
Algorithme du plus court chemin

Michel Bierlaire

`michel.bierlaire@epfl.ch`

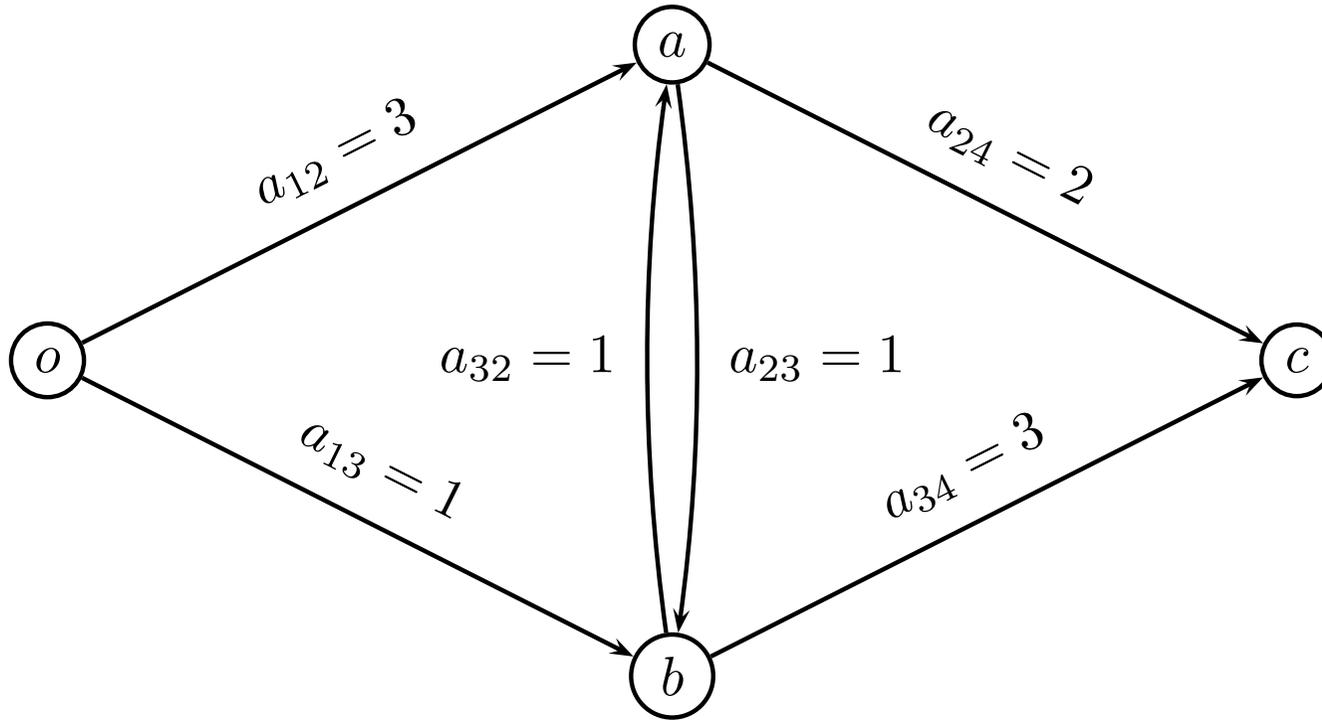
EPFL - Laboratoire Transport et Mobilité - ENAC

Le plus court chemin

Problème :

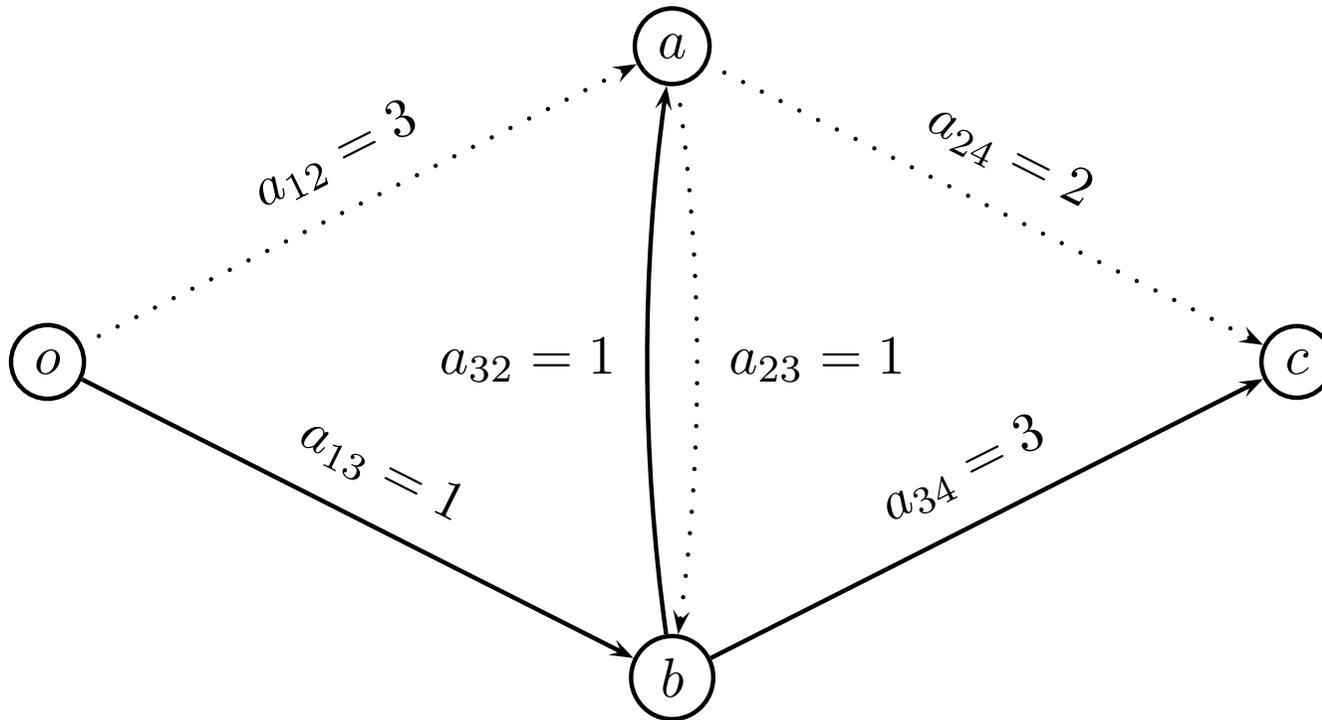
- Soit un réseau $G = (N, A)$.
- Un coût a_{ij} est associé à chaque arc $(i, j) \in A$:
 - distance,
 - temps de trajet,
 - etc.
- Soit un nœud appelé *origine*. Par convention, ce sera le nœud o .
- Nous cherchons le chemin de coût minimum reliant le nœud o à n'importe quel autre nœud du réseau.

Le plus court chemin



Le plus court chemin

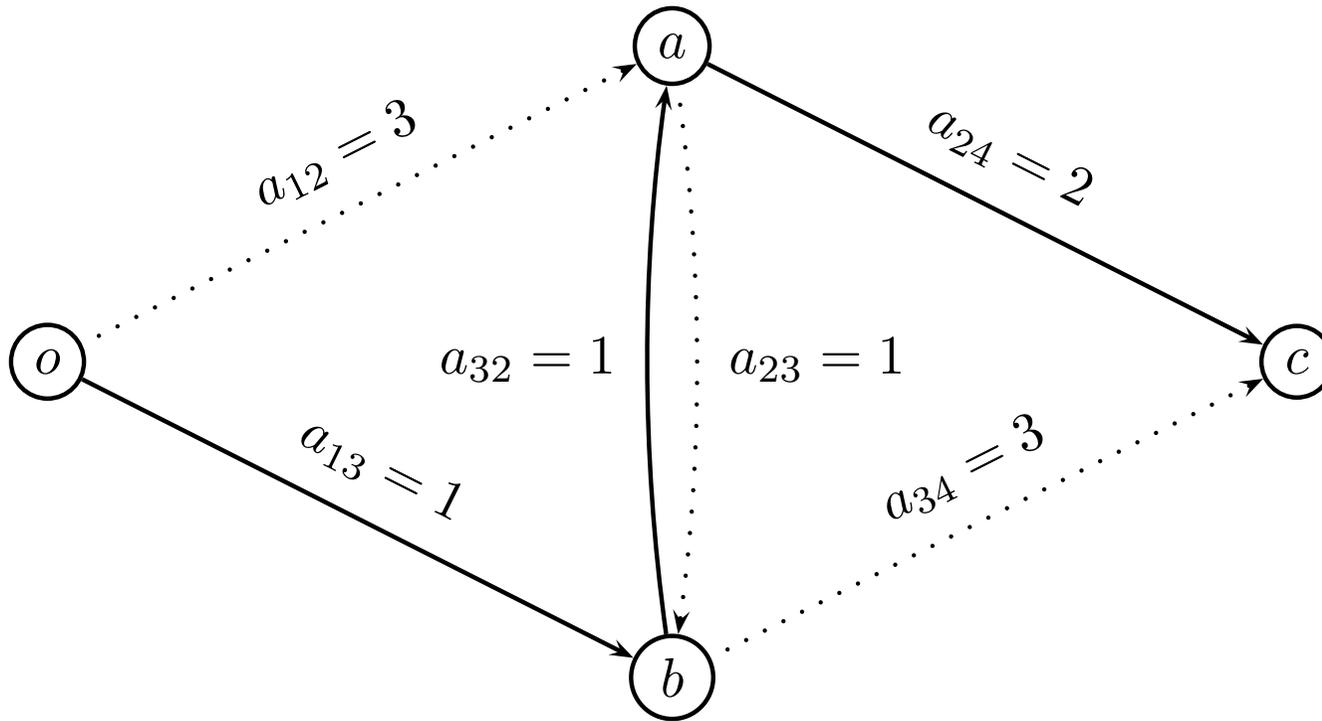
- La solution est un arbre.



Note : chaque nœud dans l'arbre a exactement un prédécesseur.

Le plus court chemin

- La solution n'est pas nécessairement unique.



Idée générale de l'algorithme

- Parcours systématique du réseau à partir de l'origine.
- A chaque nœud visité, une étiquette est associée.
- Cette étiquette est potentiellement mise à jour à chaque visite du nœud.

Conditions d'optimalité

- Soient $d_i \in \mathbb{R}$, $i \in N$ tels que

$$d_j \leq d_i + a_{ij} \quad \forall (i, j) \in A.$$

- Soit P un chemin entre l'origine o et un nœud ℓ .
- Si

$$d_j = d_i + a_{ij} \quad \forall (i, j) \in P,$$

alors P est un plus court chemin entre o et ℓ .

Conditions d'optimalité

Preuve:

- P est composé d'arcs

$$(o, i_1), (i_1, i_2), \dots, (i_k, \ell)$$

- Longueur de P :

$$L(P) = a_{oi_1} + a_{i_1i_2} + \dots + a_{i_k\ell}$$

- Comme $a_{ij} = d_j - d_i$,

$$L(P) = (d_{i_1} - d_o) + (d_{i_2} - d_{i_1}) + \dots + (d_\ell - d_{i_k}) = d_\ell - d_o.$$

Conditions d'optimalité

- Soit Q un chemin quelconque entre o et ℓ .
- Q est composé d'arcs

$$(o, j_1), (j_1, j_2), \dots, (j_n, \ell)$$

- Longueur de Q :

$$L(Q) = a_{oj_1} + a_{j_1j_2} + \dots + a_{j_n\ell}$$

- Comme $a_{ij} \geq d_j - d_i$,

$$L(Q) \geq (d_{j_1} - d_o) + (d_{j_2} - d_{j_1}) + \dots + (d_\ell - d_{j_n}) = d_\ell - d_o = L(P).$$

- La longueur de P est donc plus courte que la longueur de Q .
- Comme Q est arbitraire, P est le plus court chemin entre o et ℓ .

Algorithme

Idée :

- On démarre avec un vecteur d'étiquettes $(d_i)_{i \in N}$.
- On sélectionne un arc (i, j) qui viole les conditions d'optimalité, c.-à-d. tel que

$$d_j > d_i + a_{ij}.$$

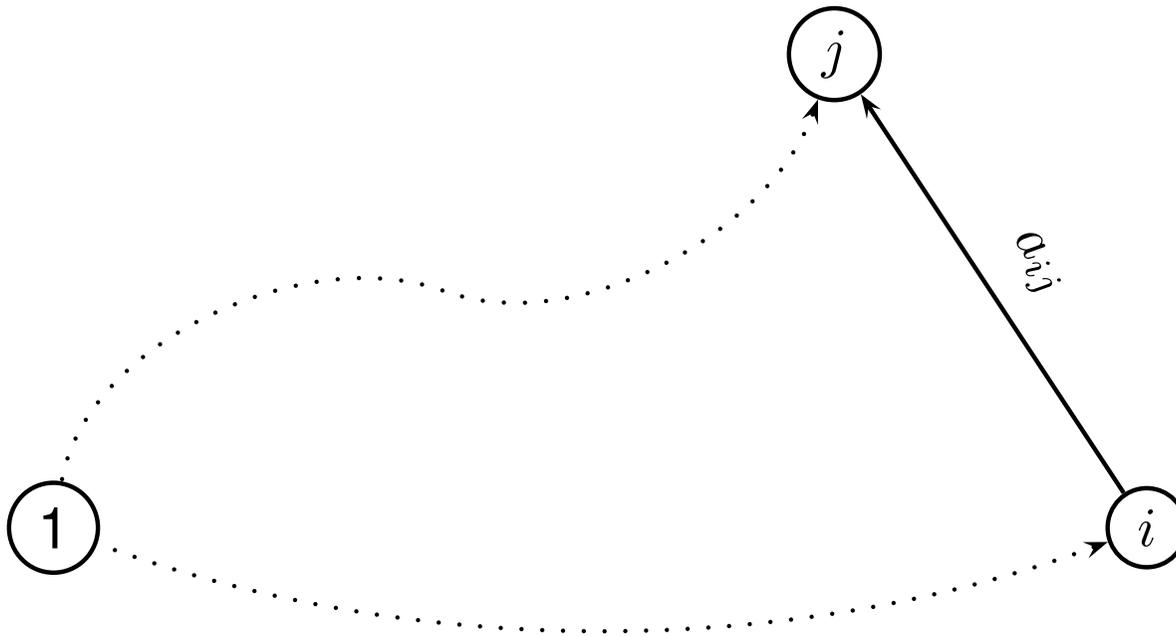
- On met à jour l'étiquette de j :

$$d_j = d_i + a_{ij}.$$

- Et ainsi de suite jusqu'à ce que tous les arcs vérifient la condition.

Interprétation

- d_i : longueur d'un chemin entre le nœud o et le nœud i .
- Si $d_j > d_i + a_{ij}$, chemin $o \rightarrow i \rightarrow j$ plus court que le chemin $o \rightarrow j$.



Exploration du graphe

- Travailler nœud par nœud.
- Pour un nœud donné, traiter tous les arcs sortants.
- Dès qu'un nœud est atteint, on l'ajoute à la liste.
- Dès qu'un nœud est traité, on le supprime de la liste.
- On arrête lorsque la liste est vide.
- Notons V la liste des nœuds à traiter.

Algorithme

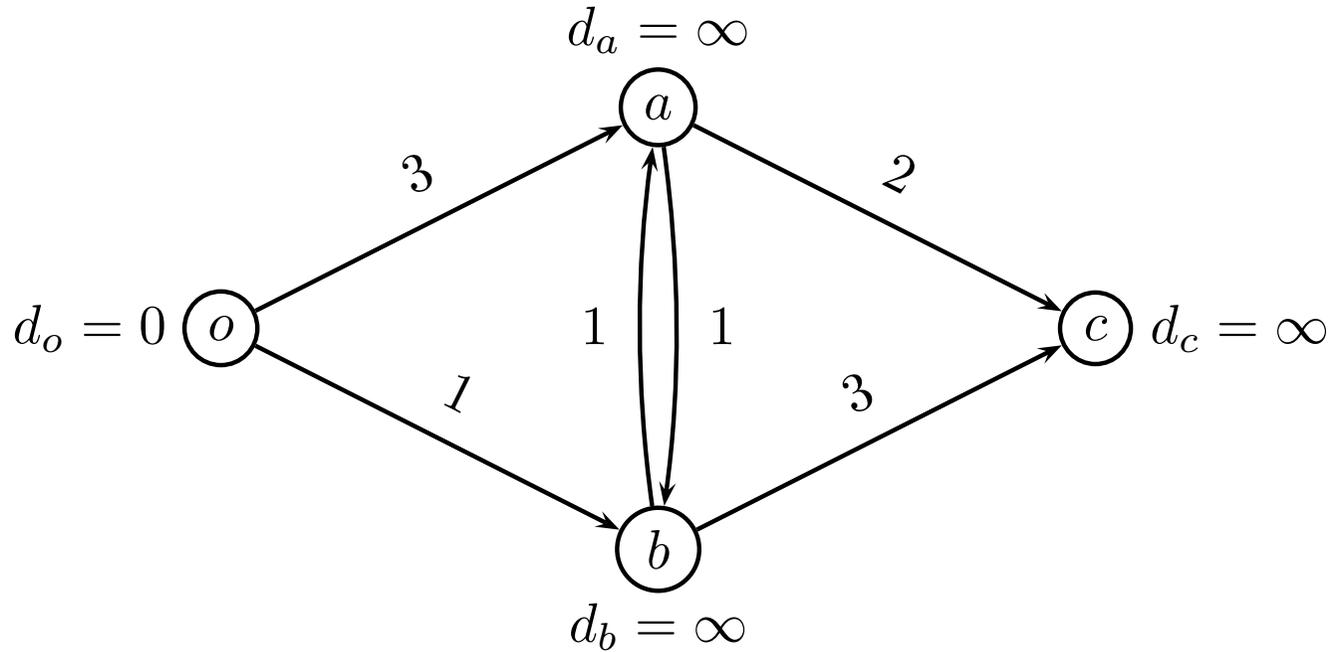
Initialisation

- Liste de nœud : $V = \{o\}$.
- Étiquettes : $d_o = 0, d_i = +\infty, \forall i \neq o$.

Itérations Tant que $V \neq \emptyset$,

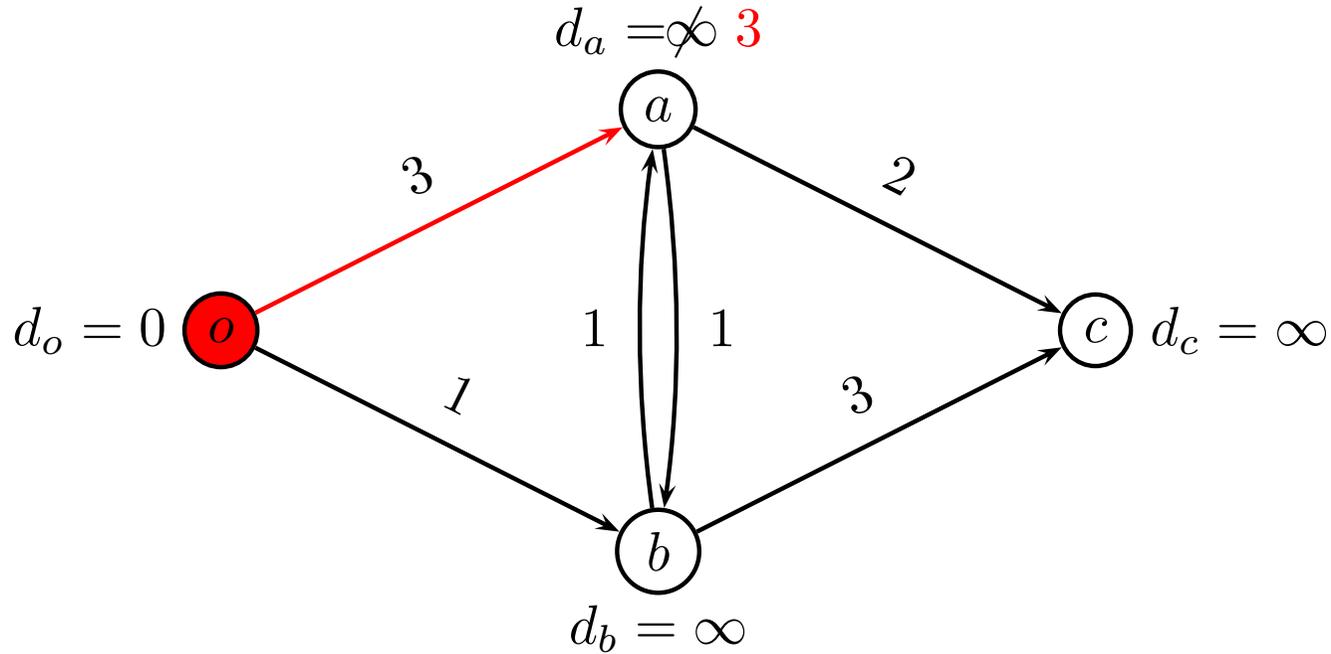
- Choisir i dans V .
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
 - Si $d_j > d_i + a_{ij}$,
 - $d_j = d_i + a_{ij}$.
 - $V = V \cup \{j\}$.

Exemple



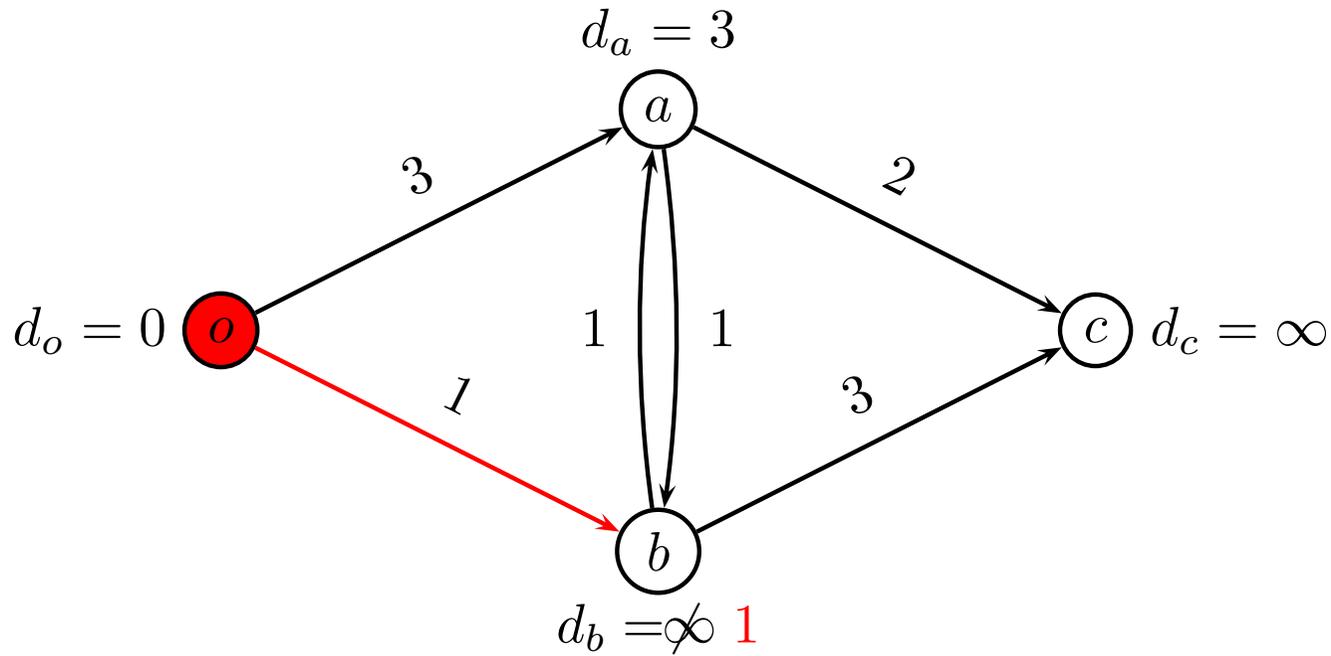
Iter	V	d_o	d_a	d_b	d_c	Traiter
0	{ o }	0	∞	∞	∞	o

Exemple



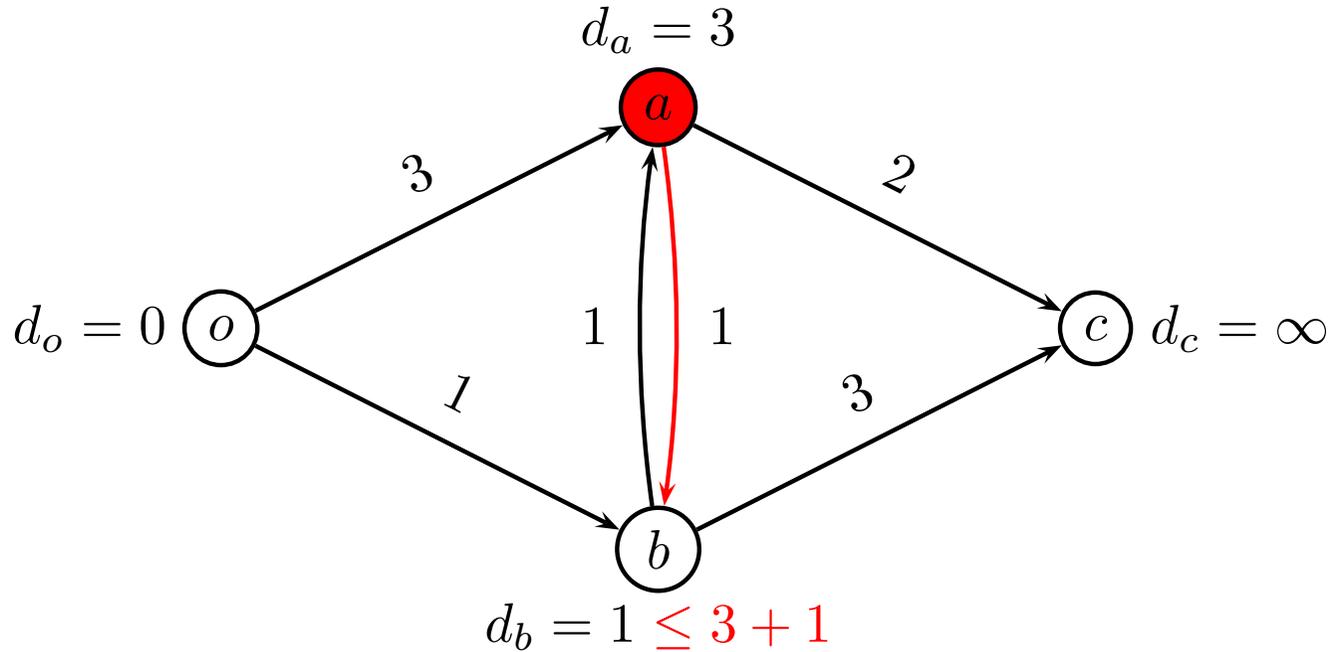
Iter	V	d_o	d_a	d_b	d_c	Traiter
0	$\{o\}$	0	∞	∞	∞	o
1	$\{a\}$	0	3	∞	∞	

Exemple



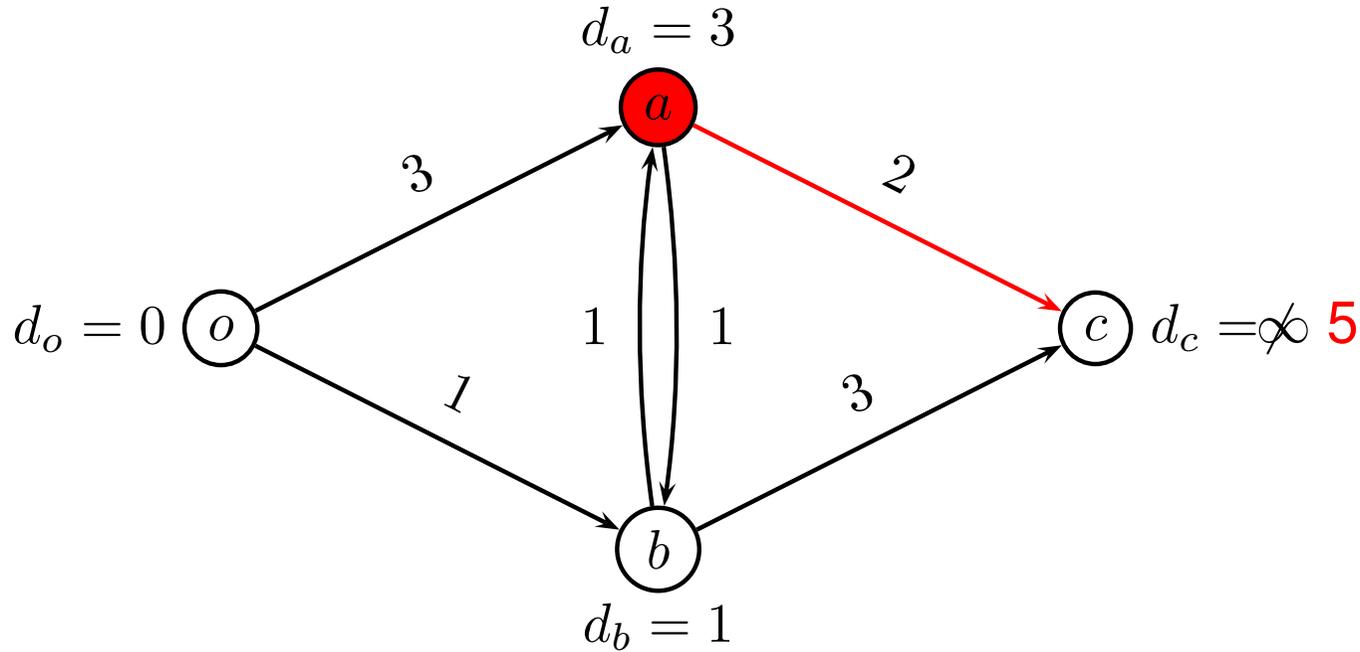
Iter	V	d_o	d_a	d_b	d_c	Traiter
0	$\{o\}$	0	∞	∞	∞	o
1	$\{a,b\}$	0	3	1	∞	a

Exemple



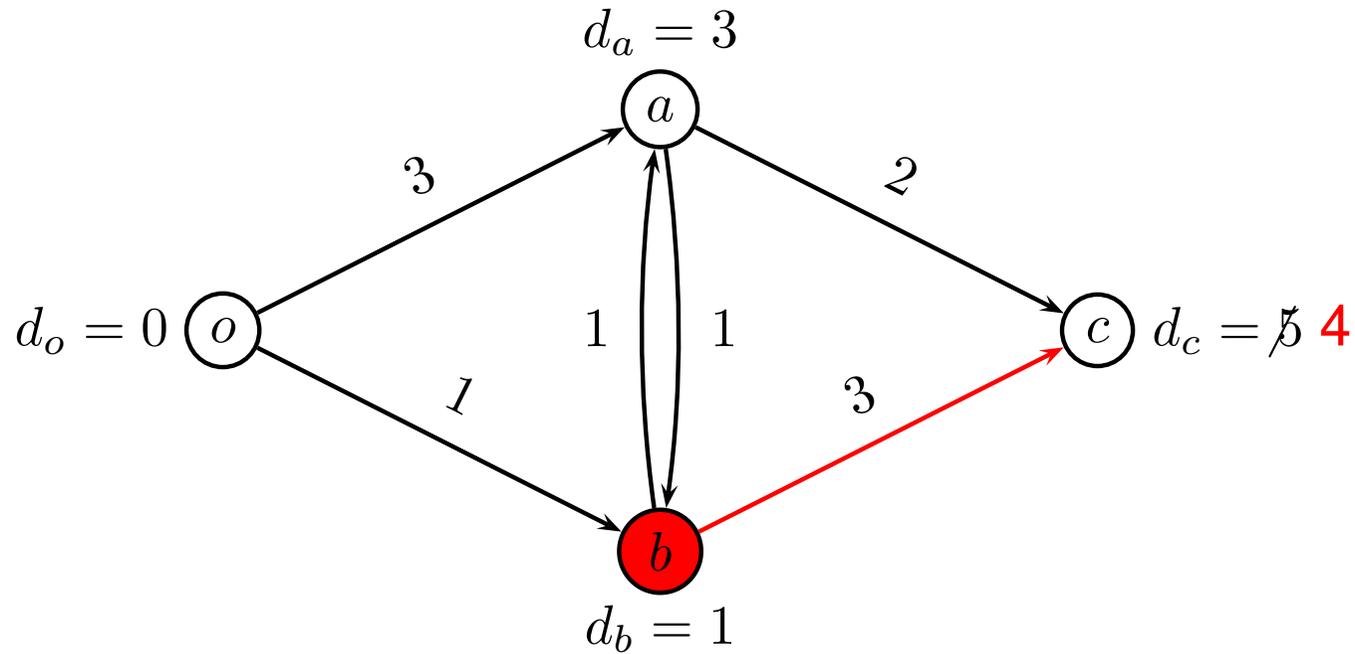
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	{o}	0	—	∞	—	∞	—	∞	—	o
1	{a, b}	0	—	3	o	1	o	∞	—	a
2	{b}	0	—	3	o	1	o	∞	—	

Exemple



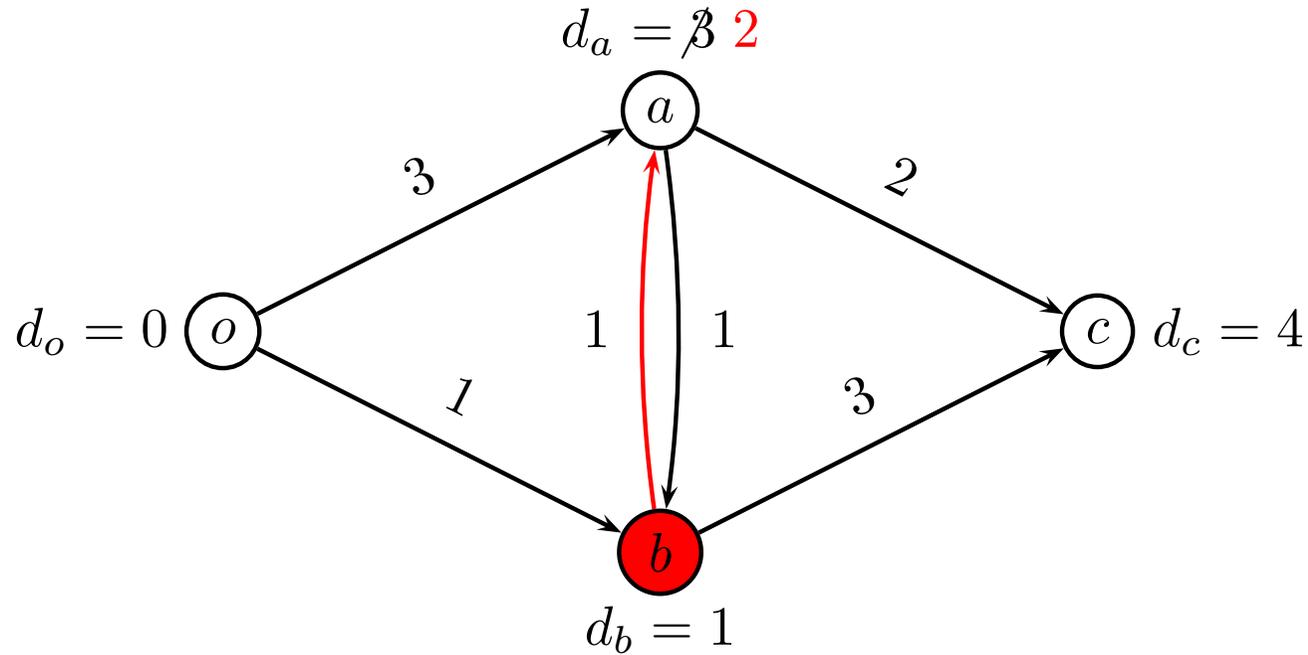
Iter	V	d_o	d_a	d_b	d_c	Traiter
0	$\{o\}$	0	∞	∞	∞	1
1	$\{a,b\}$	0	3	1	∞	a
2	$\{b,c\}$	0	3	1	5	b

Exemple



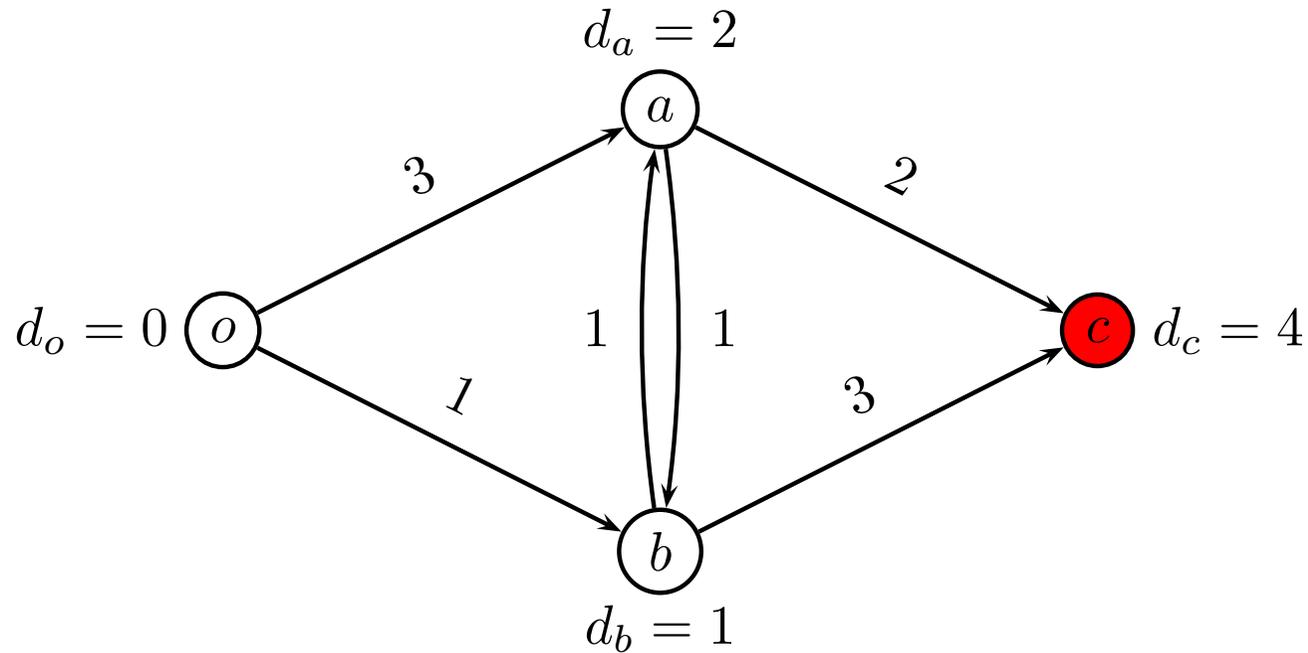
Iter	V	d_o	d_a	d_b	d_c	Traiter
0	$\{o\}$	0	∞	∞	∞	o
1	$\{a, b\}$	0	3	1	∞	a
2	$\{b, c\}$	0	3	1	5	b
3	$\{c\}$	0	3	1	4	

Exemple



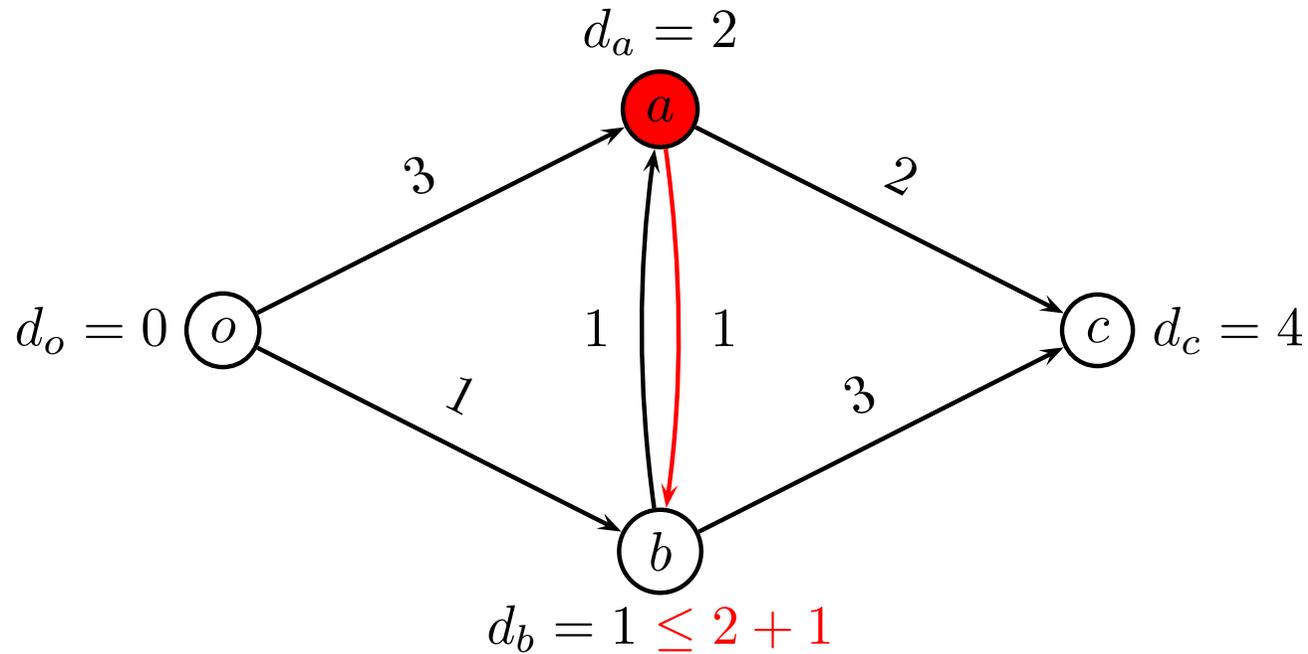
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c

Exemple



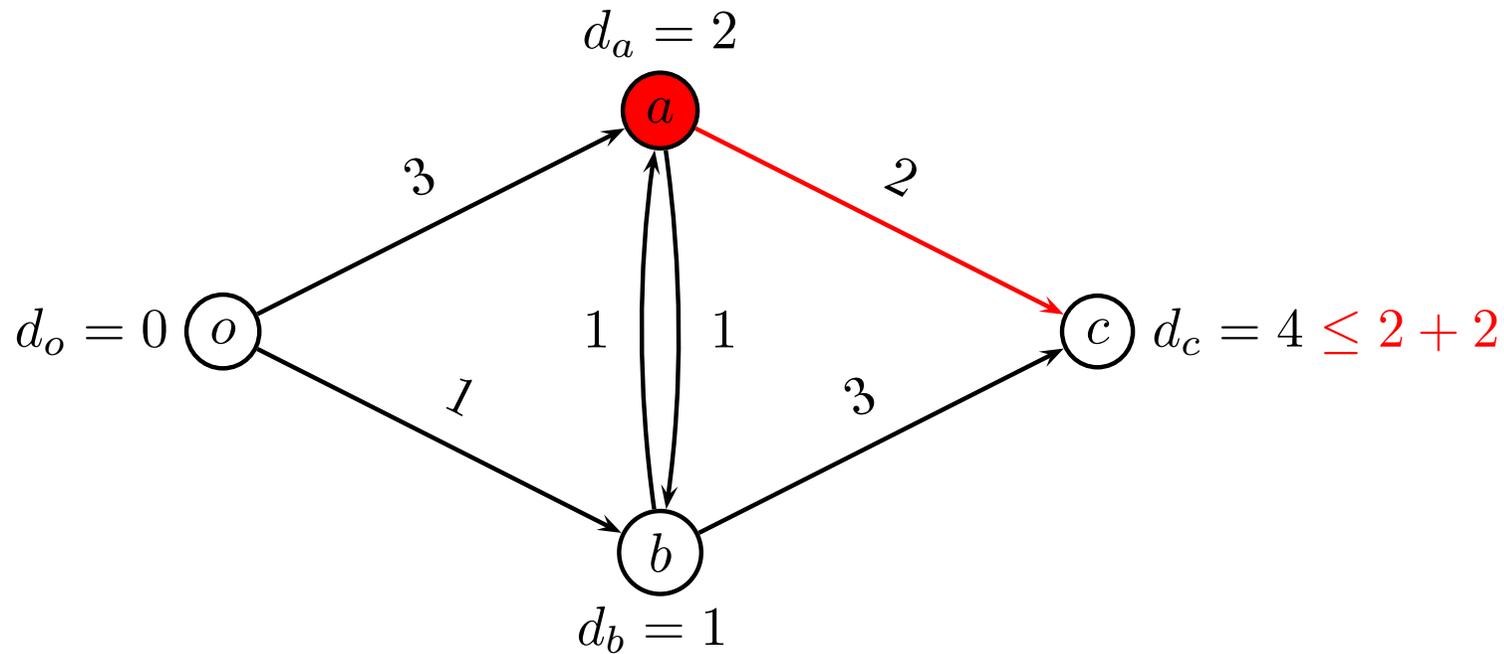
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c
4	$\{a\}$	0	—	2	b	1	o	4	b	a

Exemple



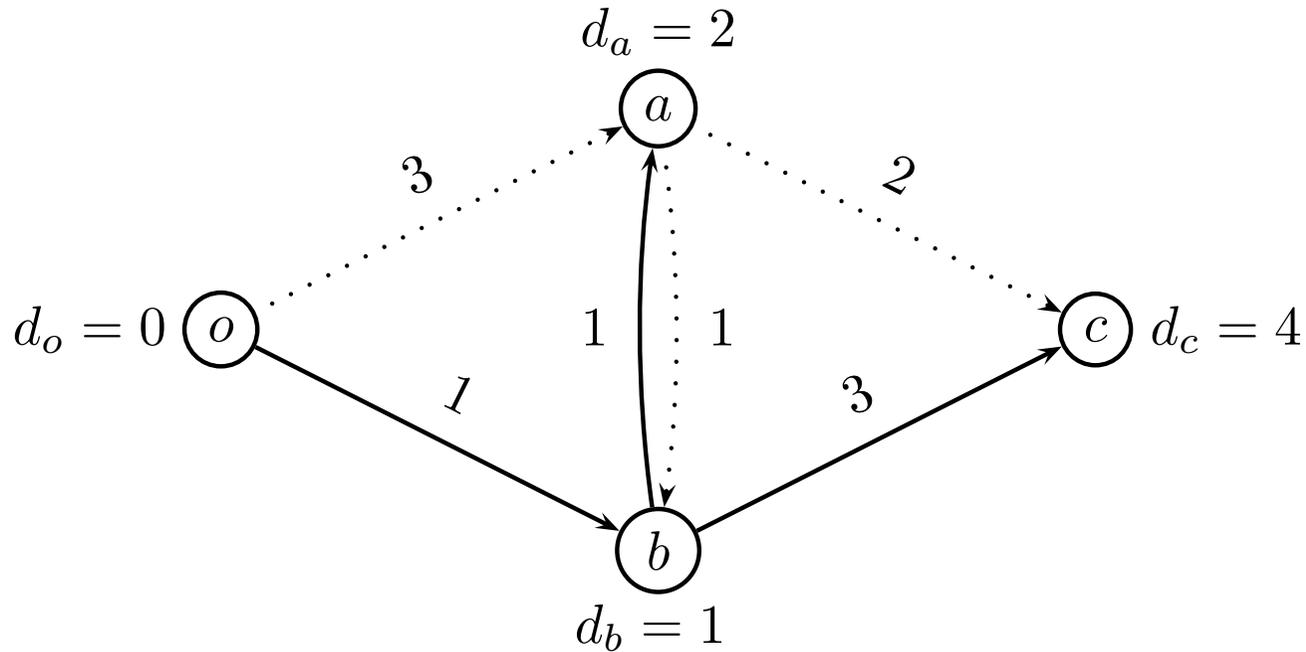
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	{o}	0	—	∞	—	∞	—	∞	—	o
1	{a,b}	0	—	3	o	1	o	∞	—	a
2	{b,c}	0	—	3	o	1	o	5	a	b
3	{c,a}	0	—	2	b	1	o	4	b	c
4	{a}	0	—	2	b	1	o	4	b	a
5	{}	0	—	2	b	1	o	4	b	

Exemple



Iter	V	d_o		d_a		d_b		d_c		Traiter
0	{ o }	0	—	∞	—	∞	—	∞	—	o
1	{ a, b }	0	—	3	o	1	o	∞	—	a
2	{ b, c }	0	—	3	o	1	o	5	a	b
3	{ c, a }	0	—	2	b	1	o	4	b	c
4	{ a }	0	—	2	b	1	o	4	b	a
5	{ }	0	—	2	b	1	o	4	b	

Exemple



Iter	V	d_o		d_a		d_b		d_c		Traiter
0	{o}	0	—	∞	—	∞	—	∞	—	o
1	{a,b}	0	—	3	o	1	o	∞	—	a
2	{b,c}	0	—	3	o	1	o	5	a	b
3	{c,a}	0	—	2	b	1	o	4	b	c
4	{a}	0	—	2	b	1	o	4	b	a
5	{}	0	—	2	b	1	o	4	b	

Propriétés à la fin de chaque itération

- Si $d_i < \infty$, alors d_i est la longueur d'un chemin reliant o à i .
- Si $i \notin V$, alors
 - soit $d_i = \infty$ (le nœud n'a pas encore été atteint),
 - soit $d_j \leq d_i + a_{ij}$, $\forall j$ tel que $(i, j) \in A$ (les arcs sortant ont été traités).

Propriétés si l'algorithme se termine

- Pour tout nœud j tel que $d_j < \infty$,
 - $d_o = 0$;
 - d_j est la longueur du plus court chemin entre 1 et j ;
 - Équation de Bellman :

$$d_j = \min_{(i,j) \in A} d_i + a_{ij} \quad \text{si } j \neq o.$$

- $d_j = \infty$ si et seulement s'il n'y a pas de chemin reliant 1 et j .
- Dans ce cas, le graphe n'est pas connexe.
- **L'algorithme se termine si et seulement s'il n'y a aucun chemin commençant en o et contenant un circuit à coût négatif.**

Algorithme de Dijkstra

- Algorithme “générique” ne précise pas comment choisir le nœud suivant à traiter.
- Dijkstra : le nœud i à traiter est celui correspondant à la plus petite étiquette.

Algorithme

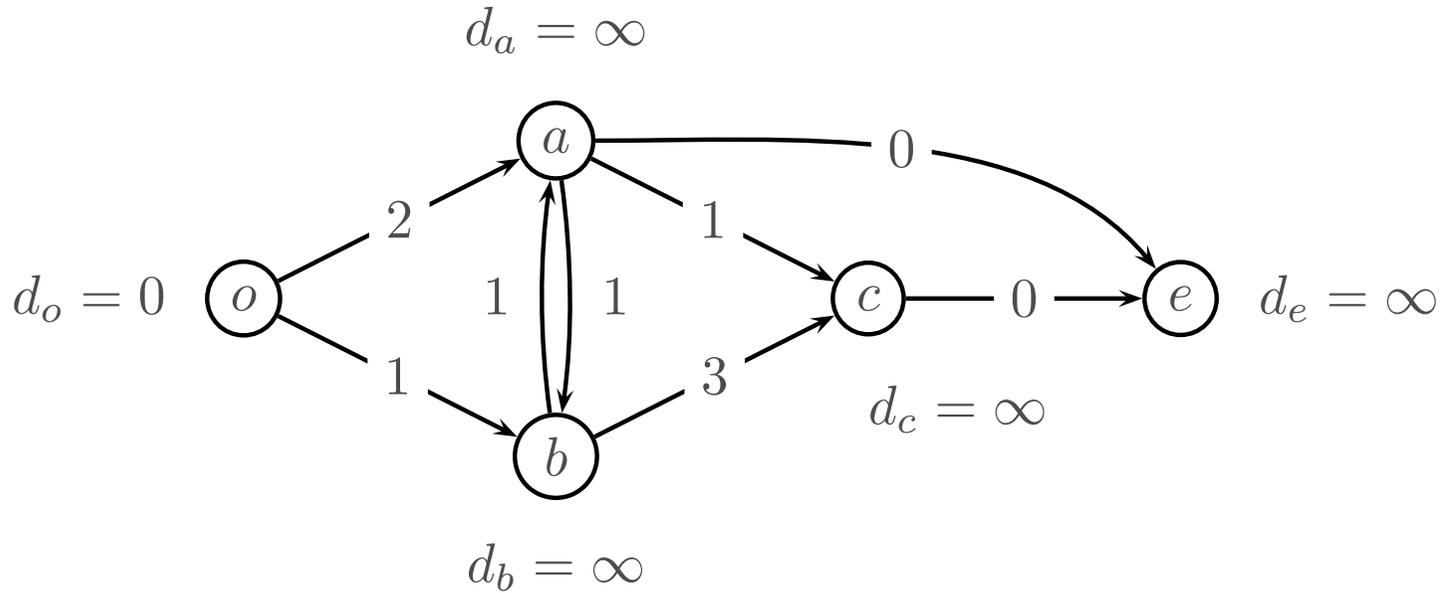
Initialisation

- Liste de nœuds: $V = \{o\}$.
- Étiquettes : $d_o = 0, d_i = +\infty, \forall i \neq o$.

Itérations Tant que $V \neq \emptyset$,

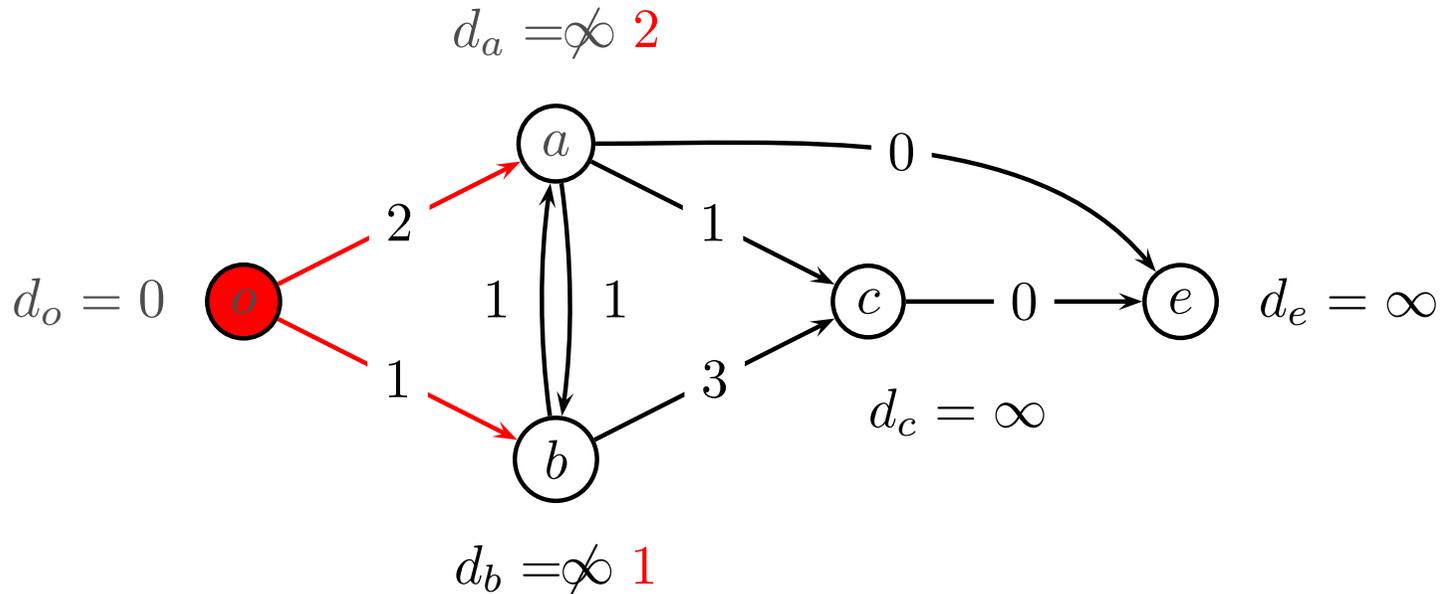
- Soit $i \in V$ tel que $d_i \leq d_j, \forall j \in V$.
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
 - Si $d_j > d_i + a_{ij}$,
 - $d_j = d_i + a_{ij}$.
 - $V = V \cup \{j\}$.

Exemple



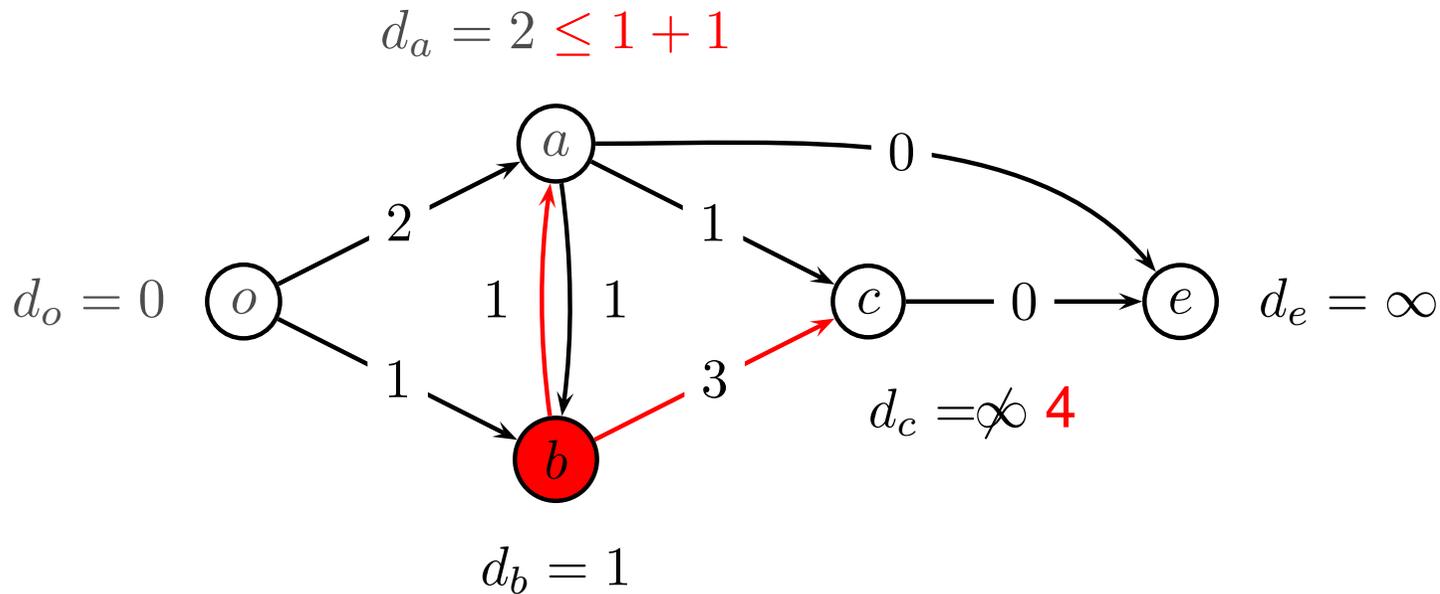
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o

Exemple



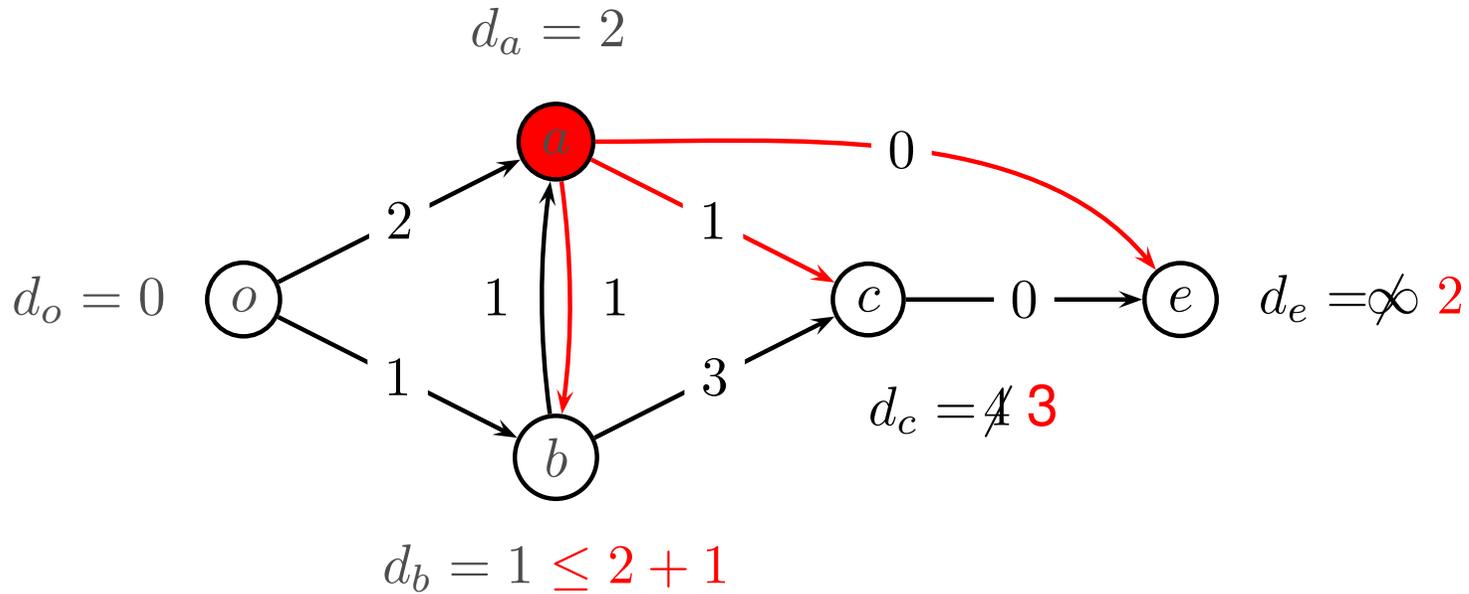
Iter	V	<i>o</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>e</i>	Traiter
0	{ <i>o</i> }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	<i>o</i>
1	{ <i>a, b</i> }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	∞ (-)	∞ (-)	<i>b</i>

Exemple



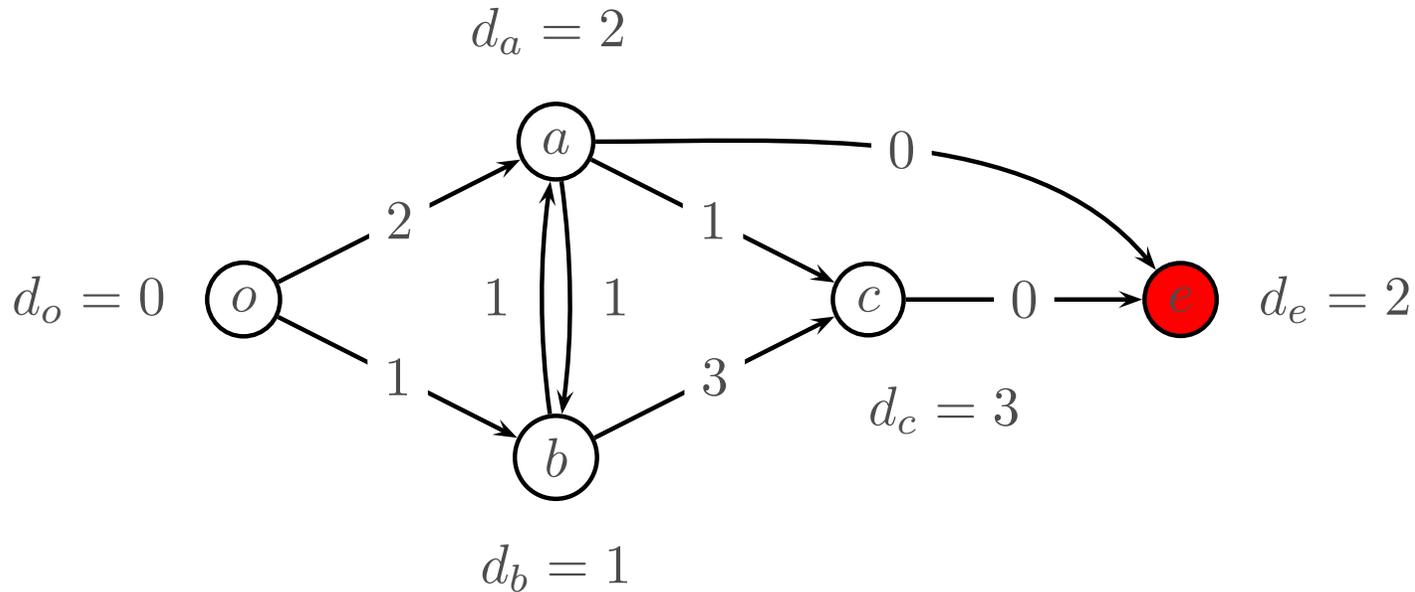
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a, b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a, c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a

Exemple



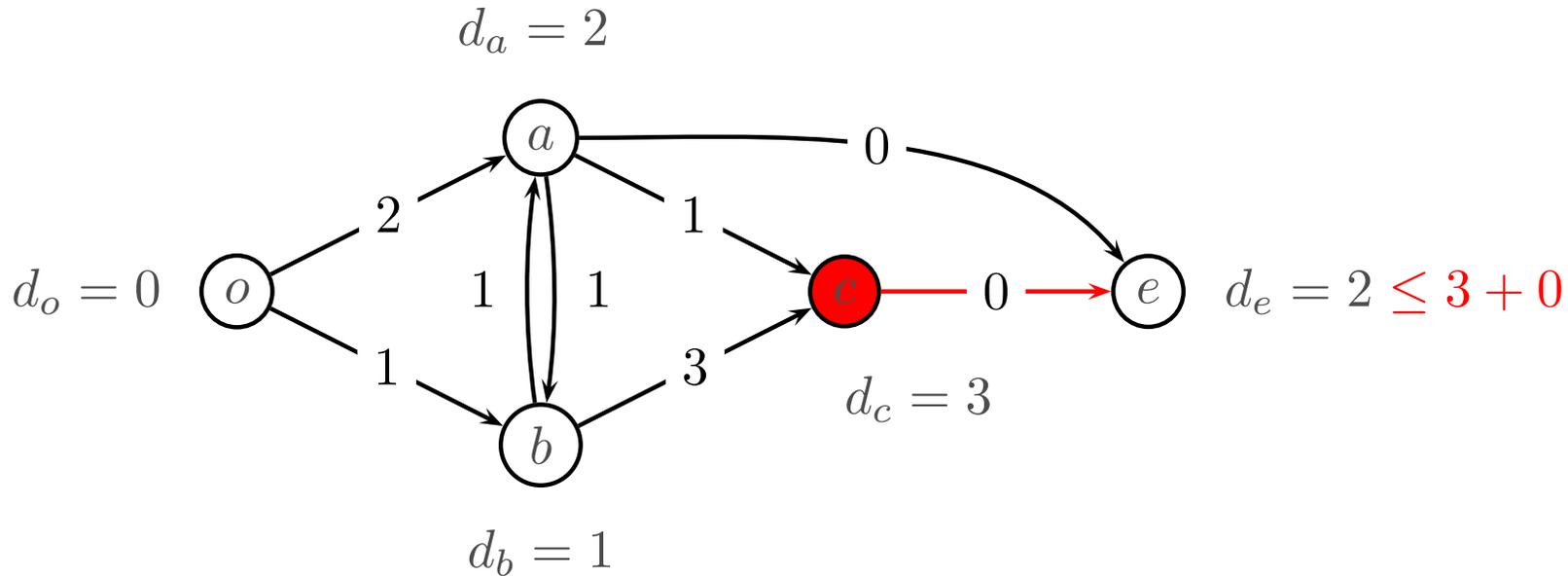
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a, b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a, c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c, e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e

Exemple



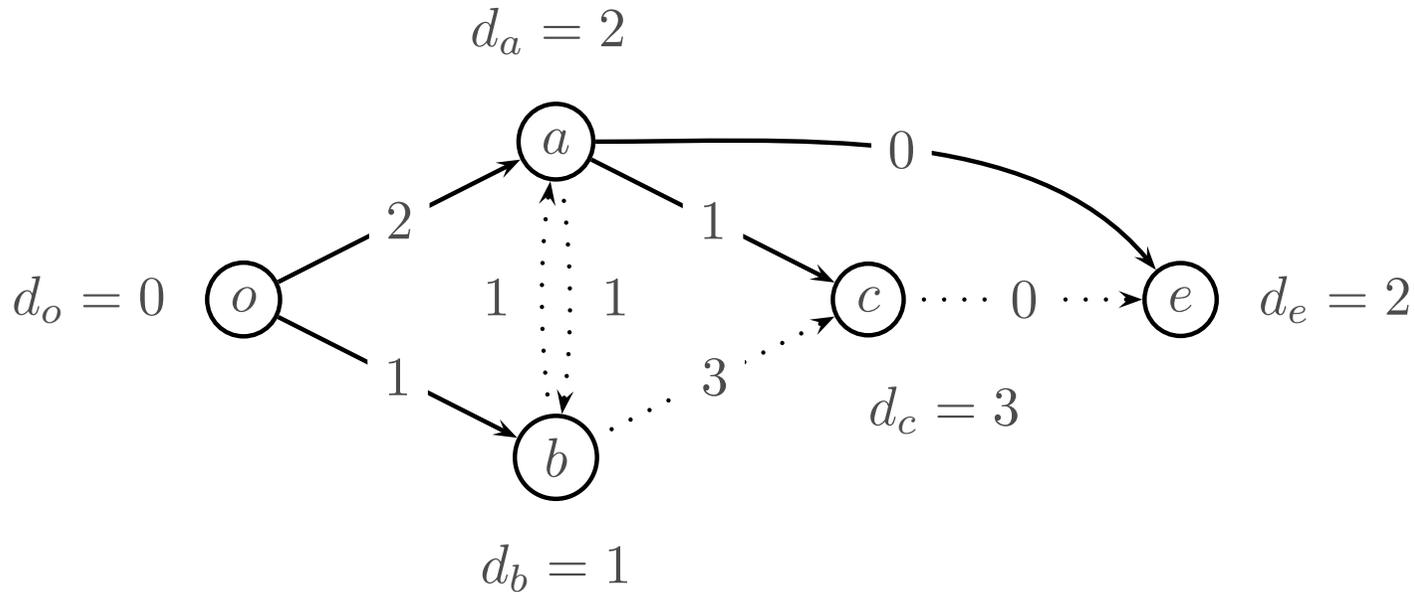
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a, b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a, c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c, e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c

Exemple



Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a, b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a, c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c, e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c
5	{ }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	

Exemple



Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a, b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a, c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c, e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c
5	{ }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	

Exemple

Iter	V	<i>o</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>e</i>	Traiter
0	{ <i>o</i> }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	<i>o</i>
1	{ <i>a</i> , <i>b</i> }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	∞ (-)	∞ (-)	<i>b</i>
2	{ <i>a</i> , <i>c</i> }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	4 (<i>b</i>)	∞ (-)	<i>a</i>
3	{ <i>c</i> , <i>e</i> }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	3 (<i>a</i>)	2 (<i>a</i>)	<i>e</i>
4	{ <i>c</i> }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	3 (<i>a</i>)	2 (<i>a</i>)	<i>c</i>
5	{ }	0 (-)	2 (<i>o</i>)	1 (<i>o</i>)	3 (<i>a</i>)	2 (<i>a</i>)	

Note : Chaque nœud n'a été traité qu'une seule fois.

Algorithme de Dijkstra

- Soit l'ensemble

$$W = \{i \mid d_i < \infty \text{ et } i \notin V\}.$$

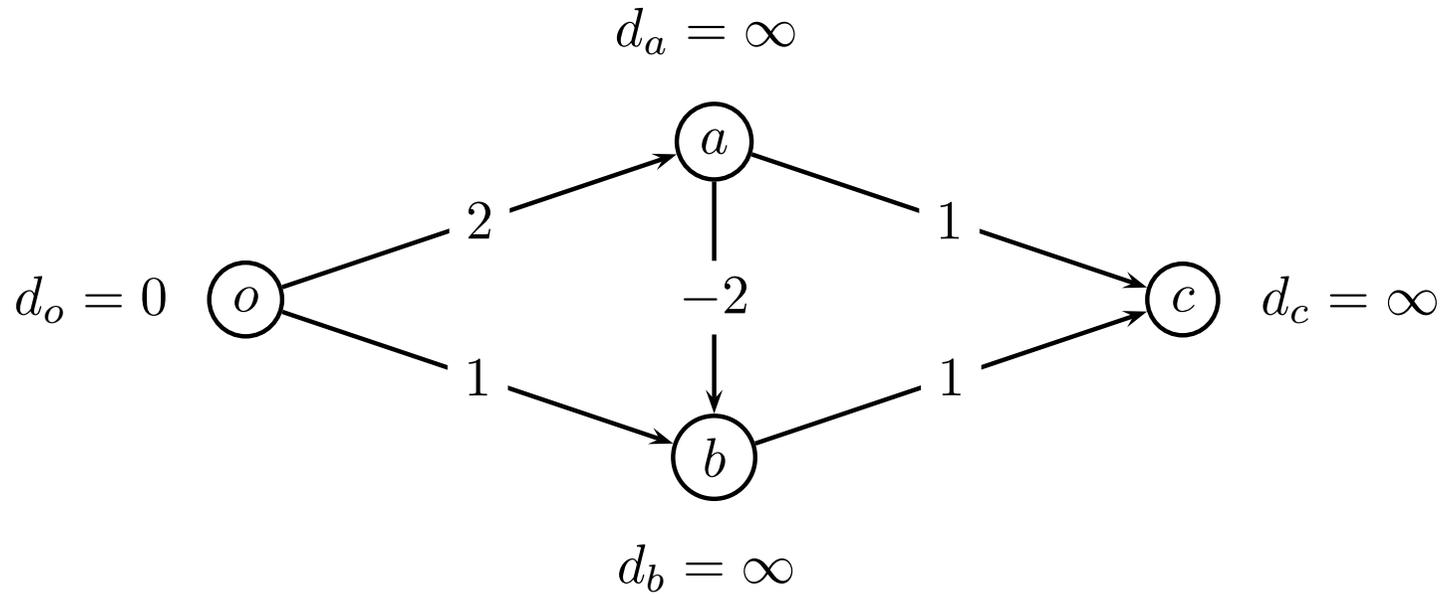
- Si les coûts sur les arcs sont non négatifs, alors à chaque itération
 - aucun nœud dans W au début de l'itération n'entre dans V lors de l'itération,
 - à la fin de l'itération, $d_i \leq d_j$ si $i \in W$ et $j \notin W$.

W : ensemble des étiquettes permanentes.

Notes

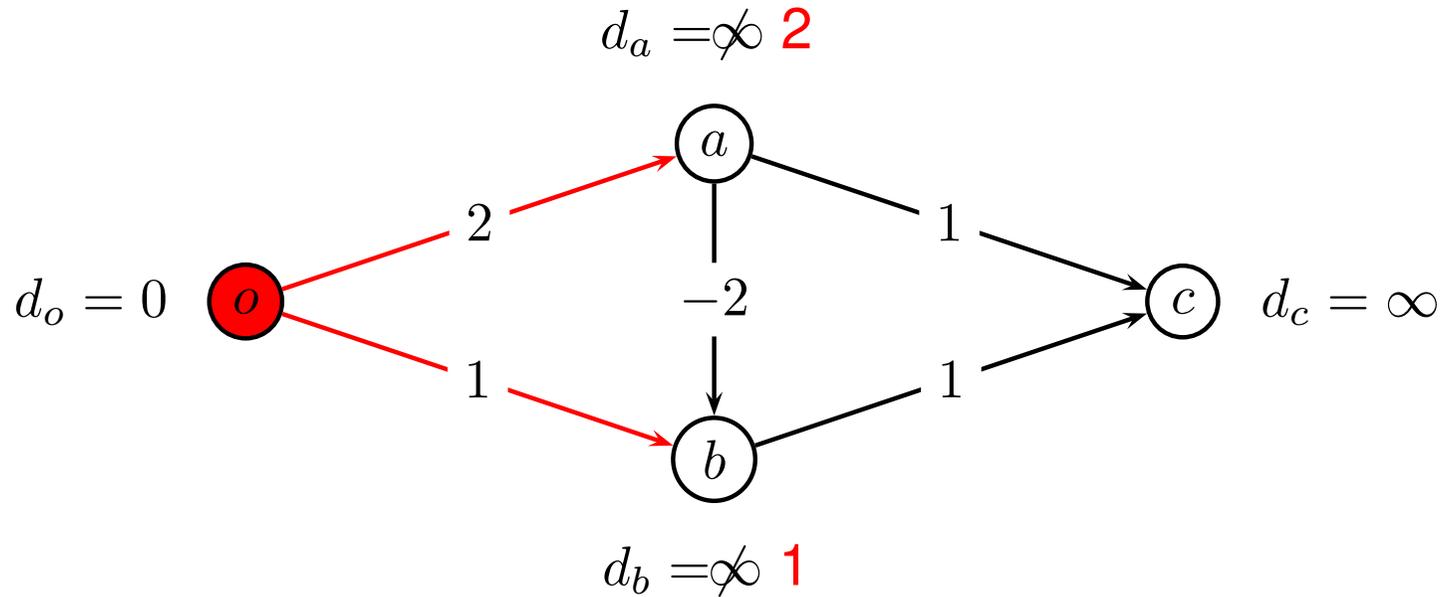
- Si l'on désire calculer le plus court chemin de o à b , on peut arrêter l'algorithme de Dijkstra dès que le nœud b est dans W .
- Si au moins un arc a un coût négatif, rien ne garantit le caractère permanent des étiquettes.

Exemple : coût négatif



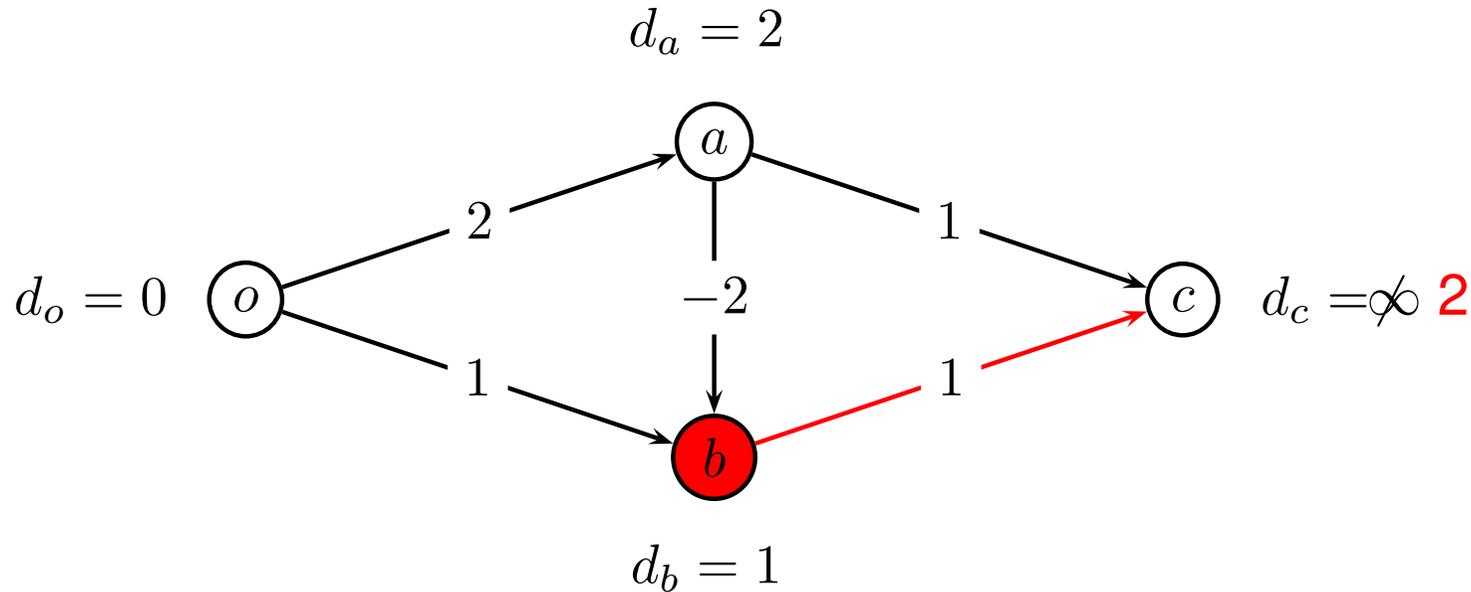
Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o

Exemple : coût négatif



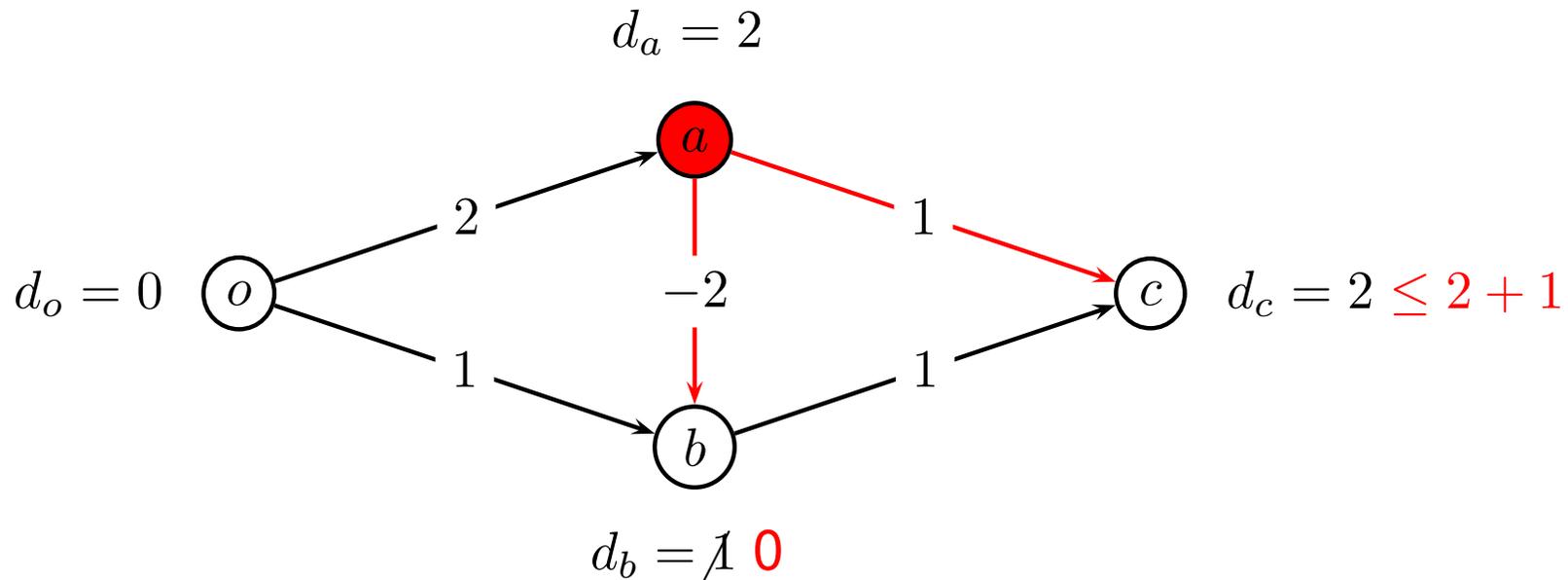
Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a,b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	b

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a, b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{a, c\}$	0 (-)	2 (o)	1 (o)	2 (b)	a

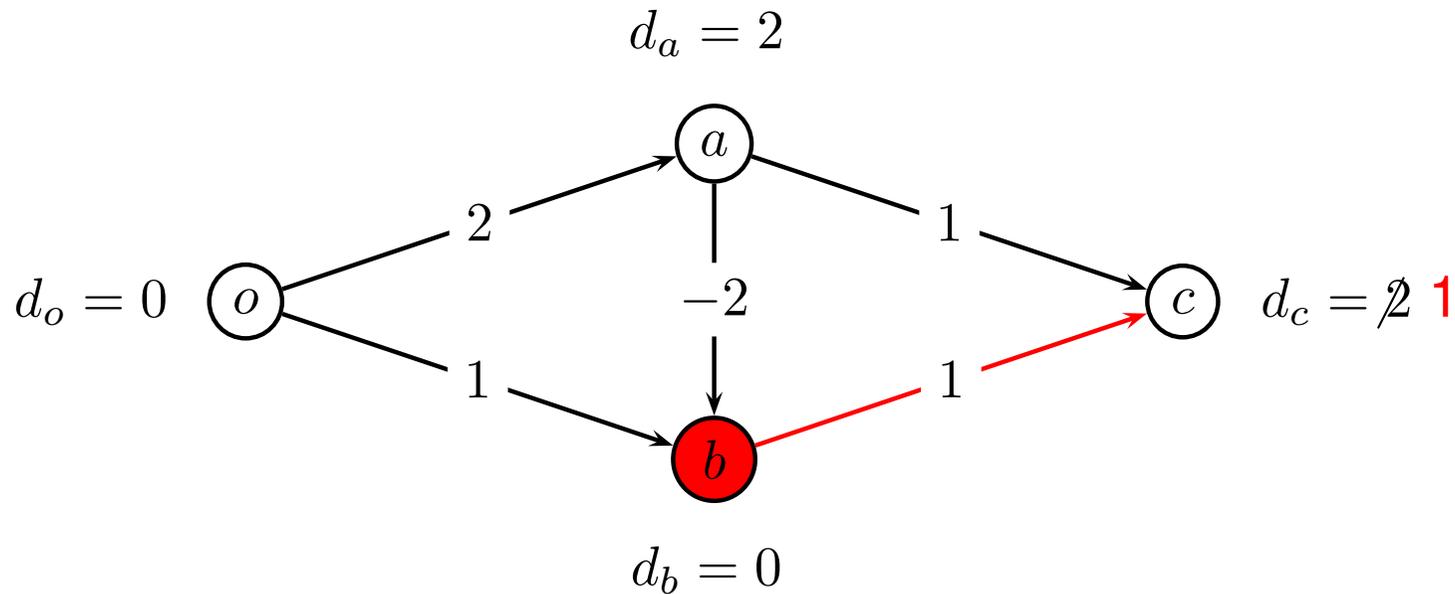
Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	{o}	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{a,b}	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	{a,c}	0 (-)	2 (o)	1 (o)	2 (b)	a
3	{b,c}	0 (-)	2 (o)	0 (a)	2 (b)	b

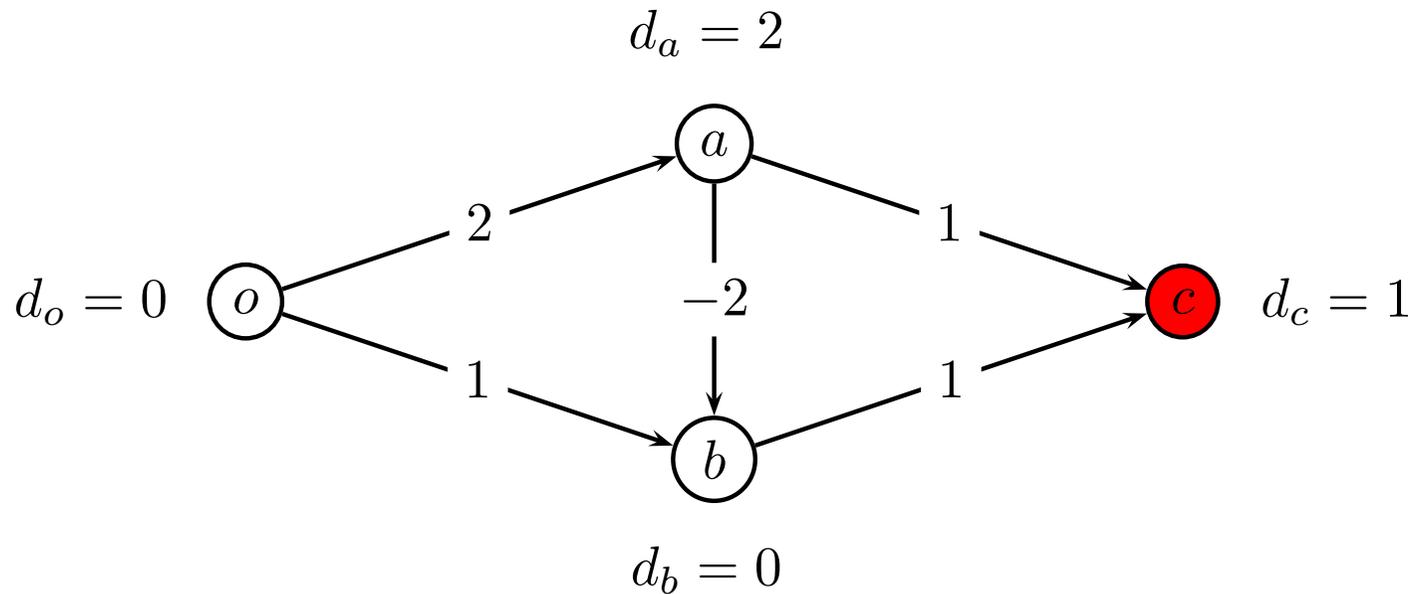
!!!

Exemple : coût négatif



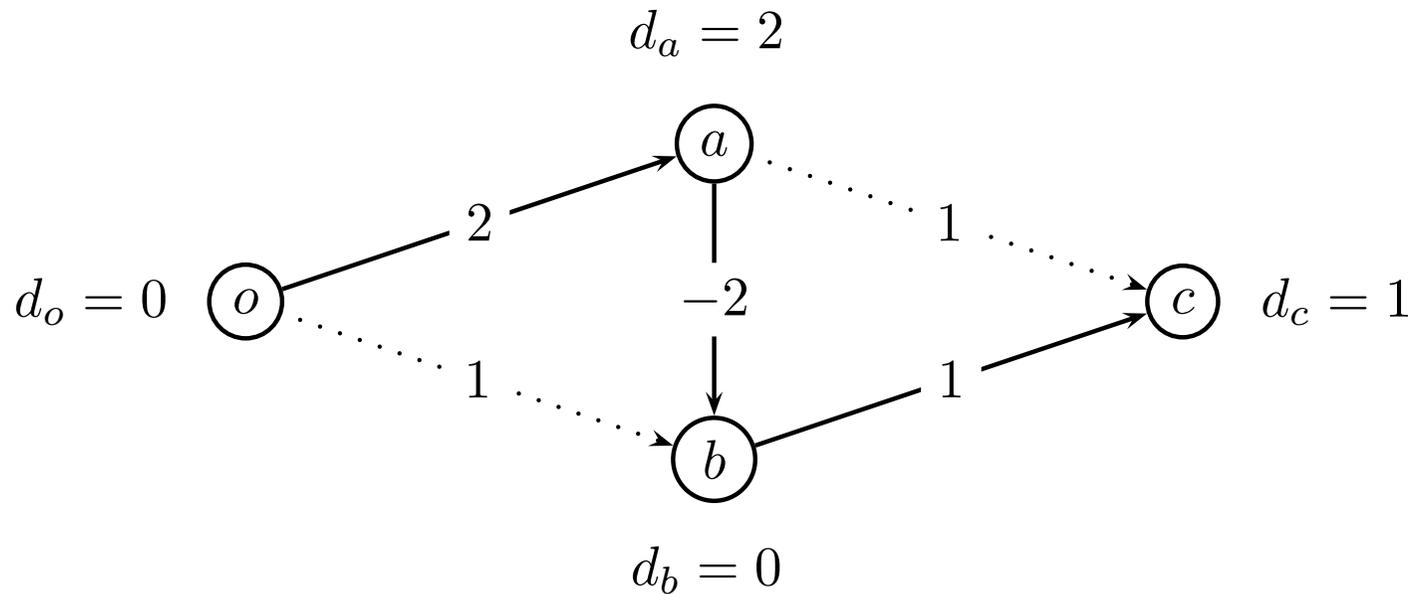
Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a,b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{a,c\}$	0 (-)	2 (o)	1 (o)	2 (b)	a
3	$\{b,c\}$	0 (-)	2 (o)	0 (a)	2 (b)	b
4	$\{c\}$	0 (-)	2 (o)	0 (a)	1 (b)	c

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a,b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{a,c\}$	0 (-)	2 (o)	1 (o)	2 (b)	a
3	$\{b,c\}$	0 (-)	2 (o)	0 (a)	2 (b)	b
4	$\{c\}$	0 (-)	2 (o)	0 (a)	1 (b)	c
5	$\{\}$	0 (-)	2 (o)	0 (a)	1 (b)	

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a,b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{a,c\}$	0 (-)	2 (o)	1 (o)	2 (b)	a
3	$\{b,c\}$	0 (-)	2 (o)	0 (a)	2 (b)	b
4	$\{c\}$	0 (-)	2 (o)	0 (a)	1 (b)	c
5	$\{\}$	0 (-)	2 (o)	0 (a)	1 (b)	

Dijkstra et coût négatif

- L'algorithme converge.
- Mais le concept d'étiquettes permanentes n'est plus pertinent.
- Toute implémentation basée sur cette propriété ne peut fonctionner qu'avec des coûts positifs.