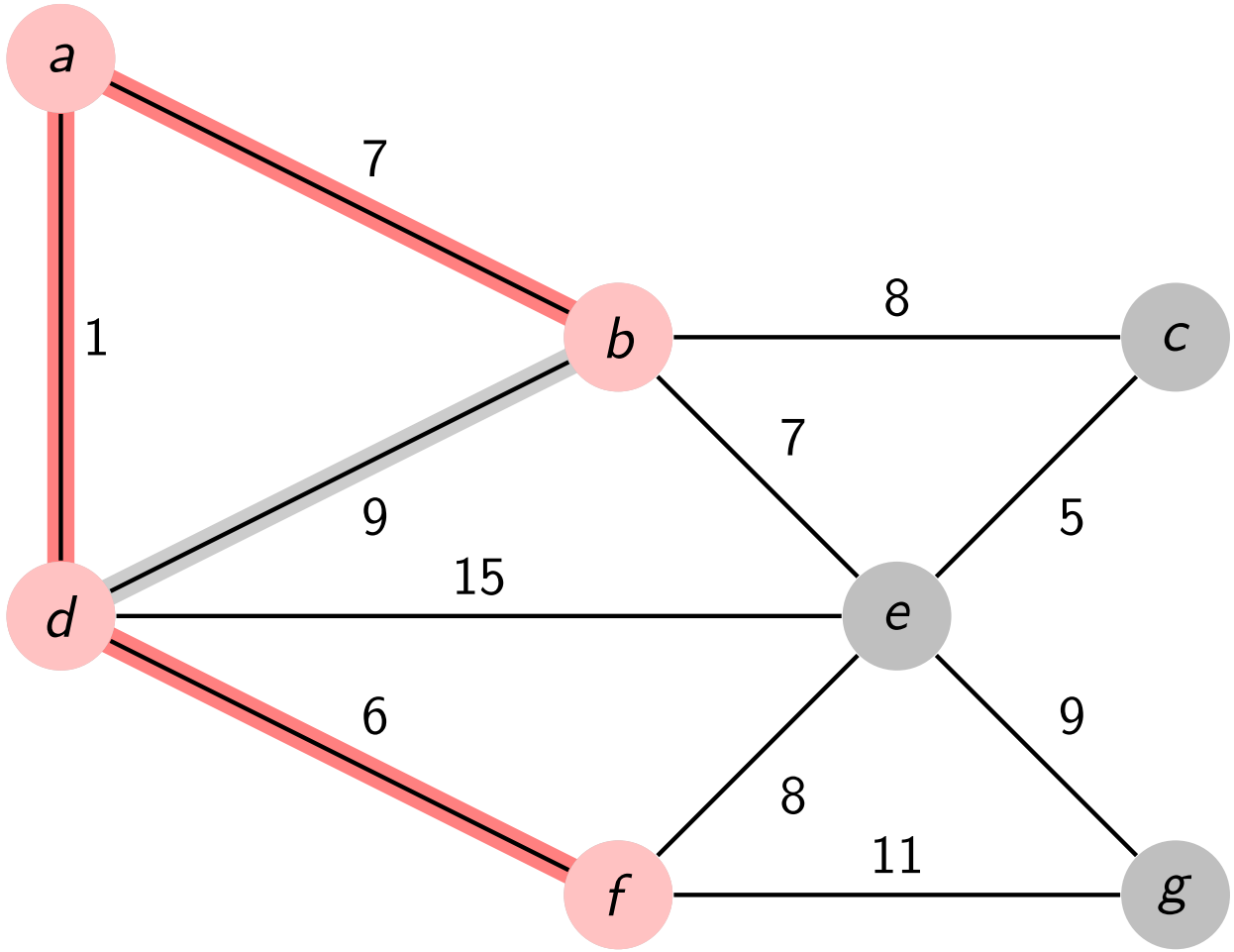
Prim's algorithm



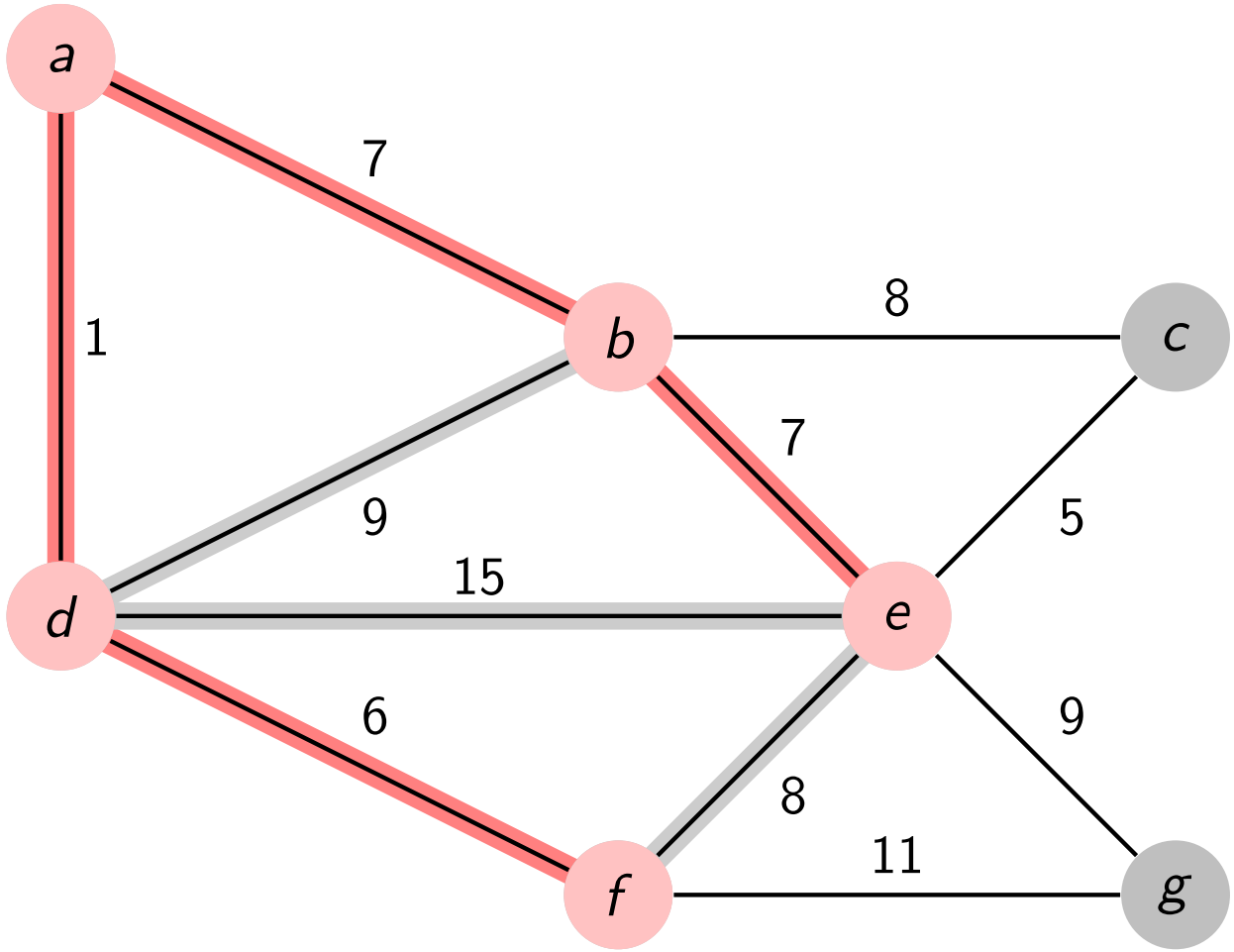
Prim's algorithm



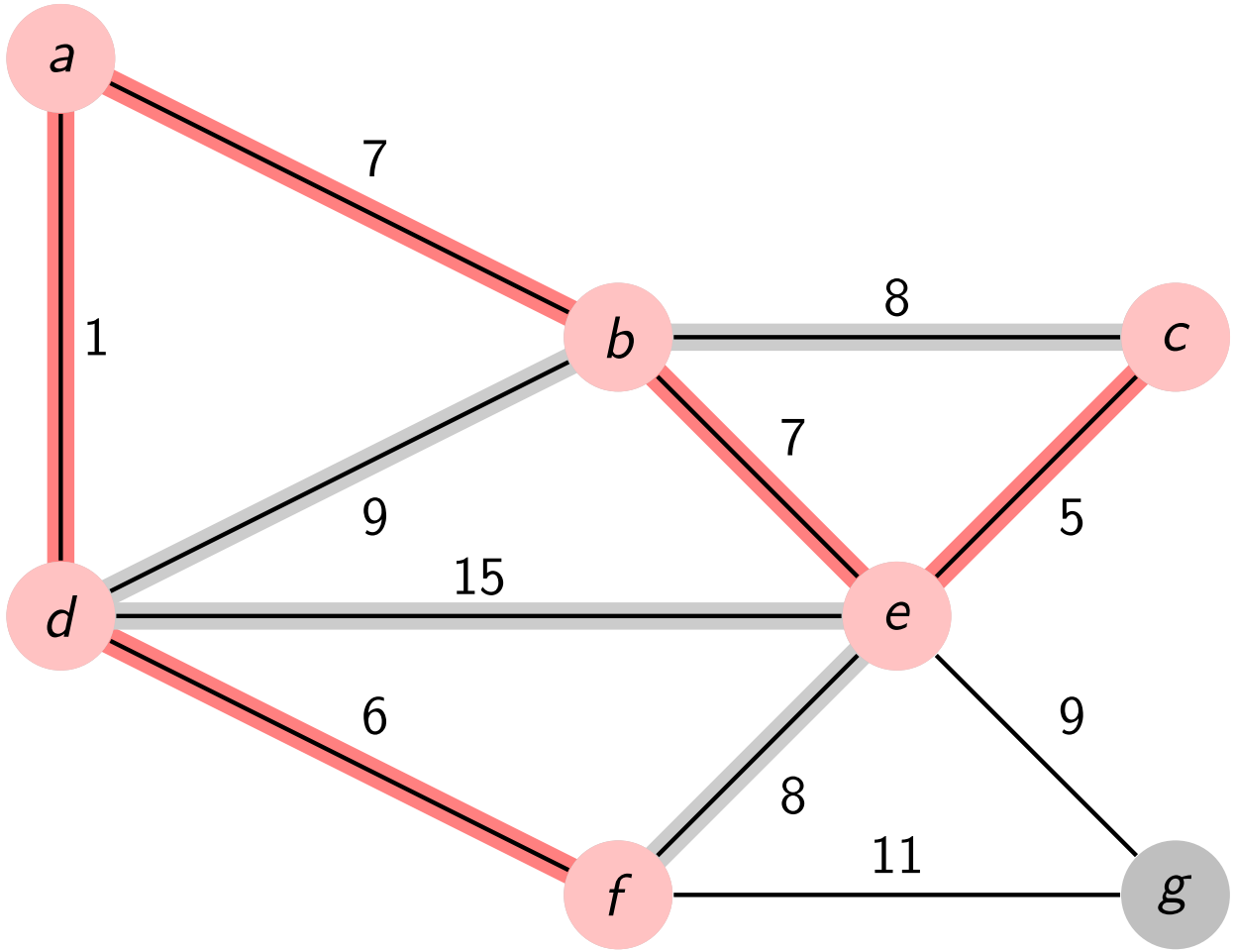
Prim's algorithm



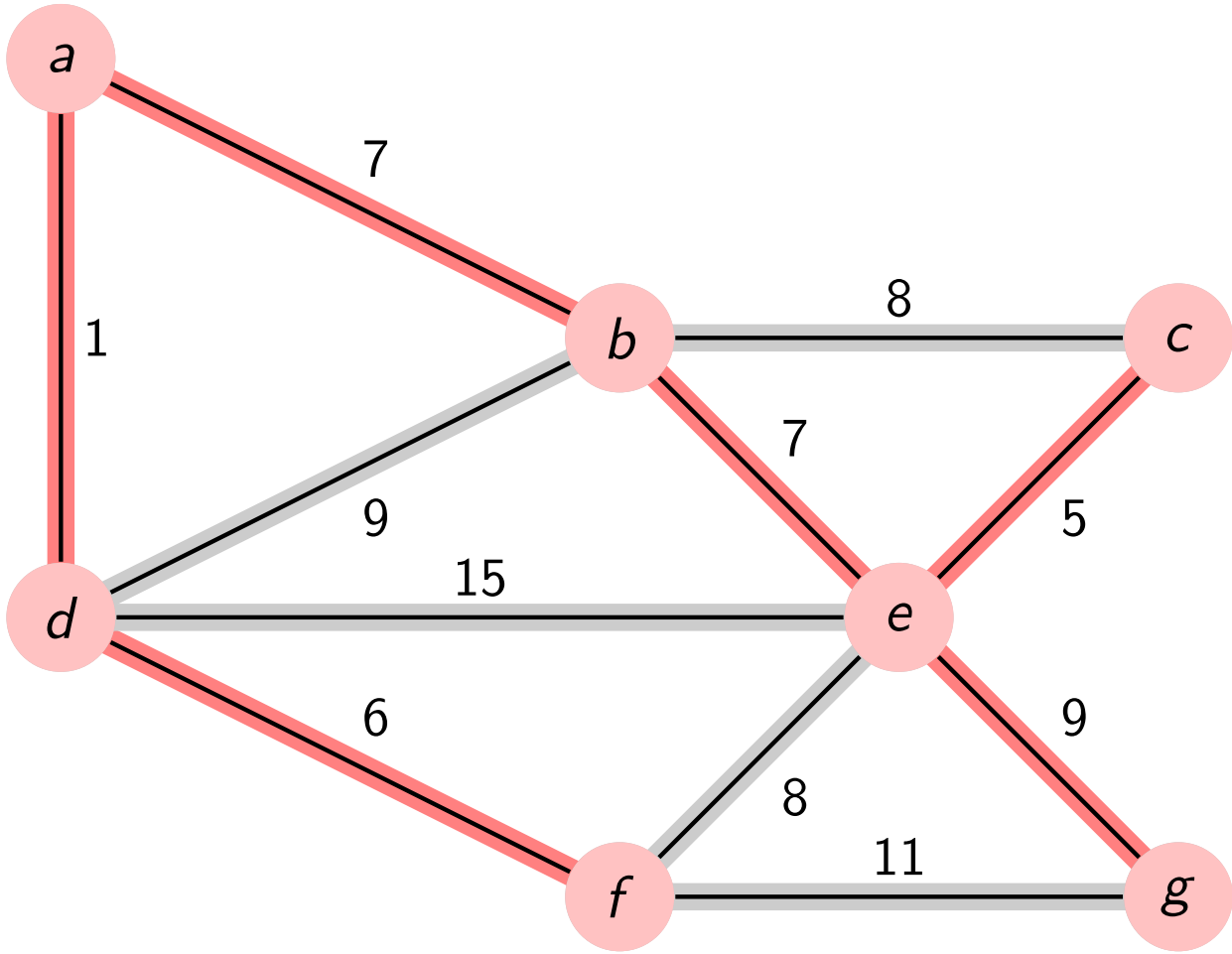
Prim's algorithm



Prim's algorithm

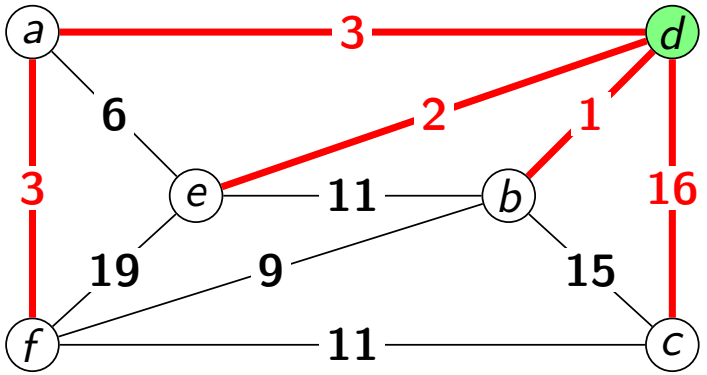


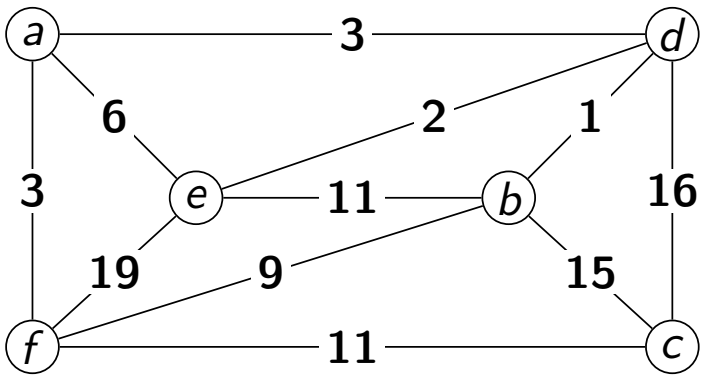
Prim's algorithm



Arbre couvrant avec Dijkstra

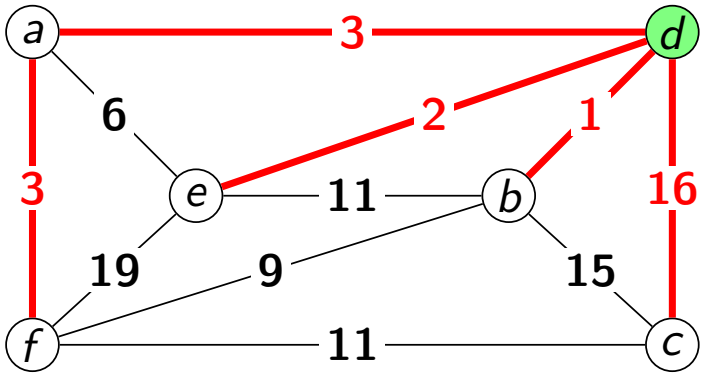
Data: graphe $\mathcal{G} = (S, A)$, W, s,t
Result: dist(s,t)
$\forall x \in S, D[x] = \infty$ $D[s] = 0$ $V = \emptyset$
  $P = S$;
while $P \neq \emptyset$ **do**
    a tq $D[a]$ min dans P;
    $V \leftarrow a$;
    $P \leftarrow P \setminus \{a\}$;
    for *b voisin de a* **do**
        if $D[b] > D[a] + W[a, b]$ **then**
            $D[b] \leftarrow D[a] + W[a, b]$

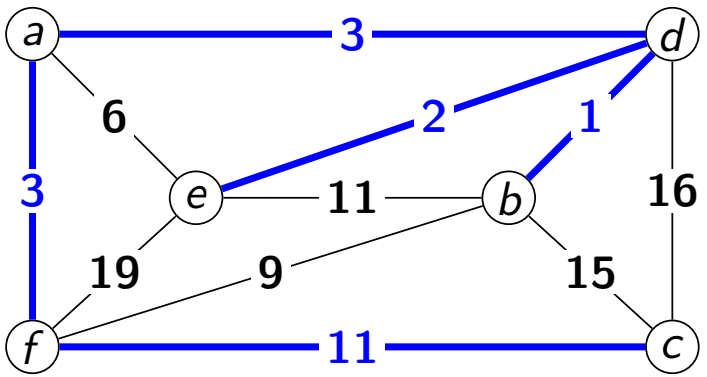




Arbre couvrant avec Dijkstra

Data: graphe $\mathcal{G} = (S, A)$, W, s,t
Result: dist(s,t)
$\forall x \in S, D[x] = \infty$ $D[s] = 0$ $V = \emptyset$
$P = S$;
while $P \neq \emptyset$ **do**
    a tq $D[a]$ min dans P;
    $V \leftarrow a$;
    $P \leftarrow P \setminus \{a\}$;
    for *b voisin de a* **do**
        if $D[b] > D[a] + W[a, b]$ **then**
            $D[b] \leftarrow D[a] + W[a, b]$

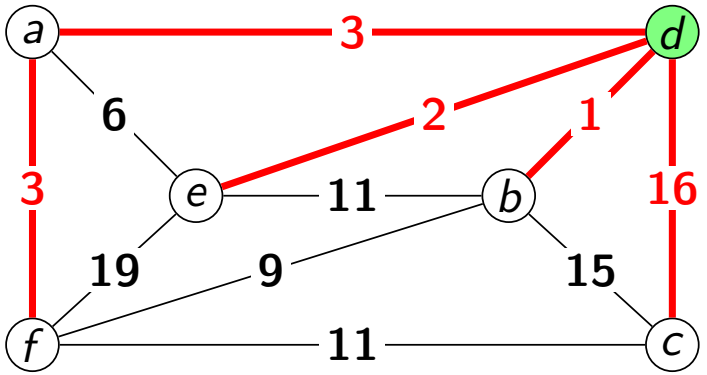


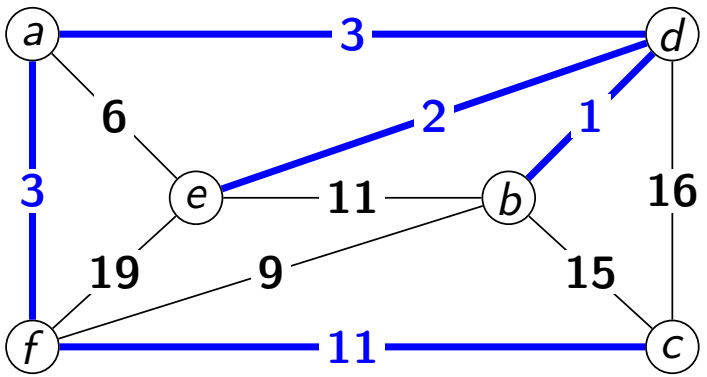


Arbre couvrant *"plus courts chemins depuis d"*

Arbre couvrant avec Dijkstra

Data: graphe $\mathcal{G} = (S, A)$, W, s,t
Result: dist(s,t)
$\forall x \in S, D[x] = \infty$ $D[s] = 0$ $V = \emptyset$
$P = S$;
while $P \neq \emptyset$ **do**
    a tq $D[a]$ min dans P;
    $V \leftarrow a$;
    $P \leftarrow P \setminus \{a\}$;
    for *b voisin de a* **do**
        if $D[b] > D[a] + W[a, b]$ **then**
            $D[b] \leftarrow D[a] + W[a, b]$

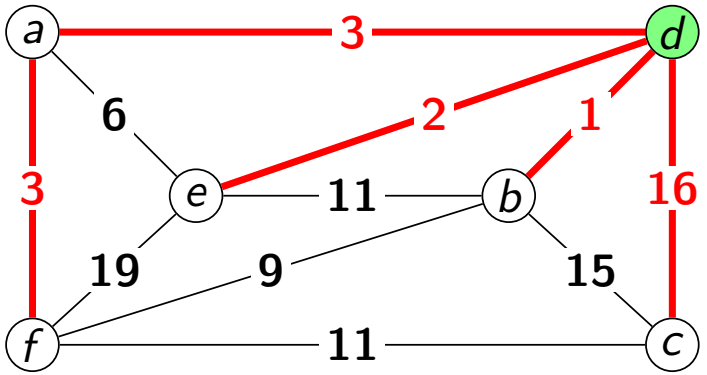


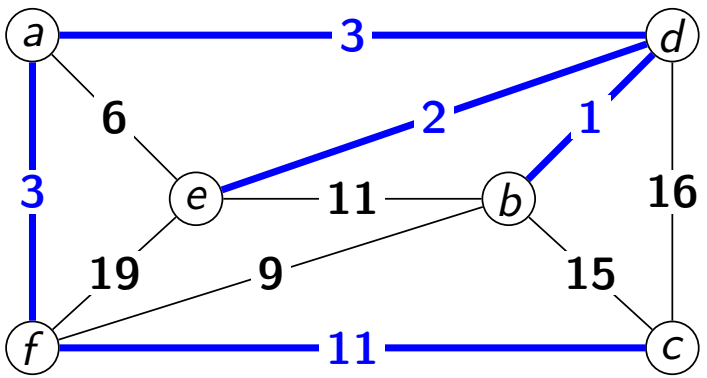


Arbre couvrant *"plus courts chemins depuis d"* $\rightsquigarrow$ pas de poids mini. en général

Arbre couvrant avec Dijkstra

Data: graphe $\mathcal{G} = (S, A)$, W, s,t
Result: dist(s,t)
$\forall x \in S, D[x] = \infty$ $D[s] = 0$ $V = \emptyset$
  $P = S$;
while $P \neq \emptyset$ **do**
    a tq $D[a]$ min dans P;
    $V \leftarrow a$;
    $P \leftarrow P \setminus \{a\}$;
    for *b voisin de a* **do**
        if $D[b] > D[a] + W[a, b]$ **then**
            $D[b] \leftarrow D[a] + W[a, b]$





Arbre couvrant *"plus courts chemins depuis d"* $\rightsquigarrow$ pas de poids mini. en général

Ex : optimisation du temps de transmission depuis un serveur

Plan

Arbre de Steiner

On va étudier une légère variation du problème de l'ACM : l'arbre de Steiner

Arbre de Steiner

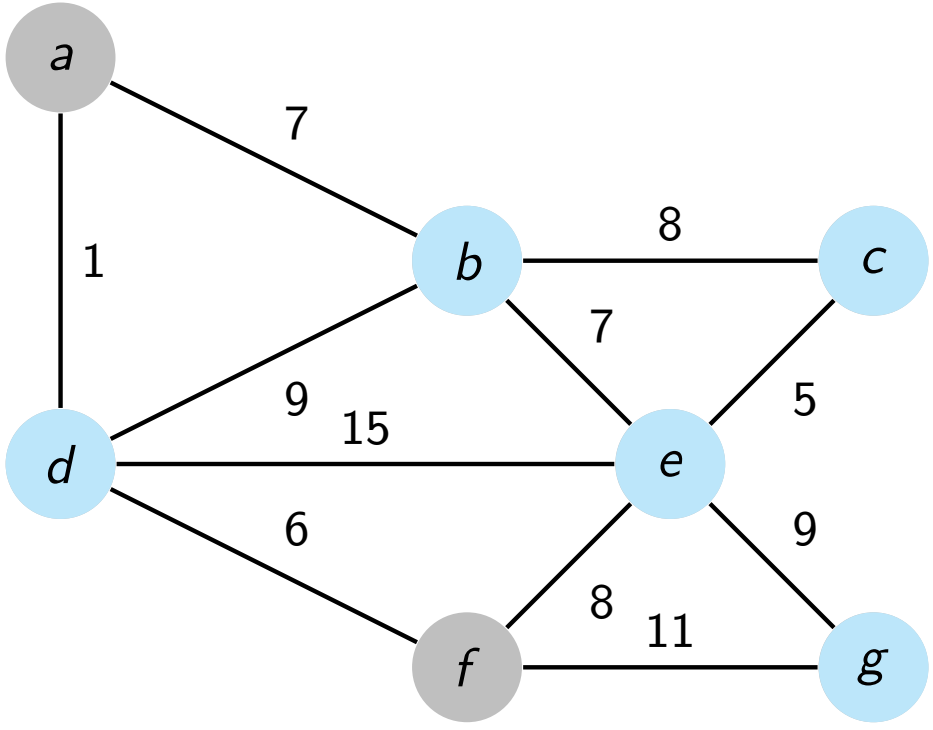
On va étudier une légère variation du problème de l'ACM : l'arbre de Steiner

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$. On cherche le graphe de poids minimal couvrant I

Arbre de Steiner

On va étudier une légère variation du problème de l'ACM : l'arbre de Steiner

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$. On cherche le graphe de poids minimal couvrant I



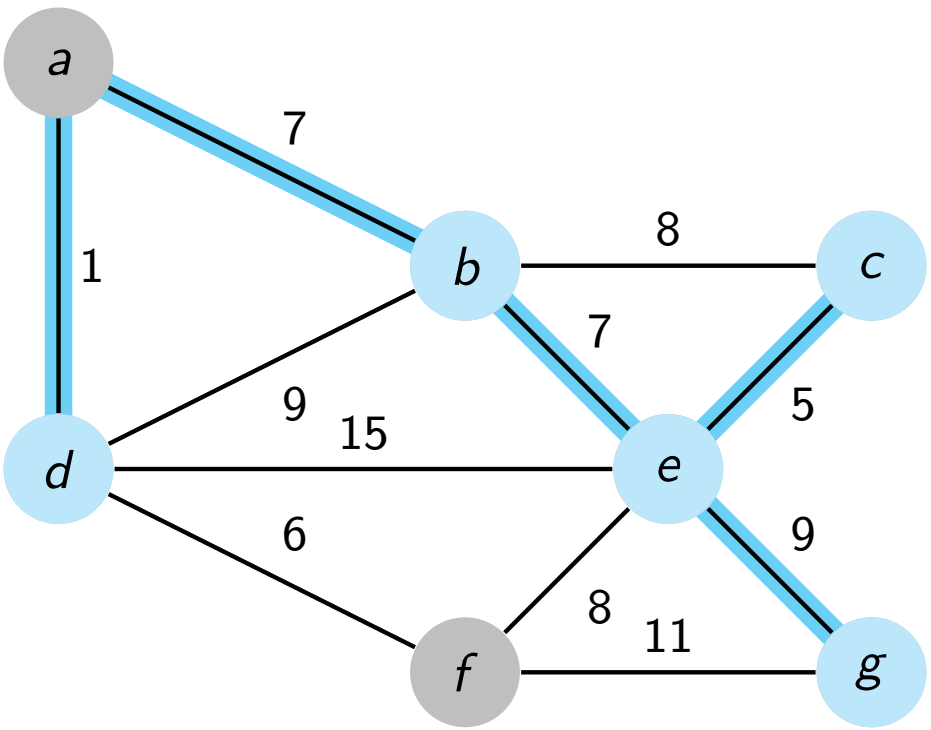
Théorème

Le problème de l'arbre de Steiner est NP-difficile.

Arbre de Steiner

On va étudier une légère variation du problème de l'ACM : l'arbre de Steiner

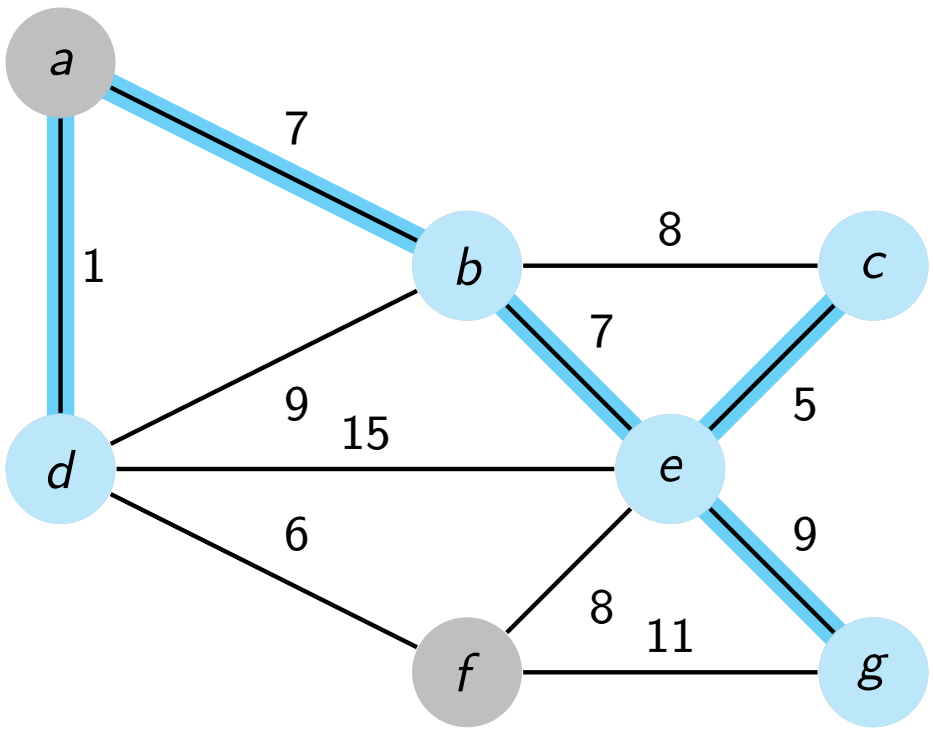
Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$. On cherche le graphe de poids minimal couvrant I



Arbre de Steiner

On va étudier une légère variation du problème de l'ACM : l'arbre de Steiner

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$. On cherche le graphe de poids minimal couvrant I



Théorème

Le problème de l'arbre de Steiner est NP-difficile.

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

Théorème

T est un arbre couvrant I et $w(T) \leq 2w(T^*)$ (où T^* est un arbre de Steiner).

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

Théorème

T est un arbre couvrant I et $w(T) \leq 2w(T^*)$ (où T^* est un arbre de Steiner).L'algorithme s'exécute en $O(|I||V|^2)$

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

Théorème

T est un arbre couvrant I et $w(T) \leq 2w(T^*)$ (où T^* est un arbre de Steiner).L'algorithme s'exécute en $O(|I||V|^2)$

Si P $\neq$ NP, il est (prouvablement) impossible d'avoir un algorithme en temps polynomial ayant un d'approximation aussi proche de 1 (ou même $< \frac{96}{95}$).

Algorithme d'approximation de Kou–Markowsky–Berman (1981)

Soit $\mathcal{G} = (S, A)$ un graphe (avec des distances positives sur les arêtes) et $I \subset S$.

1. Calculer la cloture métrique C du graphe induit par I dans G
2. Extraire un ACM T_C de C
3. Reconstruire T_C dans G (en remplaçant les arêtes par les plus courts chemins)
4. Prendre un ACM de ce graphe et l'élaguer jusqu'à obtenir un arbre T dont toutes les feuilles sont dans I

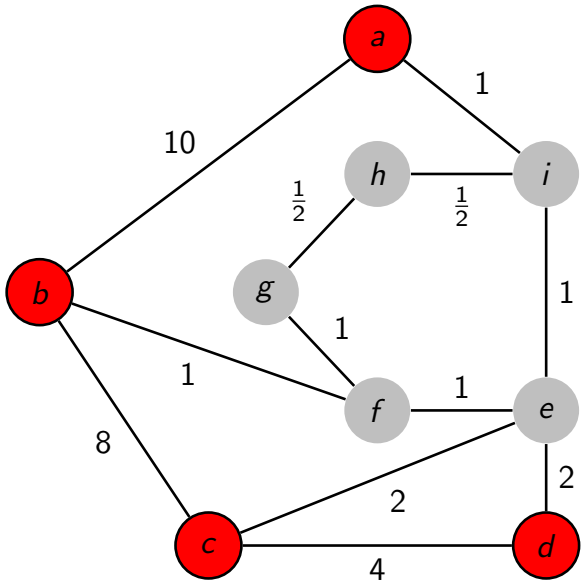
Théorème

T est un arbre couvrant I et $w(T) \leq 2w(T^*)$ (où T^* est un arbre de Steiner).L'algorithme s'exécute en $O(|I||V|^2)$

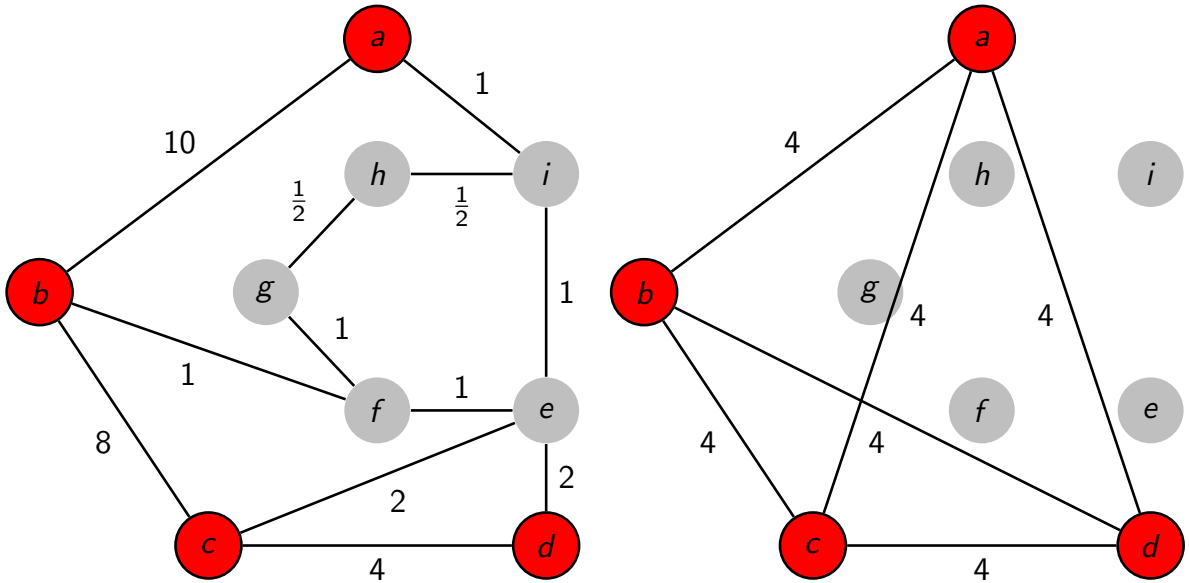
Si P $\neq$ NP, il est (prouvablement) impossible d'avoir un algorithme en temps polynomial ayant un d'approximation aussi proche de 1 (ou même $< \frac{96}{95}$).

Meilleur algo d'approx : ratio de $\ln(4) + \varepsilon \approx 1.386$ (randomizé, en moyenne)

Kou–Markowsky–Berman : exemple

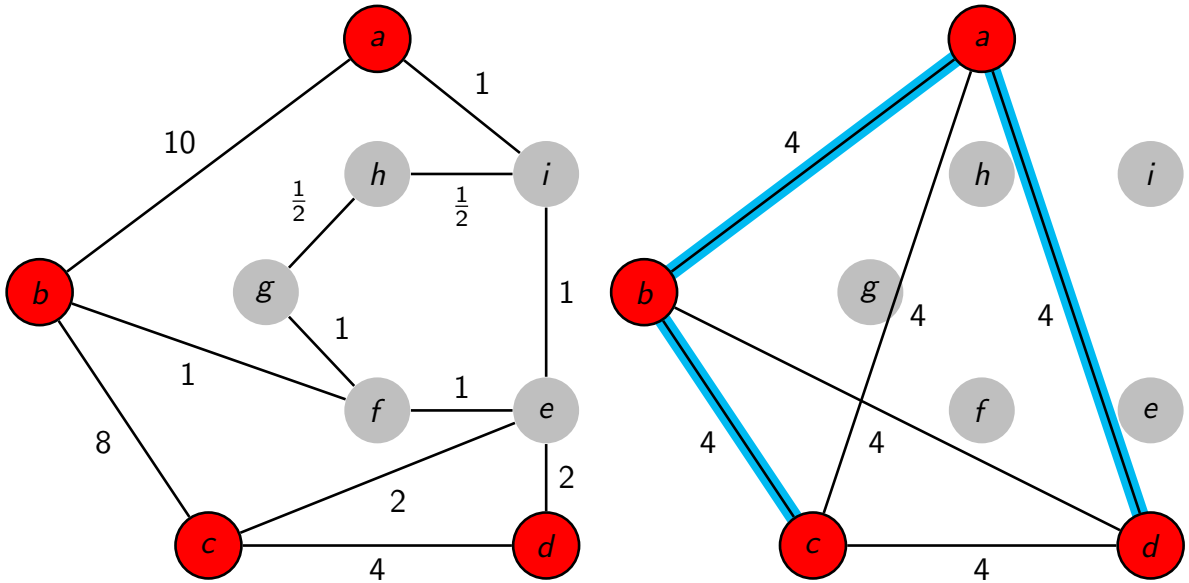


Kou–Markowsky–Berman : exemple



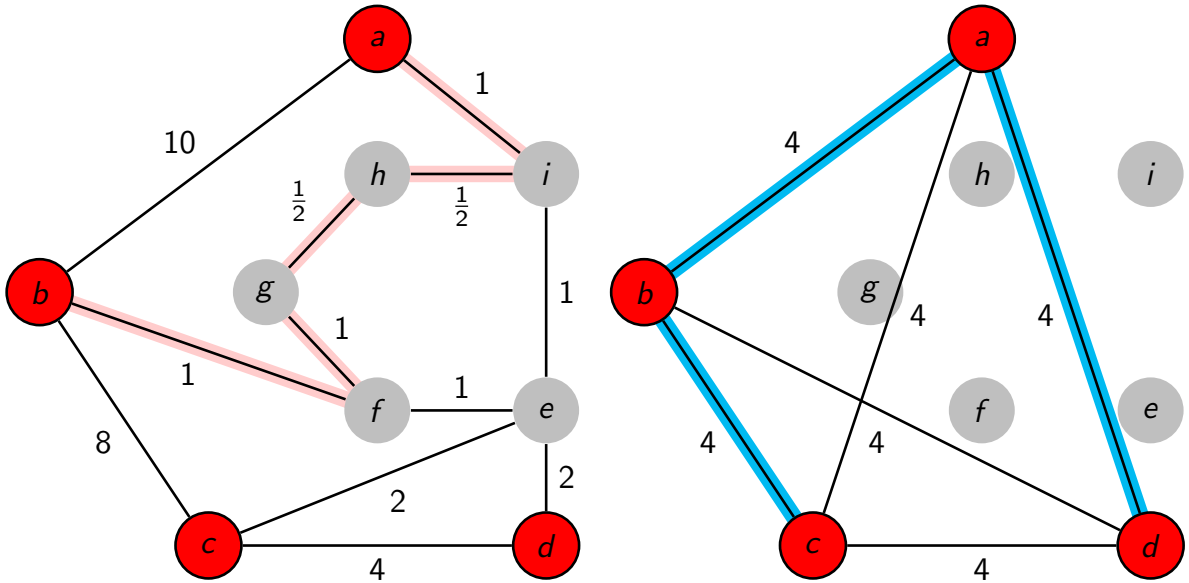
- Clôture métrique des sommets a, b, c, d

Kou–Markowsky–Berman : exemple



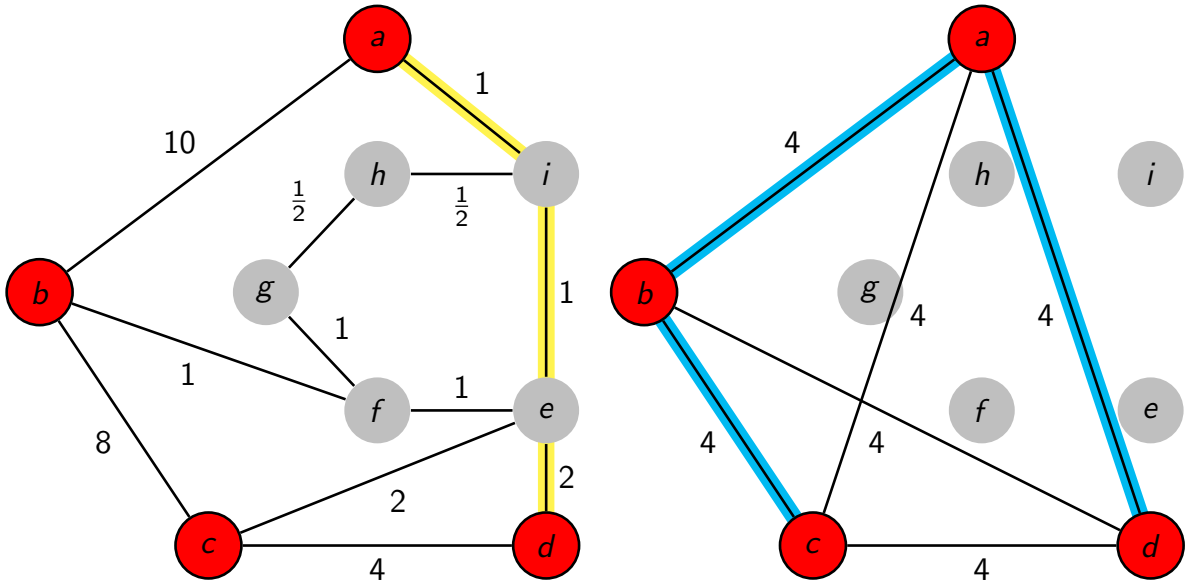
- Clôture métrique des sommets a, b, c, d
- ACM de la clôture

Kou–Markowsky–Berman : exemple



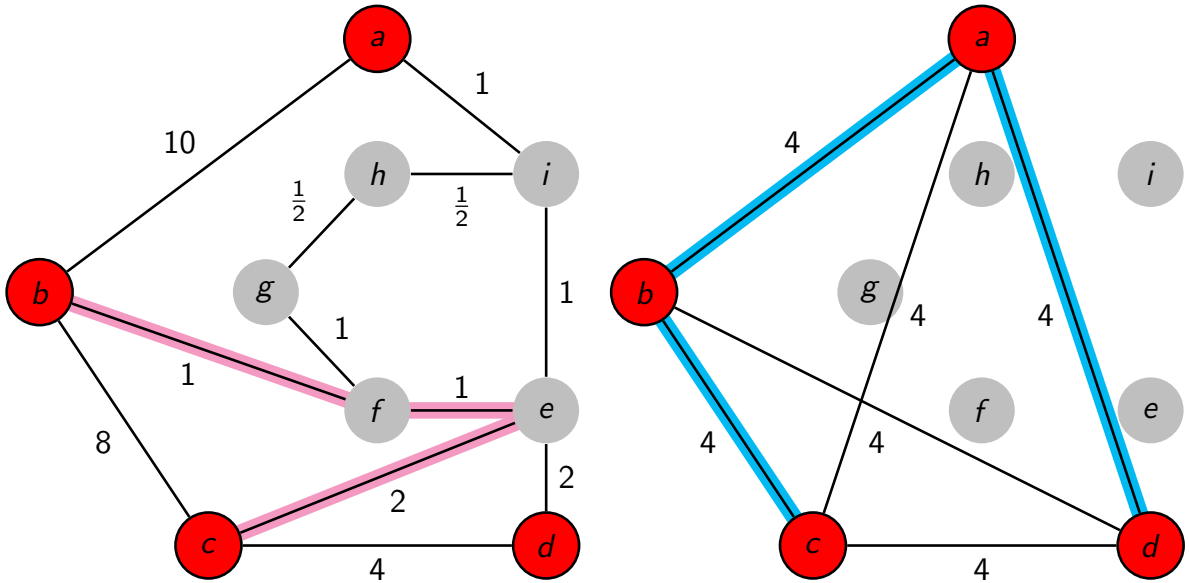
- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G

Kou–Markowsky–Berman : exemple



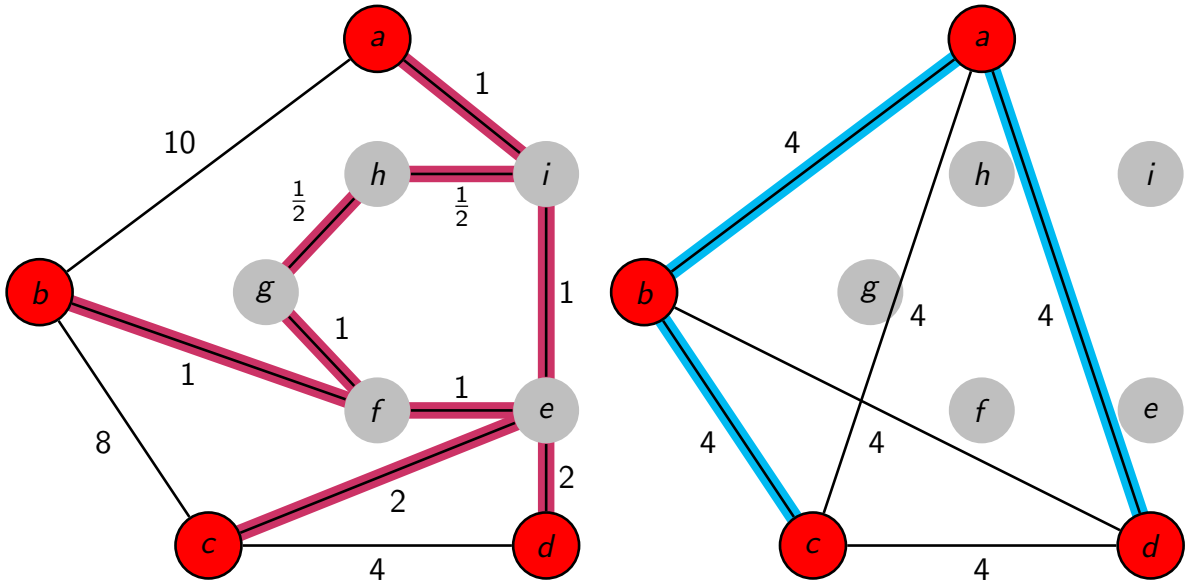
- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G

Kou–Markowsky–Berman : exemple



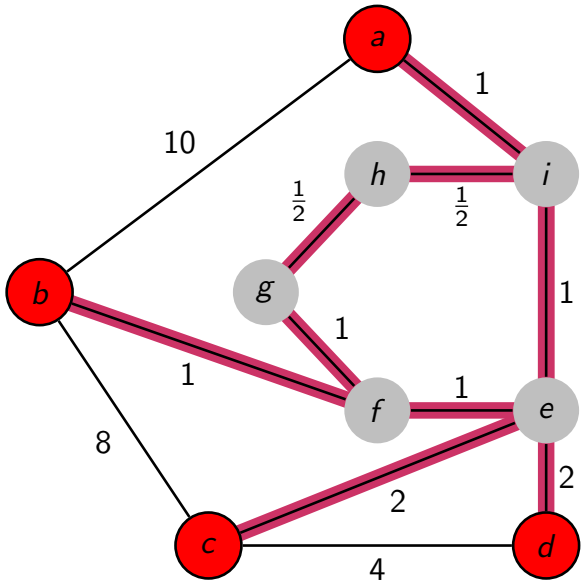
- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G

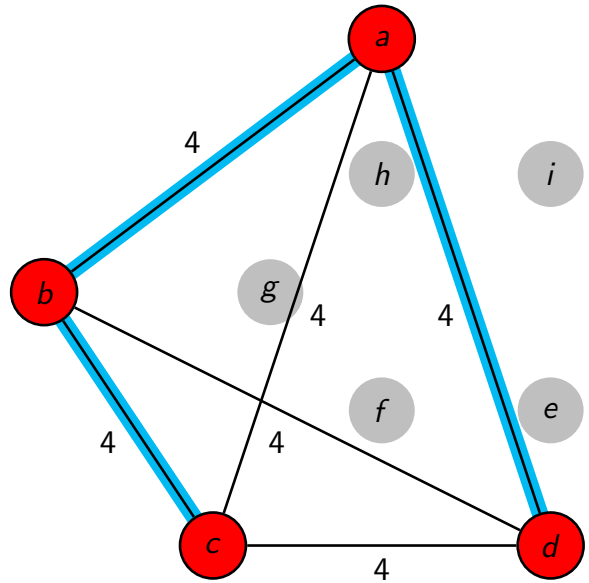
Kou–Markowsky–Berman : exemple

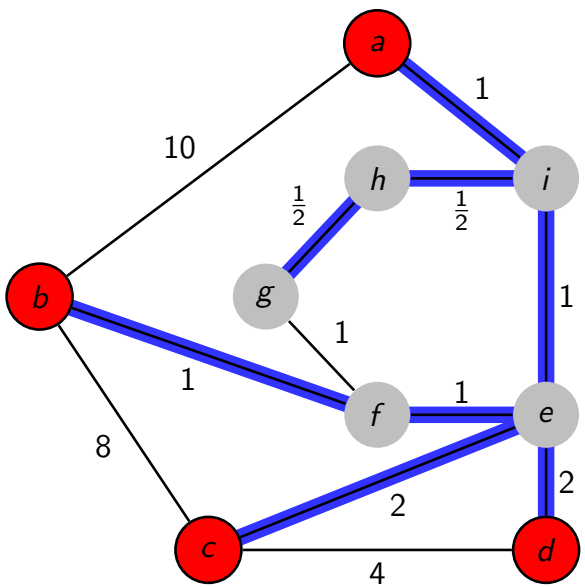


- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G

Kou–Markowsky–Berman : exemple

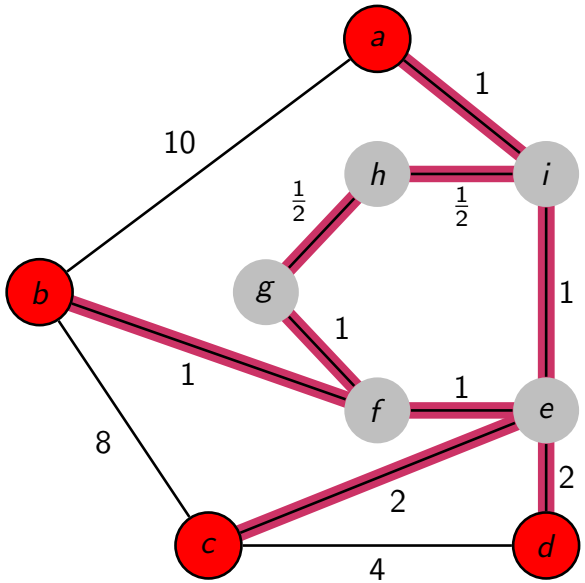


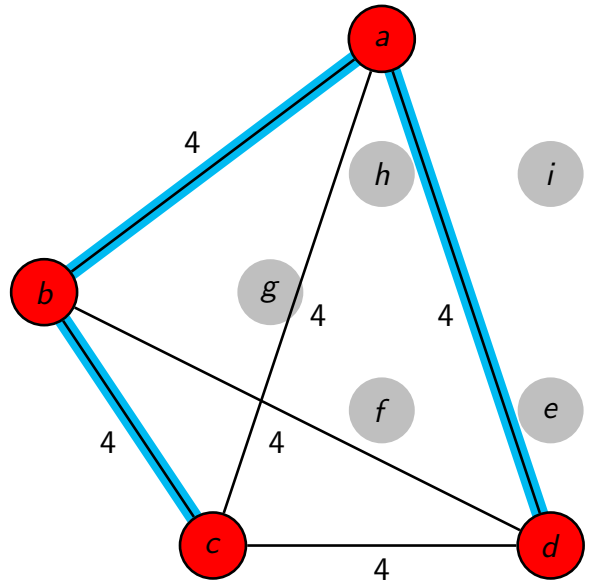


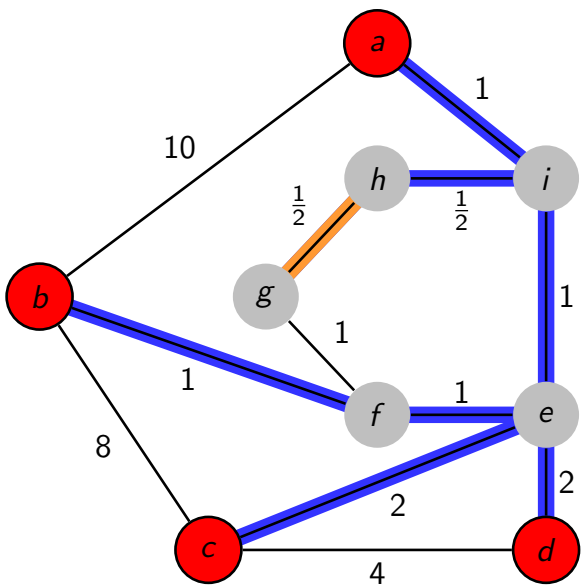


- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G
- ACM

Kou–Markowsky–Berman : exemple

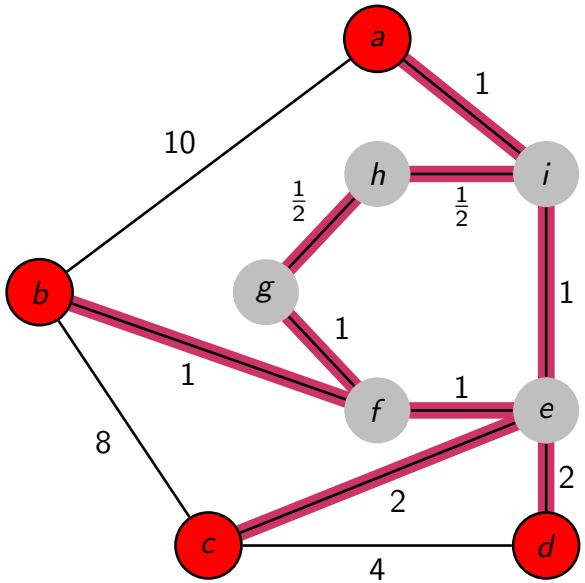


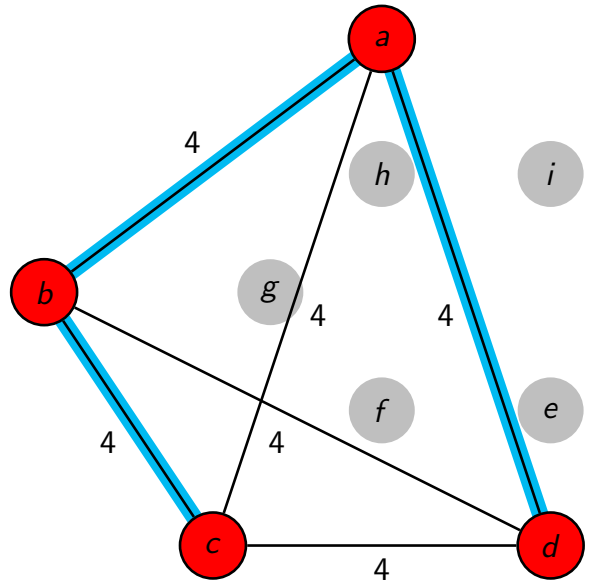


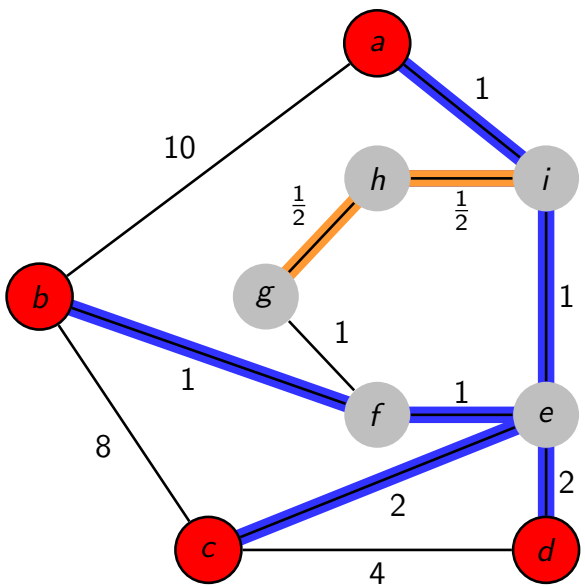


- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G
- ACM
- Élagage

Kou–Markowsky–Berman : exemple

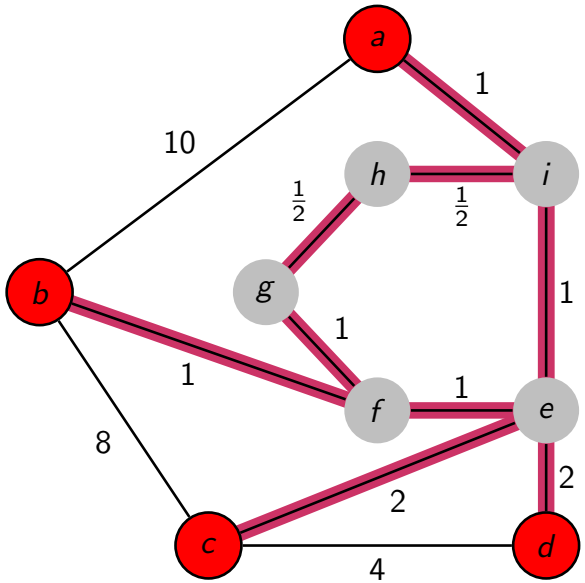


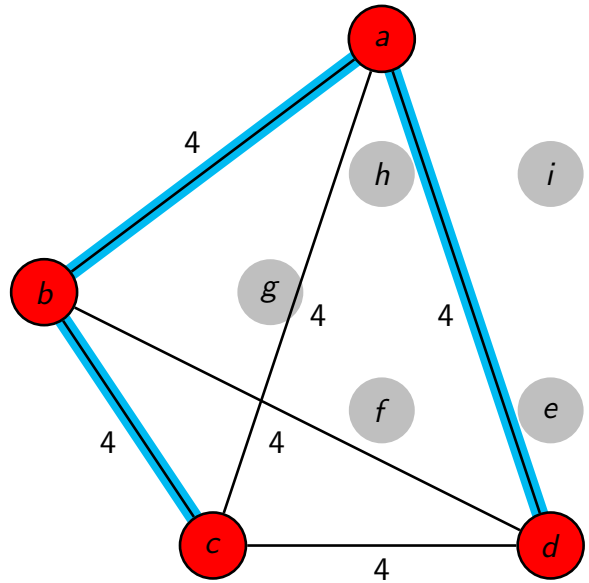


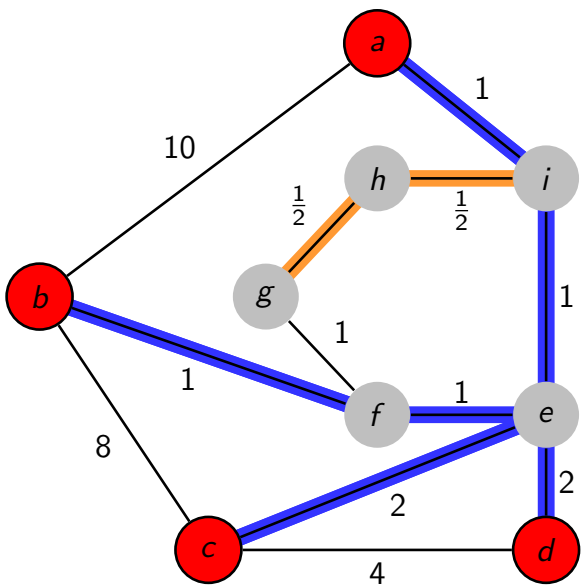


- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G
- ACM
- Élagage

Kou–Markowsky–Berman : exemple

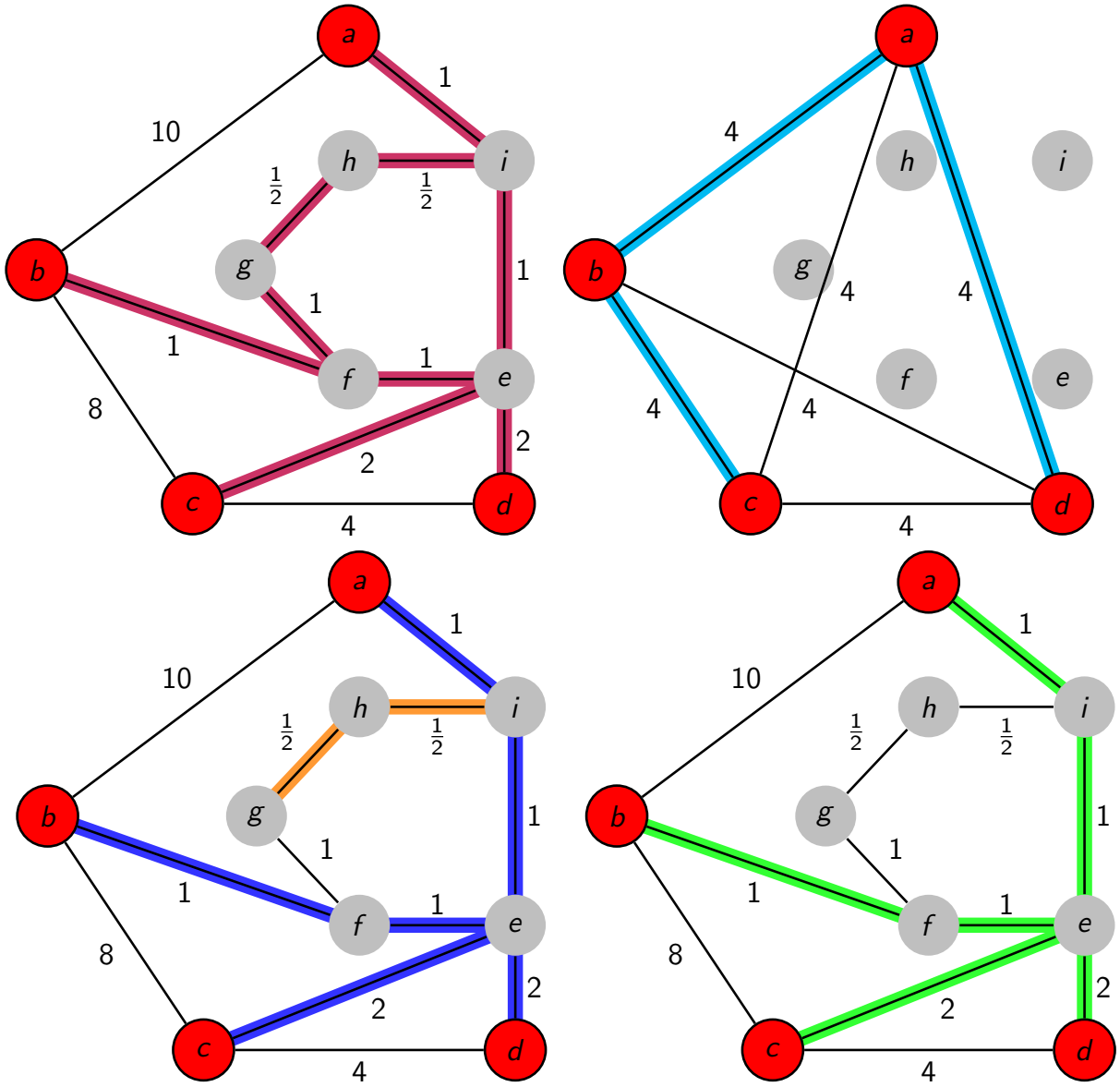






- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G
- ACM
- Élagage

Kou–Markowsky–Berman : exemple



- Clôture métrique des sommets a, b, c, d
- ACM de la clôture
- Reconstruction dans G
- ACM
- Élagage
- Approximation de l'arbre de Steiner