Théorie des Graphes

Chapitre 2 : Arbres couvrants de poids minimum

Imane EL MALKI et Pr. Imad HAFIDI
imane.elmalki@gmail.com

Master Big Data et Aide à la décision

Problème d'arbre couvrant minimale

Problématique

Lors de la phase de conception d'un circuit électrique, on a souvent besoin de relier entre elles les broches de composants électriquement équivalents.
Pour interconnecter un ensemble de n broches, on peut utiliser un arrangement de $n-1$ câbles, chacun reliant deux broches.
On peut modéliser ce problème de câblage à l'aide d'un graphe non-orienté connexe $G=(S,A)$. S représente l'ensemble des broches,et A l'ensemble des interconnections possible entree paires de broches, et pour chaque arête $(u,v)\in A$, on a un poids $w(u,v)$ qui spécifie le coût (la longueur du câble nécessaire) pour connecter u et v.

Problème d'arbre couvrant minimale

Problématique

Lors de la phase de conception d'un circuit électrique, on a souvent besoin de relier entre elles les broches de composants électriquement équivalents.
Pour interconnecter un ensemble de n broches, on peut utiliser un arrangement de $n-1$ câbles, chacun reliant deux broches.
On peut modéliser ce problème de câblage à l'aide d'un graphe non-orienté connexe $G=(S,A)$. S représente l'ensemble des broches,et A l'ensemble des interconnections possible entree paires de broches, et pour chaque arête $(u,v)\in A$, on a un poids $w(u,v)$ qui spécifie le coût (la longueur du câble nécessaire) pour connecter u et v.

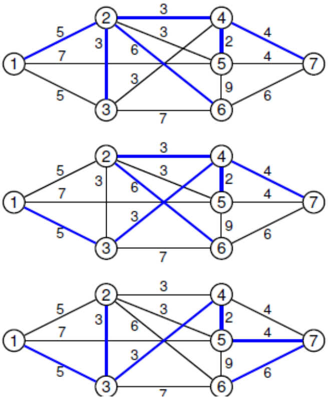
On veut alors trouver un graphe partiel acyclique et connexe $G^{'}=(S,T)$ dont le poids total :

$$w(T)=\sum_{(u,v)\in T} w(u,v)$$

soit minimum.

Non unicité de l'arbre couvrant minimale

Exemple :



Dans cette partie nous examinons deux algorithmes permettant de résoudre le problème de l'arbre couvrant :

Dans cette partie nous examinons deux algorithmes permettant de résoudre le problème de l'arbre couvrant :

1- **Algorithme de Kruskal**
2- **Algorithme de Prim**

Dans cette partie nous examinons deux algorithmes permettant de résoudre le problème de l'arbre couvrant :

1- **Algorithme de Kruskal**
2- **Algorithme de Prim**

Les deux algorithmes sont des algorithmes **Gloutons**. À Chaque étape de l'algorithme, une option parmi plusieurs possibles doit être choisie.

Dans cette partie nous examinons deux algorithmes permettant de résoudre le problème de l'arbre couvrant :

1- **Algorithme de Kruskal**
2- **Algorithme de Prim**

Les deux algorithmes sont des algorithmes **Gloutons**. À Chaque étape de l'algorithme, une option parmi plusieurs possibles doit être choisie.
La stratégie Gloutonne effectue le choix qui semble le meilleur à l'instant donné. Une telle stratégie n'aboutit pas forcément à des solutions globalement optimales.

Historique

L'algorithme de **Kruskal** doit son nom à Joseph Kruskal un mathématicien américian qui a développé cette méthode en 1956.

Historique

L'algorithme de **Kruskal** doit son nom à Joseph Kruskal un mathématicien américian qui a développé cette méthode en 1956.

Cet algorithme est resté l'un des plus simples et des plus rapides à mettre en oeuvre, ce qui explique sa popularité persistante dans divers domaines, notamment l'informatique théorique, les réseaux de télécommunication ...

Historique

L'algorithme de **Kruskal** doit son nom à Joseph Kruskal un mathématicien américian qui a développé cette méthode en 1956.
Cet algorithme est resté l'un des plus simples et des plus rapides à mettre en oeuvre, ce qui explique sa popularité persistante dans divers domaines, notamment l'informatique théorique, les réseaux de télécommunication ...

L'algorithme de Kruskal est un algorithme glouton qui repose sur le fait qu'un arbre couvrant d'un graphe d'ordre n est :

- un graphe a n-1 arêtes
- un graphe acyclique

Le choix des arêtes se fera donc sur deux critères : la conservation de l'acyclicité et un coût minimal.

Kruskal

Les étapes de l’algorithme :

Kruskal

Les étapes de l'algorithme :

étape 1: Tri des arêtes selon leur poids

Kruskal

Les étapes de l’algorithme :

étape 1: Tri des arêtes selon leur poids

étape 2: Ajout des arêtes à l’arbre couvrant.

Kruskal

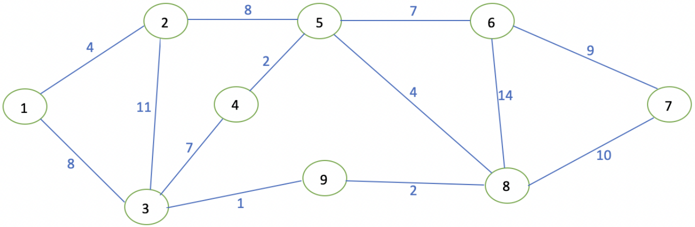
Les étapes de l'algorithme :

étape 1: Tri des arêtes selon leur poids

étape 2: Ajout des arêtes à l'arbre couvrant.

étape 3: Obtention d'un arbre couvrant

Exemple :



Kruskal

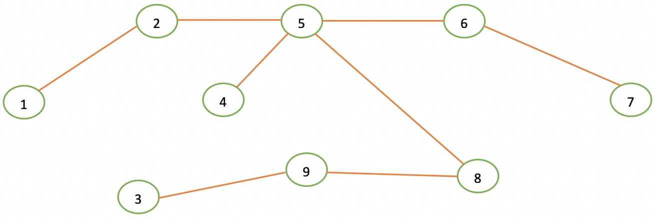
Solution :

- **(3,9) : 1**
- **(4,5) : 2**
- **(8,9) : 2**
- **(1,2) : 4**
- **(5,8) : 4**
- **(3,4) : 7**
- **(5,6) : 7**
- **(2,5) : 8**
- **(1,3) : 8**
- **(6,7) : 9**
- **(7,8) : 10**
- **(2,3) : 11**
- **(6,8) : 14**

Kruskal

Solution :

- (3,9) : 1
- (4,5) : 2
- (8,9) : 2
- (1,2) : 4
- (5,8) : 4
- (3,4) : 7
- (5,6) : 7
- (2,5) : 8
- (1,3) : 8
- (6,7) : 9
- (7,8) : 10
- (2,3) : 11
- (6,8) : 14



Kruskal

L'algorithme de Kruskal

ProcédureKruskal (G(S,A))
$E \longleftarrow \emptyset$
$F \longleftarrow A$
Tant que $Card(E) < Card(S) - 1$ faire
    Trouver $e \in F$ de poids minimal
    $F \longleftarrow F - e$
    $E \cup e$ est acyclique alors
        $E \longleftarrow E \cup e$
    Fin si
Fin Tant que

Algorithme de Prim

Historique :

L'algorithme à été développé en 1930 par le mathématicien tchèque Vojtech Jarnik, puis a été redécouvert et republié par Robert C. Prim et Edsger W. Dijkstra en 1959. Ainsi, il est parfois appelé **DJP algorithm**, **Jarnik's algorithm**, **Prim-Jarnik algorithm**, ou **Prim-Dijkstra algorithm**.

Algorithme de Prim

Historique :

L'algorithme à été développé en 1930 par le mathématicien tchèque Vojtech Jarnik, puis a été redécouvert et republié par Robert C. Prim et Edsger W. Dijkstra en 1959. Ainsi, il est parfois appelé **DJP algorithm**, **Jarnik's algorithm**, **Prim-Jarnik algorithm**, ou **Prim-Dijkstra algorithm**.

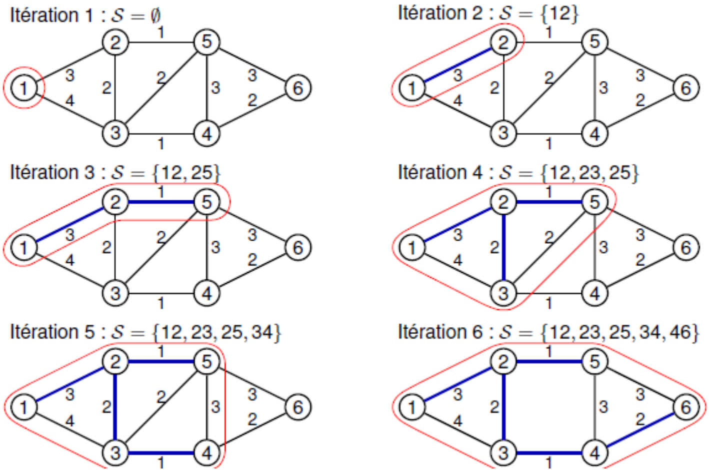
Principe :

- Choisir un sommet de départ formant l'arbre courant
- Ajouter itérativement un sommet à l'arbre en sélectionnant l'arête de plus petit coût incidente à l'arbre courant
- Arrêter l'énumération lorsque l'arbre courant est couvrant

Algorithme de Prim

Exemple :

En partant du sommet 1



Algorithme de Prim

Algorithme :

ProcédurePrim (G(S,A))
Choisir un sommet x de S
$T \longleftarrow x$ (Un ensemble des sommets)
$F \longleftarrow \emptyset$ (un ensemble d'arrêts)
Tant que $T \neq S$ faire
    Trouver $(ys) \in A$ de poids minimal tel que $y \in S - T$ et $s \in T$
    $F \longleftarrow F \cup (ys)$
    $T \longleftarrow T \cup y$
Fin Tant que

Problème du plus court chemin

Principe :

Trouver un chemin de longueur minimale entre deux sommets de graphe.

Problème du plus court chemin

Principe :

Trouver un chemin de longueur minimale entre deux sommets de graphe.

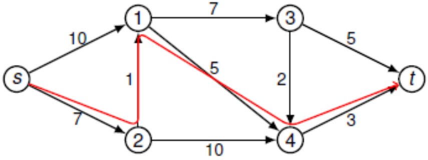
Plus court chemin

Un chemin P entre s et t est un plus court chemin si :

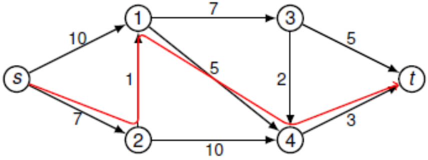
$$\sum_{(i,j)\in P} c(i,j) \leq \sum_{(i,j)\in P'} c(i,j)$$

Pour tout chemin P' entre s et t où $c(i,j)$ est le poids de l'arc entre les deux sommets i et j.

Problème du plus court chemin



Problème du plus court chemin



Intérêt

Applications dans l'industrie : chercher la route la moins coûteuse

- dans un réseau routier (GPS)
- dans un réseau de communication (transfert d'information)

Problème du plus court chemin

Problématique

soit $G = (S, A)$ un graphe pondéré orienté connexe. Nous voulons résoudre les problèmes suivants :

- Étant donné un sommet x de S fixé, on désire connaître le plus court chemin vers un sommet y.
- Étant donné un sommet x de S fixé, on désire connaître le plus court chemin vers tous le sommets de S.

Les deux problèmes sont résolus de la même façon.

Problème du plus court chemin

Problématique

soit $G = (S, A)$ un graphe pondéré orienté connexe. Nous voulons résoudre les problèmes suivants :

- Étant donné un sommet x de S fixé, on désire connaître le plus court chemin vers un sommet y.
- Étant donné un sommet x de S fixé, on désire connaître le plus court chemin vers tous le sommets de S.

Les deux problèmes sont résolus de la même façon.

Algorithmes

Il existe trois algorithmes pour résoudre le problème du plus court chemin en partant d'une source unique :

- L'algorithme de Bellman pour les graphes acycliques.
- L'algorithme de Dijkstra pour les graphes dont les poids sont positifs.
- L'algorithme de Bellman-Ford pour le cas général.

Problème du plus court chemin

Définition chemin plus court

$$w^*(x,y) = \min_{CheminsC(x,y)} w(C(x,y))$$

la valeur minimale d'un chemin de x à y.
S'il n'existe pas de chemin de x à y dans G, on pose $w^*(x,y) = +\infty$.
Notons enfin $C^*(x,y)$ un chemin tel que:

$$w^*(x,y) = w(C^*(x,y))$$

c'est à dire un chemin de valeur minimale parmi tous les chemins de x à y dans G.

Problème du plus court chemin

Représentation des plus courts chemins

Lorsque l'on cherche un plus court chemin partant d'une source x, on veut non seulement connaître la valeur de ce chemin, mais aussi les sommets présents sur ce chemin.

- Étant donné un graphe G = (S,A), on va maintenir à jour une liste de prédécesseur, notée π et une liste de valeur λ.
- Pour chaque sommet s
  - $\pi(s)$ représentera le prédécesseur de s.
  - $\lambda(s)$ représentera la valeur courante du chemin pour se rendre de la source à s.
- On supposera aussi que l'on dispose d'une méthode pour marquer les sommets.

Problème du plus court chemin

Initialisation des algorithmes

Pour tous les algorithmes, on utilise la méthode d'initialisation suivante :
Initialisation(G, s)
Pour i = 1 à n faire
$\lambda(i) \longleftarrow +\infty$
$\pi(i) \longleftarrow null$
fin pour
marquer s
$\lambda(s) \longleftarrow 0$

Problème du plus court chemin : Relâchement

Les algorithmes que nous allons présenter utilisent tous la technique dite du relâchement (Principe d'amélioration locale).

Définition :

Le processus de **relâchement** d'un arc (u, v) consiste à tester si l'on peut améliorer le plus court chemin vers v trouvé jusqu'ici en passant par u, si tel est le cas, en actualisant $\lambda(v)$ et $\pi(v)$. Une étape de relâchement peut diminuer la valeur de l'estimation de plus court chemin $\lambda(v)$ et mettre à jour le champs prédécesseur $\pi(v)$ de v.

Problème du plus court chemin : Relâchement

Les algorithmes que nous allons présenter utilisent tous la technique dite du relâchement (Principe d'amélioration locale).

Définition :

Le processus de **relâchement** d'un arc (u, v) consiste à tester si l'on peut améliorer le plus court chemin vers v trouvé jusqu'ici en passant par u, si tel est le cas, en actualisant $\lambda(v)$ et $\pi(v)$. Une étape de relâchement peut diminuer la valeur de l'estimation de plus court chemin $\lambda(v)$ et mettre à jour le champs prédécesseur $\pi(v)$ de v.

Algorithme de Relâchement

Relâcher(G, (u, v))
Si $\lambda(v) > \lambda(u) + w(u, v)$ alors
  $\lambda(v) \longleftarrow \lambda(u) + w(u, v)$
  $\pi(v) \longleftarrow u$
Fin si

Problème du plus court chemin : Tri topologique

Définition

Le tri topologique d'un graphe orienté sans circuit $G = (S, A)$ consiste à ordonner linéairement tous ses sommets de sorte que, si G contient un arc (u, v), u apparaisse avant v dans le tri.

Problème du plus court chemin : Tri topologique

Définition

Le tri topologique d'un graphe orienté sans circuit $G = (S, A)$ consiste à ordonner linéairement tous ses sommets de sorte que, si G contient un arc (u, v), u apparaisse avant v dans le tri.

Propriété

Un graphe orienté est sans circuit ssi il admet un tri topologique.

Problème du plus court chemin : Tri topologique

Définition

Le tri topologique d'un graphe orienté sans circuit $G = (S, A)$ consiste à ordonner linéairement tous ses sommets de sorte que, si G contient un arc (u, v), u apparaisse avant v dans le tri.

Propriété

Un graphe orienté est sans circuit ssi il admet un tri topologique.

Remarque

un graphe orienté sans circuit peut avoir plusieurs tris topologiques.

Exemple tri topologique

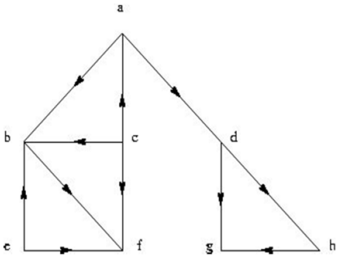


FIG – Un graphe sans circuit

- e, c, a, b, d, h, g, f
- c, e, a, b, f, d, h, g
- e, c, a, b, f, d, h, g

Algorithme de Bellman (Graphes acycliques)

Étapes de l'algorithme:

- **Étape 1 :** Tri topologique

Algorithme de Bellman (Graphes acycliques)

Étapes de l'algorithme:

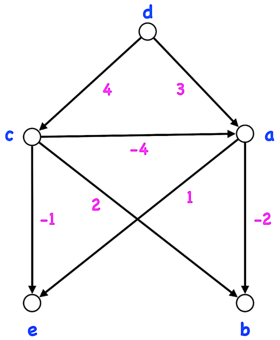
- **Étape 1 :** Tri topologique
- **Étape 2 :** Initialisation, choisir un sommet de départ r et coller des étiquettes 0 pour r, et ∞ pour les autres sommets.

Algorithme de Bellman (Graphes acycliques)

Étapes de l'algorithme:

- **Étape 1 :** Tri topologique
- **Étape 2 :** Initialisation, choisir un sommet de départ r et coller des étiquettes 0 pour r, et ∞ pour les autres sommets.
- **Étape 3 :** Relâcher tous les arcs sortant de r puis on relâche tous les arc sortant des autre sommets en suivant le tri topologique de la première étape.

Exemple :

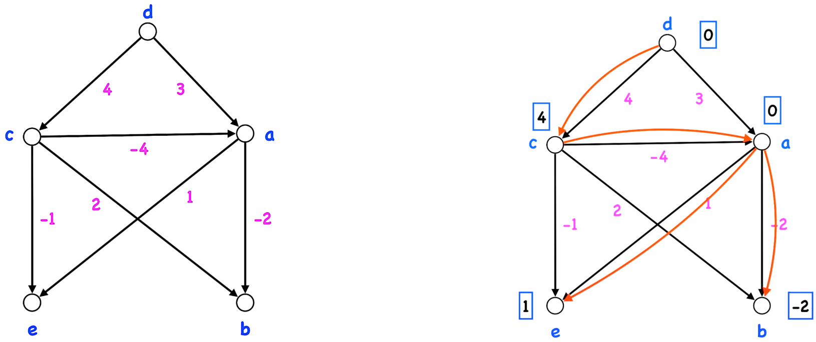


Algorithme de Bellman (Graphes acycliques)

Étapes de l'algorithme:

- **Étape 1 :** Tri topologique
- **Étape 2 :** Initialisation, choisir un sommet de départ r et coller des étiquettes 0 pour r, et ∞ pour les autres sommets.
- **Étape 3 :** Relâcher tous les arcs sortant de r puis on relâche tous les arc sortant des autre sommets en suivant le tri topologique de la première étape.

Exemple :



Algorithme de Bellman (Graphes acycliques)

Algorithme de Bellman

Bellman(G, s)
Triez topologiquement les sommets de G
Initialisation(G, s)
Pour $i = s$ à n faire
    Pour tout $j \in \Gamma^{+}(i)$ faire
        Relâcher(G, (i , j))
    Fin Pour
Fin Pour

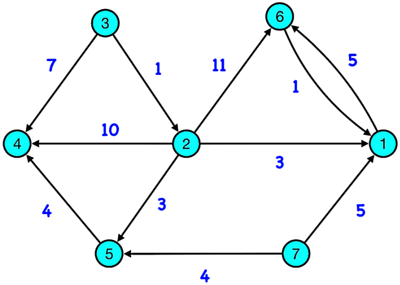
Algorithme de Dijkstra : Poids positifs

L'algorithme de Dijkstra peut être divisé en quatre étapes :

- **Étape 1 : Initialisation :** On attribue une distance infinie à tous les nœuds, sauf au nœud de départ prend 0.
- **Étape 2 : Sélection du nœud courant :** On sélectionne le nœud non visité avec la distance la plus faible
- **Étape 3 : Relâchement :** On calcule la distance du nœud courant à chacun de ses voisins non visités. Si la distance calculée est inférieure à la distance actuelle, alors la distance du voisin est mise à jour.
- **Étape 4 : Marquage du nœud courant comme visité :** On marque le nœud courant comme visité.

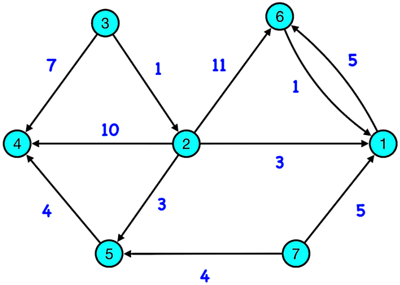
Algorithme de Dijkstra : Poids positifs

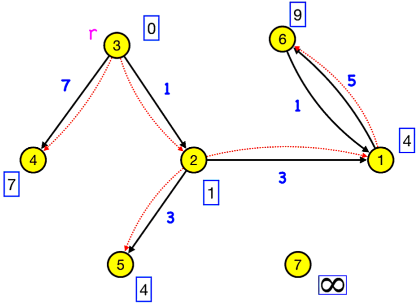
Exemple :



Algorithme de Dijkstra : Poids positifs

Exemple :





Algorithme de Dijkstra : Poids positifs

Algorithme de Dijkstra

Dijkstra(G, s)
Initialisation(G, s)
$E \longleftarrow \emptyset, F \longleftarrow S$[E et F sont des ensembles de sommets]
Tant que $E \neq S$ alors $\quad i \longleftarrow Minimum(F)$
    Pour tout $j \in \Gamma^{+}(i)$ faire
        si j n'est pas marqué alors
            Relâcher(G, (i , j))
            $E \longleftarrow E \cup i$
        Fin si
    Fin pour
Fin Tant que

Algorithme Bellman-Ford : Cas général

Introduction :

L'algorithme de Bellman-Ford est aussi un algorithme qui permet de trouver la plus petite distance possible d'un sommet vers tout autre sommet. Son application est similaire à l'algorithme de Dijkstra.

Algorithme Bellman-Ford : Cas général

Introduction :
L'algorithme de Bellman-Ford est aussi un algorithme qui permet de trouver la plus petite distance possible d'un sommet vers tout autre sommet. Son application est similaire à l'algorithme de Dijkstra.

Difference avec Dijkstra :

- L'algorithme de Bellman-Ford est plus lent par rapport à l'algorithme de Dijkstra.

Algorithme Bellman-Ford : Cas général

Introduction :

L'algorithme de Bellman-Ford est aussi un algorithme qui permet de trouver la plus petite distance possible d'un sommet vers tout autre sommet. Son application est similaire à l'algorithme de Dijkstra.

Difference avec Dijkstra :

- L'algorithme de Bellman-Ford est plus lent par rapport à l'algorithme de Dijkstra.
- Bellman-Ford peut trouver les distances les plus petites même avec les graphes ayant un poids négatif.

Algorithme Bellman-Ford : Cas général

Introduction :
L'algorithme de Bellman-Ford est aussi un algorithme qui permet de trouver la plus petite distance possible d'un sommet vers tout autre sommet. Son application est similaire à l'algorithme de Dijkstra.

Difference avec Dijkstra :

- L'algorithme de Bellman-Ford est plus lent par rapport à l'algorithme de Dijkstra.
- Bellman-Ford peut trouver les distances les plus petites même avec les graphes ayant un poids négatif.
- Bellman-Ford a aussi la possibilité de detecter les graphes avec un cycle négatif

Algorithme Bellman-Ford : Cas général

Introduction :
L'algorithme de Bellman-Ford est aussi un algorithme qui permet de trouver la plus petite distance possible d'un sommet vers tout autre sommet. Son application est similaire à l'algorithme de Dijkstra.

Difference avec Dijkstra :

- L'algorithme de Bellman-Ford est plus lent par rapport à l'algorithme de Dijkstra.
- Bellman-Ford peut trouver les distances les plus petites même avec les graphes ayant un poids négatif.
- Bellman-Ford a aussi la possibilité de detecter les graphes avec un cycle négatif

N.B : Les graphes avec un cycle négatid n'a pas de solution.

Algorithme Bellman-Ford : Cas général

Étapes de l'algorithme:

Soit $G = (S, A)$ un graphe avec n sommets. L'algorithme de Bellman-Ford suit les étapes suivantes :

- **Initialisation :** On commence par notre source, et on lui assigne la distance 0. Les autres sommets ont une distance ∞.

Algorithme Bellman-Ford : Cas général

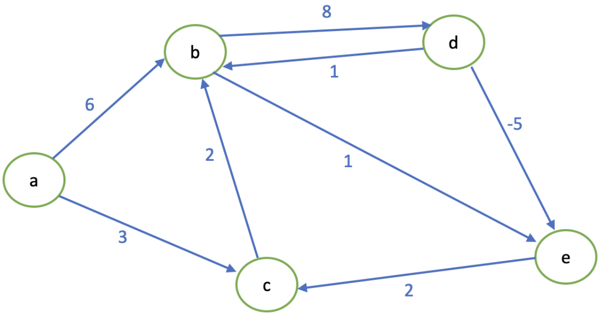
Étapes de l'algorithme:

Soit $G = (S, A)$ un graphe avec n sommets. L'algorithme de Bellman-Ford suit les étapes suivantes :

- **Initialisation :** On commence par notre source, et on lui assigne la distance 0. Les autres sommets ont une distance ∞.
- On fait $n - 1$ iterations. Pour chaque iteration, on effectue les actions suivantes :
  - Relâcher tous les arcs de G
- Vérifier l'existance d'un circuit négatif.

Algorithme Bellman-Ford : Cas général

Exemple :



Algorithme Bellman-Ford : Cas général

Algorithme de Bellman-Ford

BellmanFord(G,s)
Initialisation(G,s)
Pour i=1:n-1
    Pour tout $v \in S$
        pour tout $u \in \Gamma^{-}(v)$
        Relâcher(G(u,v))
Fin pour
Vérifier qu'il n'ya pas de circuit négatif

La planification

La planification est une activité courante de la vie moderne :

- Organiser un voyage : banque, agent de voyage, préparation des valises, réservation du taxi.
- Soirée : Inviter ses amis, faire des courses, nettoyer la maison, préparation des repas...

La planification s'appuie sur des procédés plus élaborés qui révèlent du talent organisationnelle ou des techniques de la recherche opérationnelle : le circuit répétitif qu'emprunte les éboueurs d'un quartier en a fourni un exemple

Étapes de planification

Pour une planification plus optimale, il faut :

- préciser l'objectif ;
- déterminer les opérations où les taches nécessaires pour atteindre cet objectif ;
- estimer la durée de chaque tache et les ressources exigées par chacune ;
- calculer la durée totale et le coût de totale du projet ;
- dresser un calendrier d'échelonnement des taches (ordonnancement des taches).

Exemples

La planification est dans tous les domaines :

- Lancement d'un nouveau produit,
- installation d'un nouvel équipement,
- mise en place d'une opération de communication

Méthodes d'ordonnancement

Il existe parmi les divers méthodes proposées aux planificateurs modernes, deux prétendues principales au titre de méthode reine :

- PERT (Program Evolution and Revue Technique): a pris son départ en 1958 au bureau des programmes spéciaux de la marine américaine, pour être en mise en ?uvre la première fois dans la gestion du gigantesque du projet que constituaient la conception, la fabrication et le lancement de la fusée POLARIS (coordination de l'action de près de 6000 constructeurs pour la réalisation de missiles à ogives nucléaires POLARIS) ancêtre du programme spatiale américain.
- MPM (Méthode des Potentiels Metra) : mise au point en France par Bernard Roy et son équipe utilisée en particulier pour l'aménagement des superstructures du paquebot France.

Exemple

Pour organiser la construction d'une maison on devra franchir les étapes suivantes :
Étape 1 :
Creusement du trou, Fondations, Coulage de la dalle, Montage des murs, Charpente et toiture, huisseries, montage des cloisons internes, menuiserie interne, plâtres, électricité, plomberie, ..
Étape 2 :
Préciser la durée de chaque tache.
Étape 3 :
l'ordre naturel des taches : il est déraisonnable de convoquer l'électricien après le plâtrier ou le peintre avant le plâtrier ...

Pour ne pas nous encombrer avec la signification des taches, nous allons supposer maintenant qu'il y a 7 taches à effectuer que nous appellerons A, B, C, D, E, F, G

Nous allons présenter notre problème d'ordonnancement avec le tableau suivant :

Nom du tâche	Durée de la tache	Tâches pré-requises
A	2	Aucune
B	3	A
C	5	A
D	1	A, B, C
E	5	A, C
F	6	A, D, B
G	3	B, E, F

Définition

Définition

La méthode MPM est une technique d'ordonnancement basée sur la théorie des graphes, visant à optimiser la planification des tâches d'un projet.

But

L'utilisation de la MPM permet:

- Déterminer la durée minimum du projet
- Dates de départs des différents taches pour respecter la durée minimum du projet.
- Déterminer les taches critiques (où un retard peut affecter la durée minimum du projet)

Remarque

Le recours à la méthode des potentiels Métra suppose qu'aient été identifiées préalablement les différentes tâches nécessaires à la réalisation du projet, leur durée et leurs relations d'antériorité.

Construction du graphe

Règles

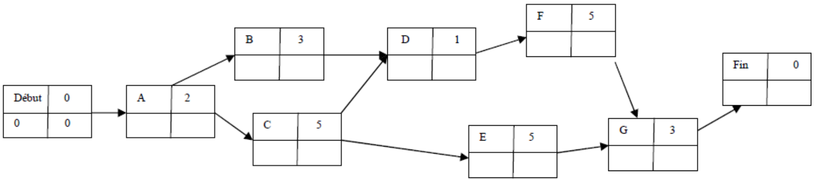
- un sommet correspond à une tâche
- un arc définit une relation d'antériorité
- on ajoute des sommets Début et Fin : D précède tout sommet et n'admettant aucun prédécesseur immédiat, tandis que F est lié comme sommet terminale à tout sommet qui ne possède aucun successeur.
- Chaque sommet de la représentation graphique est figuré par un rectangle.

Nom de la tache	Durée de la tache
Date de départ au plus tôt	Date de départ au plus tard

Graphe MPM

Exemple

Nous reprenons l'exemple cité dans le chapitre précédent.



Ce graphe permet de visualiser les relations entre les différentes taches à réaliser pour mener ce projet. Son tracé nous amène à examiner les relations d'antériorité et de postériorité entre les taches.

Moment plus tôt

Définition

Une tâche s ne pouvant débuter que lorsque toutes les tâches qui y aboutissent sont terminées, le moment plus tôt où se terminera l'ensemble des tâches aboutissant au sommet s. Ce moment au plus tôt sera noté E(s).

Elle s'obtient très simplement :

$$E(s) = Max_t(E(t) + \text{Durée tâche}(t))$$

t représente l'ensemble des tâches immédiatement antérieures à s.

La détermination des dates au plus tôt des différents sommets se fait donc par calculs successifs, à partir du sommet "Début" (dont, par convention, la date au plus tôt est fixée à 0).

$$
\begin{aligned}
E(A) &= 0 \quad E(B) = E(A) + 2 = 2 \\
E(C) &= E(A) + 2 = 2 \\
E(D) &= Max(E(A) + 2, E(B) + 3, E(C) + 5) = 7 \\
E(E) &= Max(E(A) + 2, E(C) + 5) = 7 \\
E(F) &= Max(E(D) + 1, E(B) + 3) = 8 \\
E(G) &= Max(E(D) + 1, E(E) + 5, E(F) + 6) = 14 \\
E(Fin) &= E(G) + 3 = 17
\end{aligned}
$$

Remarque

La durée minimale du projet est le moment plus tôt de la tache fin.
Le projet ne peu donc se terminer en moins de 17 jours. C'est la durée minimale du projet.

Moment plus tard

Définition

Le moment au plus tard d'un réseau MPM correspond à la date à laquelle une tâche doit être exécutée au plus tard pour ne pas remettre en cause la durée optimale totale du projet. Le moment au plus tard dénoté L(s).

Ceci sera obtenu en commençant par les sommets de niveau les plus élevés jusqu'aux sommets de niveau les plus faibles :

$$L(s) = Min_t(L(t) - \text{Durée tâche}(s))$$

t représente l'ensemble des tâches successeurs à s.

Les moments au plus tard se calculent successivement en ordre inverse que l'ordre de calcule des moments au plus tôt, à partir du sommet "Fin" (dont, par convention, la date au plus tard est fixée à la durée minimum du projet).

$$E(G) = E(Fin) - 3 = 13 \qquad E(F) = E(G) - 5 = 8$$

$$E(E) = E(G) - 5 = 8 \qquad E(D) = Min(E(F) - 1, E(G) - 1) = 7$$

$$E(C) = Min(E(D) - 5, E(E) - 5) = 2 \qquad E(B) = Min(E(D) - 3, E(F) - 3) = 2$$

$$E(A) = Min(E(C) - 2, E(B) - 2) = 0 \qquad E(\text{Début}) = E(A) - 0 = 0$$

Remarque

La date plus tard de la tache Début doit être égale au zéro.

Chemins, Taches critiques

Tache critique

Une tache critique est une tache dont la date plus tôt est égale au date plus tard.
Dans notre exemple :

Chemin critique

On appelle chemin critique la succession des tâches pour lesquels aucun retard n'est possible sans remettre en cause la durée optimale du projet (tâches pour lesquelles date au plus tôt = date au plus tard).
Dans notre exemple :