Plus courts chemins

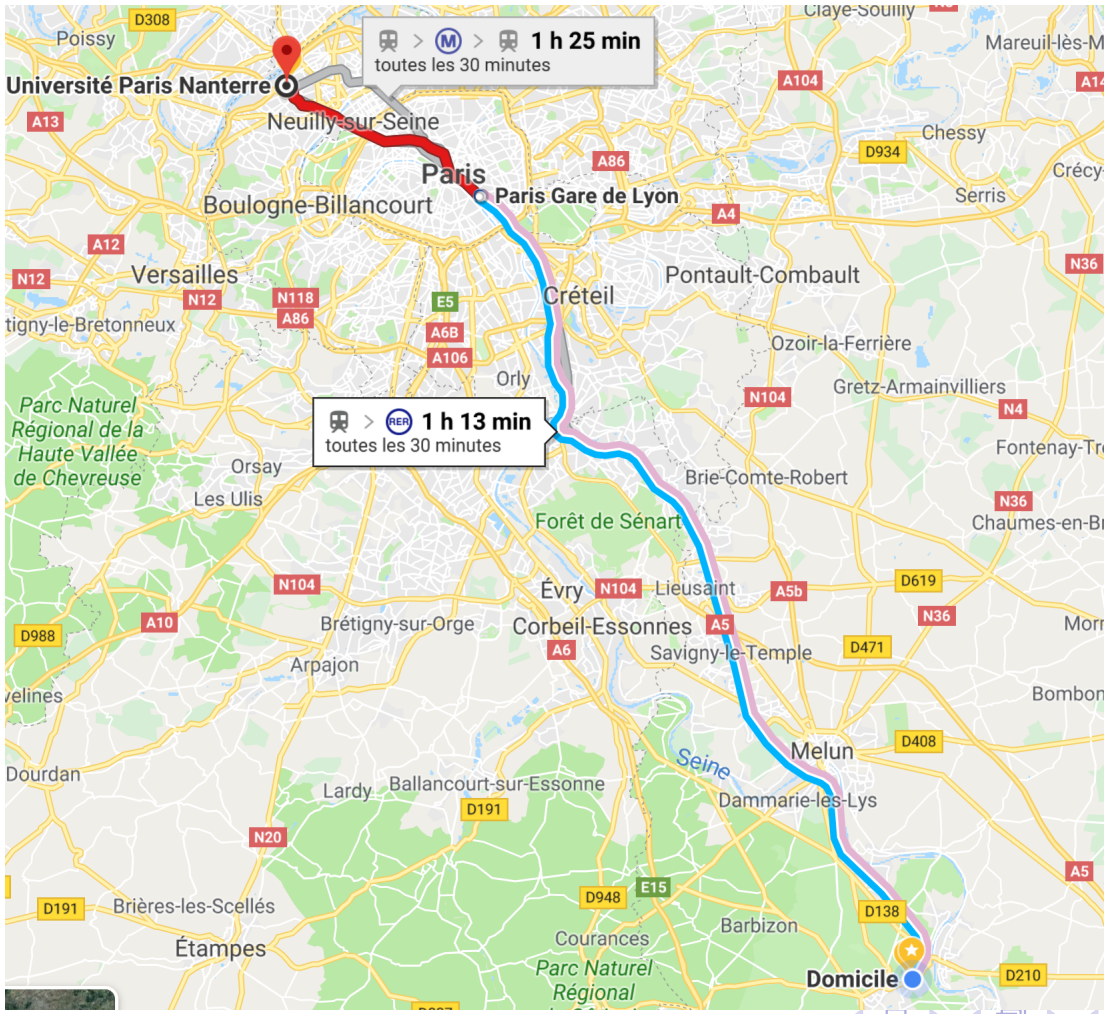
François Delbot

17 octobre 2019

Plan : 1 - Le problème du plus court chemin

Le problème du plus court chemin

Comment se rendre de Fontainebleau à l'Université de Paris Nanterre ?



Le problème du plus court chemin

Ou comment se rendre de Fontainebleau à l'Université de Paris Nanterre ?

1. Comment fonctionne votre GPS ?
2. Comment fonctionne Google Maps ?
3. Comment déterminer le plus court chemin entre deux adresses ?

Nous allons voir dans cette partie comment déterminer le plus court chemin entre deux sommets d'un graphe pondéré pour lequel les sommets peuvent représenter des villes ou des intersections, les arcs les routes existantes et le poids d'un arc la distance à parcourir.

Le problème du plus court chemin

Quelle version du problème faut-il considérer ?

Trouver le chemin le plus court allant d'un point A à un point B peut être considéré de trois manières différentes, qui semblent différentes dans leur difficulté :

1. Plus court chemin entre deux sommets du graphe.
2. Plus court chemin d'un sommet vers tous les autres.
3. Plus courts chemin de chaque sommet vers tous les autres.

Le problème du plus court chemin

Quelle version du problème faut-il considérer ?

Mais :

1. Il n'est pas possible de connaitre le plus court chemin entre deux sommets du graphe sans considérer les autres sommets.
2. Si on connait le plus court chemin d'un sommet vers tous les autres il suffit de répéter l'opération pour connaitre le plus court chemin de chaque sommet vers tous les autres.

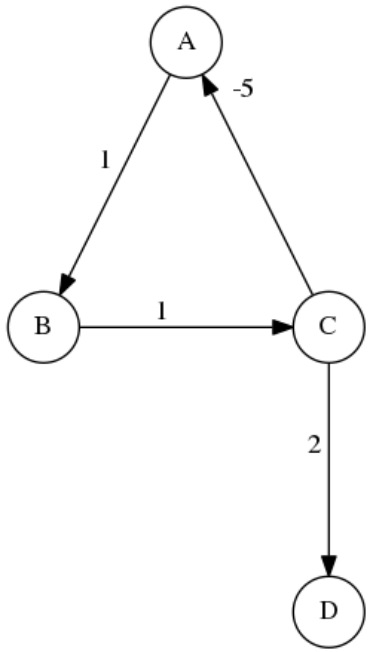
Nous allons donc nous concentrer sur la deuxième version du problème : le calcul d'un plus court chemin d'un sommet vers tous les autres.

Le problème du plus court chemin

Est-ce qu'une solution existe toujours ?

Il existe des graphes pour lesquels il n'existe pas de plus court chemin allant d'un point A à un point B :

1. Graphe n'est pas connexe.
2. Cycle de poids négatif.



Plan : 2 - Algorithme de Dijkstra

Un algorithme de plus court chemin

Algorithme de Dijkstra

Permet de déterminer le plus court chemin d'un sommet origine vers tous les autres. Il est conçu pour s'appliquer sur des graphes orientés et pondérés.

- Ne fonctionne que si $\forall (u, v) \in E$ le poids de l'arc $(u, v) \geq 0$.
- Si le graphe est non pondéré, on peut considéré que chaque arc possède un poids de 1.
- Si le graphe est non orienté, on peut considérer que chaque arête (u, v) représente les deux arcs (u, v) et (v, u).

L'algorithme est donc utilisable sur les graphes non pondérés et/ou non orienté.

Description de l'algorithme

Algorithm 8: Dijkstra(G, s)

Data: $G = (V, E)$ un graphe orienté pondéré et $s \in V$ un sommet du graphe
Result: Un arbre des plus courts chemin d'origine s
foreach $u \in V$ **do**
    $Distance[u] = \infty$;
    $Pere[u] = NIL$;
end
$Distance[s] = 0$;
$F = V$;
while $F \neq \emptyset$ **do**
    $u = Extraire - Min(F)$;
    Choisir un sommet $u \in F$ tel que $Distance[u]$ soit minimum;
    $F = F - u$;
    foreach $v \in N(u)$ **do**
        if $Distance[v] > Distance[u] + Poids(u, v)$ **then**
            $Distance[v] = Distance[u] + Poids(u, v)$;
            $Pere[v] = u$;
        end
    end
end

Algorithme de Dijkstra

Explications

- Phase d'initialisation.
  - Chaque sommet est à une distance infinie ne possède pas de père.
  - Le tableau des distances va nous permettre de maintenir la plus courte distance allant de l'origine vers chaque sommet.
  - le tableau des pères nous permettra de savoir depuis quel sommet on arrive. A la fin, ce tableau permet de construire un arbre des plus courts chemins ayant pour racine le sommet d'origine.
  - Le sommet de départ est à une distance de 0 de lui-même.
  - L'ensemble F contient la liste des sommets non traités. A chaque étape, nous allons en choisir un. Il y a donc autant d'étape que de sommet.

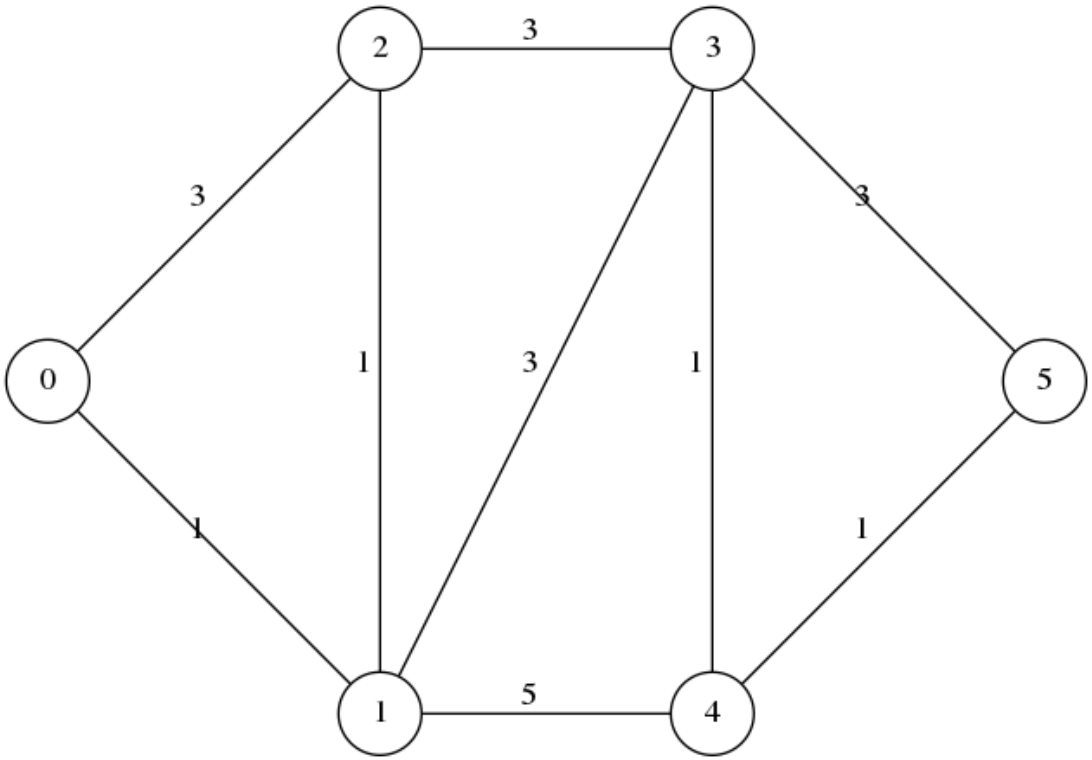
Algorithme de Dijkstra

Explications

- Boucle principale. Continue tant que l'ensemble F n'est pas vide.
  - La fonction Extraire-Min(F) parcours les sommets présents dans F. On sélectionne le sommet u qui possède la distance la plus faible.
  - Le sommet est retiré de F.
  - Pour chaque arc sortant on va regarder si cet arc permet d'améliorer un chemin déjà connu.
  - Si c'est le cas, on met à jour la distance du sommet atteint. Son père devient u, puisque c'est grâce à cet arc que l'on améliore le chemin.

Déroulement de l'algorithme

Graphe initial

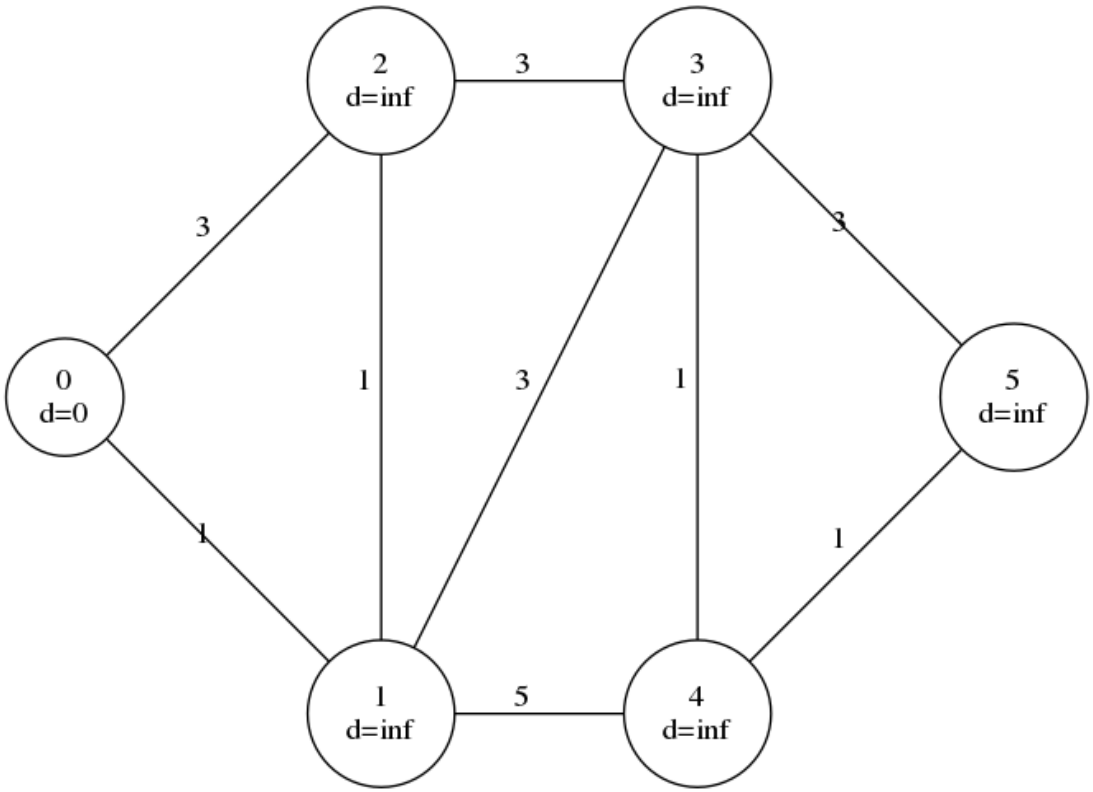


Déroulement de l'algorithme

Phase d'initialisation

Sommet de départ : 0

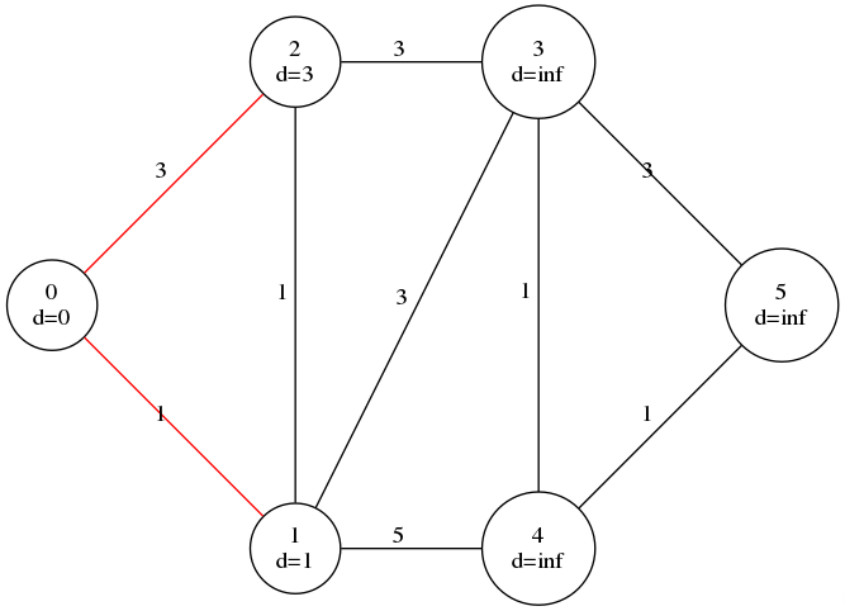
	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Init.		0		∞		∞		∞		∞		∞



Déroulement de l'algorithme

Étape 1

- Sommet a distance minimum : 0 distance : 0
- Parcourons les sommets voisins de 0
  1. Le chemin vers le sommet 1 est amélioré en passant par le sommet 0 car $\infty > 0 + 1$
  2. Le chemin vers le sommet 2 est amélioré en passant par le sommet 0 car $\infty > 0 + 3$



Déroulement de l'algorithme

Étape 1

- Sommet a distance minimum : 0 distance : 0
- Parcourons les sommets voisins de 0
  1. Le chemin vers le sommet 1 est amélioré en passant par le sommet 0 car $\infty > 0 + 1$
  2. Le chemin vers le sommet 2 est amélioré en passant par le sommet 0 car $\infty > 0 + 3$

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞

Déroulement de l'algorithme

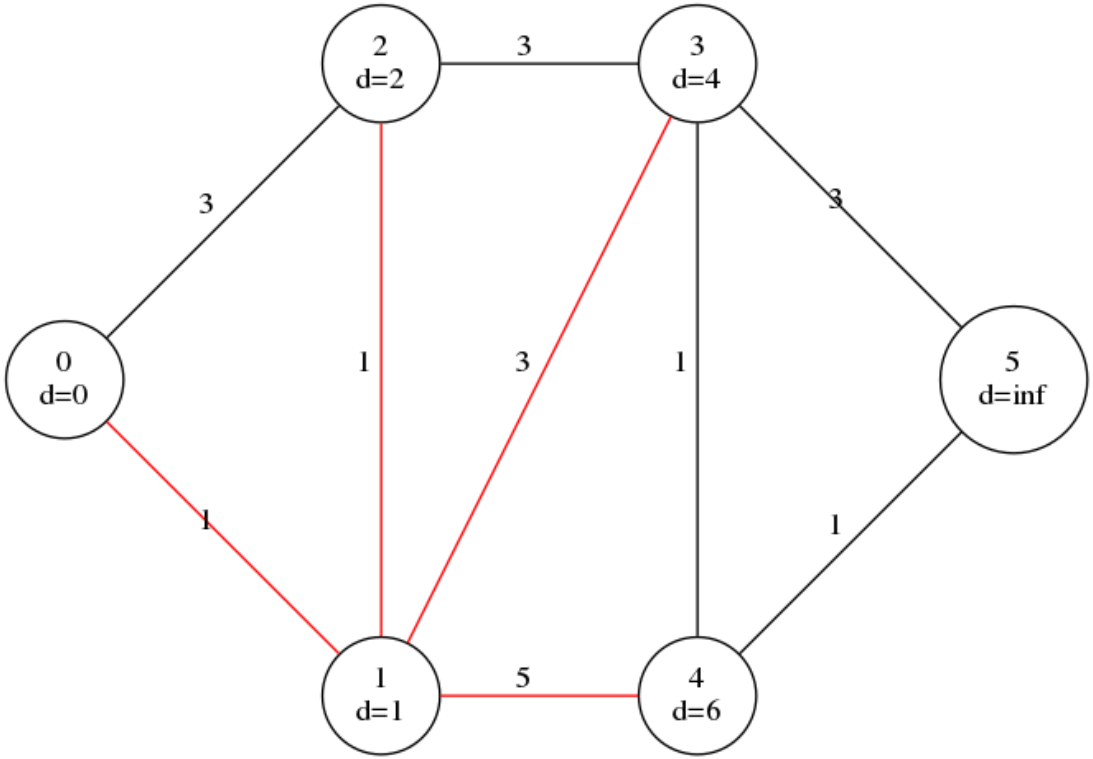
Étape 2

- Sommet a distance minimum : 1 distance : 1
- Parcourons les sommets voisins de 1
  1. Le chemin vers le sommet 0 n'est pas amélioré en passant par le sommet 1
  2. Le chemin vers le sommet 2 est amélioré en passant par le sommet 1 car $3 > 1 + 1$
  3. Le chemin vers le sommet 3 est amélioré en passant par le sommet 1 car $\infty > 1 + 3$
  4. Le chemin vers le sommet 4 est amélioré en passant par le sommet 1 car $\infty > 1 + 5$

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞
Étape 2	n/a	0	0	1	1	2	1	4	1	6	n/a	∞

Déroulement de l'algorithme

Étape 2



Déroulement de l'algorithme

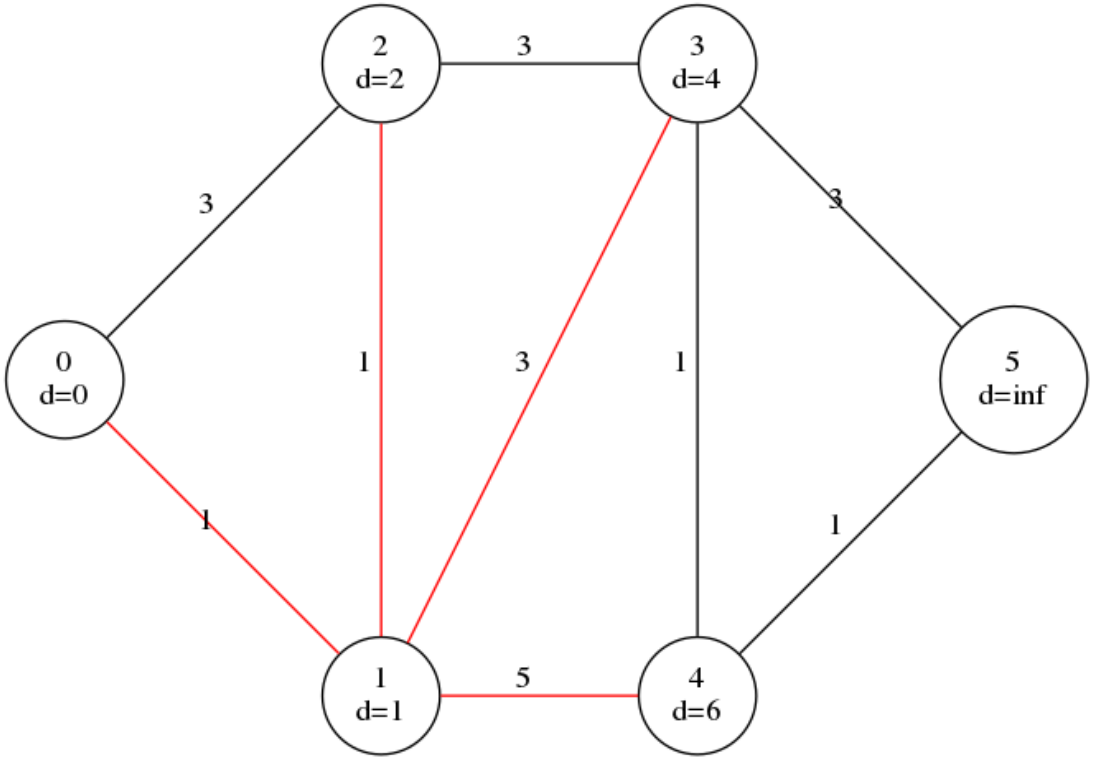
Étape 3

- Sommet a distance minimum : 2 distance : 2
- Parcourons les sommets voisins de 2
  1. Le chemin vers le sommet 0 n'est pas amélioré en passant par le sommet 2
  2. Le chemin vers le sommet 1 n'est pas amélioré en passant par le sommet 2
  3. Le chemin vers le sommet 3 n'est pas amélioré en passant par le sommet 2

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞
Étape 2	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 3	n/a	0	0	1	1	2	1	4	1	6	n/a	∞

Déroulement de l'algorithme

Étape 3



Déroulement de l'algorithme

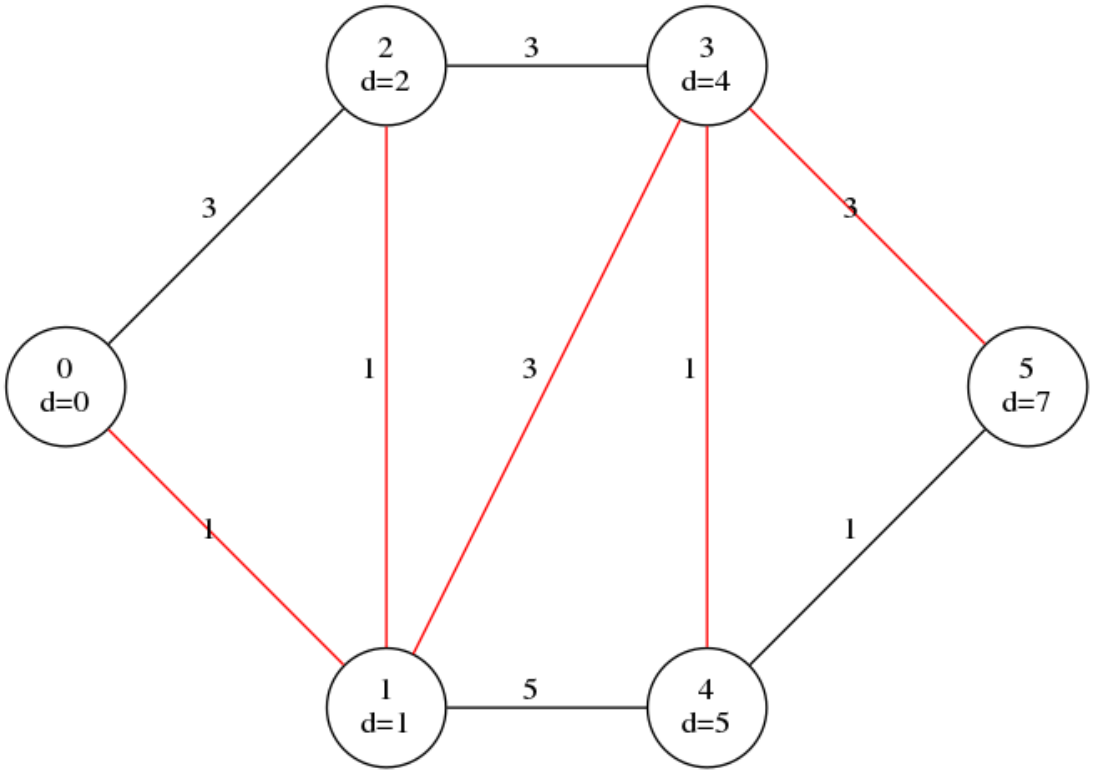
Étape 4

- Sommet a distance minimum : 3 distance : 4
- Parcourons les sommets voisins de 3
  1. Le chemin vers le sommet 1 n'est pas amélioré en passant par le sommet 3
  2. Le chemin vers le sommet 2 n'est pas amélioré en passant par le sommet 3
  3. Le chemin vers le sommet 4 est amélioré en passant par le sommet 3 car $6 > 4 + 1$
  4. Le chemin vers le sommet 5 est amélioré en passant par le sommet 3 car $\infty > 4 + 3$

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞
Étape 2	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 3	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 4	n/a	0	0	1	1	2	1	4	3	5	3	7

Déroulement de l'algorithme

Étape 4



Déroulement de l'algorithme

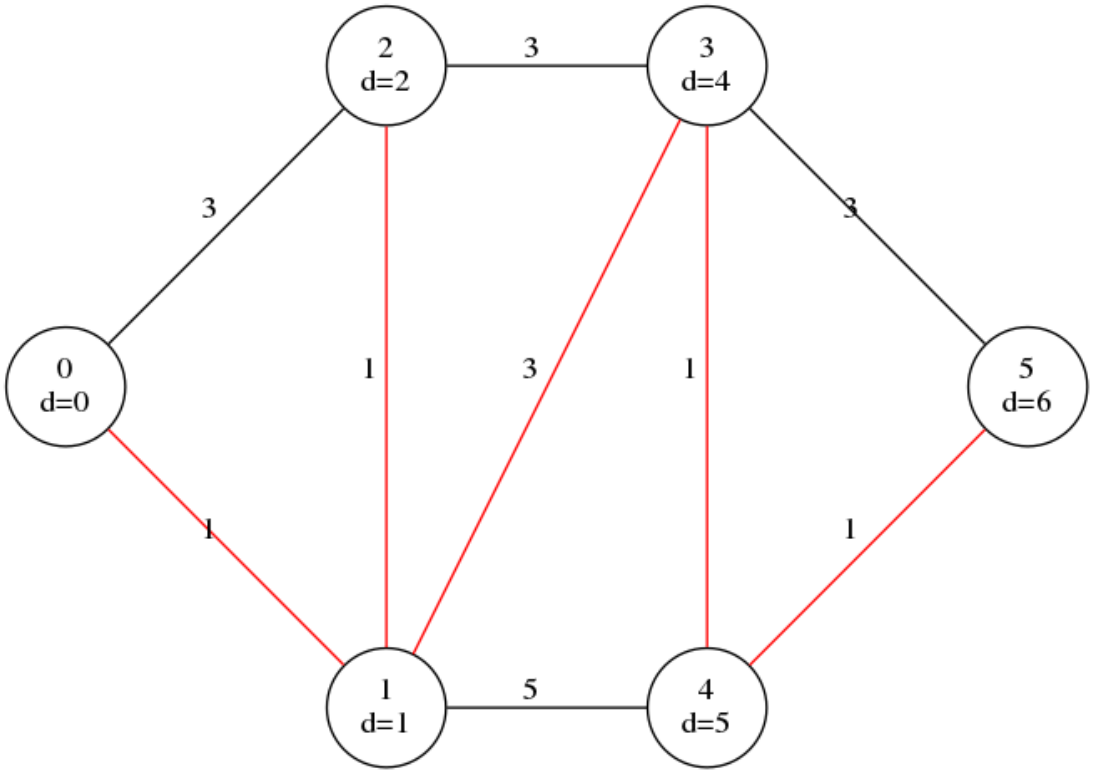
Étape 5

- Sommet a distance minimum : 4 distance : 5
- Parcourons les sommets voisins de 4
  1. Le chemin vers le sommet 1 n'est pas amélioré en passant par le sommet 4
  2. Le chemin vers le sommet 3 n'est pas amélioré en passant par le sommet 4
  3. Le chemin vers le sommet 5 est amélioré en passant par le sommet 4 car $7 > 5 + 1$

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞
Étape 2	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 3	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 4	n/a	0	0	1	1	2	1	4	3	5	3	7
Étape 5	n/a	0	0	1	1	2	1	4	3	5	4	6

Déroulement de l'algorithme

Étape 5



Déroulement de l'algorithme

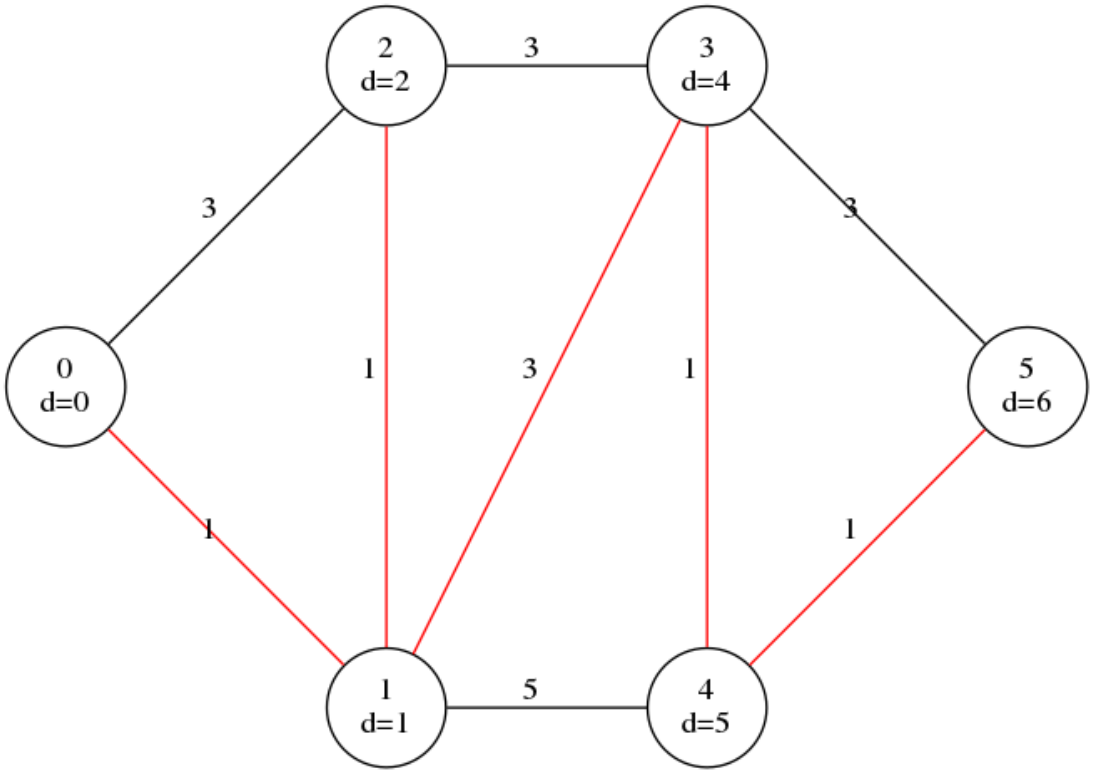
Étape 6

- Sommet a distance minimum : 5 distance : 6
- Parcourons les sommets voisins de 5
  1. Le chemin vers le sommet 3 n'est pas amélioré en passant par le sommet 5
  2. Le chemin vers le sommet 4 n'est pas amélioré en passant par le sommet 5

	P[0]	D[0]	P[1]	D[1]	P[2]	D[2]	P[3]	D[3]	P[4]	D[4]	P[5]	D[5]
Initialisation	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Étape 1	n/a	0	0	1	0	3	n/a	∞	n/a	∞	n/a	∞
Étape 2	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 3	n/a	0	0	1	1	2	1	4	1	6	n/a	∞
Étape 4	n/a	0	0	1	1	2	1	4	3	5	3	7
Étape 5	n/a	0	0	1	1	2	1	4	3	5	4	6
Étape 6	n/a	0	0	1	1	2	1	4	3	5	4	6

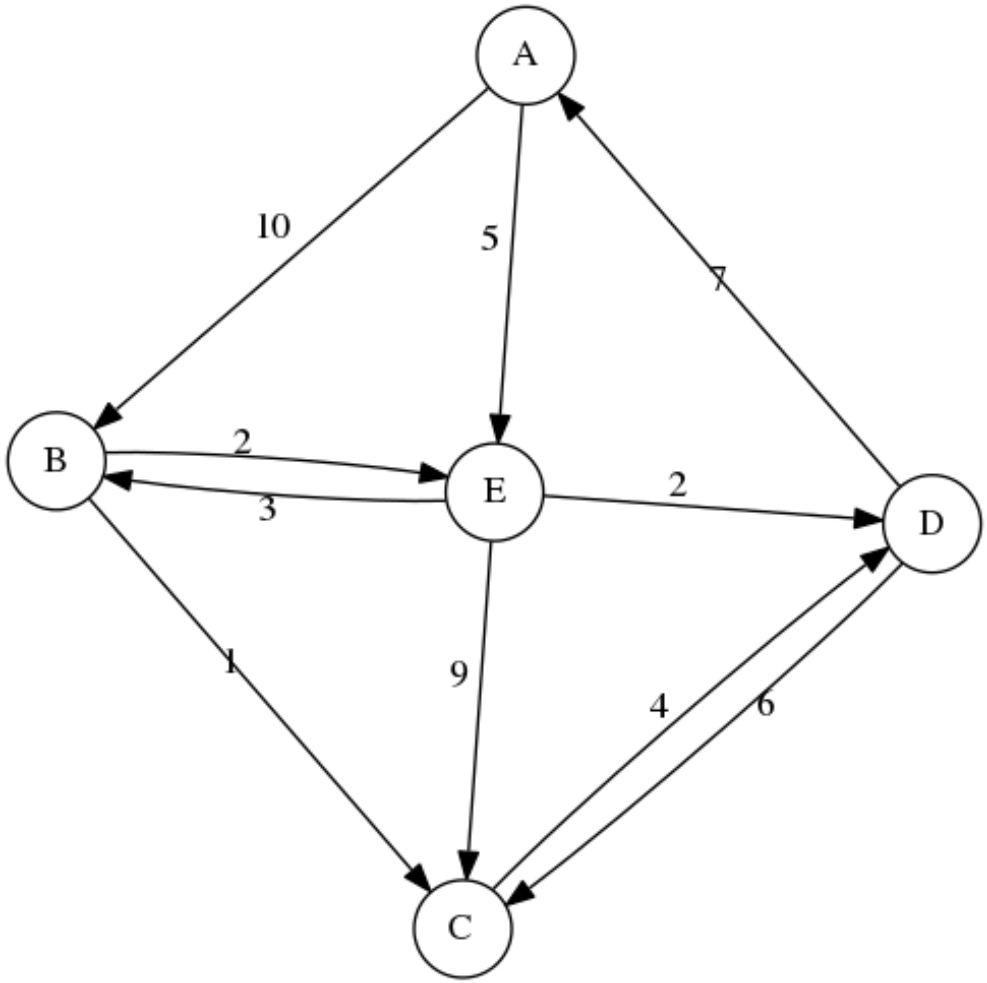
Déroulement de l'algorithme

Étape 6



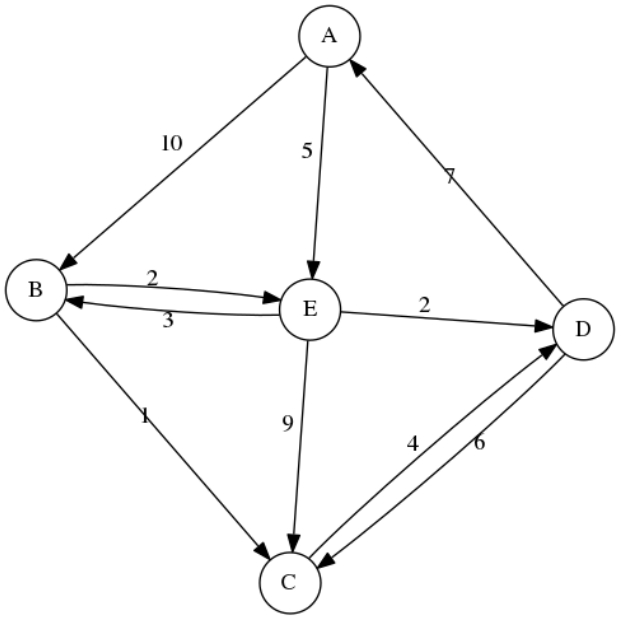
Déroulement de l'algorithme

Appliquez l'algorithme de Dijkstra sur le graphe suivant en partant du sommet A



Déroulement de l'algorithme

Tableau résumé de l'algorithme



	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
Init.	n/a	0	n/a	∞	n/a	∞	n/a	∞	n/a	∞
Ét.1 (A)	n/a	0	A	10	n/a	∞	n/a	∞	A	5
Ét.2 (E)	n/a	0	E	8	E	14	E	7	A	5
Ét.3 (D)	n/a	0	E	8	D	13	E	7	A	5
Ét.4 (B)	n/a	0	E	8	B	9	E	7	A	5
Ét.5 (C)	n/a	0	E	8	B	9	E	7	A	5

Plan : 3 - Algorithme de Bellman-Ford

Algorithme de Bellman-Ford

Une version alternative de l'algorithme de Dijkstra

L'algorithme de Dijkstra

Avantages

Fonctionne très bien ! Et rapidement !

Inconvénients

- Ne fonctionne pas en présence d'arêtes de poids négatif.
- Ne permet pas de détecter les cycles de poids négatif.

L'algorithme de Bellman-Ford est plus lent, mais il fonctionne en présence d'arcs de poids négatifs, et permet de détecter la présence d'un cycle de poids négatif.

Description de l'algorithme

Algorithm 9: Bellman-Ford(G, s)

Data: $G = (V, E)$ un graphe orienté pondéré et $s \in V$ un sommet du graphe
Result: Un arbre des plus courts chemin d'origine s
foreach $u \in V$ **do**
    $Distance[u] = \infty$;
    $Pere[u] = NIL$;
end
$Distance[s] = 0$;
for $|V|$ *fois* **do**
    foreach $(u, v) \in E$ **do**
        if $Distance[v] > Distance[u] + Poids(u, v)$ **then**
            $Distance[v] = Distance[u] + Poids(u, v)$;
            $Pere[v] = u$;
        end
    end
end
foreach $(u, v) \in E$ **do**
    if $Distance[v] > Distance[u] + Poids(u, v)$ **then**
        return Faux;
    end
end
return Vrai;

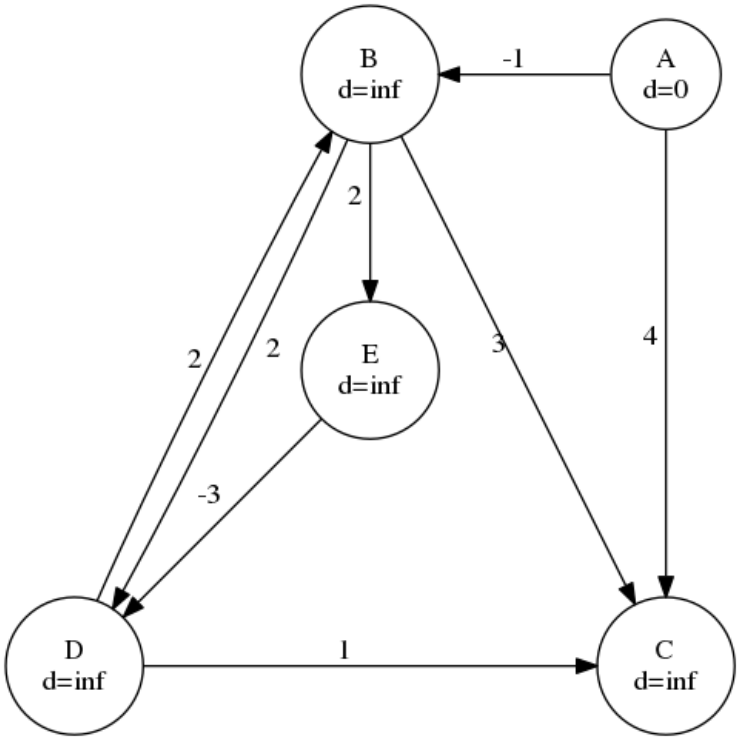
Algorithme de Bellman-Ford

Explications

- Phase d'initialisation. Identique à celle de l'algorithme de Dijkstra.
- Boucle principale. Le nombre de répétitions permet de garantir que les plus courts chemins se seront correctement propagés.
  - Ne fonctionne pas de manière gloutonne.
  - On va propager des informations à chaque itération. Peut importe les sommets impliqués. On va réaliser suffisamment d'itérations pour que l'ensemble des informations soient prises en compte.
  - Pour chaque itération, on va considérer tous les arcs. Si un arc permet d'améliorer un chemin, alors on l'utilise.
- Dernière boucle. On refait une passe. Si une amélioration est encore possible, cela signifie que le graphe contient un cycle de poids négatif.

Déroulement de l'algorithme de Bellman-Ford

Initialisation

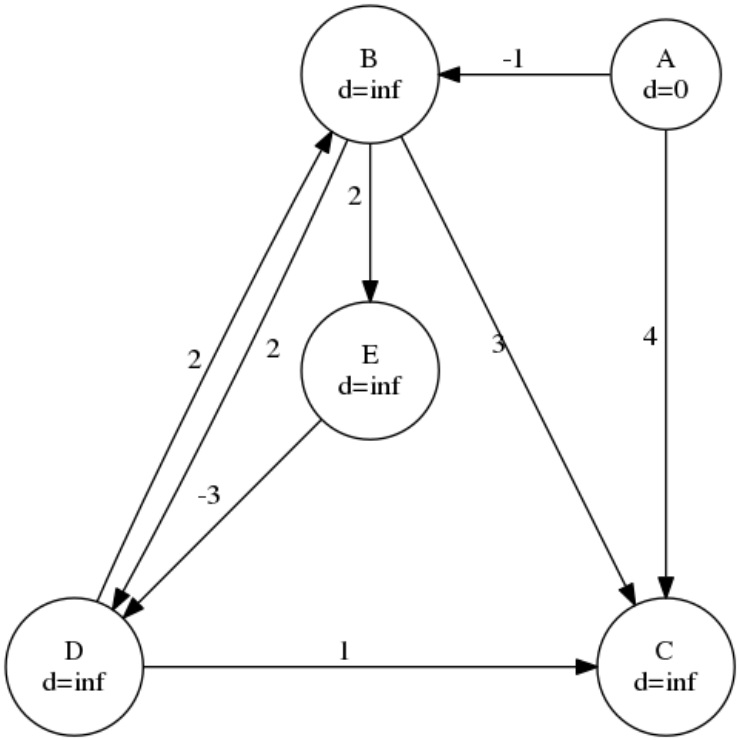


Sommet de départ : A

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf

Déroulement de l'algorithme de Bellman-Ford

Initialisation



Nous allons considérer les arcs dans l'ordre suivant :
$(D, C), (D, B), (E, D), (B, C), (B, E), (B, D), (A, B), (A, C)$

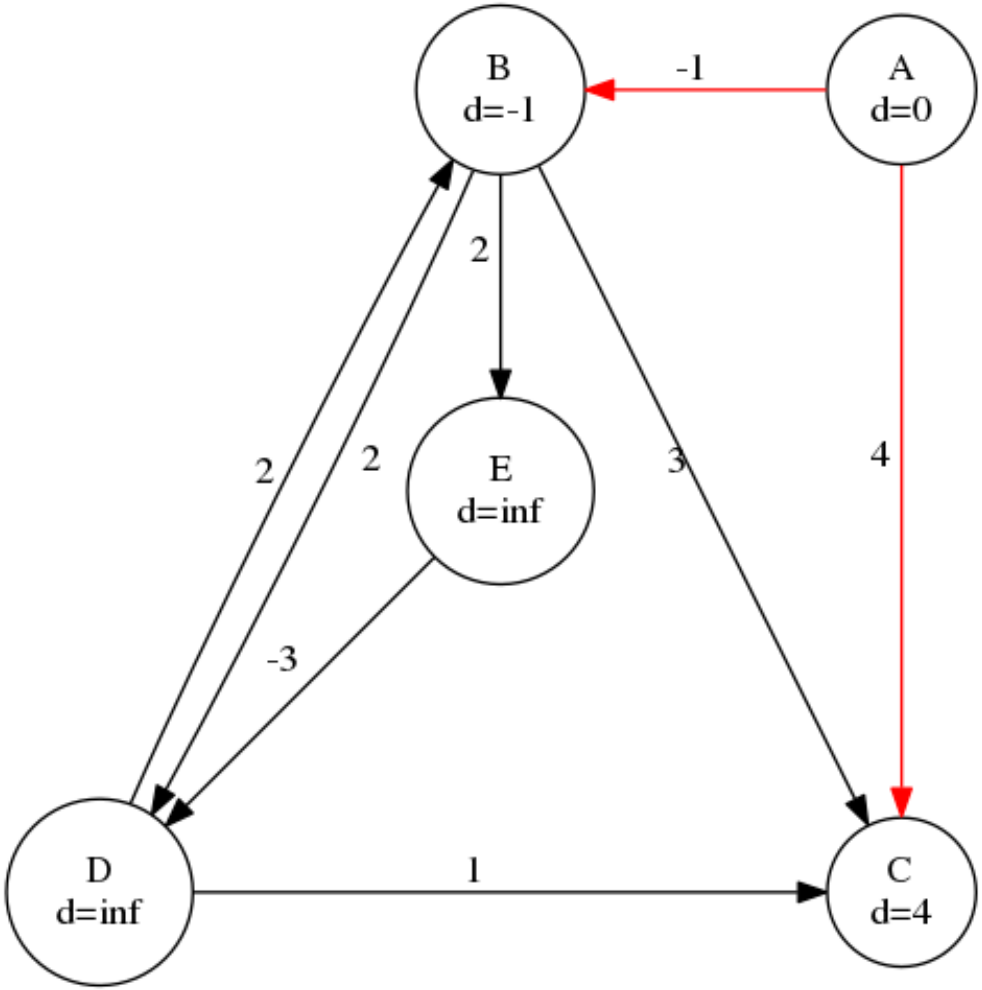
Déroulement de l'algorithme de Bellman-Ford

Étape 1

- Chemin vers le sommet C : pas d'amélioration (D , C)
- Chemin vers le sommet B : pas d'amélioration (D , B)
- Chemin vers le sommet D : pas d'amélioration (E , D)
- Chemin vers le sommet C : pas d'amélioration (B , C)
- Chemin vers le sommet E : pas d'amélioration (B , E)
- Chemin vers le sommet D : pas d'amélioration (B , D)
- Chemin vers le sommet B : amélioration en passant par le sommet A car $inf > 0 + -1$
- Chemin vers le sommet C : amélioration en passant par le sommet A car $inf > 0 + 4$

Déroulement de l'algorithme de Bellman-Ford

Étape 1



Déroulement de l'algorithme de Bellman-Ford

Étape 1

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf
Etape 1	n/a	0	A	-1	A	4	n/a	inf	n/a	inf

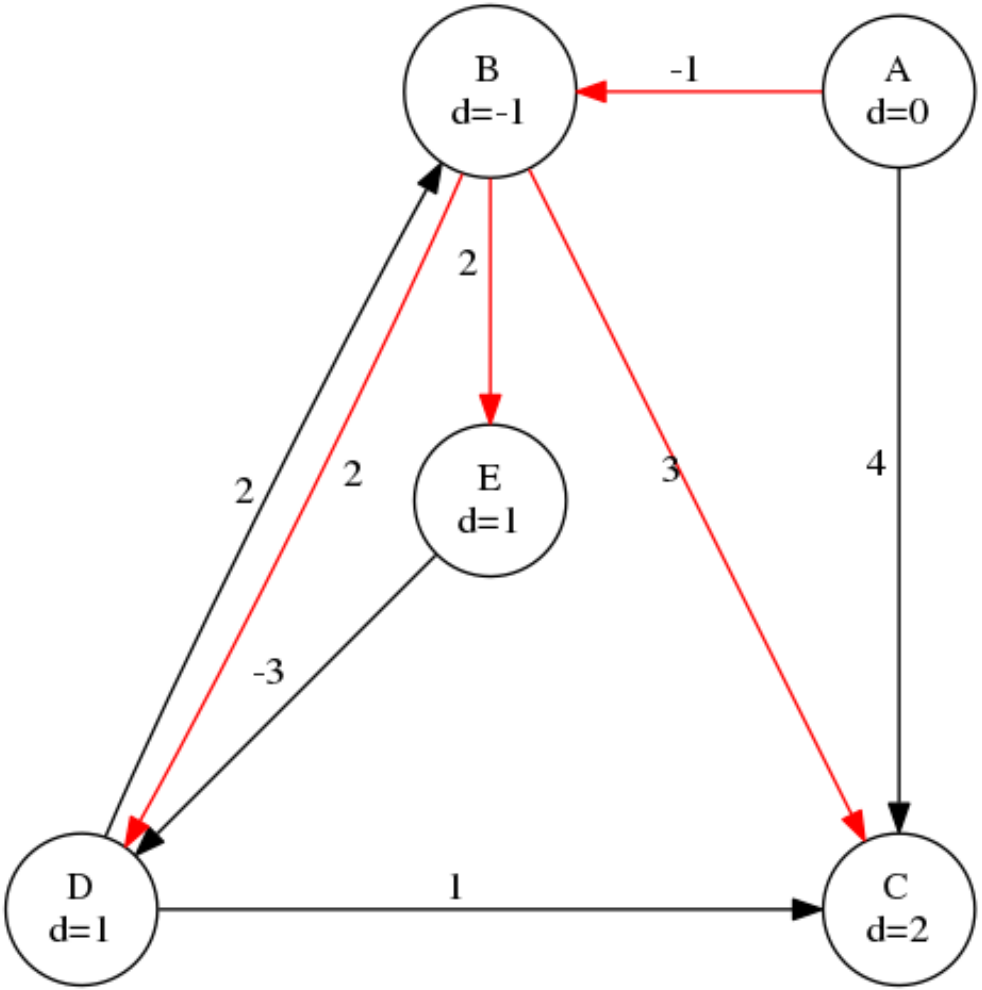
Déroulement de l'algorithme de Bellman-Ford

Étape 2

- Chemin vers le sommet C : pas d'amélioration (D , C)
- Chemin vers le sommet B : pas d'amélioration (D , B)
- Chemin vers le sommet D : pas d'amélioration (E , D)
- Chemin vers le sommet C : amélioration en passant par le sommet B car $4 > -1 + 3$
- Chemin vers le sommet E : amélioration en passant par le sommet B car $inf > -1 + 2$
- Chemin vers le sommet D : amélioration en passant par le sommet B car $inf > -1 + 2$
- Chemin vers le sommet B : pas d'amélioration (A , B)
- Chemin vers le sommet C : pas d'amélioration (A , C)

Déroulement de l'algorithme de Bellman-Ford

Étape 2



Déroulement de l'algorithme de Bellman-Ford

Étape 2

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf
Etape 1	n/a	0	A	-1	A	4	n/a	inf	n/a	inf
Etape 2	n/a	0	A	-1	B	2	B	1	B	1

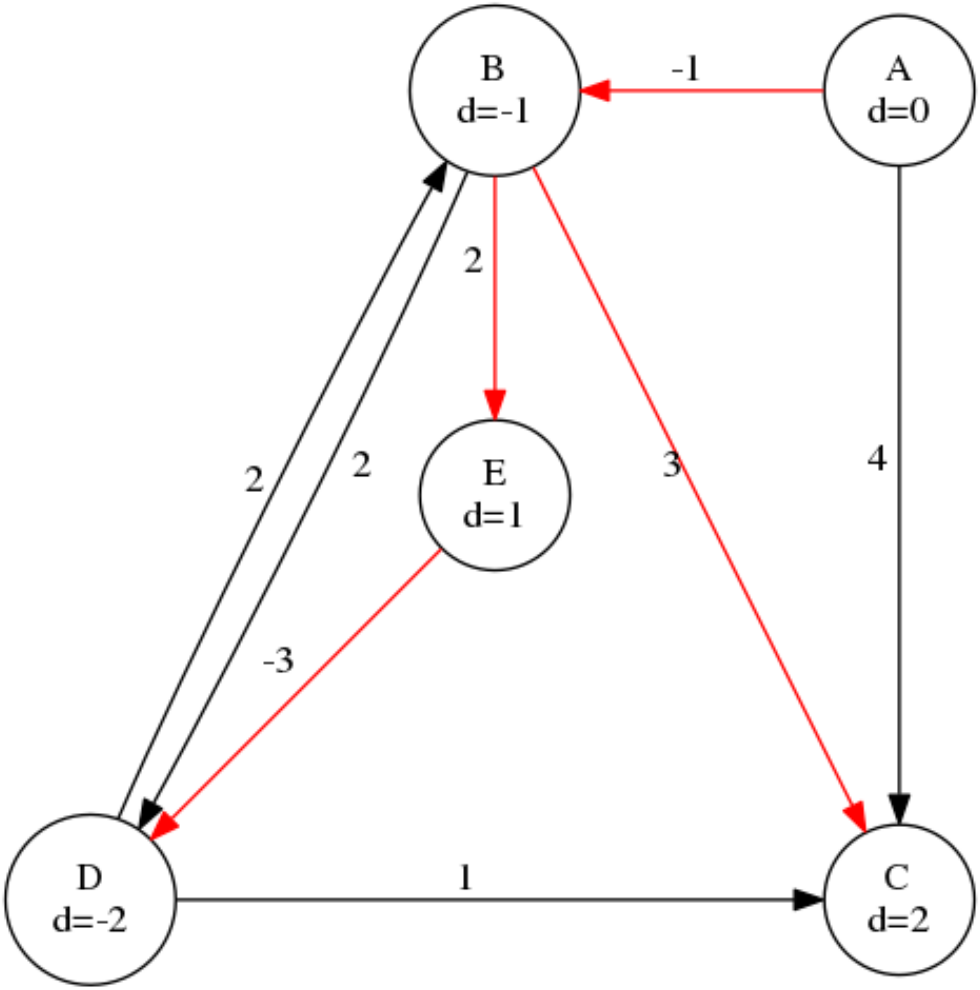
Déroulement de l'algorithme de Bellman-Ford

Étape 3

- Chemin vers le sommet C : pas d'amélioration (D , C)
- Chemin vers le sommet B : pas d'amélioration (D , B)
- Chemin vers le sommet D : amélioration en passant par le sommet E car $1 > 1 + -3$
- Chemin vers le sommet C : pas d'amélioration (B , C)
- Chemin vers le sommet E : pas d'amélioration (B , E)
- Chemin vers le sommet D : pas d'amélioration (B , D)
- Chemin vers le sommet B : pas d'amélioration (A , B)
- Chemin vers le sommet C : pas d'amélioration (A , C)

Déroulement de l'algorithme de Bellman-Ford

Étape 3



Déroulement de l'algorithme de Bellman-Ford

Étape 3

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf
Etape 1	n/a	0	A	-1	A	4	n/a	inf	n/a	inf
Etape 2	n/a	0	A	-1	B	2	B	1	B	1
Etape 3	n/a	0	A	-1	B	2	E	-2	B	1

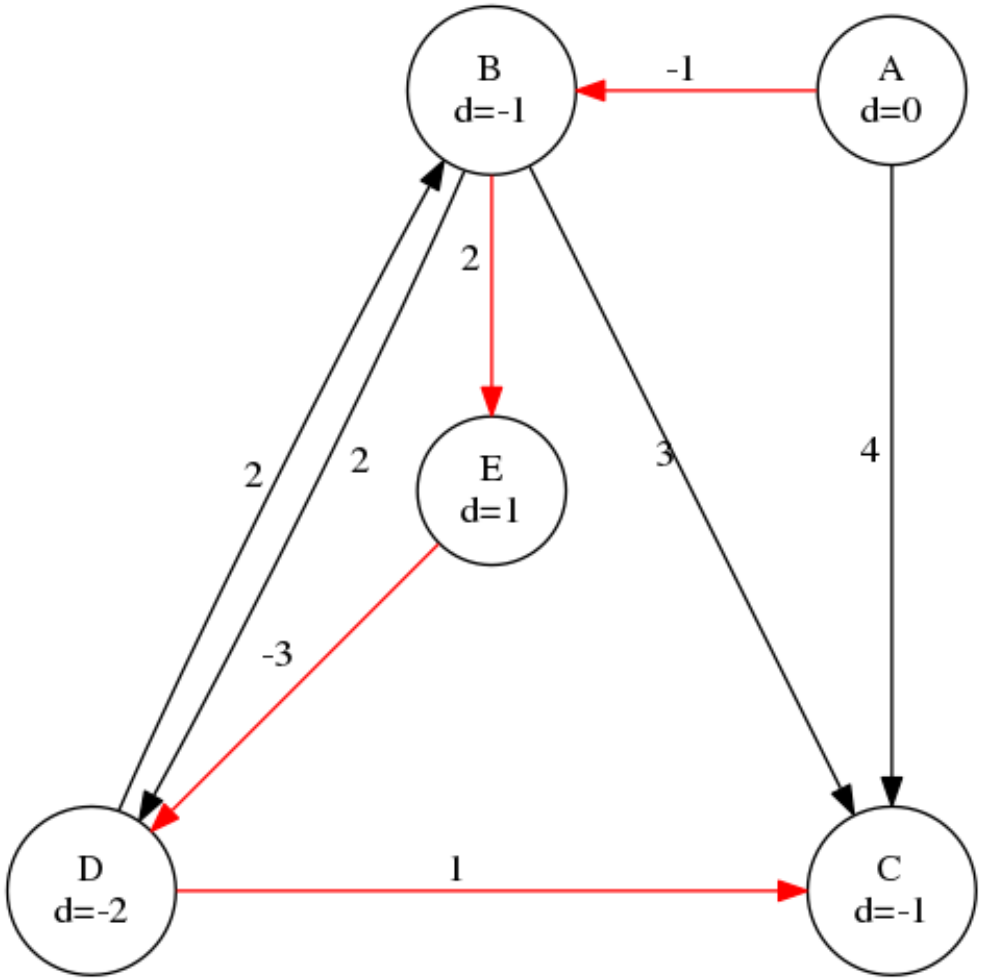
Déroulement de l'algorithme de Bellman-Ford

Étape 4

- Chemin vers le sommet C : amélioration en passant par le sommet D car $2 > -2 + 1$
- Chemin vers le sommet B : pas d'amélioration (D , B)
- Chemin vers le sommet D : pas d'amélioration (E , D)
- Chemin vers le sommet C : pas d'amélioration (B , C)
- Chemin vers le sommet E : pas d'amélioration (B , E)
- Chemin vers le sommet D : pas d'amélioration (B , D)
- Chemin vers le sommet B : pas d'amélioration (A , B)
- Chemin vers le sommet C : pas d'amélioration (A , C)

Déroulement de l'algorithme de Bellman-Ford

Étape 4



Déroulement de l'algorithme de Bellman-Ford

Étape 4

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf
Etape 1	n/a	0	A	-1	A	4	n/a	inf	n/a	inf
Etape 2	n/a	0	A	-1	B	2	B	1	B	1
Etape 3	n/a	0	A	-1	B	2	E	-2	B	1
Etape 4	n/a	0	A	-1	D	-1	E	-2	B	1

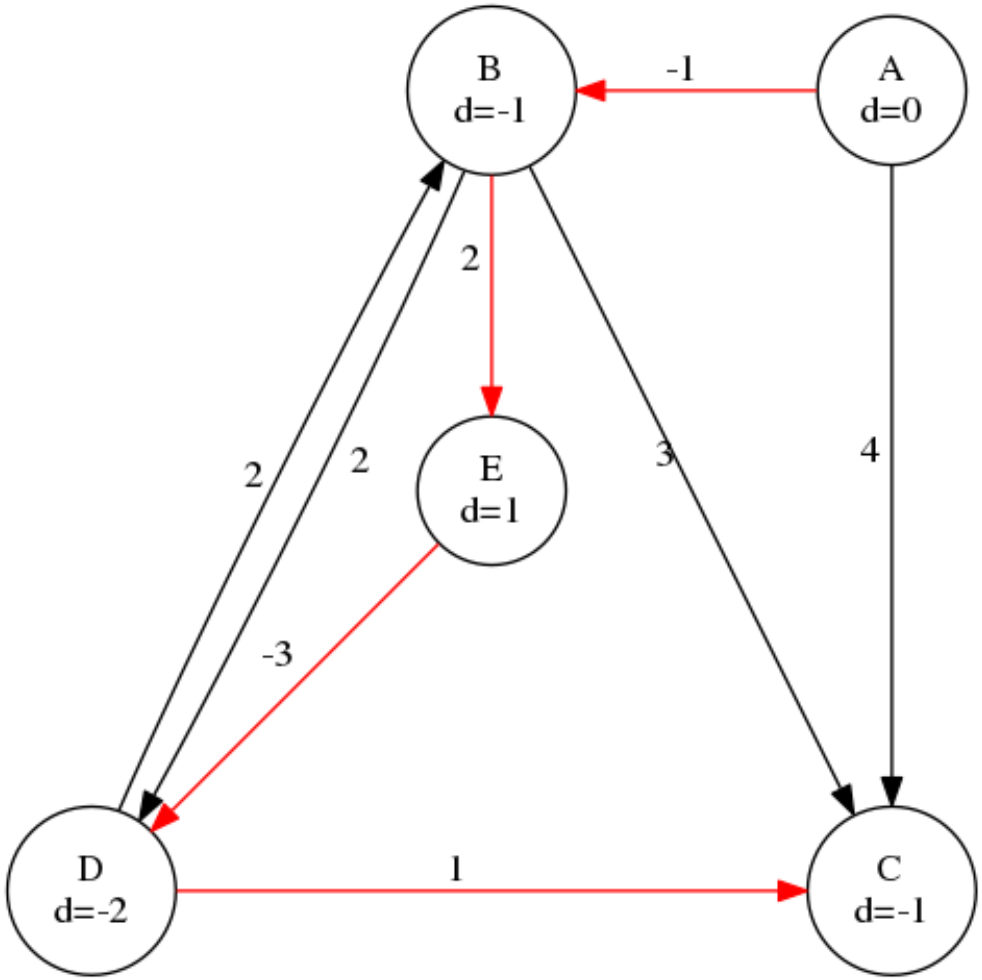
Déroulement de l'algorithme de Bellman-Ford

Étape 5

- Chemin vers le sommet C : pas d'amélioration (D , C)
- Chemin vers le sommet B : pas d'amélioration (D , B)
- Chemin vers le sommet D : pas d'amélioration (E , D)
- Chemin vers le sommet C : pas d'amélioration (B , C)
- Chemin vers le sommet E : pas d'amélioration (B , E)
- Chemin vers le sommet D : pas d'amélioration (B , D)
- Chemin vers le sommet B : pas d'amélioration (A , B)
- Chemin vers le sommet C : pas d'amélioration (A , C)

Déroulement de l'algorithme de Bellman-Ford

Étape 5



Déroulement de l'algorithme de Bellman-Ford

Étape 5 et bilan

	P[A]	D[A]	P[B]	D[B]	P[C]	D[C]	P[D]	D[D]	P[E]	D[E]
	n/a	0	n/a	inf	n/a	inf	n/a	inf	n/a	inf
Etape 1	n/a	0	A	-1	A	4	n/a	inf	n/a	inf
Etape 2	n/a	0	A	-1	B	2	B	1	B	1
Etape 3	n/a	0	A	-1	B	2	E	-2	B	1
Etape 4	n/a	0	A	-1	D	-1	E	-2	B	1
Etape 5	n/a	0	A	-1	D	-1	E	-2	B	1

Plan : 4 - Comparaison des deux algorithmes

Comparaison des deux algorithmes

Temps d'exécution

Soit n le nombre de sommets du graphe et m le nombre d'arcs du graphe. Avec une certaine implémentation, il est possible d'obtenir la complexité suivante :

- L'algorithme de Dijkstra s'exécute en $O((m+n)log(n))$.
- L'algorithme de Bellman-Ford s'exécute en $O(nm)$.

Comparaison des deux algorithmes

Graphes admissibles

- L'algorithme de Dijkstra sélectionne de manière gloutonne le nœud de poids minimum qui n'a pas encore été traité, et effectue ce processus de relaxation sur tous ses arcs sortants.
- L'algorithme de Bellman-Ford effectue la relaxation sur tous les arcs, et fait cela n fois. A chacune de ces itérations, le nombre de sommets avec des distances correctement calculées augmente. Il en résulte que tous les sommets auront finalement leurs distances correctes.

L'algorithme de Bellman-Ford peut donc s'appliquer à une classe d'entrées plus large que l'algorithme de Dijkstra.