Algorithme du plus court chemin

Michel Bierlaire

michel.bierlaire@epfl.ch

EPFL - Laboratoire Transport et Mobilité - ENAC

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Le plus court chemin

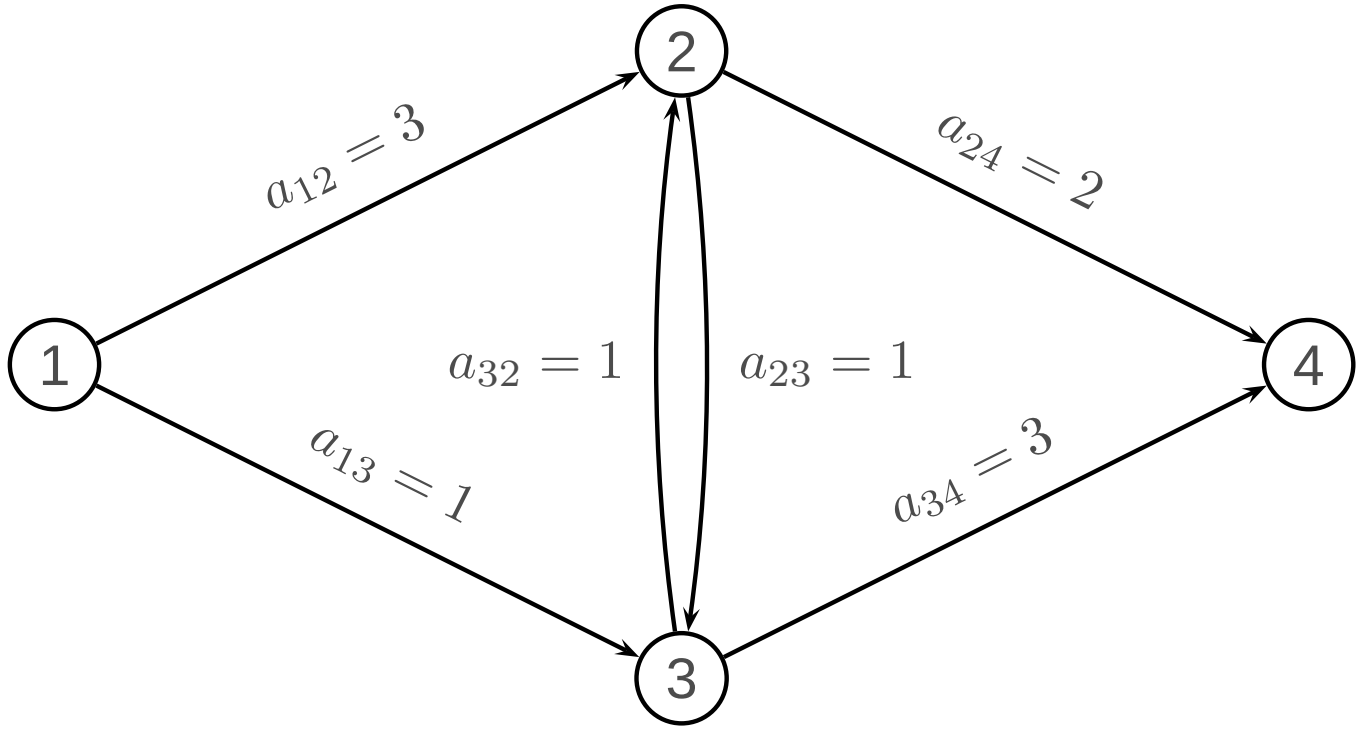
- Le problème du plus court chemin consiste à déterminer le chemin de coût minimum reliant un nœud a à un nœud b.
- On peut le voir comme un problème de transbordement.
- Cependant, il est plus efficace d'utiliser des algorithmes spécialisés.

Le plus court chemin

Problème :

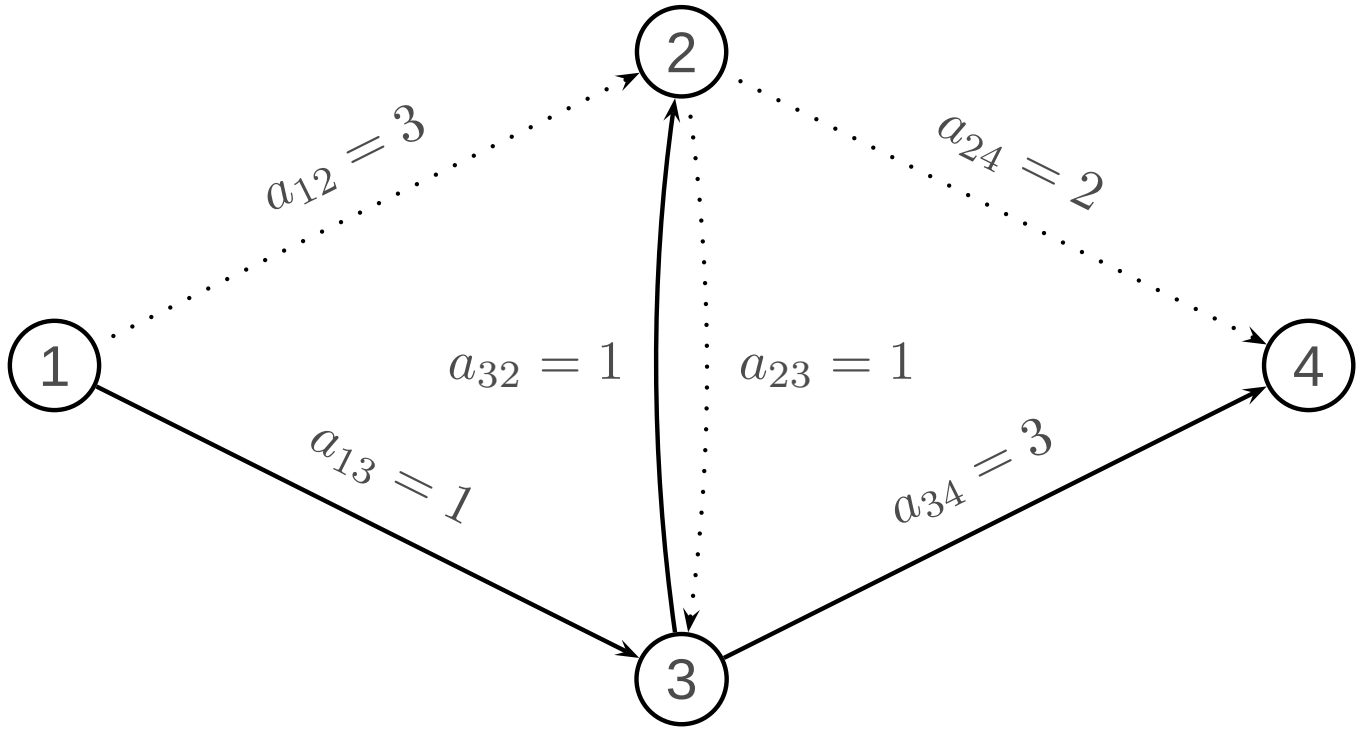
- Soit un réseau $G = (N, A)$.
- Un coût a_{ij} est associé à chaque arc $(i, j) \in A$:
  - distance,
  - temps de trajet,
  - etc.
- Soit un nœud appelé *origine*. Par convention, ce sera le nœud 1.
- Nous cherchons le chemin de coût minimum reliant le nœud 1 à n'importe quel autre nœud du réseau.

Le plus court chemin



Le plus court chemin

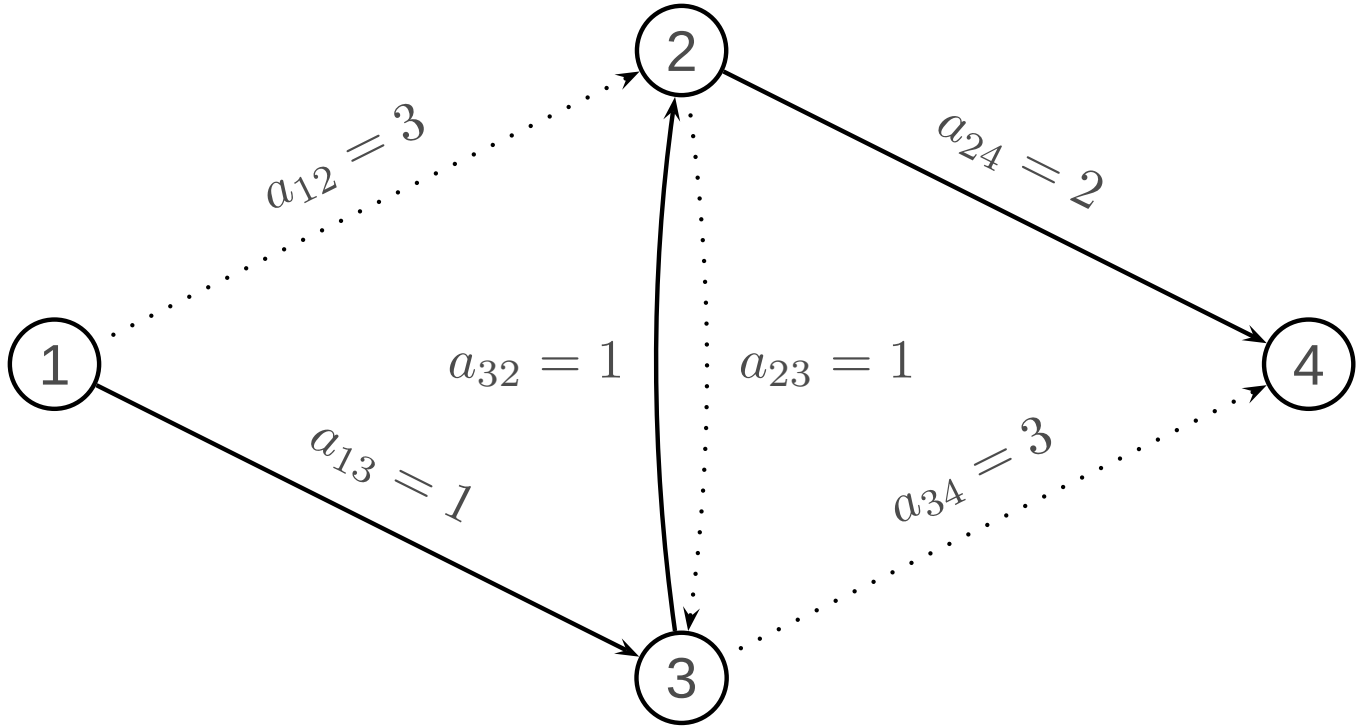
- La solution est un arbre.



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 un nœud 1 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 1 et ℓ.

Conditions d'optimalité

Preuve:

- P est composé d'arcs

$$(1, i_1), (i_1, i_2), \ldots, (i_k, \ell)$$

- Longueur de P:

$$L(P) = a_{1i_1} + a_{i_1 i_2} + \cdots + a_{i_k \ell}$$

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

$$L(P) = (d_{i_1} - d_1) + (d_{i_2} - d_{i_1}) + \cdots + (d_\ell - d_{i_k}) = d_\ell - d_1.$$

Conditions d'optimalité

- Soit Q un chemin quelconque entre 1 et ℓ.
- Q est composé d'arcs

$$(1, j_1), (j_1, j_2), \ldots, (j_n, \ell)$$

- Longueur de Q:

$$L(Q) = a_{1j_1} + a_{j_1 j_2} + \cdots + a_{j_n \ell}$$

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

$$L(Q) \geq (d_{j_1} - d_1) + (d_{j_2} - d_{j_1}) + \cdots + (d_\ell - d_{j_m}) = d_\ell - d_1 = L(P).$$

- La longueur de P est donc plus courte que la longueur de Q.
- Comme Q est arbitraire, P est le plus court chemin entre 1 et ℓ.

CQFD

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

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

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

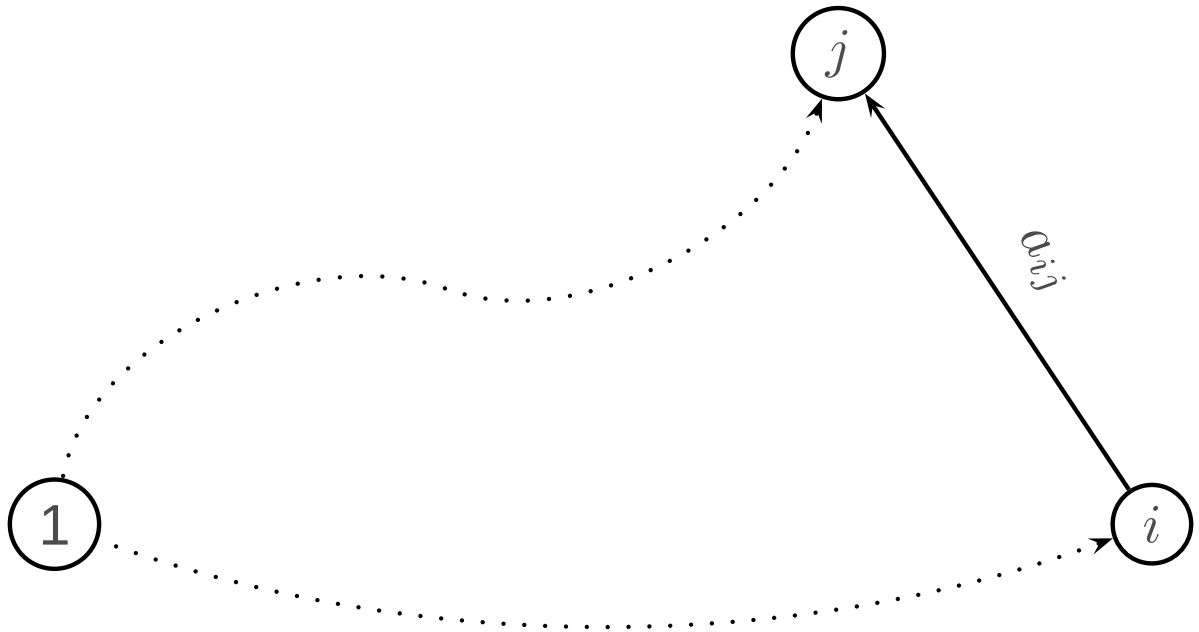
- On met à jour l'étiquette de j:

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

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

Interprétation

- d_i : longueur d'un chemin entre le nœud 1 et le nœud i.
- Si $d_j > d_i + a_{ij}$, chemin $1 \to i \to j$ plus court que le chemin $1 \to j$.



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

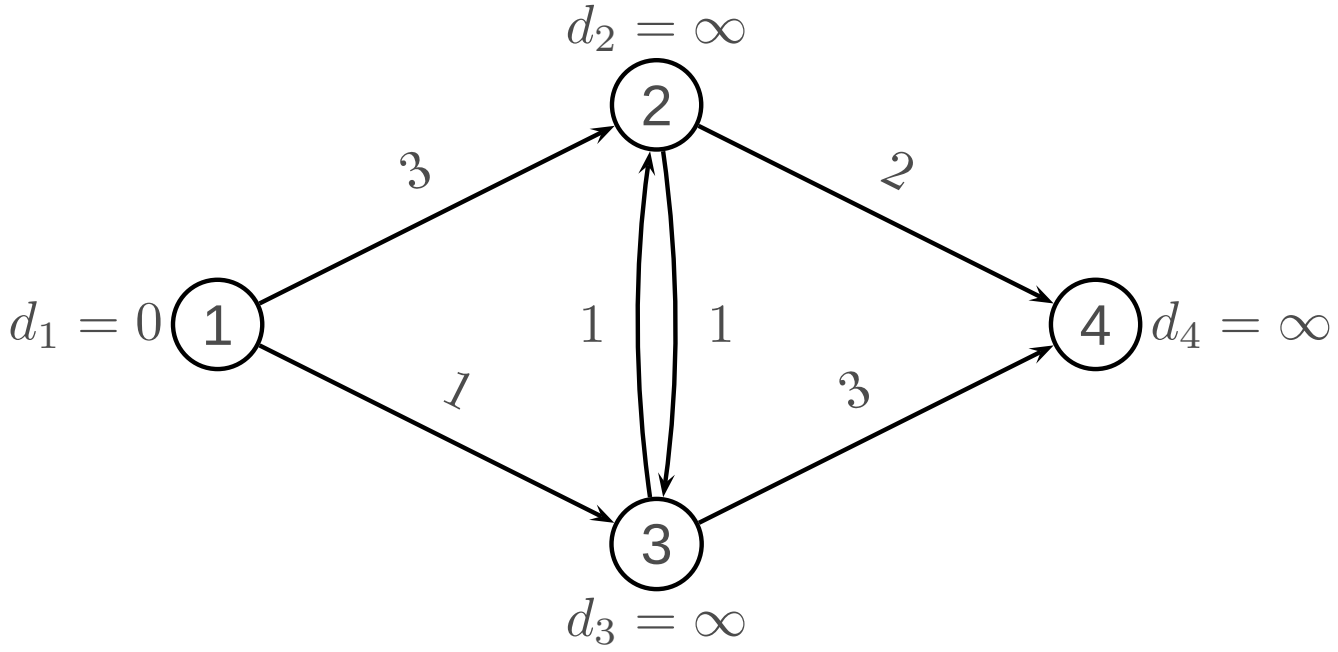
Initialisation

- Liste de nœuds: $V = \{1\}$.
- Étiquettes : $d_1 = 0$, $d_i = +\infty$, $\forall i \neq 1$.

Itérations Tant que $V \neq \emptyset$,

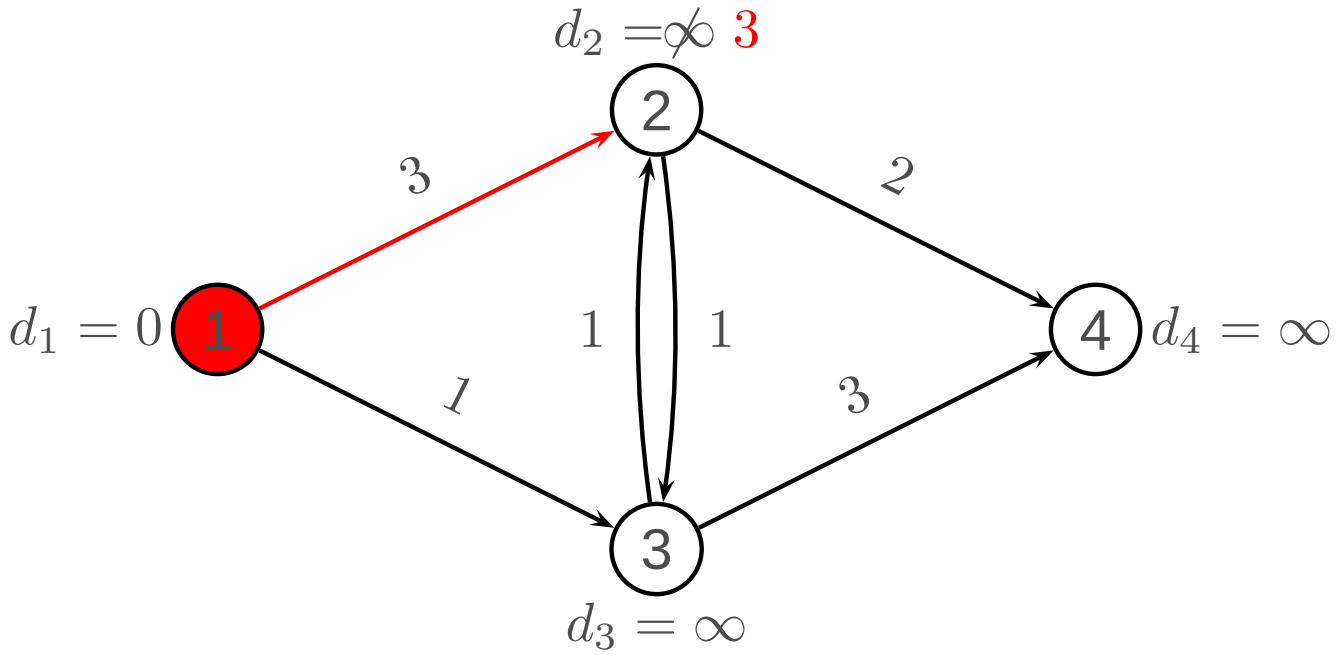
- Choisir i dans V.
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
  - Si $d_j > d_i + a_{ij}$,
    - $d_j = d_i + a_{ij}$.
    - $V = V \cup \{j\}$.

Exemple



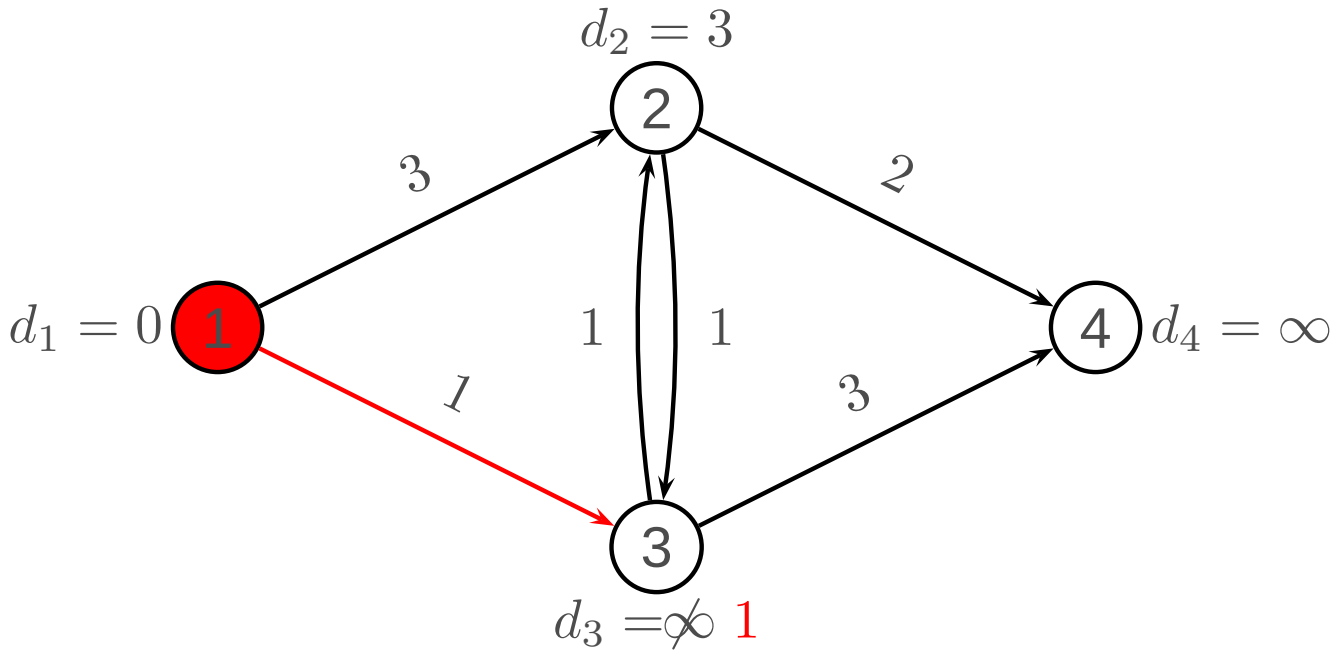
Iter	V	d_1	d_2	d_3	d_4	Traiter
0	{ 1 }	0 —	∞ —	∞ —	∞ —	1

Exemple



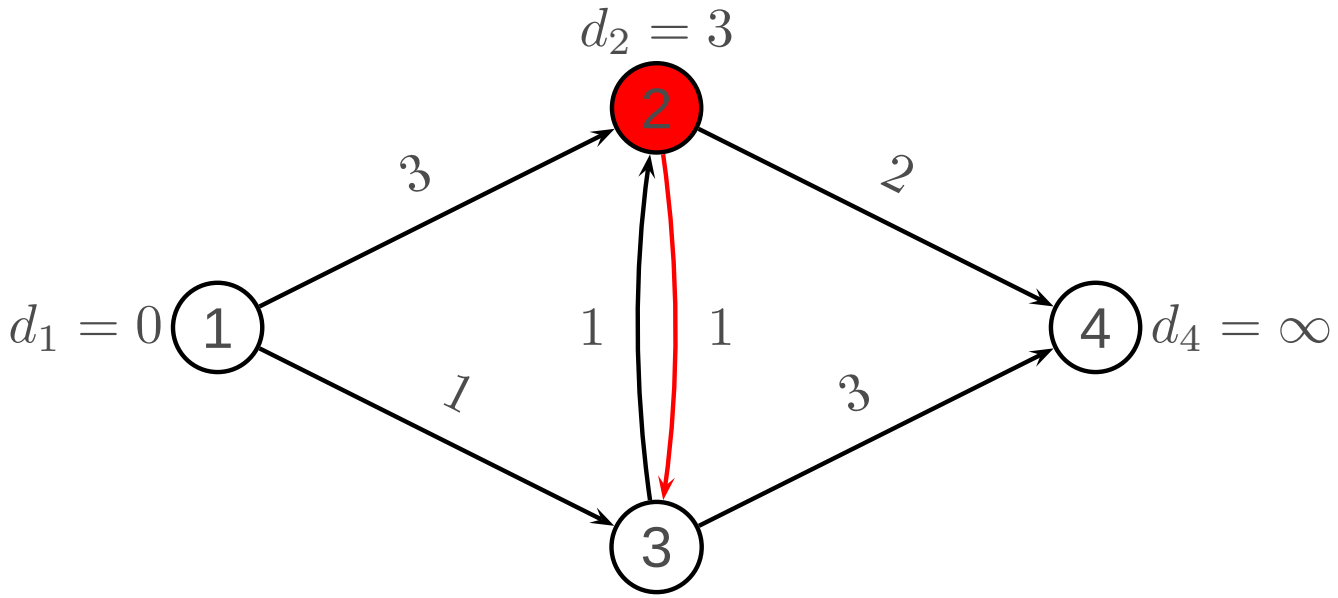
Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2 }	0	—	3	1	∞	—	∞	—	

Exemple



Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2

Exemple

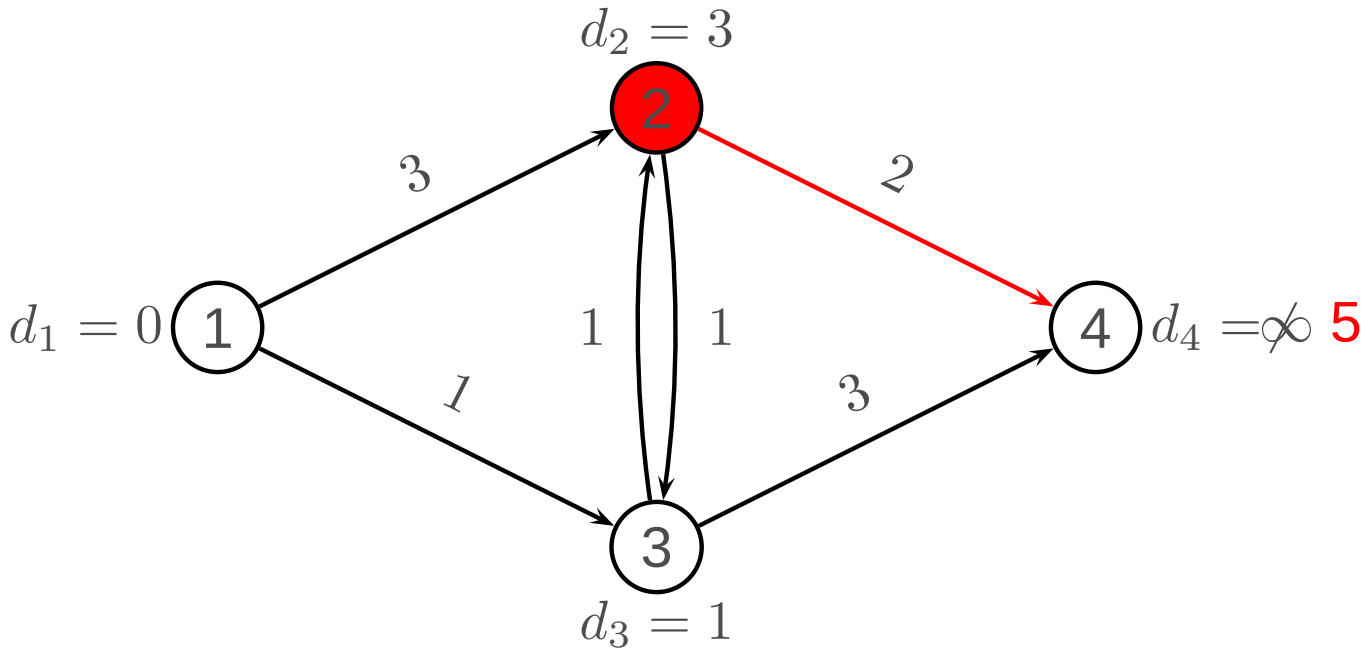


Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3 }	0	—	3	1	1	1	∞	—	

TRANSP-OR

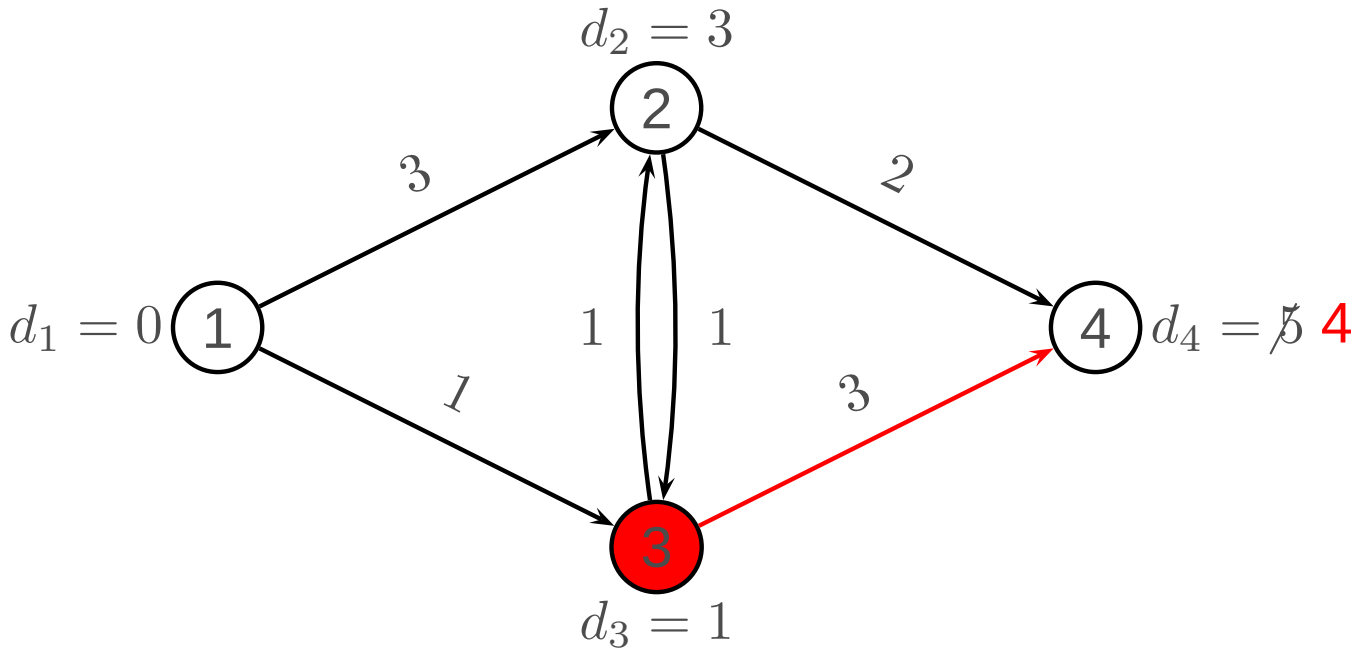
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple



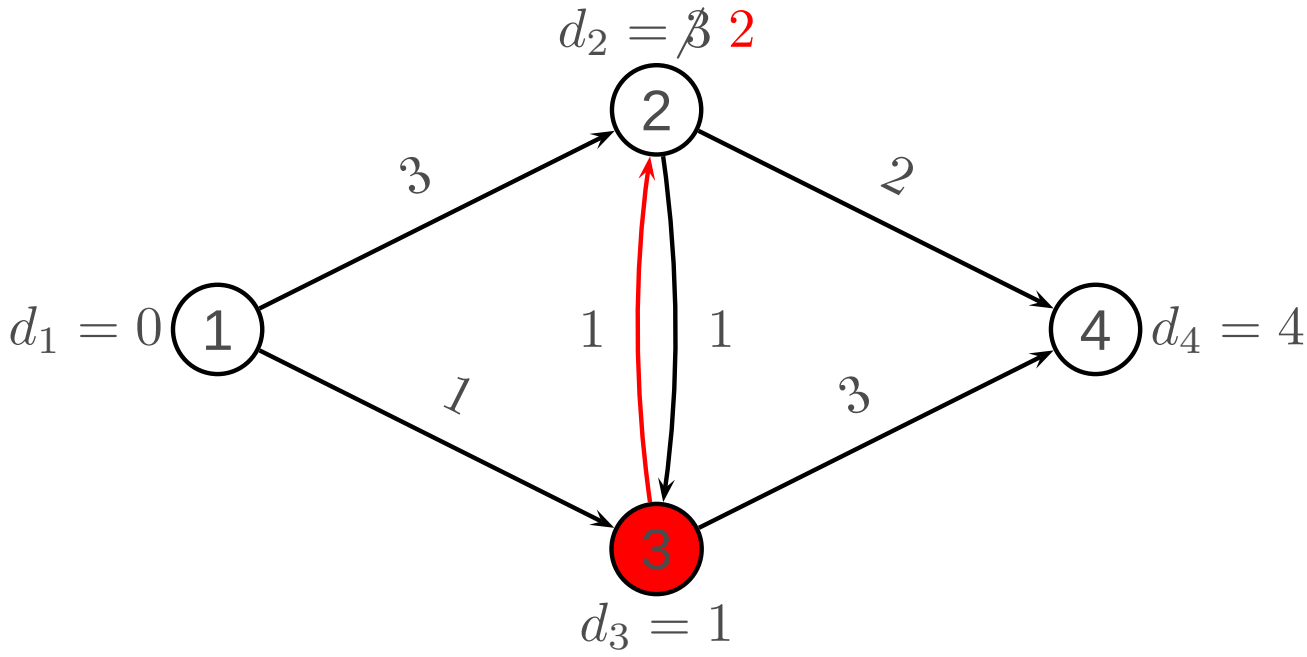
Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3

Exemple



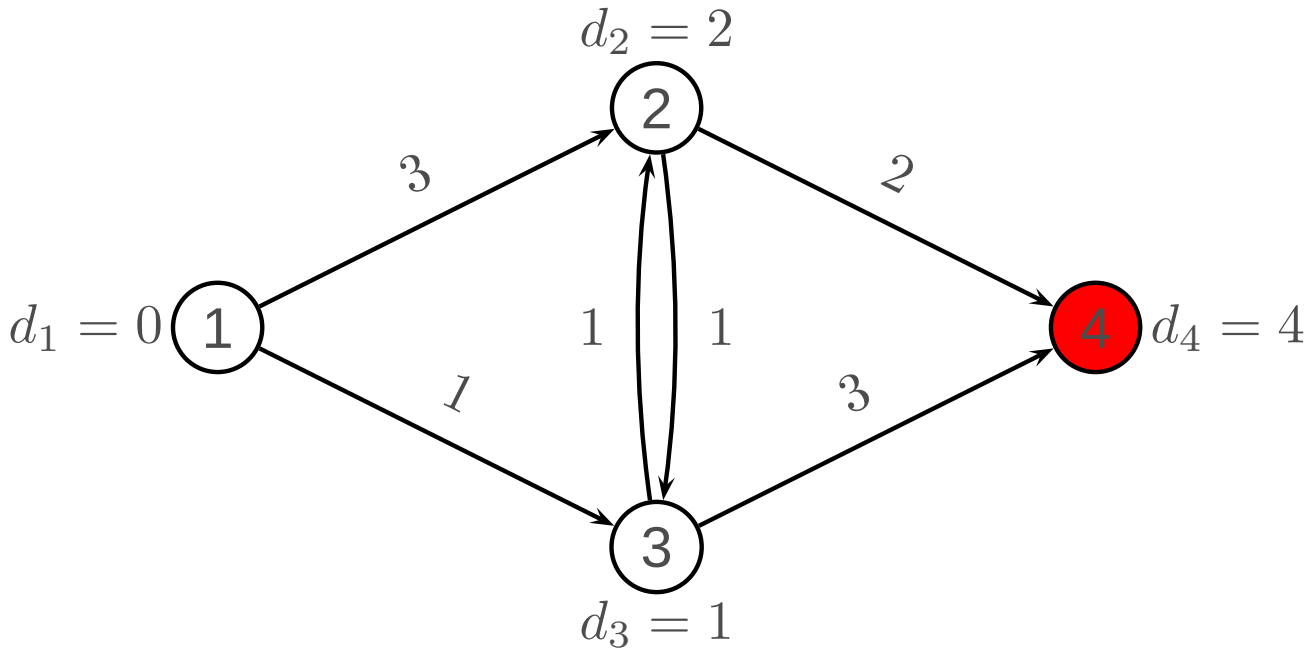
Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4 }	0	—	3	1	1	1	4	3	

Exemple



Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4,2 }	0	—	2	3	1	1	4	3	4

Exemple

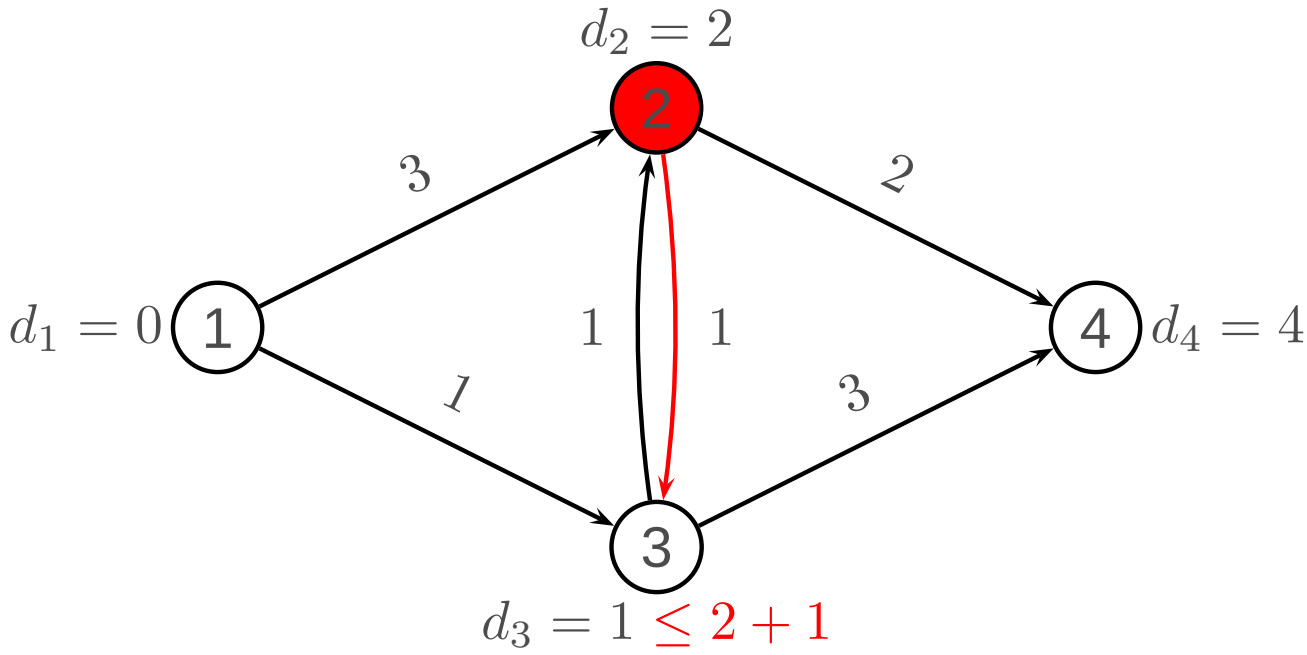


Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4,2 }	0	—	2	3	1	1	4	3	4
4	{ 2 }	0	—	2	3	1	1	4	3	2

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple

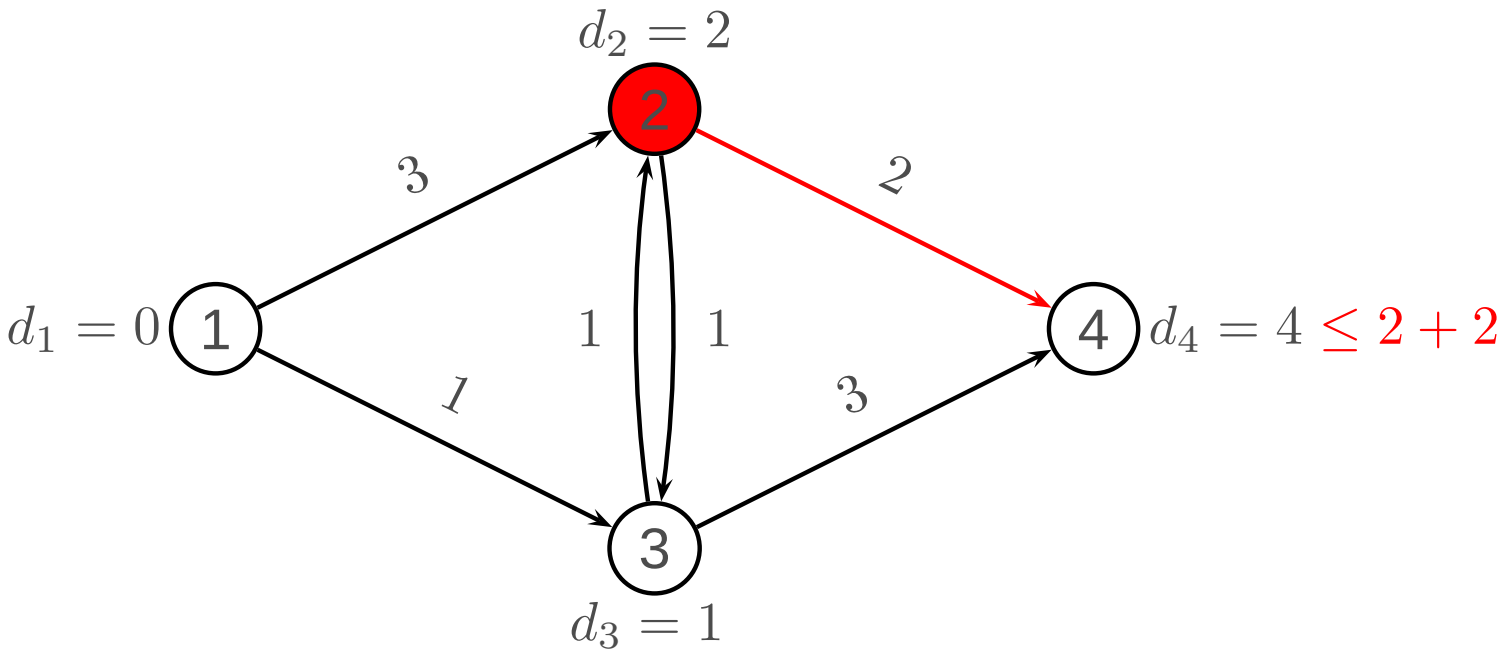


Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4,2 }	0	—	2	3	1	1	4	3	4
4	{ 2 }	0	—	2	3	1	1	4	3	2
5	{ }	0	—	2	3	1	1	4	3	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple

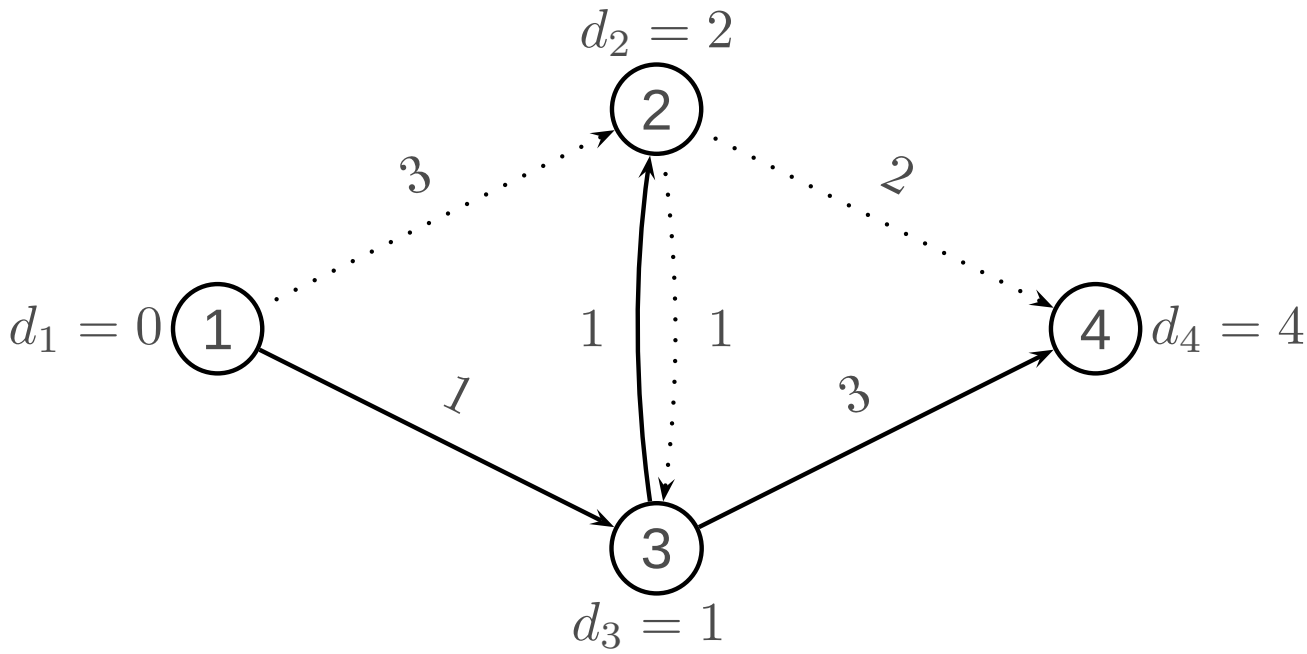


Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4,2 }	0	—	2	3	1	1	4	3	4
4	{ 2 }	0	—	2	3	1	1	4	3	2
5	{ }	0	—	2	3	1	1	4	3	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple



Iter	V	d_1		d_2		d_3		d_4		Traiter
0	{ 1 }	0	—	∞	—	∞	—	∞	—	1
1	{ 2,3 }	0	—	3	1	1	1	∞	—	2
2	{ 3,4 }	0	—	3	1	1	1	5	2	3
3	{ 4,2 }	0	—	2	3	1	1	4	3	4
4	{ 2 }	0	—	2	3	1	1	4	3	2
5	{ }	0	—	2	3	1	1	4	3	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Propriétés à la fin de chaque itération

- Si $d_i < \infty$, alors d_i est la longueur d'un chemin reliant 1 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 $d_j < \infty$,
  - $d_1 = 0$;
  - d_j est la longueur du plus court chemin entre 1 et j;
  - Équation de Bellman :

$$d_j = \min_{(i,j) \in A} d_i + a_{ij} \quad \text{si } j \neq 1.$$

- $d_j = \infty$ 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 1 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

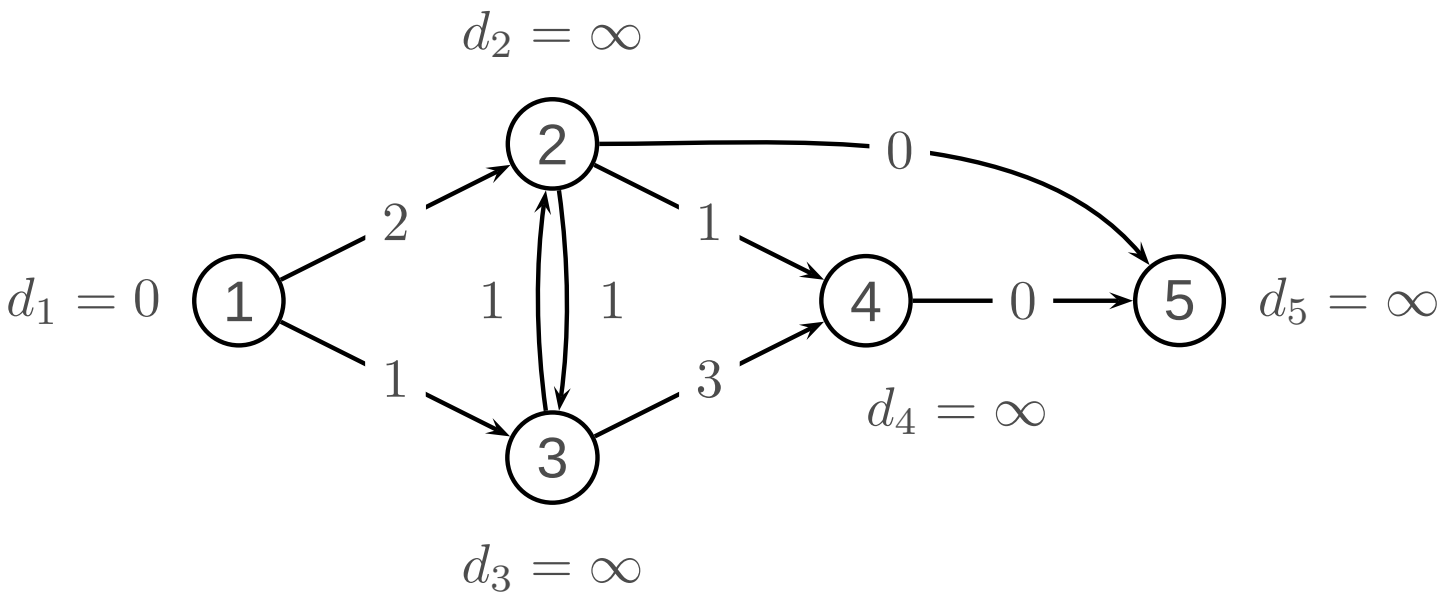
Initialisation

- Liste de nœuds: $V = \{1\}$.
- Étiquettes : $d_1 = 0$, $d_i = +\infty$, $\forall i \neq 1$.

Itérations Tant que $V \neq \emptyset$,

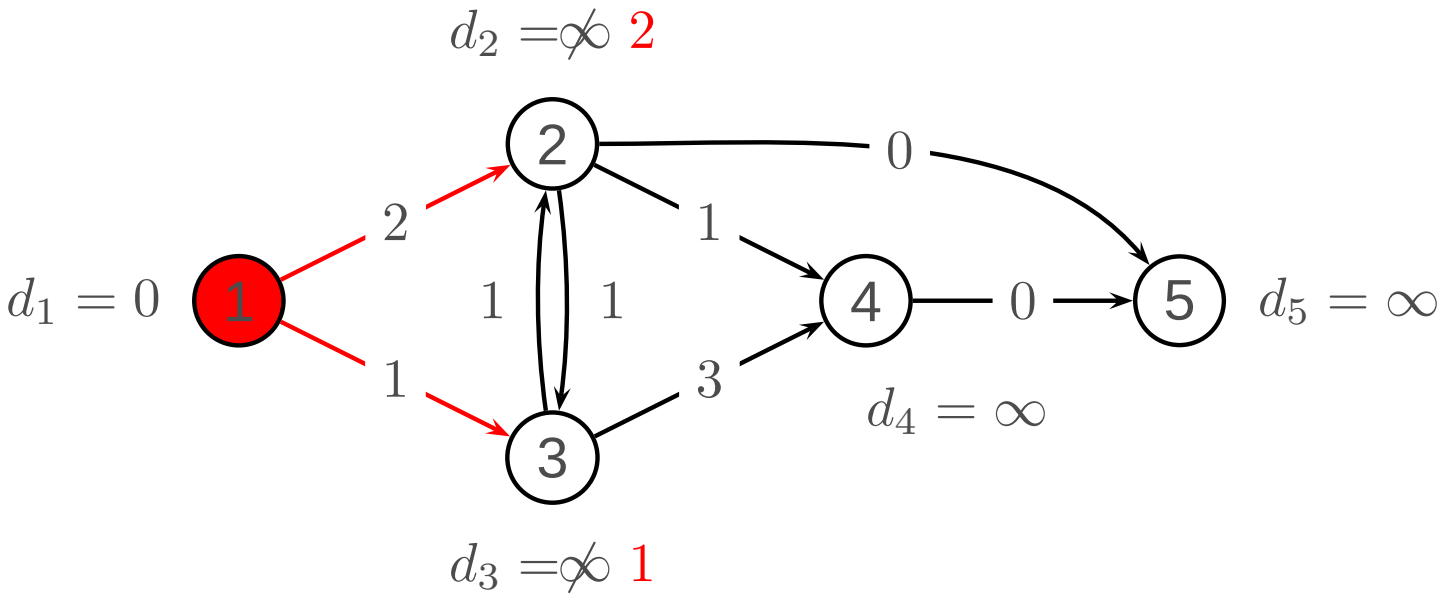
- Soit $i \in V$ tel que $d_i \leq d_j$, $\forall j \in V$.
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
  - Si $d_j > d_i + a_{ij}$,
    - $d_j = d_i + a_{ij}$.
    - $V = V \cup \{j\}$.

Exemple



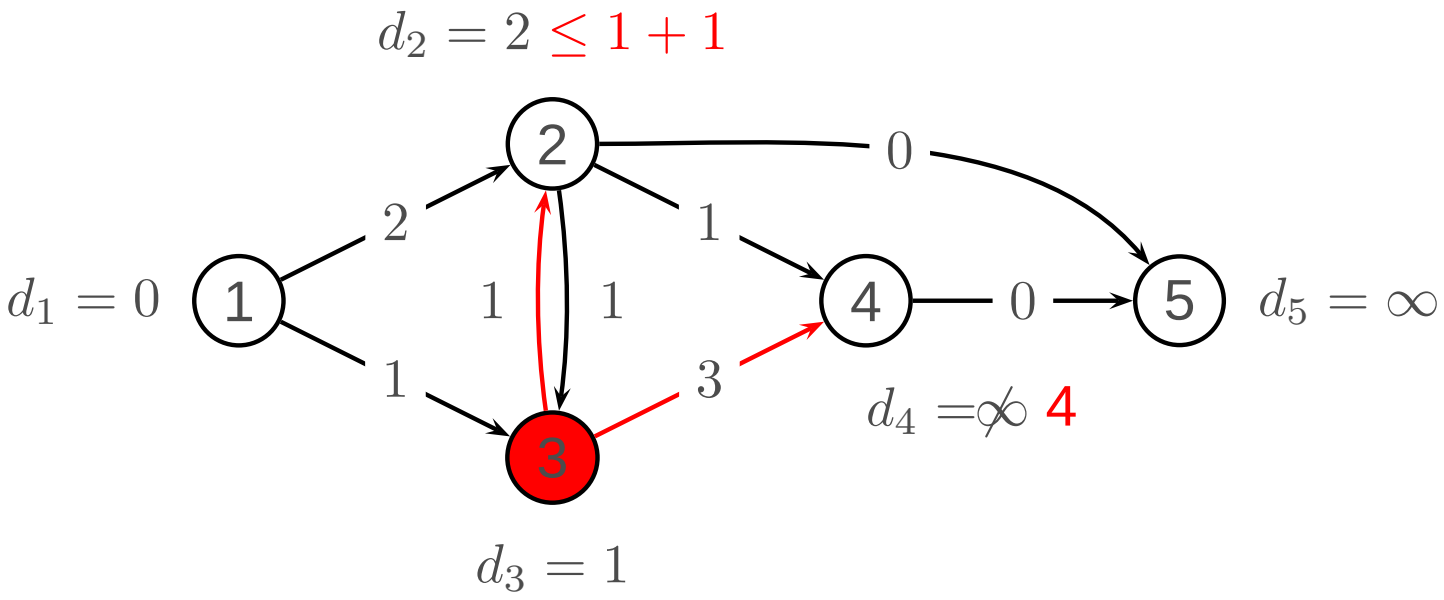
Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1

Exemple



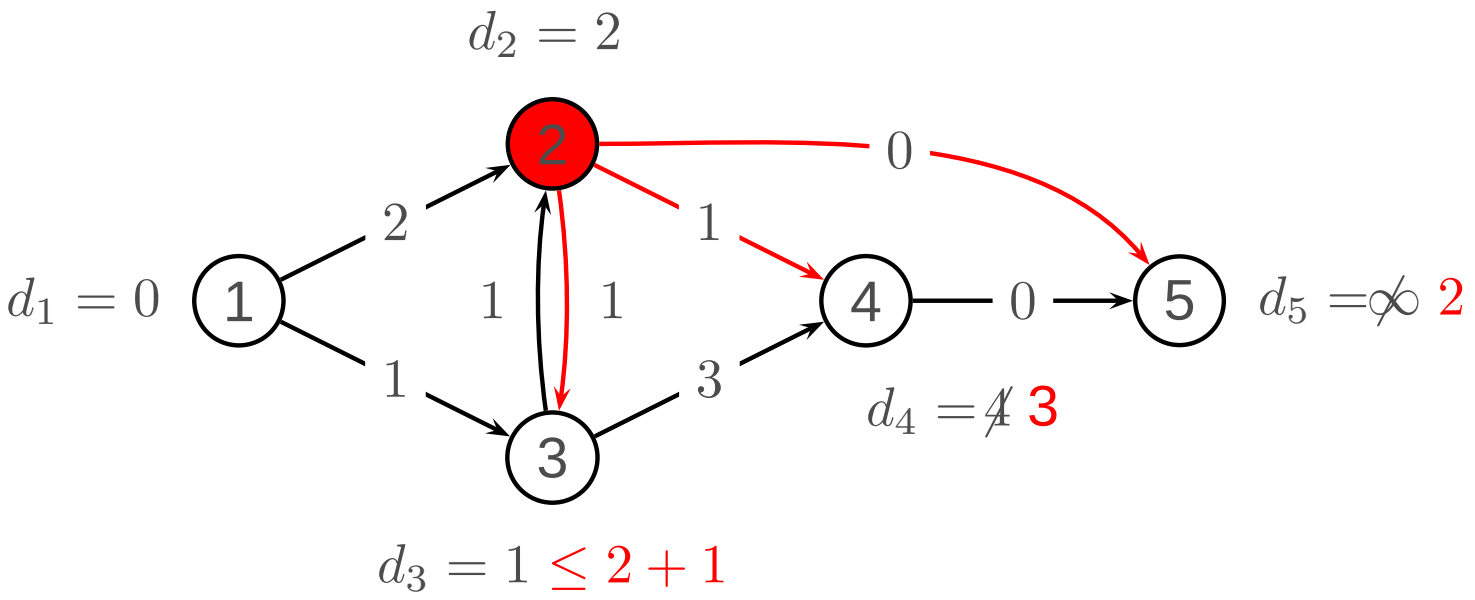
Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3

Exemple



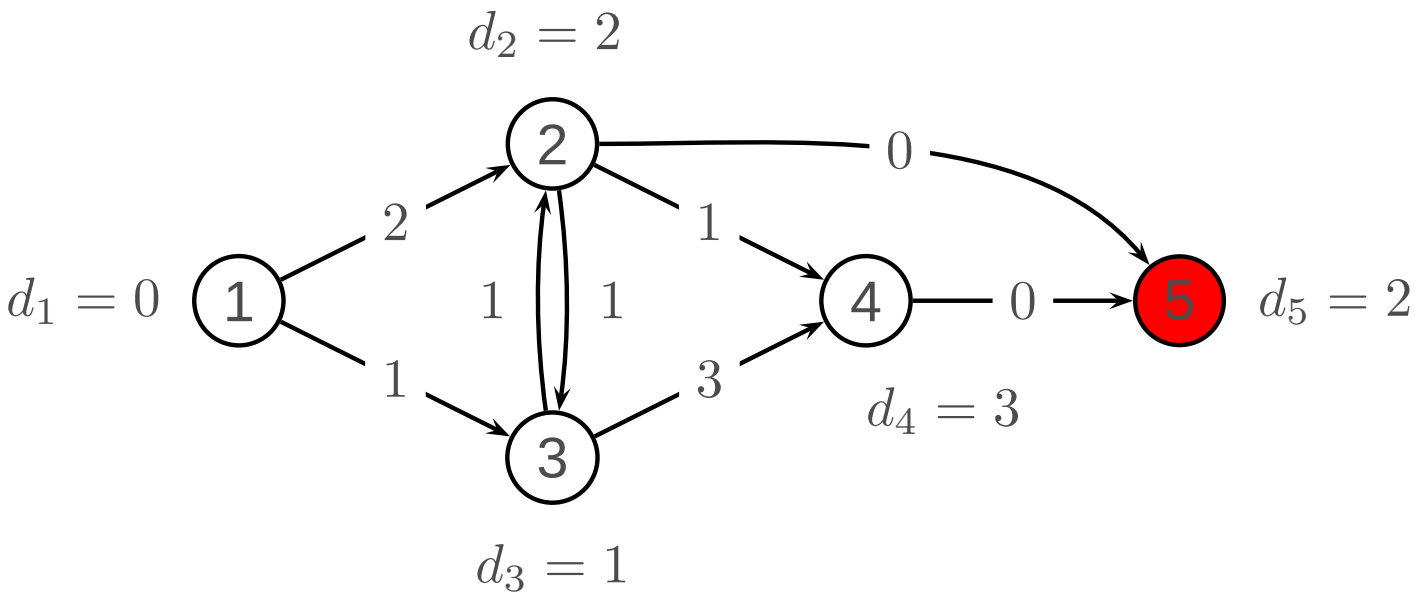
Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2

Exemple



Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2
3	{ 4,5 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	5

Exemple

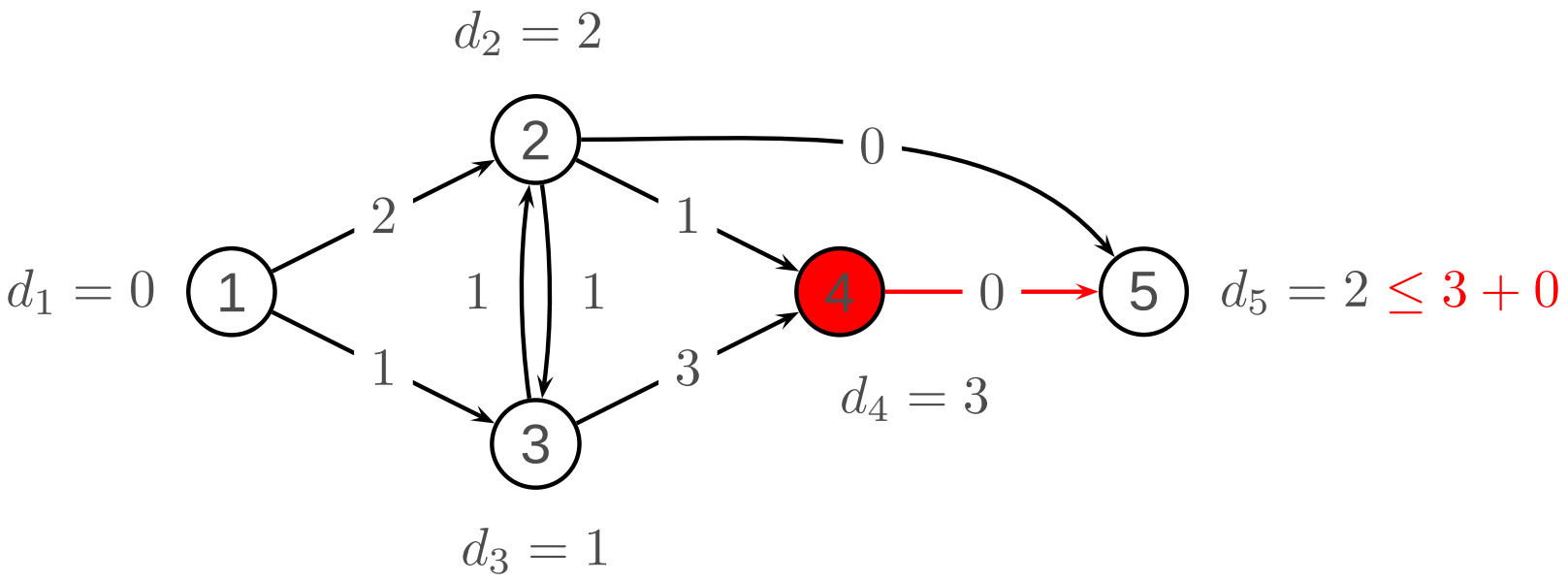


Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2
3	{ 4,5 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	5
4	{ 4 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	4

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple

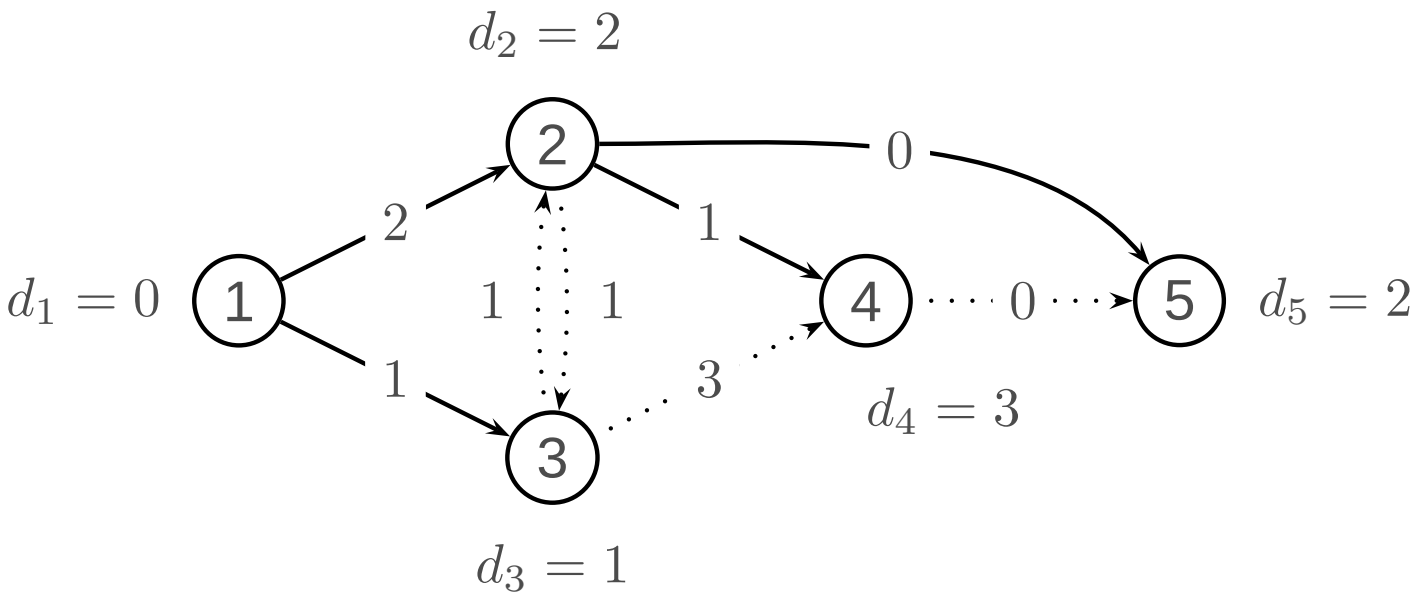


Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2
3	{ 4,5 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	5
4	{ 4 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	4
5	{ }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple



Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2
3	{ 4,5 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	5
4	{ 4 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	4
5	{ }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple

Iter	V	1	2	3	4	5	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	4 (3)	∞ (-)	2
3	{ 4,5 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	5
4	{ 4 }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	4
5	{ }	0 (-)	2 (1)	1 (1)	3 (2)	2 (2)	

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

Algorithme de Dijkstra

- Soit l'ensemble

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

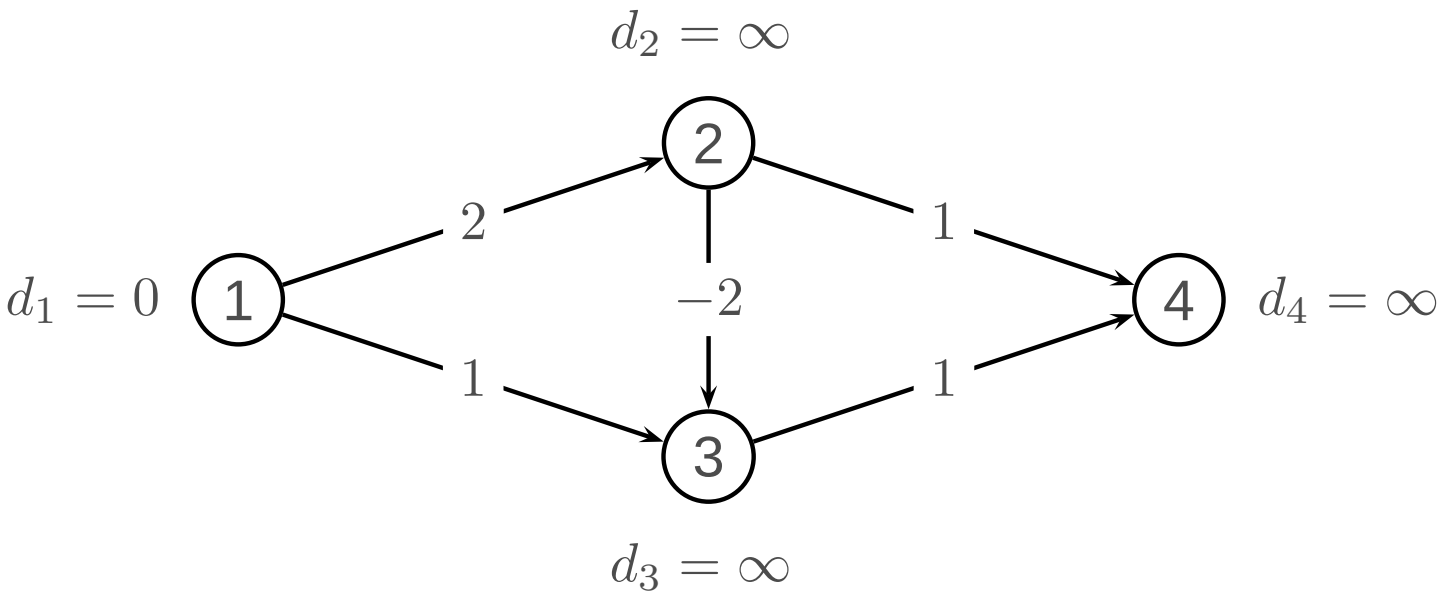
- Si les coûts sur les arcs sont non négatifs, alors à chaque itération
  - aucun nœud dans W au début de l'itération n'entre dans V lors de l'itération,
  - à la fin de l'itération, $d_i \leq d_j$ si $i \in W$ et $j \notin W$.

W : ensemble des étiquettes permanentes.

Notes

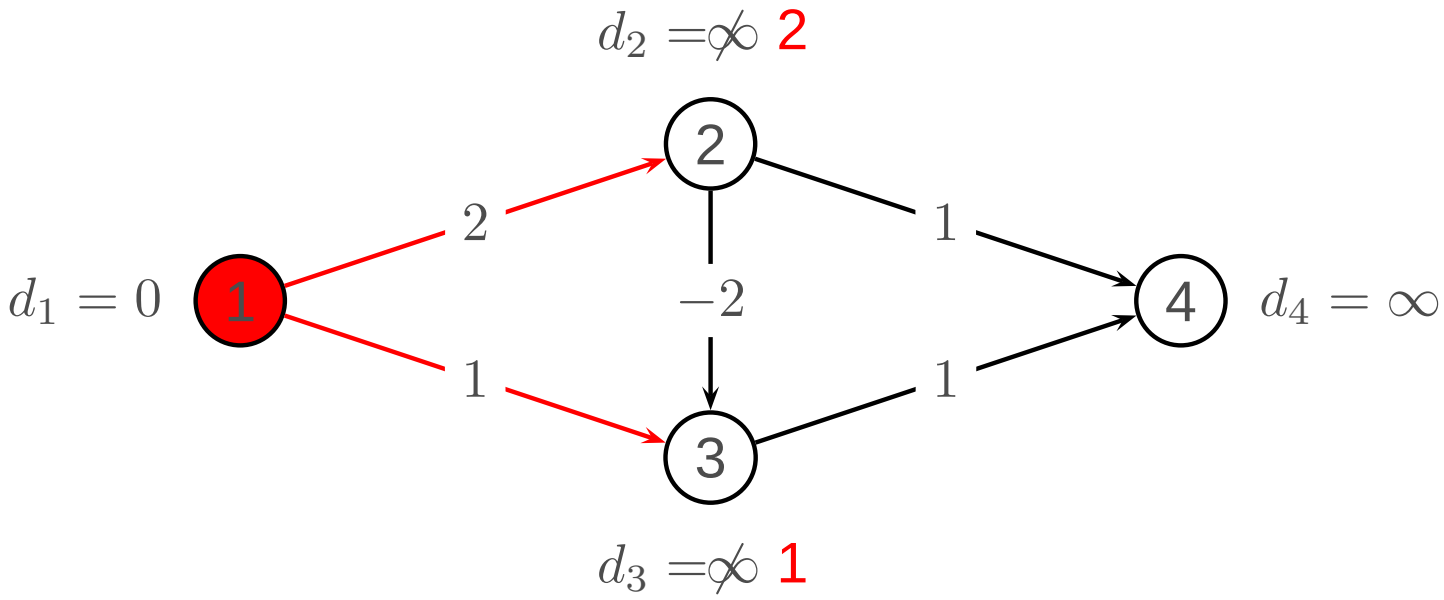
- Si l'on désire calculer le plus court chemin de 1 à b, on peut arrêter l'algorithme de Dijkstra dès que le nœud b est dans W.
- Si au moins un arc a un coût négatif, rien ne garantit le caractère permanent des étiquettes.

Exemple : coût négatif



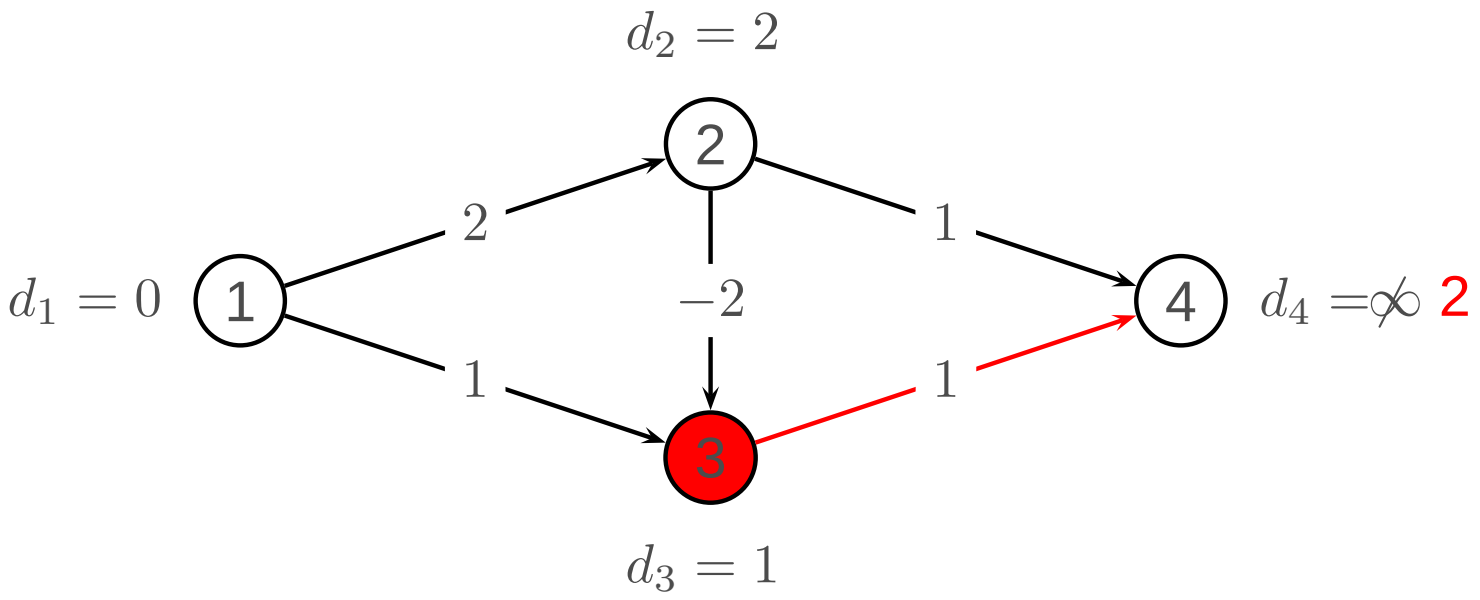
Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1

Exemple : coût négatif



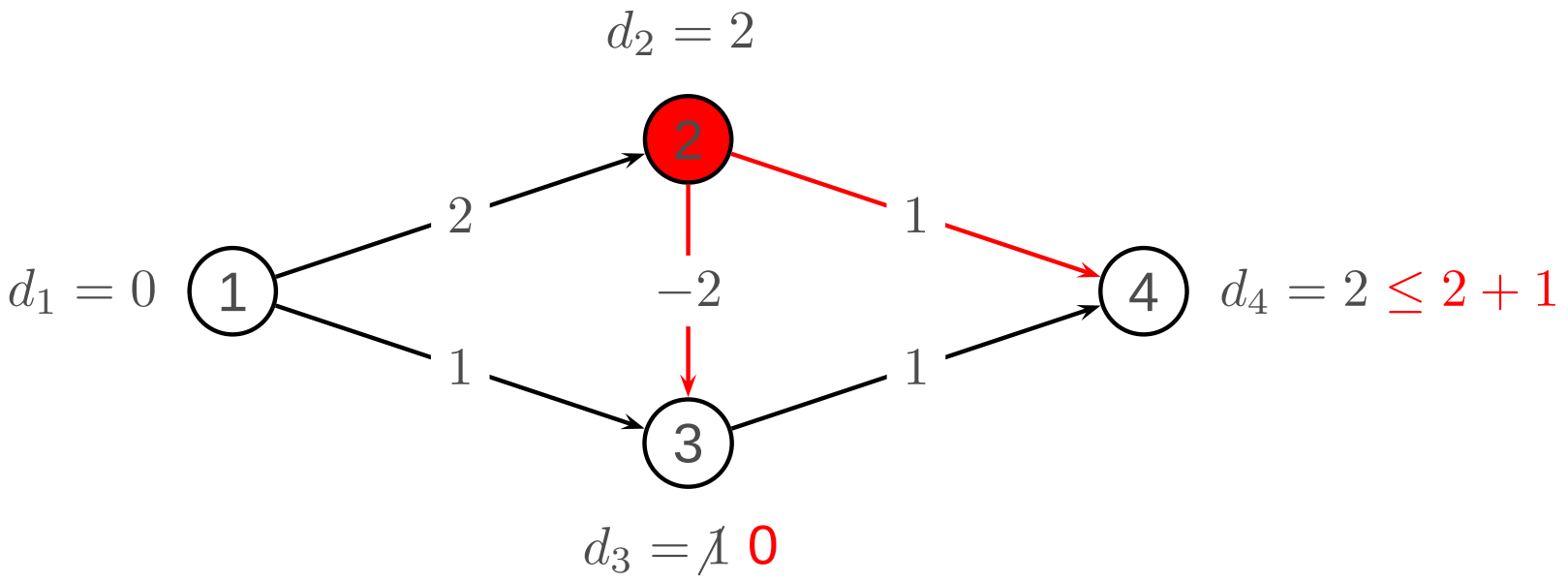
Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3

Exemple : coût négatif



Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	2 (3)	2

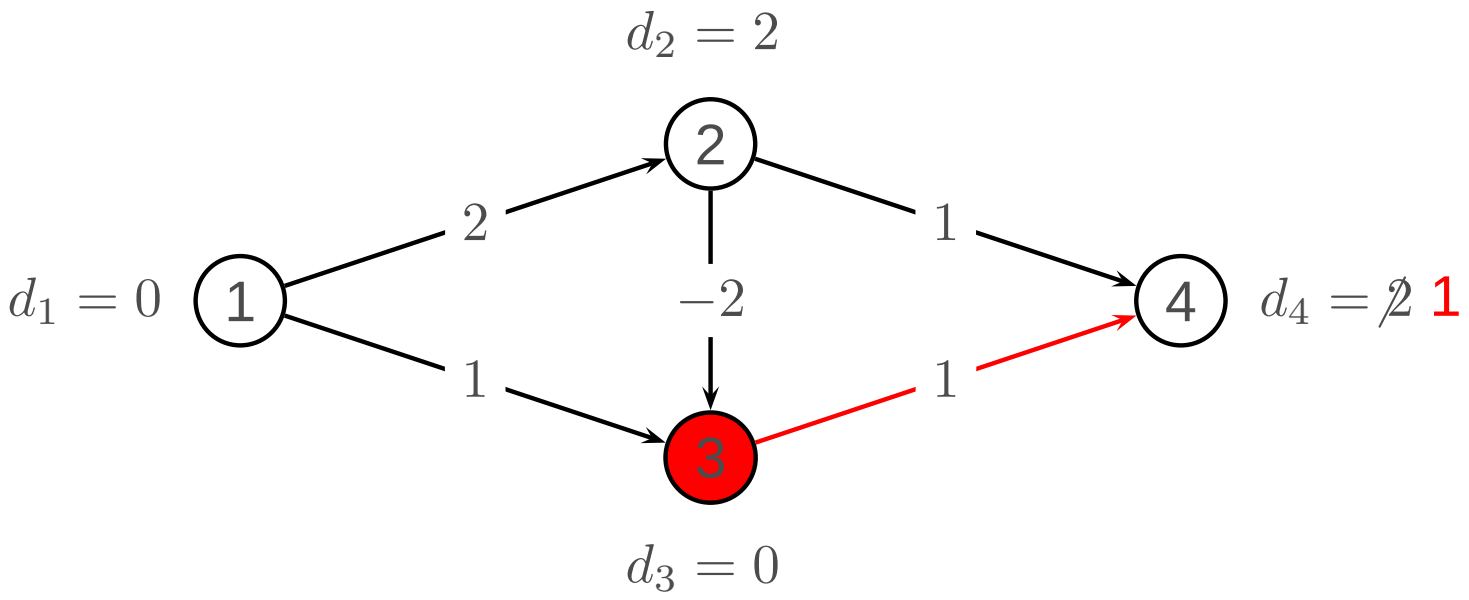
Exemple : coût négatif



Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	2 (3)	2
3	{ 3,4 }	0 (-)	2 (1)	0 (2)	2 (3)	3

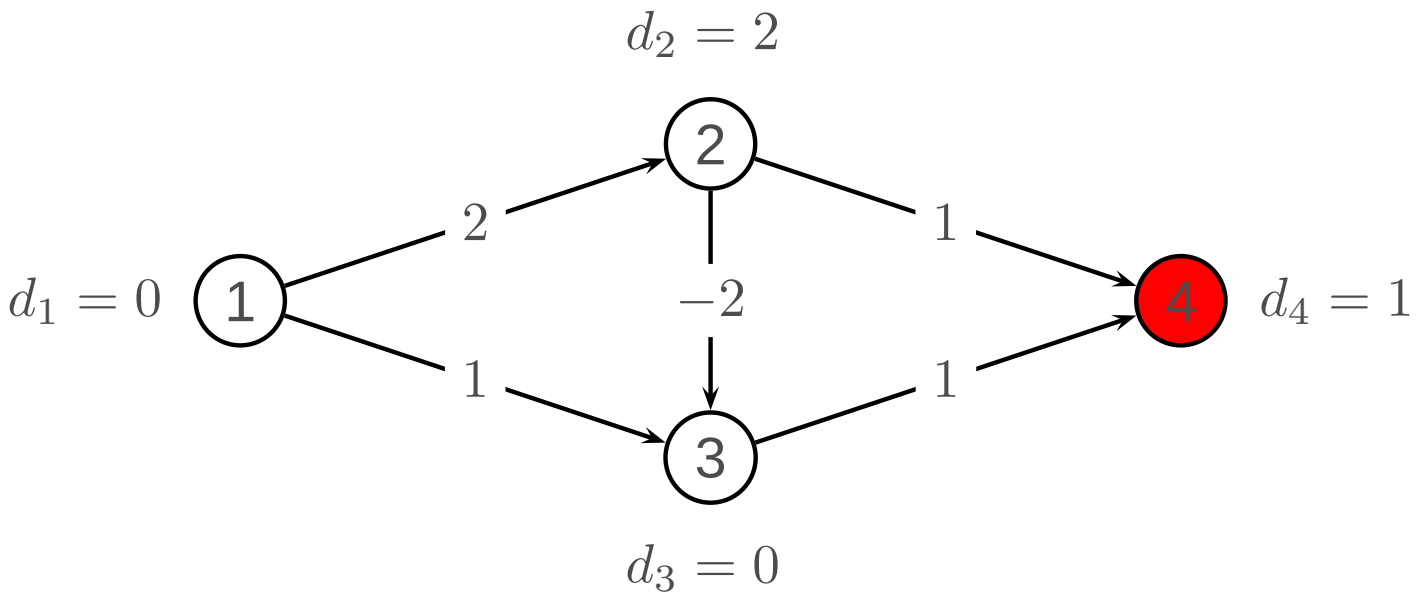
!!!

Exemple : coût négatif



Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	2 (3)	2
3	{ 3,4 }	0 (-)	2 (1)	0 (2)	2 (3)	3
4	{ 4 }	0 (-)	2 (1)	0 (2)	1 (3)	4

Exemple : coût négatif

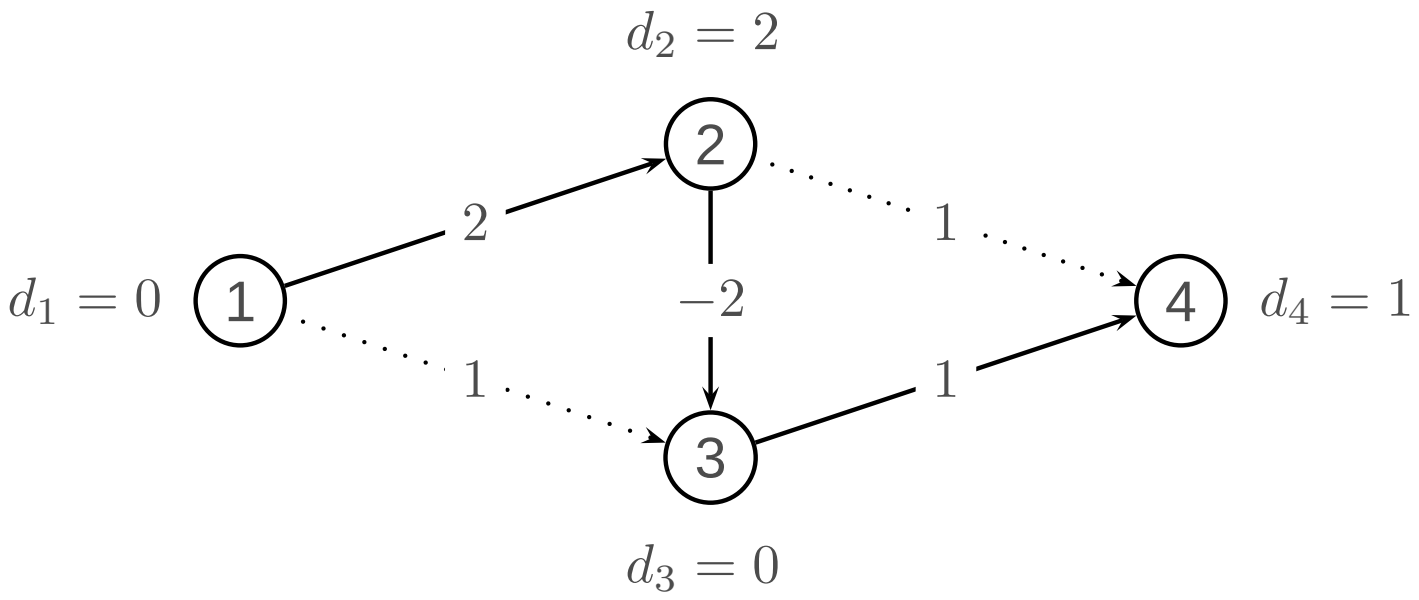


Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	2 (3)	2
3	{ 3,4 }	0 (-)	2 (1)	0 (2)	2 (3)	3
4	{ 4 }	0 (-)	2 (1)	0 (2)	1 (3)	4
5	{ }	0 (-)	2 (1)	0 (2)	1 (3)	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Exemple : coût négatif



Iter	V	1	2	3	4	Traiter
0	{ 1 }	0 (-)	∞ (-)	∞ (-)	∞ (-)	1
1	{ 2,3 }	0 (-)	2 (1)	1 (1)	∞ (-)	3
2	{ 2,4 }	0 (-)	2 (1)	1 (1)	2 (3)	2
3	{ 3,4 }	0 (-)	2 (1)	0 (2)	2 (3)	3
4	{ 4 }	0 (-)	2 (1)	0 (2)	1 (3)	4
5	{ }	0 (-)	2 (1)	0 (2)	1 (3)	

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

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.