ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Optimisation dans les réseaux

Recherche Opérationnelle

GC-SIE

Le problème du plus court chemin

Introduction

- Plusieurs versions :
  - d'un nœud α vers un nœud β
  - de tous les nœuds vers un nœud β
  - d'un nœud α vers tous les nœuds
  - de tous les nœuds vers tous les nœuds
- Nous allons étudier le problème du plus court chemin d'un nœud α vers tous les autres.
- Souvent, on supposera que α=1.

Introduction

Idée générale :

- Parcourir le réseau à partir de l'origine
- Appliquer et mettre à jour des étiquettes $(d_1,\ldots,d_N)$ à chaque nœud.
- Chaque étiquette est soit un scalaire soit $+\infty$.

Conditions d'optimalité

- Soient $d_1,d_2,\ldots,d_N$ des scalaires tels que

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

- Soit P un chemin entre un nœud α et un nœud β.
- Si

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

- Alors P est un plus court chemin entre α et β.

Conditions d'optimalité

Preuve :

- P est composé des arcs

$$(\alpha,i_1),(i_1,i_2),\ldots,(i_P,\beta)$$

- Longueur de P =

$$a_{\alpha,i_1}+a_{i_1,i_2}+\ldots+a_{i_P,\beta}$$

- Comme $a_{ij} = d_j - d_i \; \forall (i,j) \in P$, on a
- Longueur de P =

$$(d_\beta - d_{i_P}) + (d_{i_P} - d_{i_{P-1}})+\ldots+(d_{i_2}-d_{i_1})+(d_{i_1}-d_\alpha)=$$
$$d_\beta - d_\alpha$$

Conditions d'optimalité

Soit Q un chemin quelconque entre α et β.

- Q est composé des arcs

$$(\alpha,j_1),(j_1,j_2),\ldots,(j_Q,\beta)$$

- Longueur de Q =

$$a_{\alpha,j_1}+a_{j_1,j_2}+\ldots+a_{j_Q,\beta}$$

- Comme $a_{ij} \geq d_j - d_i \ \forall (i,j) \in$ Q, on a
- Longueur de Q $\geq$

$$(d_\beta - d_{j_Q}) + (d_{j_Q} - d_{j_{Q-1}})+\ldots+(d_{j_2}-d_{j_1})+(d_{j_1}-d_\alpha)=$$
$$d_\beta - d_\alpha = \text{Longueur de P.}$$

CQFD

Algorithme générique

Idée :

- On démarre avec un vecteur d'étiquettes $(d_1,\ldots,d_N)$
- 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

$$d_j \leftarrow d_i + a_{ij}$$

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

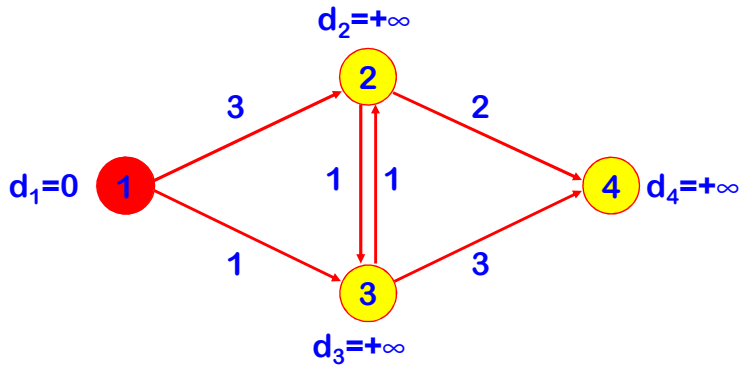
Algorithme générique

- A tout moment, l'étiquette d'un nœud i peut être interprétée comme la longueur d'un chemin reliant α à i.
- Si $d_j > d_i + a_{ij}$, cela signifie que le chemin arrivant en j à partir de i est plus court que le chemin actuel reliant α à j.
- Il est plus facile d'effectuer le traitement nœud par nœud.
- Pour chaque nœud considéré, on traitera tous ses arcs sortant.
- V= liste des nœuds à traiter

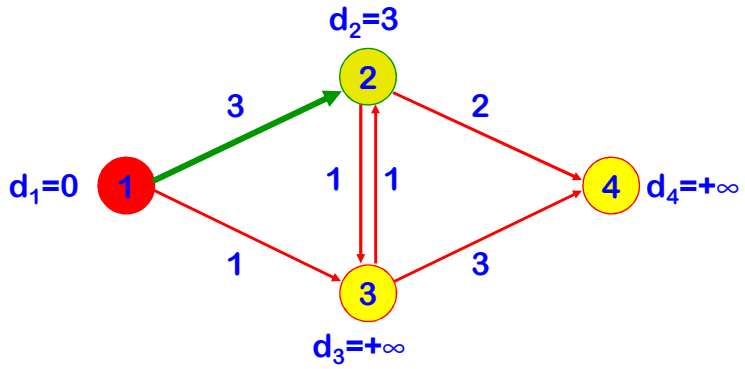
Algorithme générique

Algorithme :

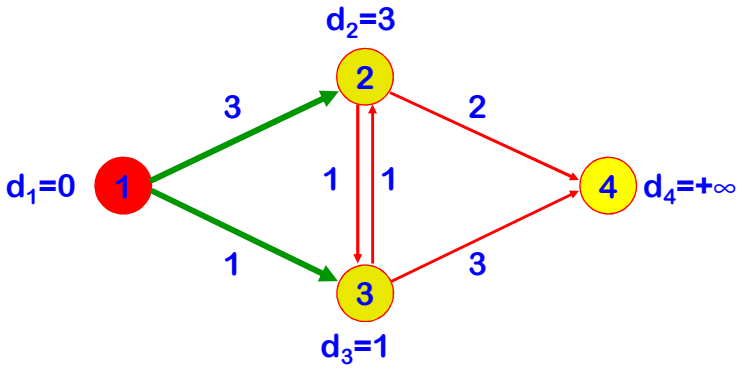
- Initialisation :
  - $V=\{\alpha\}$
  - $d_\alpha = 0$, $d_i = \infty \ \forall i \neq \alpha$
- Itérations. Tant que $V \neq \emptyset$
  - Sélectionner un nœud i dans V et l'en retirer
  - Pour chaque arc (i,j) sortant de i,
    - Si $d_j > d_i + a_{ij}$, alors
      - $d_j \leftarrow d_i + a_{ij}$
      - Ajouter j à V



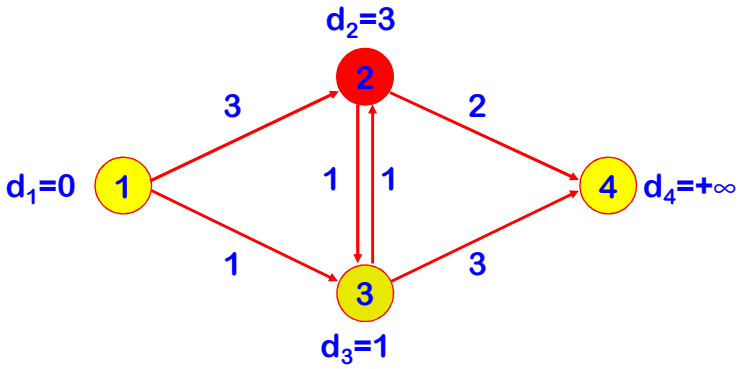
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



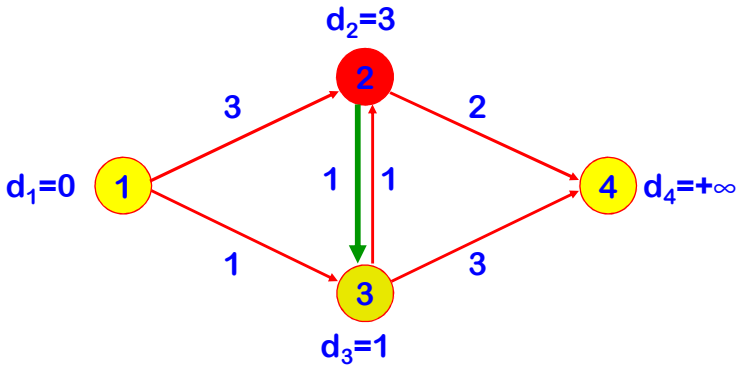
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



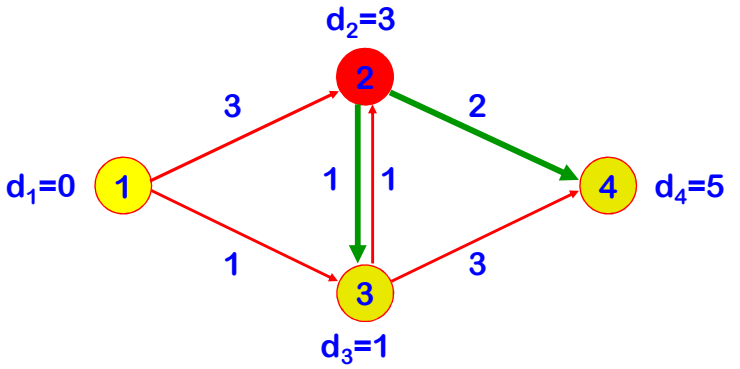
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2



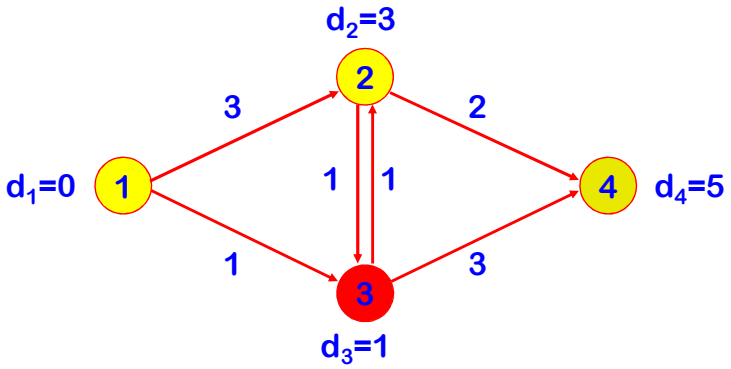
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2



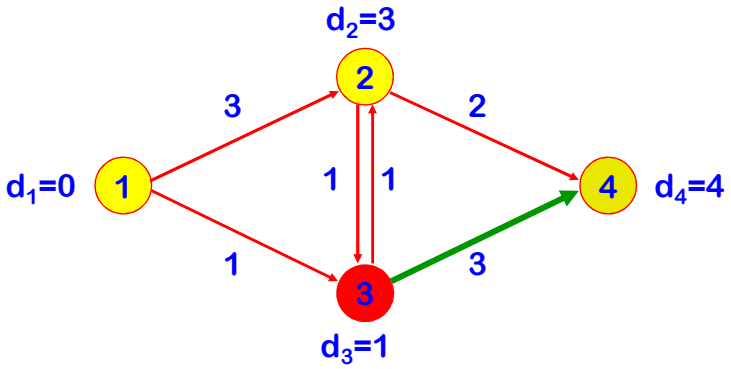
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2



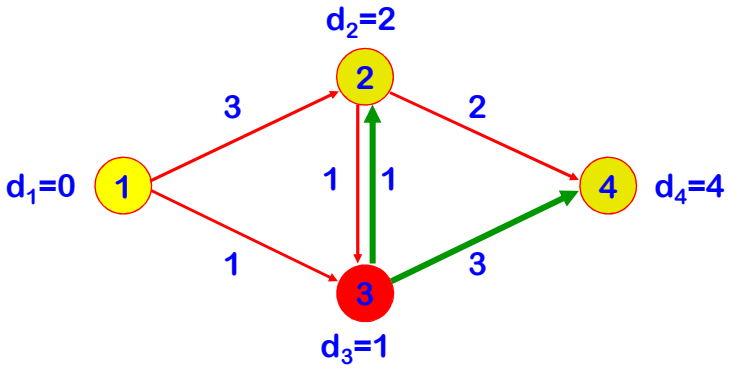
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3



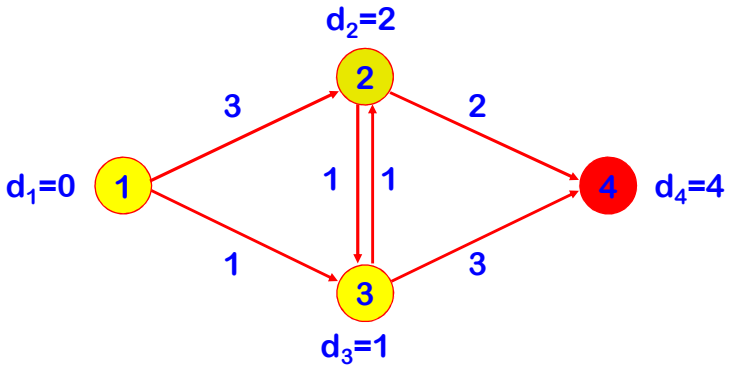
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3



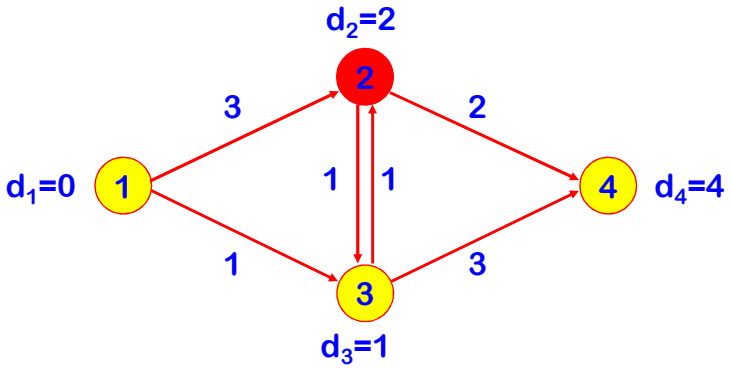
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3



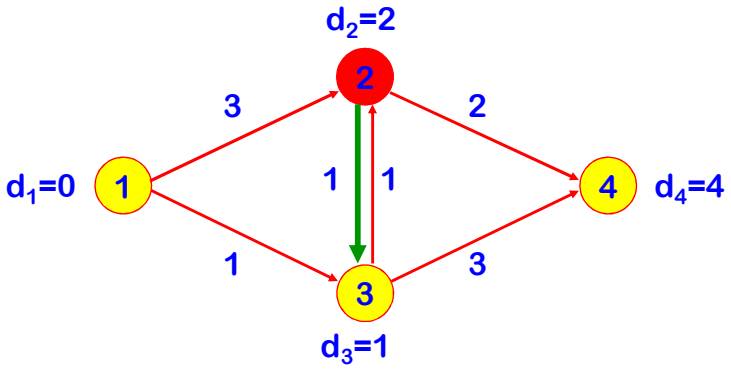
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3
4	{4,2}	0	2	1	4	4



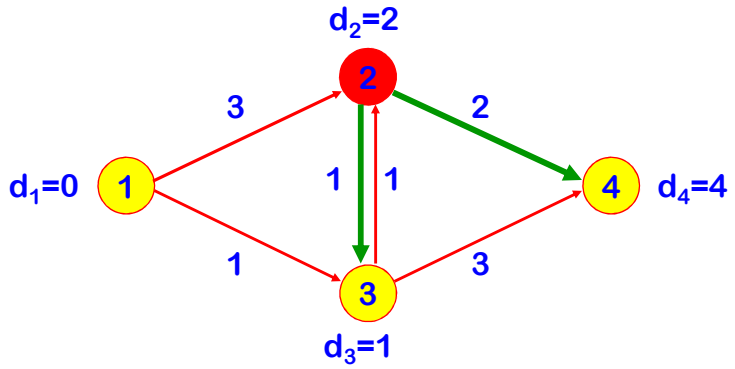
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3
4	{4,2}	0	2	1	4	4
5	{2}	0	2	1	4	2



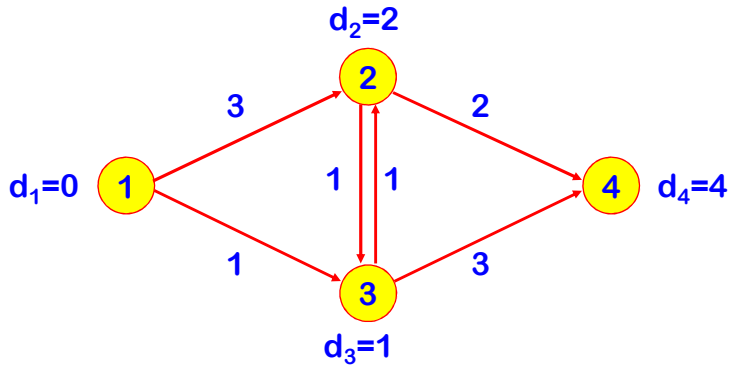
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	3	1	∞	2
3	{3,4}	0	3	1	5	3
4	{4,2}	0	2	1	4	4
5	{2}	0	2	1	4	2



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



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



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

Algorithme générique

Propriétés à la fin de chaque itération

- Si $d_j < \infty$, alors d_j est la longueur d'un chemin reliant α à j.
- Si $i \notin V$, alors soit $d_i = +\infty$, soit

$$d_j \leq d_i + a_{ij} \quad \forall j \text{ tel que } (i,j) \in \mathcal{A}$$

Algorithme générique

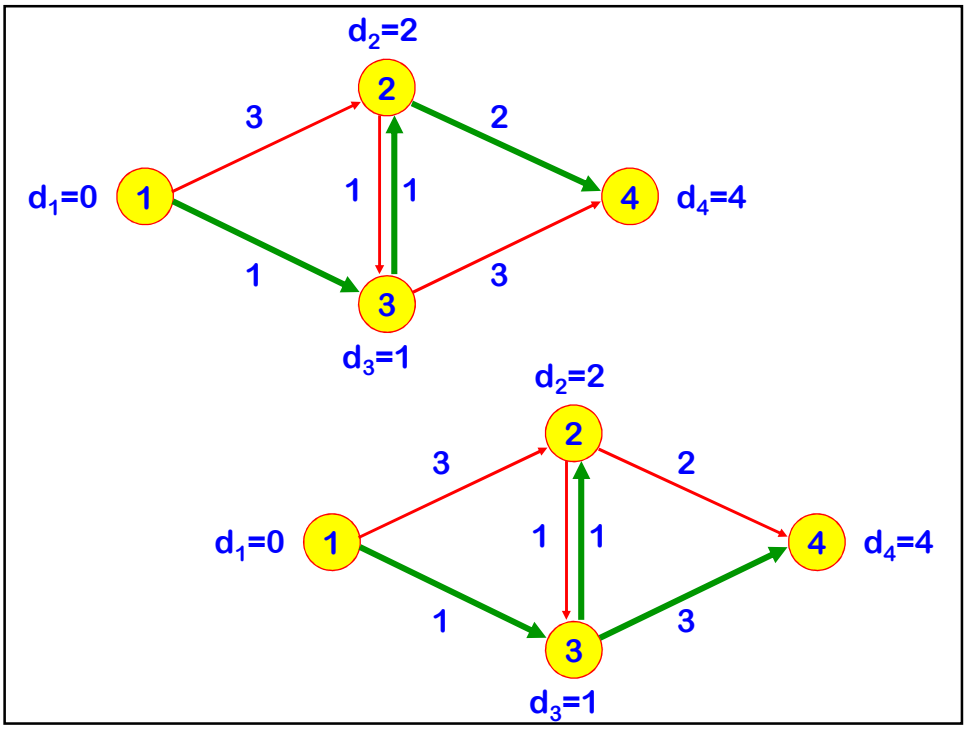
Propriétés si l'algorithme se termine

- $\forall j$ t.q. $d_j < \infty$,
  - d_j est la longueur du plus court chemin entre α et j.
  - $d_j = \min_{(i,j) \in \mathcal{A}} d_i + a_{ij}$ si $j \neq \alpha$ [Eq. de Bellman]
  - $d_\alpha = 0$
- $d_j = +\infty$ ssi il n'y a pas de chemin reliant α et j.

L'algorithme se termine ssi il n'y a aucun chemin commençant en α et contenant un cycle à coût négatif

Algorithme générique

- Les équations de Bellman permettent de reconstituer les plus courts chemins à partir des étiquettes.
- Le prédécesseur du nœud j dans le plus court chemin est celui qui réalise le minimum de l'équation de Bellman.



Algorithme de Dijkstra

- L'algorithme générique ne spécifie pas comment choisir dans V le nœud à traiter.
- L'algorithme de Dijkstra impose que le nœud i à traiter soit tel que

$$d_i = \min_{j \in V} d_j$$

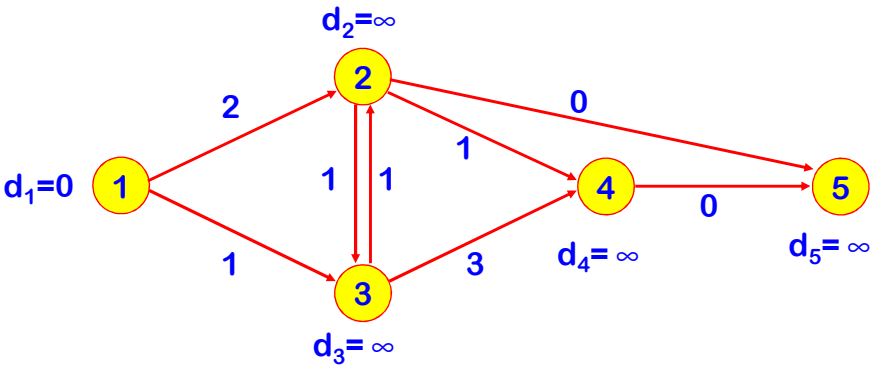
Algorithme de Dijkstra

Algorithme :

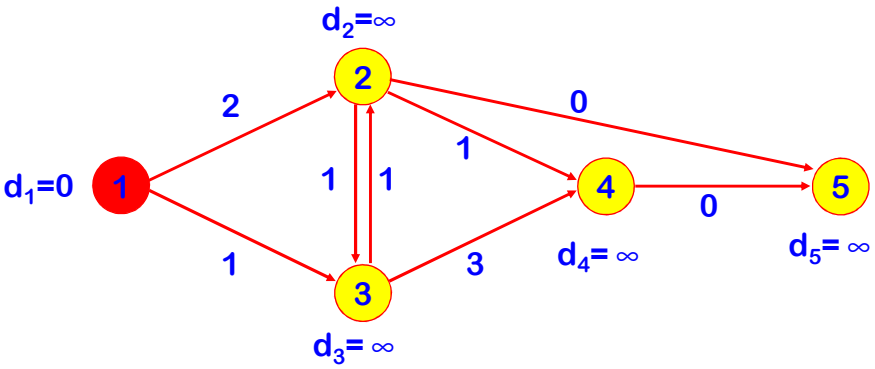
- Initialisation :
  - $V=\{\alpha\}$
  - $d_\alpha = 0$, $d_i = \infty \ \forall i \neq \alpha$
- Itérations. Tant que $V \neq \emptyset$
  - Sélectionner un nœud i dans V tel que

$$d_i = \min_{j \in V} d_j$$

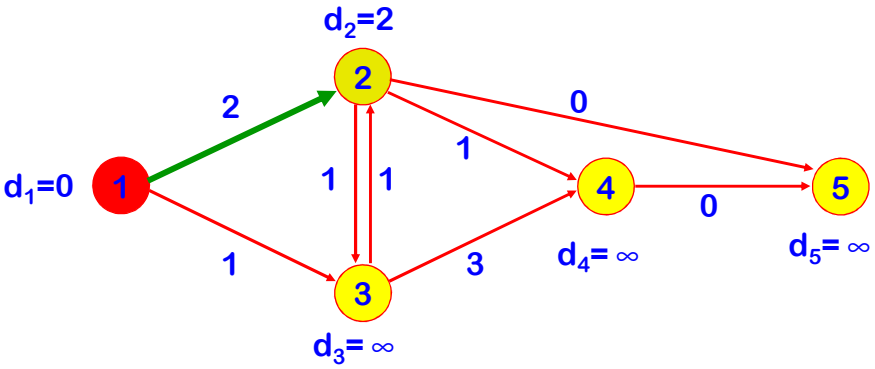
  - Retirer i de V
  - Pour chaque arc (i,j) sortant de i,
    - Si $d_j > d_i + a_{ij}$, alors
      - $d_j \leftarrow d_i + a_{ij}$
      - Ajouter j à V



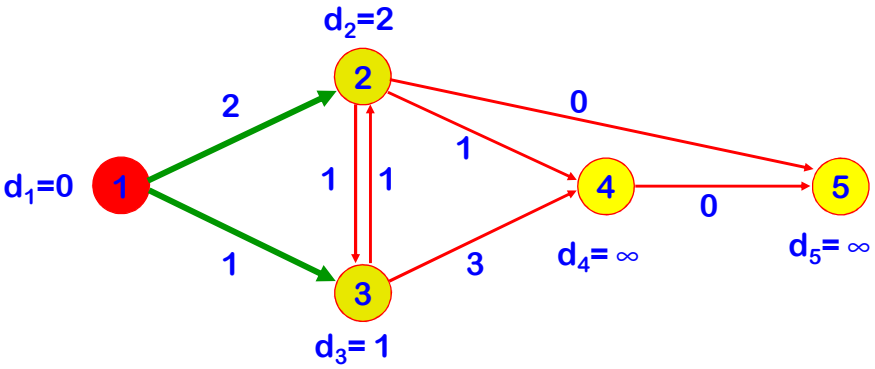
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1



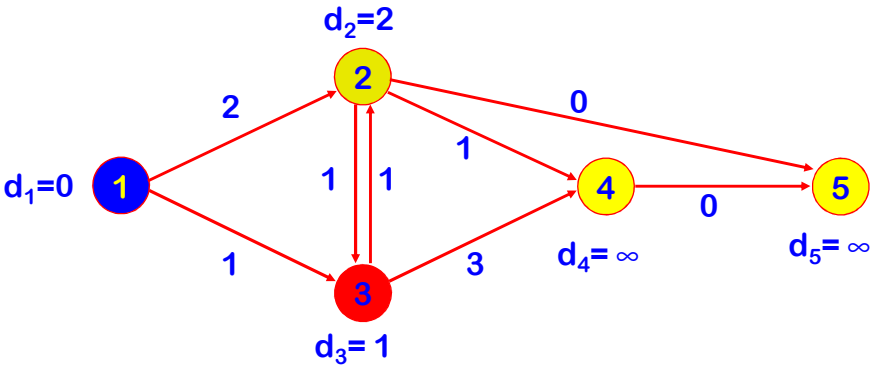
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1



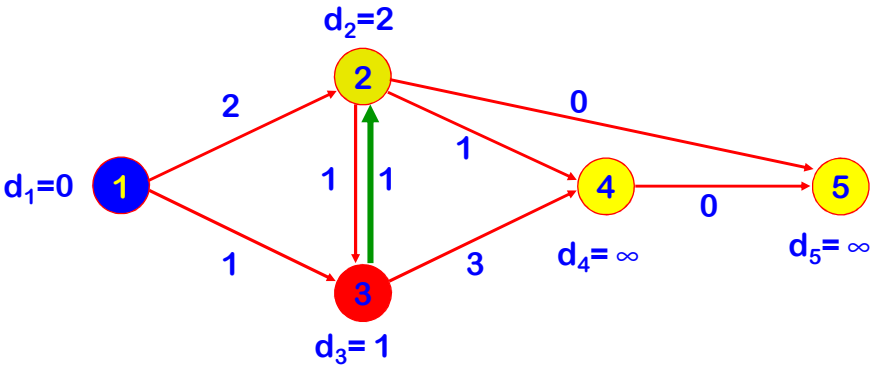
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1



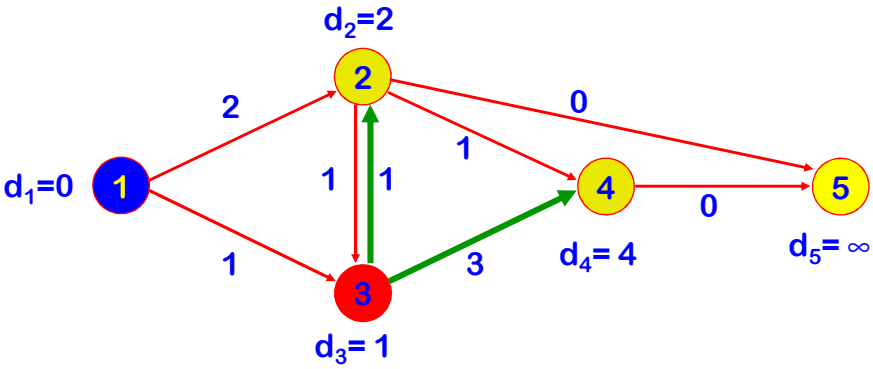
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3



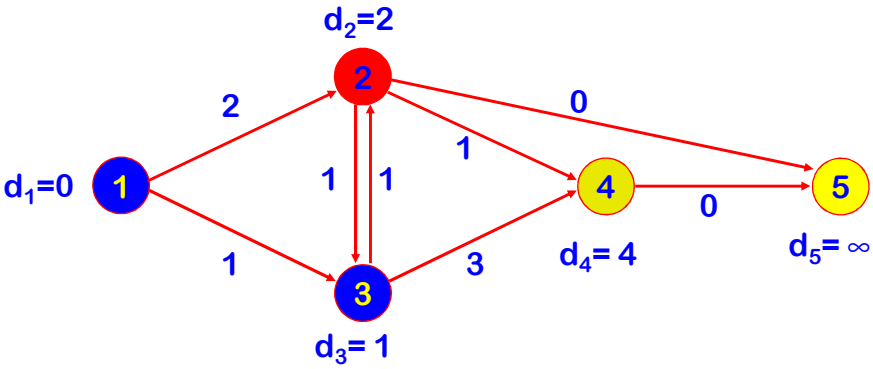
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3



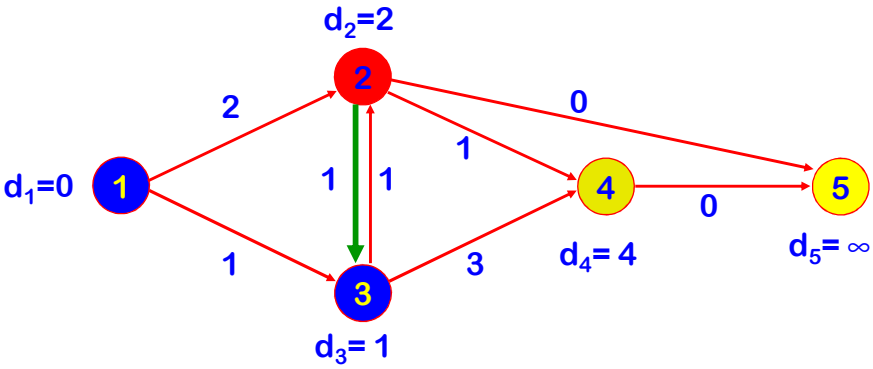
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3



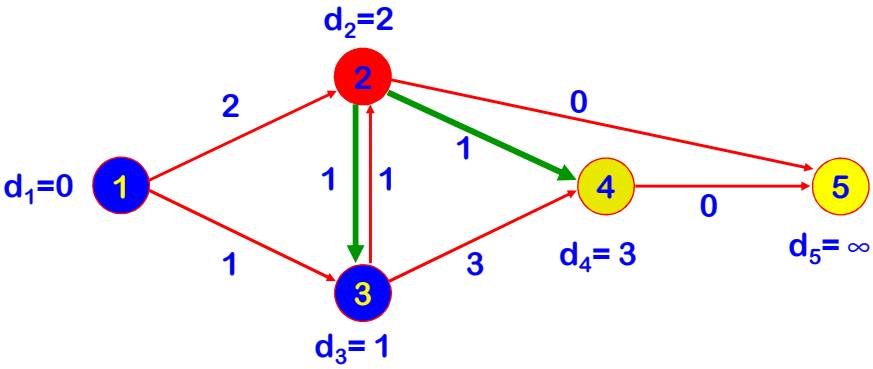
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2



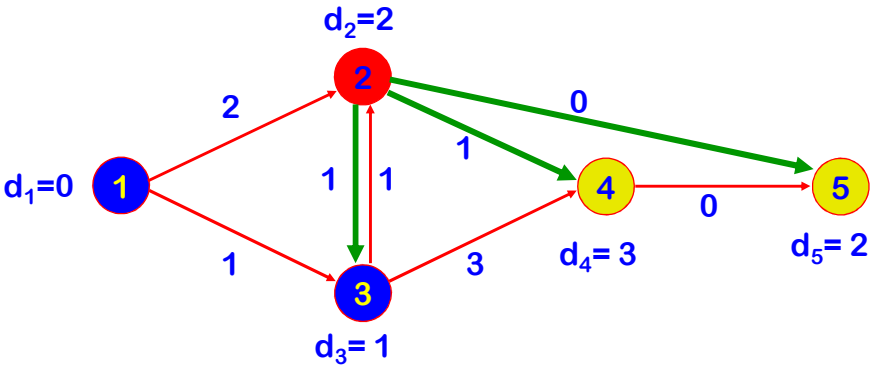
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2



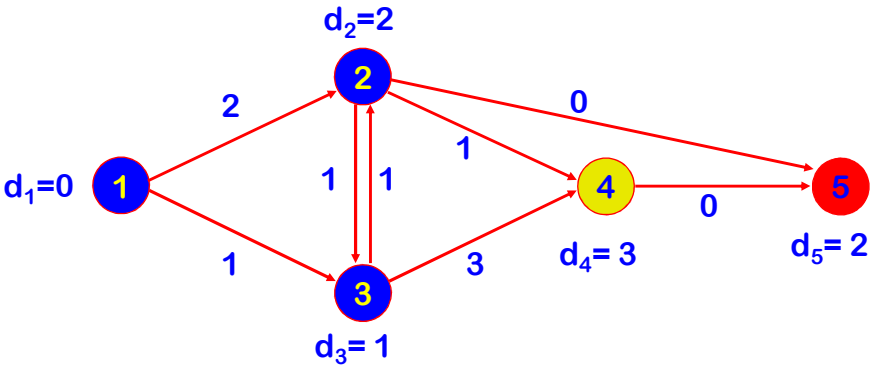
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2



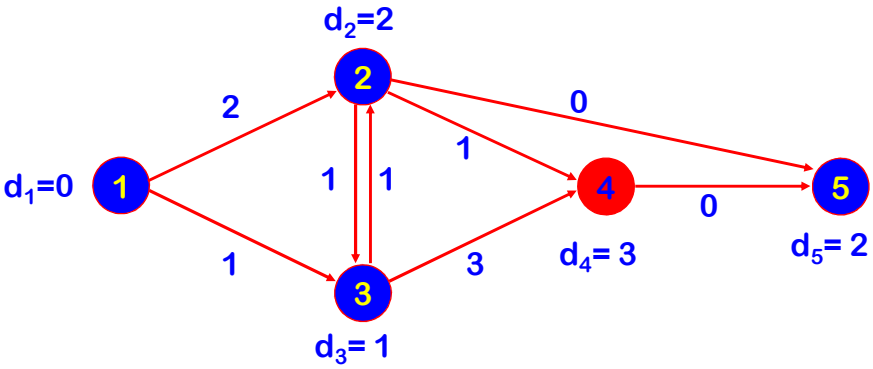
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2



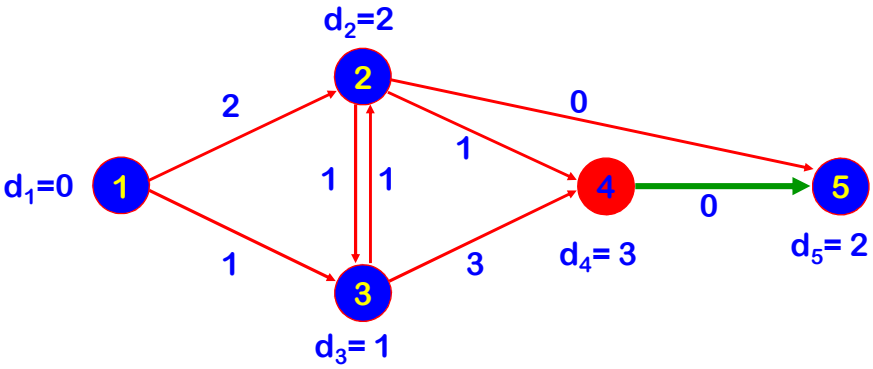
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2
4	{4,5}	0	2	1	3	2	5



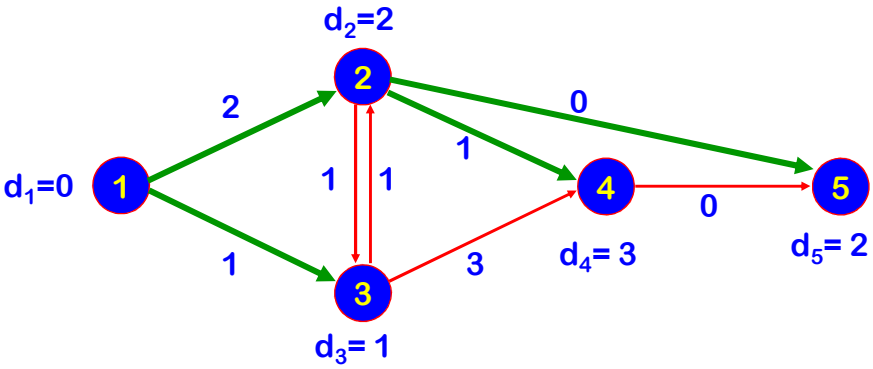
Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2
4	{4,5}	0	2	1	3	2	5
5	{4}	0	2	1	3	2	4



Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2
4	{4,5}	0	2	1	3	2	5
5	{4}	0	2	1	3	2	4



Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2
4	{4,5}	0	2	1	3	2	5
5	{4}	0	2	1	3	2	4
	{}	0	2	1	3	2	



Iter	V	d_1	d_2	d_3	d_4	d_5	traiter
1	{1}	0	∞	∞	∞	∞	1
2	{2,3}	0	2	1	∞	∞	3
3	{2,4}	0	2	1	4	∞	2
4	{4,5}	0	2	1	3	2	5
5	{4}	0	2	1	3	2	4
	{}	0	2	1	3	2	

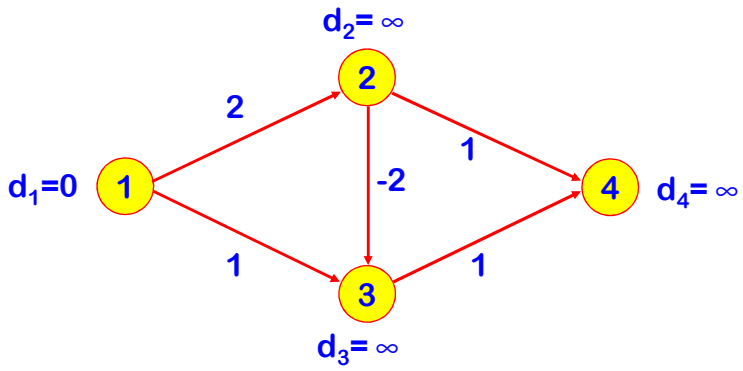
Algorithme de Dijkstra

Propriété des étiquettes permanentes

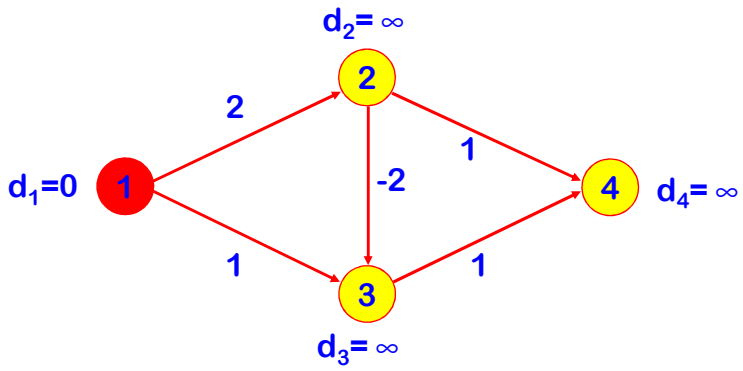
- Si les coûts sur tous les arcs sont non négatifs
- Considérons W={i ¦ d_i <∞ et i∉ V}
- Alors, pour chaque itération,
  - aucun nœud appartenant à W au début de l'itération n'entrera dans V pendant l'itération
  - à la fin de l'itération, $d_i \leq d_j$ si i∈ W et j∉ W.

Notes

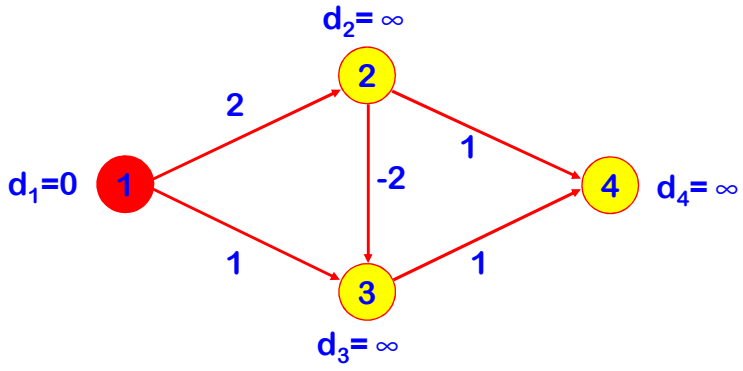
- Si l'on désire calculer le plus court chemin de α à β, on peut arrêter l'algorithme de Dijkstra dès que le nœud β est dans W.
- Si au moins un arc a un coût négatif, rien ne garantit le caractère permanent des étiquettes



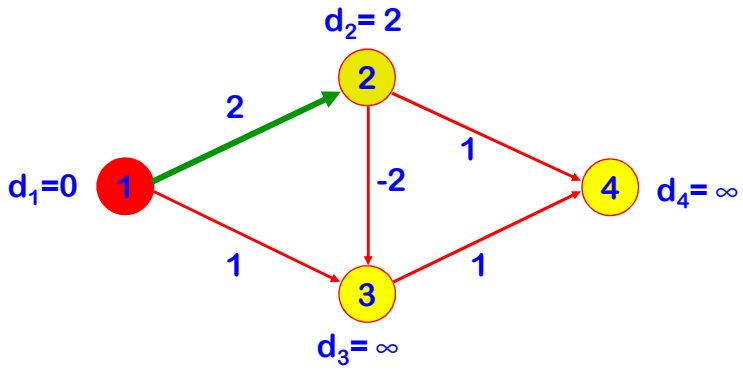
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



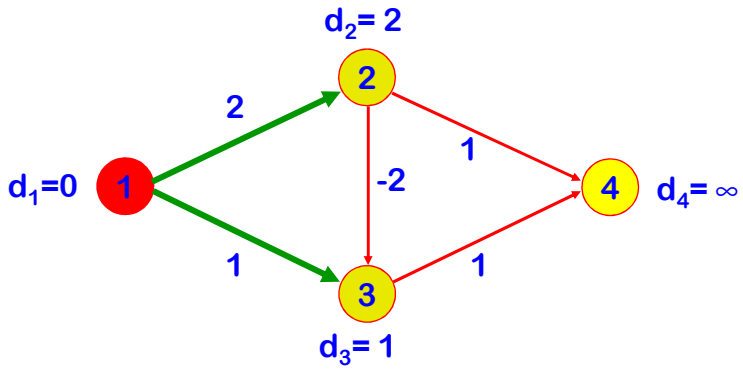
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



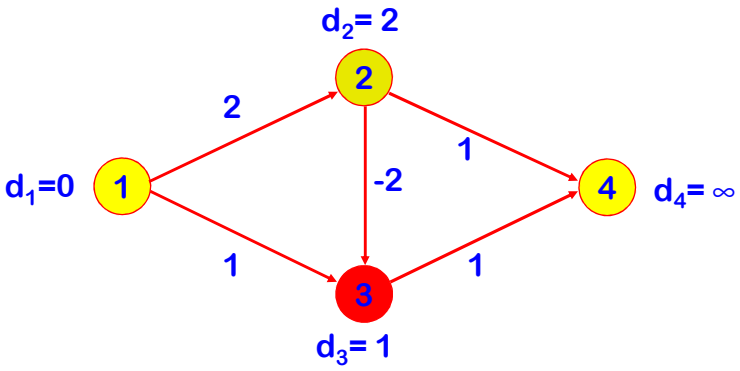
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



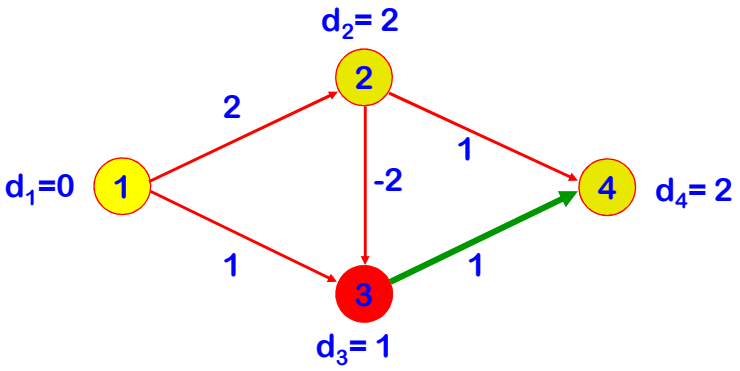
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1



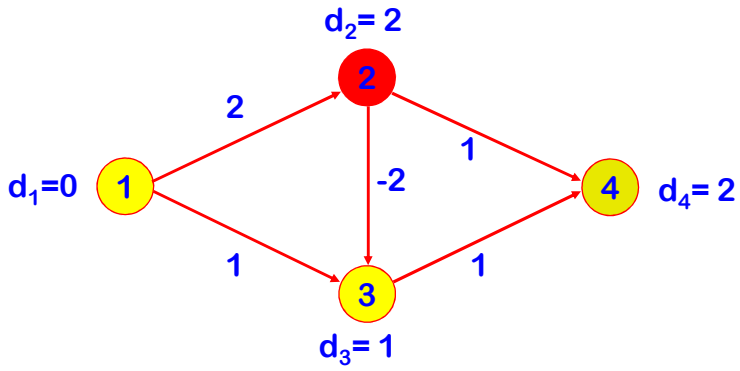
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3



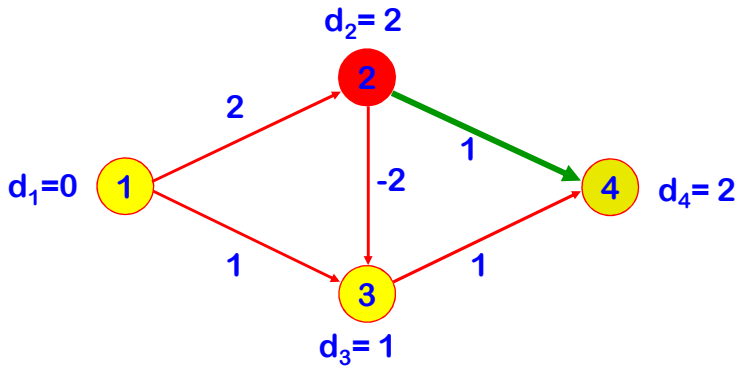
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3



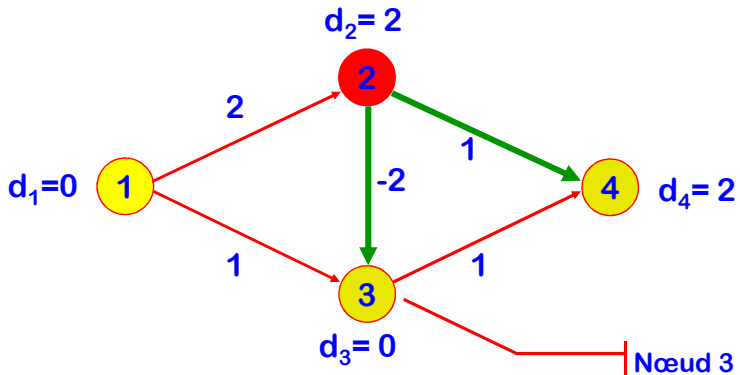
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3
3	{2,4}	0	2	1	2	2



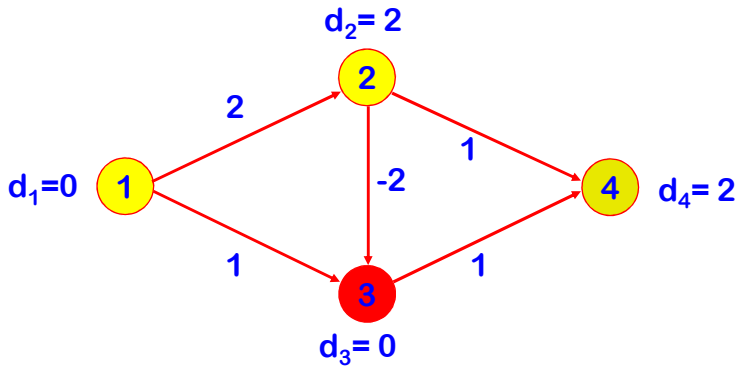
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3
3	{2,4}	0	2	1	2	2



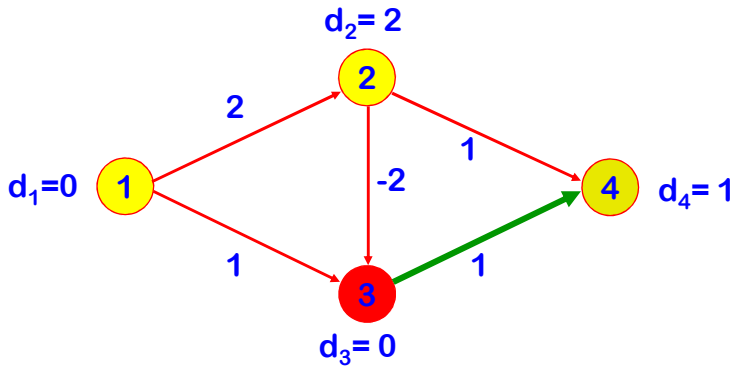
Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3
3	{2,4}	0	2	1	2	2



Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3
3	{2,4}	0	2	1	2	2
4	{3,4}	0	2	0	2	3



Iter	V	d_1	d_2	d_3	d_4	traiter
1	{1}	0	∞	∞	∞	1
2	{2,3}	0	2	1	∞	3
3	{2,4}	0	2	1	2	2
4	{3,4}	0	2	0	2	3



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