Cheminement dans les graphes

Graphe pondéré : On appelle *graphe pondéré* un graphe tel que, à chaque arête *e* est associé un *poids* P_e. Les applications des graphes pondérés sont nombreuses : cartes routières avec des indications de durée, de tarif ou de distance portée sur des routes entre deux lieux, par exemple.

Réseau : On appel réseau un graphe pondéré ayant un seul sommet *S* avec $d^-g(S)=0$ appelé *source* et un seul sommet *P* avec $d^+g(P)=0$ appelé *puit*.

Remarque : Dans le cas ou on a un graphe pondéré avec plusieurs sommets x_i ayant $dg^-(x_i)=0$ et plusieurs sommets y_i ayant $d^+g(y_i)=0$, alors il suffit d'ajouter deux sommets fictifs *S* et *P* tels que : *S* sera relié à tous les sommets x_i et tous les sommets y_i seront reliés *à P*. les pondérations des arcs fictifs ajoutés seront égales à 0 où l'infini (∞) suivant le sens des pondérations du graphe initial.

Chemin: Un chemin *Ch* est une suite d'arcs adjacents parcouru dans le même sens. La longueur de ce chemin *l(Ch)* peut être définit de deux manières différentes:

1/ La longueur en termes d'arcs, et la on donne le nombre d'arcs constituants ce chemin. *l(Ch)*=nombre d'arcs de *ch*.

2/ La longueur en termes de pondérations, et la on donne la somme des pondérations des arcs constituants ce chemin.$l(Ch)= \sum P_i \ \forall i \in Ch$.

***Circuit* :** Un circuit est un chemin simple dont les deux extrémités coïncident. Ainsi on défini la longueur d'un circuit de la même manière, par la longueur en terme d'arcs et la longueur en termes de pondérations.

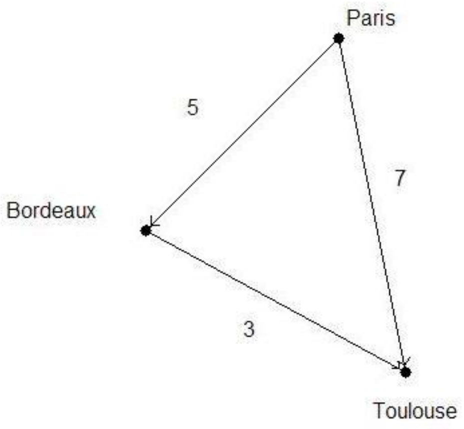
Circuit absorbant : Un circuit *C* est dit *absorbant* ssi, $l(C) = \sum P_i < 0 \ \ \forall i \in C$.

Plus court chemin : On appel ***plus court chemin*** dans un réseau R, le chemin reliant *S* à *P* dans R, ayant la somme des pondérations minimales de tous les chemins reliant *S* à *P* dans R.

***Le Problème du plus court chemin* :** Le problème du plus court chemin est un problème ancien, présent dans de nombreux domaines (trafics autoroutiers, ferroviaires, maritimes, investissement et gestion des stocks, optimisation d'un réseau, intelligence artificielle, etc…).

Le problème est simple : Comment, en partant d'un point, arriver à un autre point en faisant le moins de chemin possible ?

Pour vous aider, prenons un exemple : Une voiture partant de Paris souhaite se rendre à Toulouse. Mais l'autoroute entre Paris et Toulouse est bouchée. Le conducteur aimerait savoir si, en passant par Bordeaux, il mettra plus ou moins de temps que pour aller à Toulouse par l'autoroute. Sachant que le conducteur mettra 7 heures en passant par l'autoroute, et qu'il mettra 5 heures pour aller à Bordeaux, et 3 heures pour relier Bordeaux à Toulouse. Le graphe ci-dessous résume le problème.



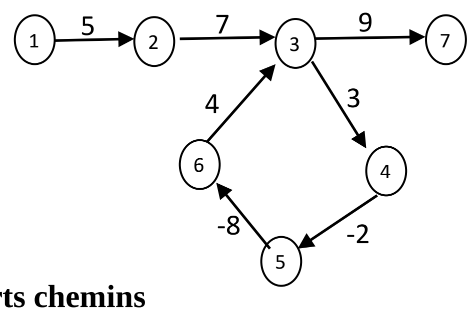
Chaque arc possède une pondération représentant le temps nécessaire de déplacement d'un sommet vers un autre. On dit que ce graphe est un graphe « orienté », car les arcs ont un sens. Pour déterminer le plus court chemin entre Paris et Toulouse, il suffit d'additionner les longueurs d'arcs des différents passages pour aller de Paris à Toulouse, et de les comparer.
Ici, on a l (P => T) = 7 et l (P => B) + l (B => T) = 5 + 3 = 8

On peut donc voir que l'automobiliste ferait mieux de passer par l'autoroute Paris-Toulouse, plutôt que de faire un détour par Bordeaux, où il perdrait une heure !

Existence un plus court chemin : Soit R=(X,U,d) un réseau, on dit que R admet un plus court chemin, si et seulement si, R n'admet pas de circuit absorbant.

Exemple :

Dans le graphe ci-dessous, le plus court chemin n'existe pas, car ce graphe admet un circuit absorbant (3,4,5,6,3). La somme des pondérations de ce circuit est égale à (-3), à chaque fois quand fait un passage sur ce circuit absorbant la valeur du plus court chemin diminue jusqu'a -∞.



Propriétés des plus courts chemins

Propriété1 : Tout sous-chemin d'un plus court chemin est un plus court chemin.

Propriété2 : S'il existe un plus court chemin entre deux sommets x et y, alors il existe un plus court chemin élémentaire entre x et y.

Il existe plusieurs algorithmes de recherche de plus cours chemin, on peut citer à titre d'exemple : l'algorithme de Bellman, l'algorithme de Disjktra, l'algorithme de Bellman-Ford…etc.

Algorithmes de recherche de plus court chemin :

1/ Algorithme de Bellman :

L'algorithme de Bellman s'applique sur les graphes (réseaux) sans circuit, avec des pondérations quelconques. Ainsi nous devons assurer l'inexistence de circuit dans le graphe en question, pour cela nous décrivons dans ce qui suit un algorithme simple permettant de vérifier si un graphe G est sans circuit ou non. L'algorithme s'appel algorithme de mise à niveau.

1.1. Algorithme de mise à niveau :

Le principe de cette algorithme est simple, il s'agit de classer les sommets du graphe G par niveau ou par rang, si on arrive à le faire alors le graphe est sans circuit, sinon on sera bloquer au niveau du circuit. L'algorithme commence par chercher tous les sommets x de G ayant les $dg^-(x)=0$ (càd, les sommets sans successeurs) et les mettres dans le même niveau k, ensuite il faut supprimer tous les arcs sortants des sommets x du niveau k; et refaire la même chose càd, chercher les nouveaux sommets ayant les $dg^-(x)=0$, cette fois-ci on va les mettre au niveau $k+1$ et ainsi de suite, si on arrive à classé tous les sommets de G donc le graphe est sans circuit, par contre, si on bloque quelque part dans G, cela veut dire que le graphe contient un circuit.

Algorithme mise à niveau

1- Soit un graphe quelconque G=(X,U), et k=1. $V=X$.
2- Choisir tous les sommets x n'ayant aucun prédécesseurs (c à d $dg^-(x)=0)$.

Si ses sommets existent alors les mettre

dans le niveau k, $N(k)=N(k)\cup\{x\}$ et $V=V-\{x\}$.

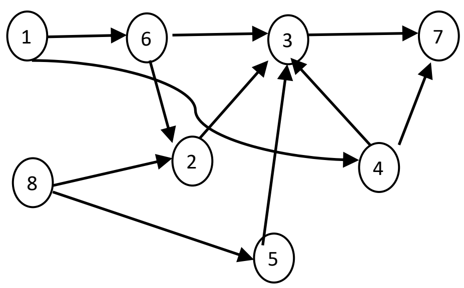
Si non Si V=∅ alors le graphe est sans circuit

Si non le graphe contient un circuit.

3- Supprimer tous les arcs sortants de tous les sommets du niveau k. Mettre $k=k+1$ et aller à (2).

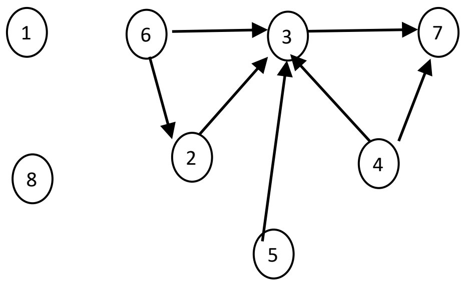
Exemple1:

Soit le graphe suivant, contient il un circuit ?

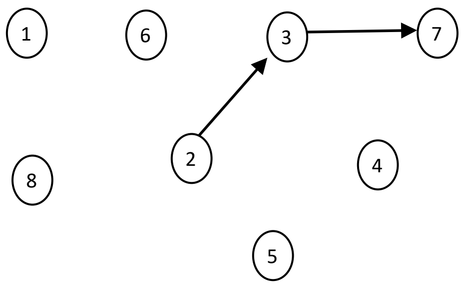


Appliquons l'algorithme, nous remarquons que les deux sommets 1 et 8 n'ont pas de prédécesseurs, car $dg^-(1)=0$ et $dg^-(8)=0$. Ainsi : N(1)={1,8}.

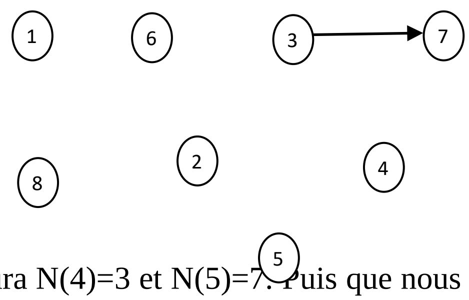
On va supprimer tous les arcs sortant de 1 et 8, ont aura ainsi le graphe suivant :



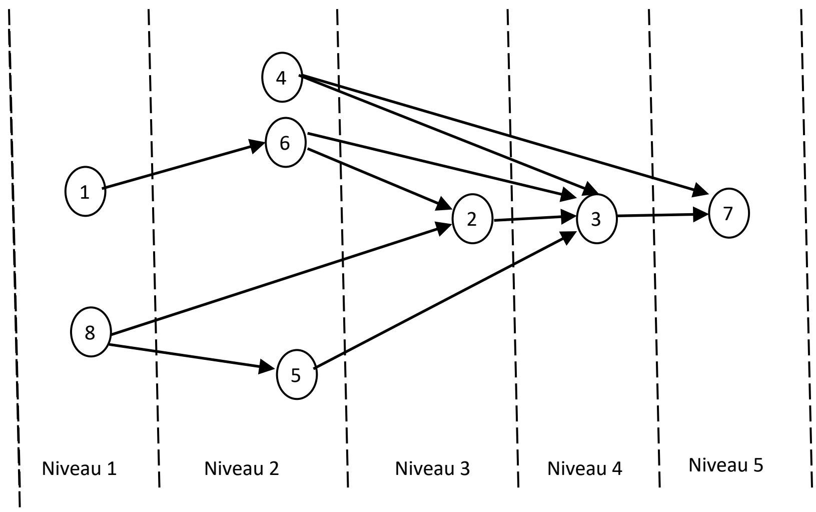
A cette étape nous constatons qu'il n'ya que les sommets 4,5 et 6 ayants $dg^-(4)=0$, $dg^-(5)=0$ et $dg^-(6)=0$. Ainsi on aura : N(2)={4,5,6}, de la même manière on va supprimer les arcs sortant de 4,5 et 6, on aura le graphe suivant :



A cette étape c'est le sommet 2 qui n'a pas de prédécesseurs, donc : N(3)={2}. De la même manière on aura le graphe suivant :

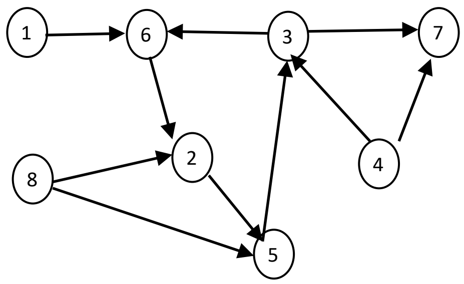


De la même manière on aura N(4)=3 et N(5)=7. Puis que nous avons réussi à classer tous les sommets de G, alors G n'admet pas de circuit. Et le graphe mis à niveau aura l'allure suivante:



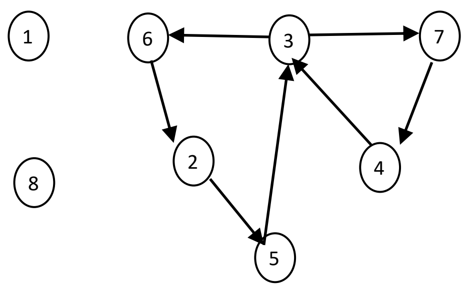
Exemple2 :

Soit le graphe suivant, admet il un circuit ?



Appliquons l'algorithme : V=X={1,2,3,4,5,6,7,8}

Dans ce graphe nous remarquons que : $dg^{-}(1)=0$ et $dg^{-}(8)=0$ ce qui veut dire que, le niveau 1 est égale à : N(1)={1,8}, d'où le nouveau V=V-{1,8}={2,3,4,5,6,7}. Ensuite d'après l'algorithme de mis à niveau, en supprimant les arcs sortant de N(1), on aura le graphe suivant:



Et le V={2,3,4,5,6,7} et non vide et il n'existe aucun sommet *x* avec les $dg^{-}(x)=0$, donc ce graphe contient un circuit.

1.2 *Algorithme de Bellman :*

L'algorithme de Bellman est un algorithme de programmation dynamique qui permet de trouver des plus courts chemins, dans un graphe G orienté pondéré (accepte même les valeurs négatives) sans circuit, depuis un sommet source donné. Contrairement à l'algorithme de Dijkstra, qui ne peut être utilisé dans un graphe avec circuit et tous les arcs ont des poids positifs ou nuls.

Définition : On appel $\pi(x)$, le plus court chemin du sommet S source du graphe G, jusqu'à x.

Le principe de cet algorithme est simple, premièrement on initialise $\pi(S)$ à 0, et les autres sommets de G à $+\infty$, ie. $\pi(x)= +\infty \quad \forall x \in X$. Ensuite on prend les sommets dans l'ordre topologique, cet ordre peut être déduit à partir de l'algorithme de mise à niveau présenté ci-dessus. Ce choix nous permettra de ne choisir que les sommets x *ayant* y_i comme prédécesseurs tel que $\pi(y_i) \neq +\infty$, ie. Que les plus cours chemins jusqu'à y_i sont déjà calculés.

Soit x un sommet pour lequel on va calculer $\pi(x)$, et soit $Y=\{ y_{i,\ i=1..p}\}$ l'ensemble des prédécesseurs direct de x. Une seule condition doit être respectée c'est que $\pi(y_i)$ sont déjà calculé $\forall i,\ i=1..p$. De la nous calculons $\pi(x)$ par la formule suivante :

$$\pi(x)=\min(\pi(y_i)+d(y_i,x)) \qquad \forall i,\ i=1..p$$

Tel que : $d(y_i,x)$ représente la pondération de l'arc (y_i,x).

Algorithme

Soit R un réseau ayant une source S et un puit P.

1. Initialiser $\pi(S)=0$ et $\pi(x)= +\infty \ \forall x \in X$. T=$\varnothing$.
2. *Choisir les sommets x ayant seulement S* pour

   prédécesseurs, et calculer $\pi(x)= \pi(S)+d(S,x)$. $T=T\cup\{x\}$.
3. Tantque (X≠T)

   Faire

   Choisir un sommet Z de X ayant pour W_i prédécesseurs,

   tel que $\pi(W_i)$ sont déjà calculé.

   $\pi(Z)=\min(\pi(W_i)+d(W_i,Z)$ *pour tous* W_i prédécesseurs de Z.

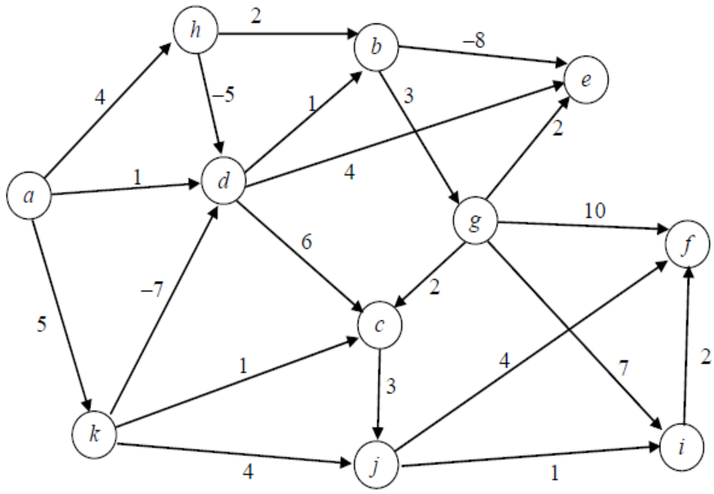
   $T=T\cup\{Z\}$.

   Fait.

Fin de l'algorithme

Exemple :

On considère le graphe orienté G = (X, U) ci-dessous pondéré par des longueurs d'arcs. On cherche à déterminer les plus courts chemins de a à tout autre sommet.



On peut appliquer l'algorithme de Bellman parce que le graphe est sans circuit, ce qu'on peut vérifier en effectuant la numérotation topologique (la mise à niveau du graphe) qui a été fait ci-dessous. Les numéros topologiques sont encadrés.

Après avoir posé distance(a) = 0, on calcule les distances par numéros topologiques croissants en appliquant la formule :

distance(x) = MIN prédécesseurs de x (distance(y)+ longueur(y,x)) .

Les distances sont indiquées en gras à côté des sommets et les plus courts chemins sont indiqués par les arcs en gras.