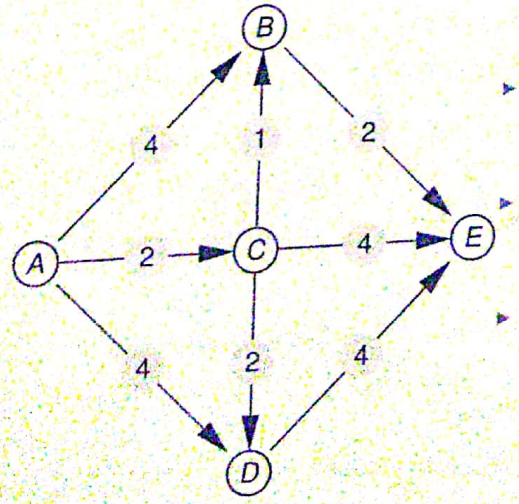
Algorithmes de plus court chemin dans les graphes orientés valués

Définition (Distance et plus court chemin)

Soit $G = (S, A, f)$ un graphe orienté valué et soient x, y deux éléments de S.

- *On appelle* **distance** *de x à y et on note $d(x, y)$ le minimum des valuations des chemins allant de x à y.*
- *On appelle* **plus court chemin** *de x à y tout chemin dont la valuation est égale à $d(x, y)$. Autrement dit, un plus court chemin entre deux sommets est un chemin de valuation minimale.*



- distance de A à E : 5
  plus court(s) chemin(s) : A C BE
- distance de A à D : 4
  plus court(s) chemin(s) : AD et ACD
- distance de E à A : ~~∅~~
  plus court(s) chemin(s) : ?

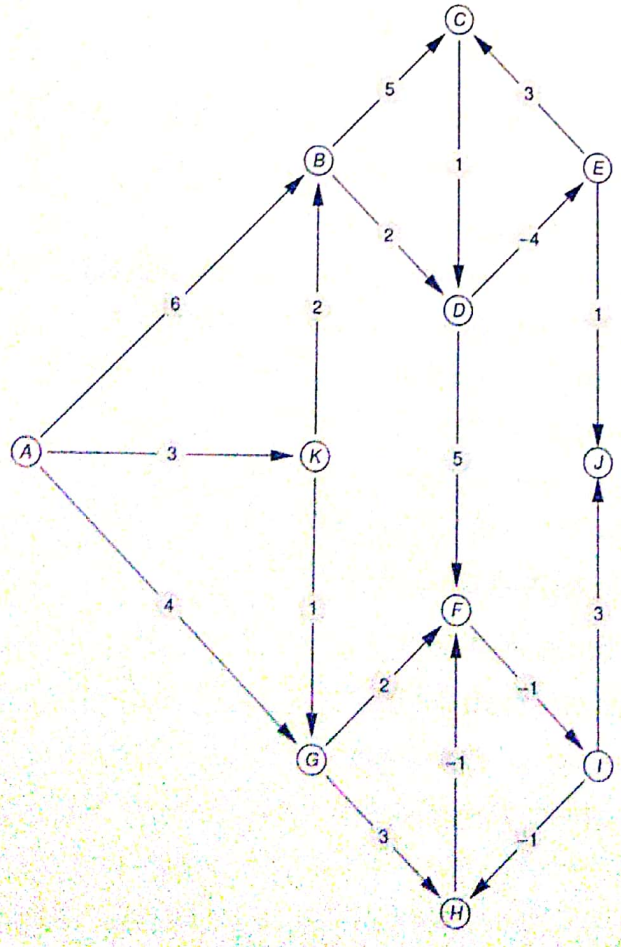
Dans un graphe orienté valué, étant donnés deux sommets x et y, plusieurs cas se présentent :

- il n'y a pas de chemin de x à y ;
- il existe un ou plusieurs plus court(s) chemin(s)[3] de x à y ;
- il existe des chemins de x à y, mais pas de plus court.

Déterminer un exemple illustrant chacune des situations ci-dessus...

3. voire une infinité de plus courts chemins !

Algorithmes de plus court chemin | Distance, plus court chemin



- **de A à B :** 5 AKB
- **de A à G :** ~~infinité~~ 4 AKG ou AG
- **de E à A :** ∅
- **de A à E :** infinité 3 AKDE
- **de A à J :** 4 AKDEJ
  $-\infty$

Définition (Circuit absorbant)

Un **circuit absorbant** *est un circuit de valuation négative.*

Remarque. Si un graphe possède un circuit absorbant, alors il n'existe pas de plus court(s) chemin(s) entre certains de ses sommets. □

Théorème

Soit G un graphe orienté valué ne possédant pas de circuit absorbant. Soient x, y deux sommets de G. S'il existe un chemin allant de x à y, alors la distance $d(x, y)$ est bien définie et il existe au moins un plus court chemin de x à y.

Dans la suite, les graphes seront donc *sans circuits absorbants.*

L'**algorithme de Dijkstra** est un algorithme de recherche de distance et de plus court chemin d'un sommet choisi à tous les autres sommets. Il ne fonctionne que si toutes les valuations des arcs sont positives.

Son principe est le suivant :

- On construit petit à petit, à partir d'un sommet choisi s_0, un ensemble M de sommets marqués. Pour tout sommet marqué s, l'estimation $d(s)$ est égale à la distance entre s_0 et s.
- À chaque étape, on sélectionne un sommet non marqué x dont la distance estimée $d(x)$ est la plus petite parmi les distances estimées des sommets non marqués.
- On "marque" alors x (i.e. on ajoute x à M), puis on met à jour à partir de x les distances estimées des successeurs non marqués de x.
- On recommence, jusqu'à épuisement des sommets non marqués.

Supposons que l'on ait déterminé la distance du sommet 1 à tous les autres sommets en mettant en œuvre l'algorithme de Dijskstra. Pour connaître la distance d'un sommet $i \neq 1$ à tous les autres sommets, on ne peut éviter une nouvelle mise en œuvre de l'algorithme de Dijskstra.

```
INITIALISATION :
    M = {1} ;
    d(1) = 0 ;
    P(1) = ∅ ;
    POUR i = 2, ..., n :
        d(i) = f(1,i) ET P(i) = 1 S'IL EXISTE UN ARC DE 1 VERS i ;
        d(i) = +∞ ET P(i) = ∅ SINON ;
    FIN (POUR)
TANT QUE M ≠ S :
    CHOISIR x ∈ S \ M TEL QUE d(x) = min_{i∈S\M} d(i) ;
    M = M ∪ {x} ;
    POUR TOUT i ∈ S \ M SUCCESSEUR DE x :
        SI d(i) > d(x) + f(x,i), ALORS d(i) = d(x) + f(x,i) ET P(i) = x ;
    FIN (POUR)
FIN (TANT QUE)
```

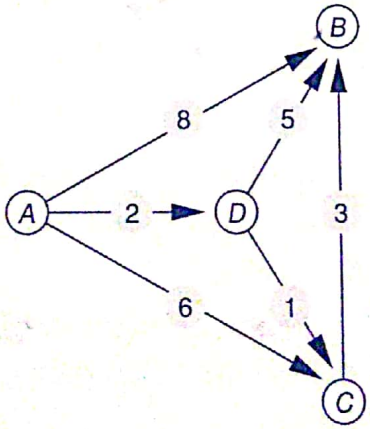
La liste M est la liste des sommets "marqués" (i.e. traités par l'algorithme).

Le nombre réel $d(i)$ fournit la **distance** du sommet 1 au sommet i.

Le sommet $P(i)$ définit le **prédécesseur** (ou **père**) du sommet i : cette information est essentielle pour construire a posteriori un plus court chemin entre le sommet 1 et tout autre sommet.

Algorithmes de plus court chemin | Algorithme de Dijkstra

Considérons le graphe orienté suivant :

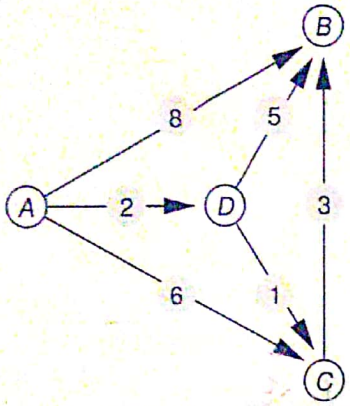


La mise en œuvre de l'algorithme fournit le résultat suivant :

M	$d(A)$	$d(B)$	$d(C)$	$d(D)$	$P(A)$	$P(B)$	$P(C)$	$P(D)$
{A}	0 ∅	8 A	6 A	2 A				
{A,D}		9 D	3 D					
{A,C}		6 C						
	0 ∅	6 C	3 D	2 A				

Algorithmes de plus court chemin | Algorithme de Dijkstra

L'algorithme de Dijkstra permet de déterminer les *distances* mais aussi *les plus courts chemins* d'un sommet choisi (par exemple, le sommet 1) à tout autre sommet *i*. Pour cela, on utilise le *tableau des pères (ou prédécesseurs)* comme une table de routage : en partant de la fin, on remonte le chemin en sélectionnant successivement chaque prédécesseur du sommet considéré.



s	A	B	C	D
$d(s)$ $P(s)$	0 ∅	6 C	3 D	2 A

Pour déterminer un plus court chemin de A à B :

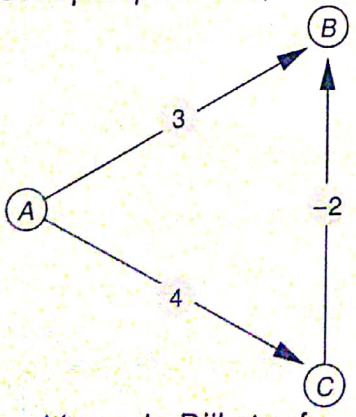
P(B) = C P(C) = D P(D) = A

Un plus court chemin de A à B est

A, D, C, B

Algorithmes de plus court chemin | Algorithme de Dijkstra

L'algorithme de Dijkstra peut être mis en défaut si toutes les valuations *ne sont pas positives*, comme le montre l'exemple suivant.

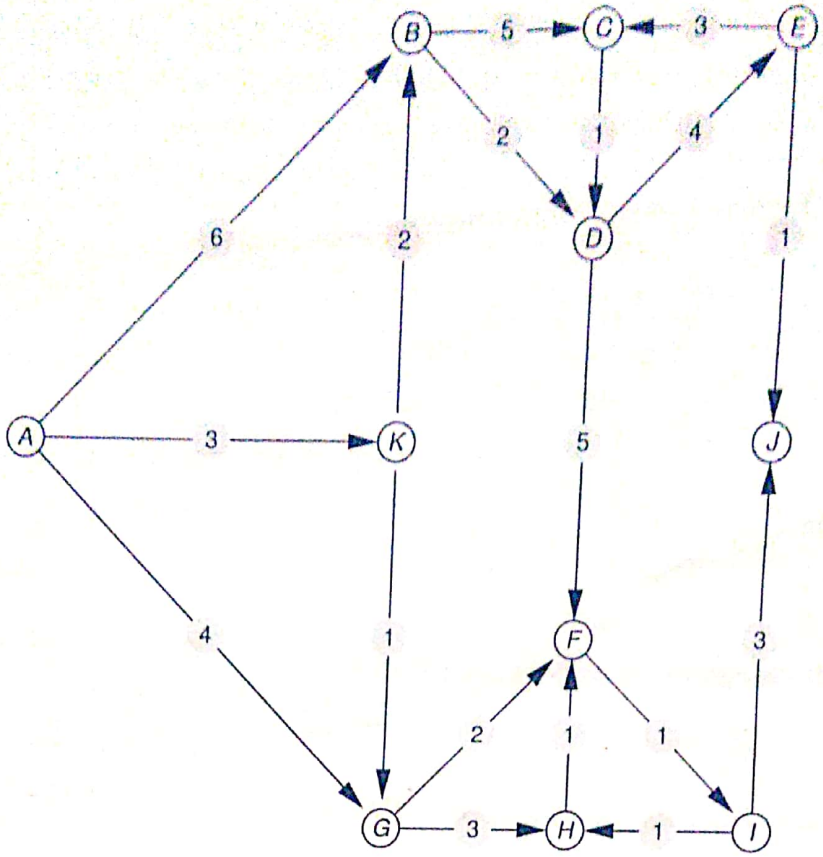


Il est facile de voir que la distance de A aux autres sommets est la suivante :

s	A	B	C
$d(s)$	0	2	4

L'algorithme de Dijkstra fournit le résultat *erroné* suivant :

M	$d(A)$ $P(A)$	$d(B)$ $P(B)$	$d(C)$ $P(C)$
[illegible]	0 ∅	3 A	4 A
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	0 ∅	3 A	4 A



Algorithmes de plus court chemin | Algorithme de Bellman

Les graphes orientés **sans circuit** possèdent des propriétés spécifiques : en particulier il existe dans un graphe sans circuit une notion de hiérarchie entre les sommets. C'est ce qu'on appelle la **décomposition en niveaux** ou aussi un "tri topologique".

Proposition (Décomposition en niveaux d'un graphe sans circuit)

Si G est un graphe sans circuit, alors on peut définir pour chaque sommet un niveau de la manière suivante :

- *les sommets sans prédécesseurs sont de rang 0 ;*
- *tout sommet x a un rang supérieur aux rangs de ses prédécesseurs :*

$$\text{rang}(x) = \max_{y \in N^-(x)} \text{rang}(y) + 1.$$

Remarque. En utilisant ce tri, on peut redessiner le graphe G en disposant les sommets de gauche à droite dans l'ordre croissant des rangs. □

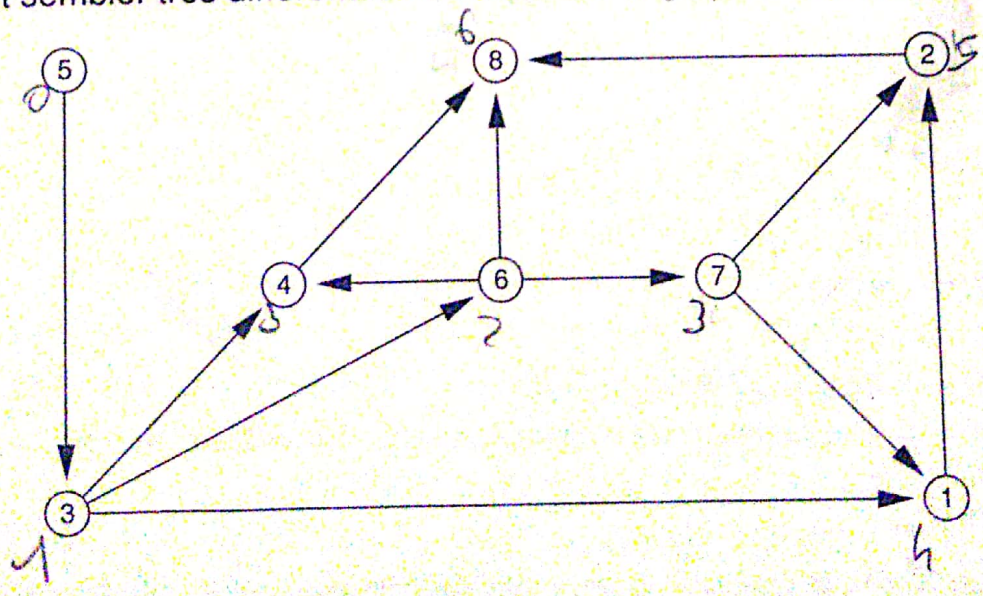
Si G possède un circuit $C = (x_0, x_1, ..., x_{n-1}, x_0)$, il ne peut admettre de décomposition en niveaux, puisque l'on aurait alors la propriété :

$$\text{rang}(x_0) < \text{rang}(x_1) < ... < \text{rang}(x_{n-1}) < \text{rang}(x_0).$$

Il existe des algorithmes qui permettent de déterminer la décomposition en niveaux d'un graphe sans circuit. Par souci de simplicité, ces méthodes de décomposition ne sont pas au programme et nous considérerons des graphes décomposés en niveaux.

Algorithmes de plus court chemin | Algorithme de Bellman

Pour rappel, un graphe peut être défini par des représentations sagittales qui peuvent sembler très différentes. Considérons le graphe orienté suivant :



Remarque. L'algorithme de Bellman permet aussi de déterminer les **plus longs chemins** (i.e. les chemins de longueur maximale) dans un graphe. Dans ce cas il suffit de modifier l'instruction

$$d(i) = \min_{j \in N^-(i)} (d(j) + f(j,i))$$

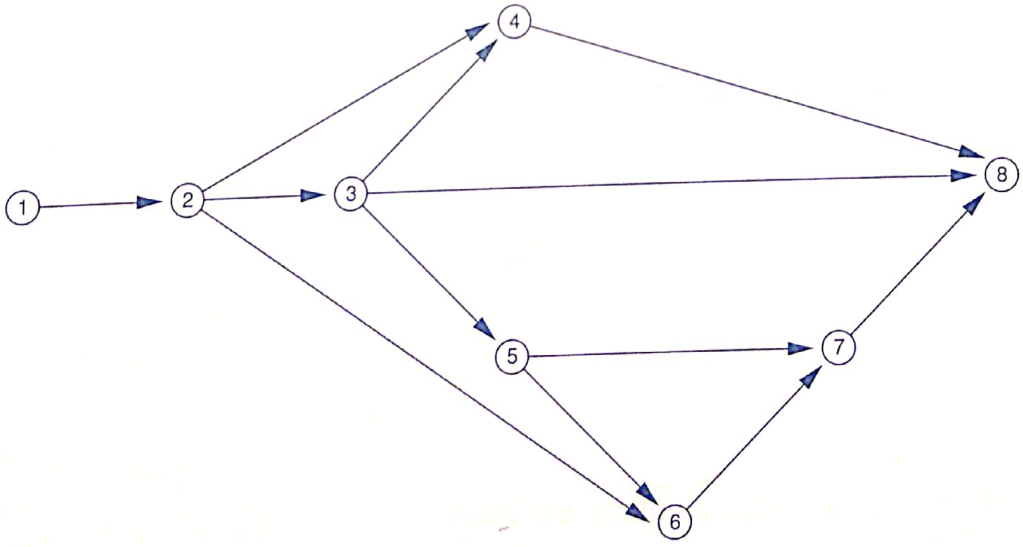
en

$$d(i) = \max_{j \in N^-(i)} (d(j) + f(j,i))$$

et d'initialiser les distances à $-\infty$ au lieu de $+\infty$. □

Algorithmes de plus court chemin | Algorithme de Bellman

Considérons le graphe suivant. Déterminons la distance et un plus court chemin du sommet 1 aux autres sommets en nombre d'arcs :



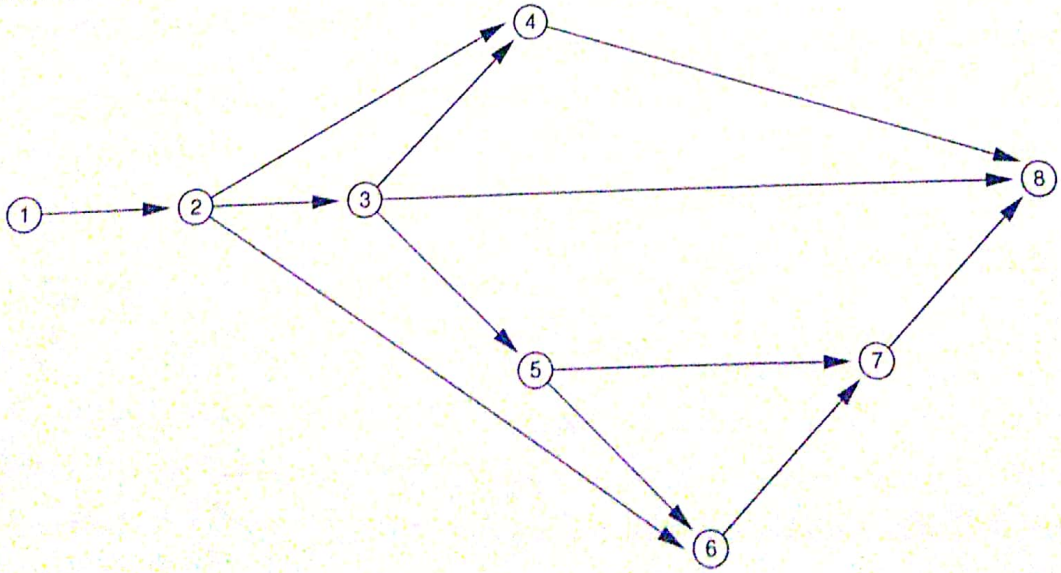
n	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$	$d(7)$	$d(8)$
1								
2								
3								
4								
5								
6								
7								
8								

n	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	$P(6)$	$P(7)$	$P(8)$
1								
2								
3								
4								
5								
6								
7								
8								

TABLE: Résultat de la recherche de plus court chemin par l'algorithme de Bellman.

Algorithmes de plus court chemin | Algorithme de Bellman

Considérons le graphe suivant. Déterminons la distance et un plus court chemin du sommet 1 aux autres sommets en nombre d'arcs :

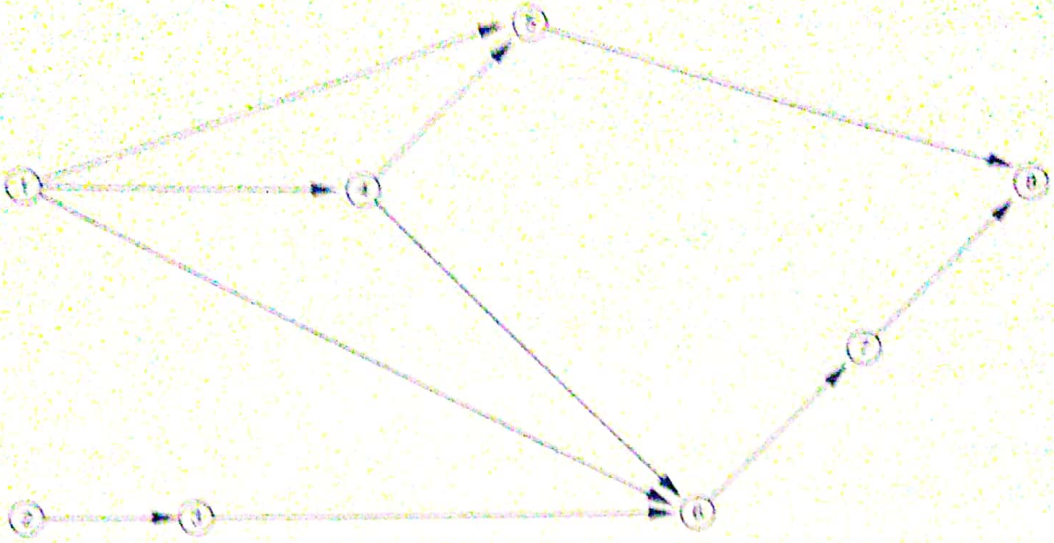


Mise en œuvre de l'algorithme de Bellman :

i	1	2	3	4	5	6	7	8
$d(i)\,(P(i))$	0(Ø)	1(1)	2(2)	2(2)	3(3)	3(2)	3(6)	3(3)

Algorithmes de plus court chemin | Algorithme de Bellman

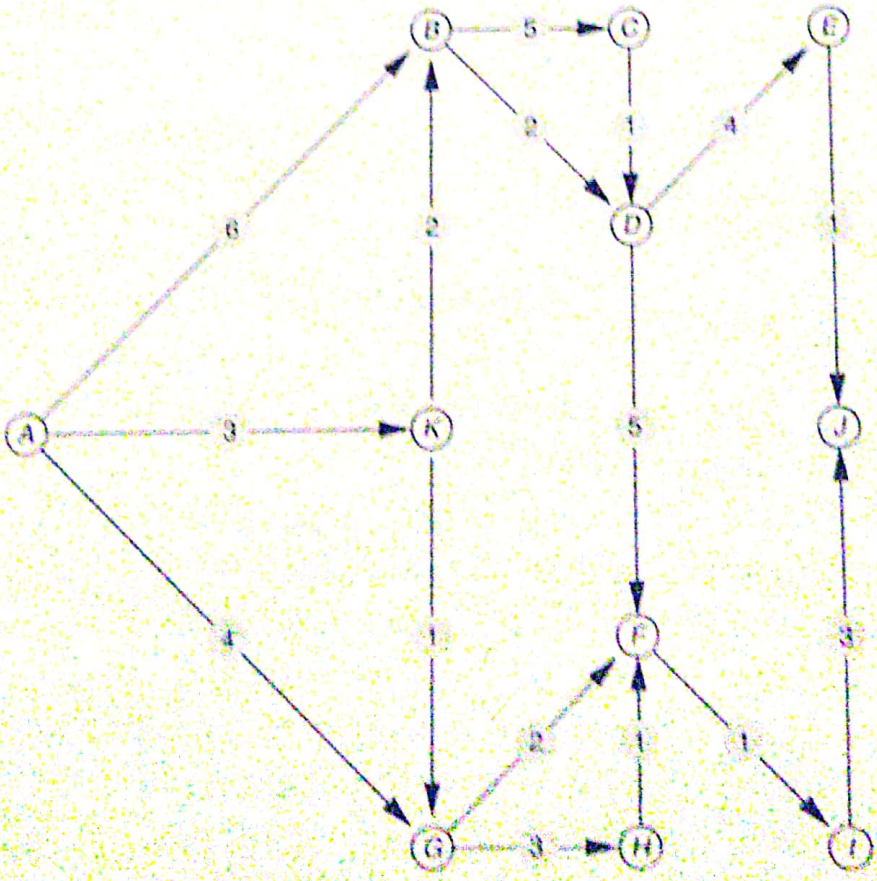
Considérons le graphe suivant. Déterminons la distance et un plus court chemin du sommet 1 aux autres sommets en nombre d'arcs :



Mise en œuvre de l'algorithme de Bellman :

i	1	2	3	4	5	6	7	8
d(i) (P(i))								

Algorithmes de plus court chemin | Algorithme de Bellman



Les notions de distance et plus courte *chaîne*, dans le cadre des graphes *non orientés* valués, sont similaires aux notions définies précédemment dans le cadre des graphes orientés :

Définition (Distance et plus courte chaîne)

Soit $G = (S, A, f)$ un graphe non orienté valué et soient $x, y \in S$.

- *On appelle* **distance de** x **à** y *et on note $d(x, y)$ le minimum des valuations des chaînes reliant x à y.*
- *On appelle* **plus courte chaîne de** x **à** y *toute chaîne dont la valuation est égale à $d(x, y)$.*

Remarque. On a toujours $d(x, y) = d(y, x)$, et toute plus courte chaîne de x à y parcourue à l'envers est une plus courte chaîne de y à x. □

Remarque. Soient deux sommets x et y. Plusieurs cas se présentent :

- il n'y a pas de chaîne de x à y ;
- il existe une ou plusieurs plus courte(s) chaîne(s) de x à y ;
- il existe des chaînes de x à y, mais pas de plus courte(s). □

Question : Peut-on utiliser les algorithmes de Dijkstra et Bellman ?