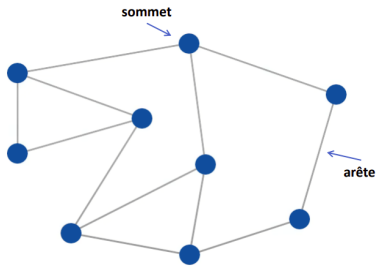
Théorie des graphes

Qu'est-ce qu'un graphe ?

Un graphe est une structure mathématique composée de :

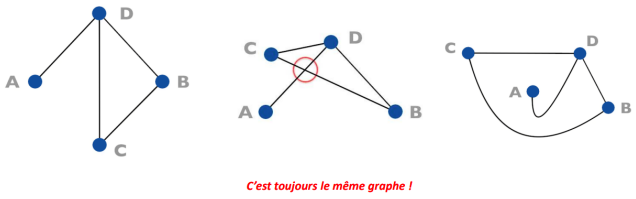
- Un ensemble de **sommets** (ou nœuds, ou points)
- Un ensemble d'**arêtes** (ou arcs, ou liens) reliant ces sommets

Définition formelle : Un graphe G est défini par un couple (S, A) où S est un ensemble fini de sommets et A un ensemble de paires de sommets (arêtes). Chaque arête est une paire (u, v) où u et v sont des sommets de S.
Remarque : Il peut y avoir zéro ou une arête entre chaque paire de sommets. Les arêtes peuvent être orientées ou non, et peuvent être pondérées (associées à un poids, par exemple une distance ou un coût).



Comment dessiner un graphe

Un graphe peut être dessiné de multiples façons sans changer sa structure. Par exemple, pour les sommets A, B, C, D et les arêtes AD, BD, BC, CD, on obtient toujours le même graphe, quelle que soit la disposition des sommets dans le plan.



À quoi sert un graphe ?

Les graphes permettent de modéliser des relations entre des éléments dans de nombreux domaines :

- **Réseaux sociaux** : Les sommets représentent des personnes, les arêtes des relations d'amitié.
- **Réseaux informatiques** : Les sommets sont des ordinateurs ou des routeurs, les arêtes des connexions.
- **Réseaux de transport** : Les sommets sont des villes ou des intersections, les arêtes des routes ou des voies.
- **Distribution d'énergie** : Les sommets sont des centrales ou des foyers, les arêtes des câbles ou des canalisations.

Les types de graphe

Graphe complet

Tous les sommets sont reliés entre eux par une arête. Il y a donc une arête entre chaque paire de sommets.

Graphe non connexe

Un graphe est non connexe s'il existe au moins deux sommets qui ne sont pas reliés par un chemin.

Graphe vide

Un graphe vide ne contient aucune arête, ce qui signifie qu'il n'y a aucune relation entre les sommets.

Notions de voisinage et degré

Le **degré** d'un sommet est le nombre d'arêtes qui lui sont incidentes. Deux sommets sont **voisins** s'ils sont reliés par une arête.

Notion de chemin

Un **chemin** est une suite de sommets reliés par des arêtes. La **longueur** d'un chemin est le nombre d'arêtes qu'il contient.

Notion de cycle

Un **cycle** est un chemin dont les deux extrémités sont reliées, formant ainsi une boucle.

Notion de connectivité

Un graphe est **connexe** si, pour toute paire de sommets U et V, il existe un chemin entre U et V.

Notion d'orientation

Un graphe est **orienté** si ses arêtes ont un sens (on parle alors d'arcs). Dans un graphe **non orienté**, les arêtes n'ont pas de sens.

Notion de pondération

Les arêtes peuvent être caractérisées par des **poids**, qui représentent par exemple :

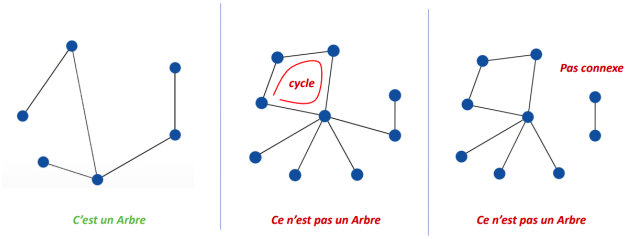
- Une distance (kilomètres entre deux villes)
- Un pourcentage (niveau d'affinité entre deux personnes)
- Un coût (prix du transport entre deux destinations)

Les arbres

Définition d'un arbre

Un **arbre** est un graphe **connexe** et **sans cycle**. Il existe plusieurs définitions équivalentes :

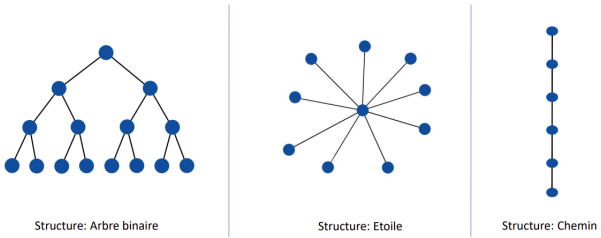
- Un arbre est un graphe connexe et sans cycle.
- Un arbre est un graphe connexe avec n sommets et $n-1$ arêtes.
- Un arbre est un graphe sans cycle avec n sommets et $n-1$ arêtes.



Types d'arbres

Il existe plusieurs types d'arbres, selon leur structure :

- Arbre binaire
- Arbre en étoile
- Arbre chemin

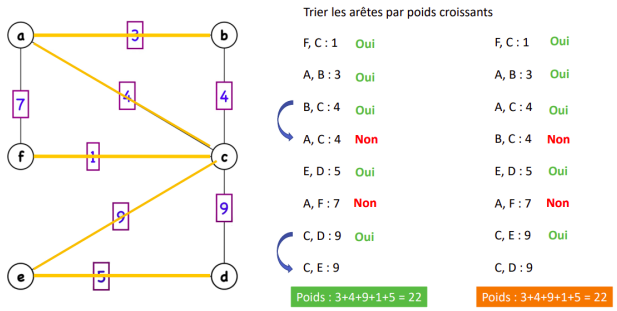


Arbres couvrants

Un **arbre couvrant** d'un graphe connexe est un arbre qui passe par tous les sommets du graphe. Il est utilisé en informatique pour éviter les boucles dans les réseaux.

Algorithme de Kruskal

L'algorithme de Kruskal permet de trouver un arbre couvrant de poids minimal dans un graphe pondéré. Il fonctionne en triant les arêtes par poids croissants et en ajoutant celles qui ne forment pas de cycle.



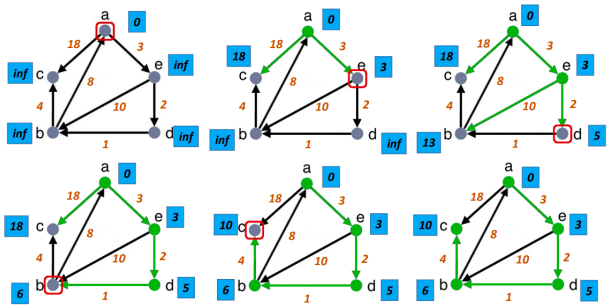
Parcours eulérien d'un graphe

Un **parcours eulérien** est un chemin qui passe par toutes les arêtes d'un graphe une et une seule fois. Un graphe connexe admet un parcours eulérien si et seulement si tous ses sommets sont de degré pair, sauf éventuellement deux.

Plus court chemin entre deux sommets

Algorithme de Dijkstra

L'algorithme de Dijkstra permet de trouver le plus court chemin entre deux sommets dans un graphe pondéré. Il est largement utilisé dans les systèmes de navigation (GPS) et les réseaux informatiques.



Introduction à l'ordonnancement

L'ordonnancement de tâches est une méthode essentielle en gestion de projet pour organiser et optimiser l'exécution des tâches dans un projet. L'objectif principal est de minimiser la durée totale du projet tout en respectant les contraintes de dépendance entre les tâches.

Objectifs

- Compresser au maximum le temps du planning.
- Identifier les tâches non compressibles.
- Identifier les tâches pouvant être parallélisées.
- Identifier les marges "au plus tôt" et "au plus tard" pour calculer le battement possible entre les tâches.

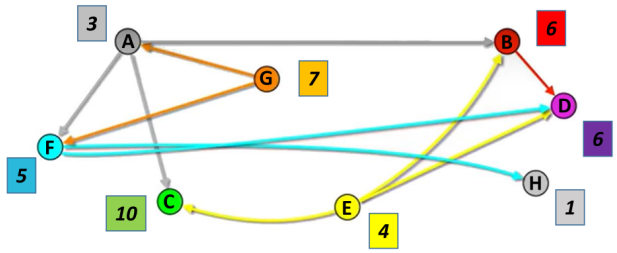
Explication des méthodes d'ordonnancement

L'ordonnancement de tâches repose sur des méthodes comme la **méthode PERT (Program Evaluation and Review Technique)** et la **méthode du chemin critique (CPM - Critical Path Method)**. Ces méthodes permettent de modéliser les tâches et leurs dépendances sous forme de graphe, où chaque tâche est représentée par un nœud et les dépendances par des arcs. Pour plus de détails sur ces méthodes, voir Wikipédia - Méthode PERT.

Définir un planning de tâche

Représentation graphique des tâches

Les tâches sont représentées par des sommets pondérés par leurs durées, et les arêtes correspondent aux dépendances entre les tâches. Voici un exemple de planning de tâches :



Niveaux des tâches

Les niveaux des tâches sont déterminés en fonction des dépendances. Une tâche de niveau 0 n'a pas de dépendance (pas d'arc entrant). Une tâche de niveau 1 dépend uniquement de tâches de niveau 0, et ainsi de suite.

Dates "au plus tôt"

Les dates "au plus tôt" indiquent le moment le plus précoce auquel une tâche peut commencer, en tenant compte des dépendances. Elles sont calculées en partant du début du projet et en ajoutant les durées des tâches précédentes.

Dates "au plus tard"

Les dates "au plus tard" indiquent le moment le plus tardif auquel une tâche peut commencer sans retarder la fin du projet. Elles sont calculées en partant de la fin du projet et en soustrayant les durées des tâches suivantes.

Marges et battements

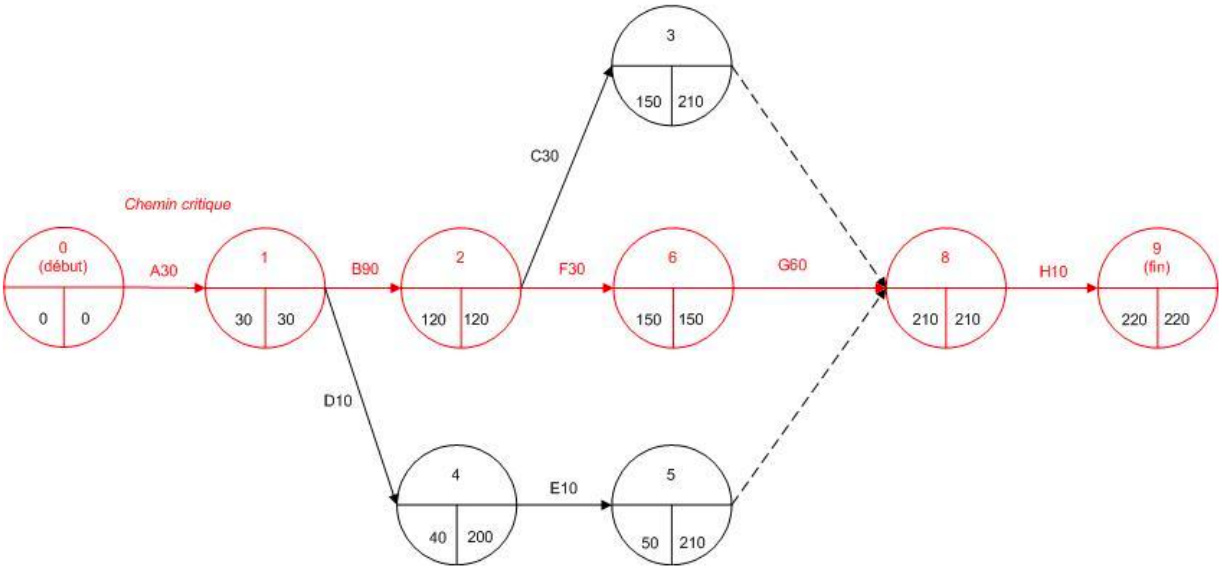
La marge d'une tâche est la différence entre sa date "au plus tard" et sa date "au plus tôt". Un battement de 0 signifie que la tâche est critique et ne peut pas être retardée sans affecter la durée totale du projet.

Chemin critique

Le chemin critique est la séquence de tâches critiques (avec un battement de 0) qui détermine la durée minimale du projet. Identifier le chemin critique permet de concentrer les efforts sur les tâches qui impactent directement la durée totale du projet.

Diagramme PERT

Le meilleur moyen de faire un exercice d'ordonnancement de tâches est de réaliser le diagramme PERT.



Voici un exemple de ce diagramme. Il contient un sommet de début et un sommet de fin. Les arêtes sont pondérées par la durée des tâches et contiennent le nom des tâches, et les sommets doivent référencer la date au plus tôt et la date au plus tard. Il est ainsi immédiat de trouver le chemin critique et les temps de battements.

Graphes - Notions de base des flots

Définition et vocabulaire

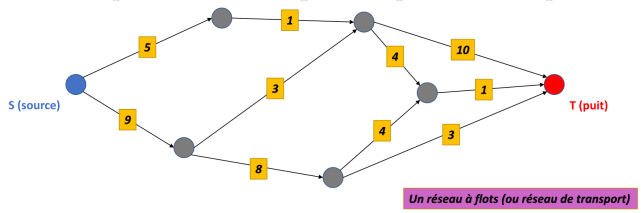
Un **réseau** est un graphe orienté où chaque arc est associé à une **capacité**, représentant la quantité maximale de flot qui peut circuler sur cet arc. Un réseau est généralement défini par :

- Un ensemble de **sommets** (ou nœuds).
- Un ensemble d'**arcs** orientés reliant ces sommets.
- Une **source** s (sommet sans arcs entrants).
- Un **puits** t (sommet sans arcs sortants).

Ajout : Un réseau est formellement défini comme un quadruplet (G, s, t, c), où G est un graphe orienté, s est la source, t est le puits, et c est une fonction de capacité associant à chaque arc une valeur réelle positive. *Source : [Cormen et al., "Introduction to Algorithms", 3rd Edition, MIT Press, 2009, p. 726].*

Un graphe orienté

Un graphe orienté est un ensemble de sommets reliés par des arcs ayant une direction. Dans le contexte des flots, chaque arc a une **capacité** qui limite la quantité de flot pouvant y circuler.



Hypothèses pour simplifier

Pour simplifier l'analyse des réseaux de flots, on suppose souvent que :

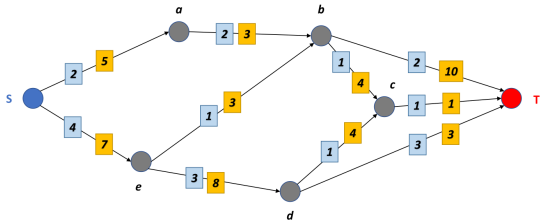
- La source s n'a pas d'arcs entrants.
- Le puits t n'a pas d'arcs sortants.

Un flot

Un **flot** est une fonction f qui associe à chaque arc (u, v) une valeur réelle $f(u, v)$ représentant la quantité de flot circulant de u vers v. Un flot doit respecter deux contraintes fondamentales :

1. **Contrainte de capacité** : $f(u, v) \leq c(u, v)$ pour tout arc (u, v).
2. **Contrainte de conservation** : Pour tout sommet u différent de s et t, la somme des flots entrants est égale à la somme des flots sortants.

Ajout : La contrainte de conservation signifie que le flot est "conservé" à travers chaque sommet intermédiaire, c'est-à-dire qu'il ne peut y avoir ni accumulation ni perte de flot dans ces sommets. *Source : [Ahuja et al., "Network Flows: Theory, Algorithms, and Applications", Prentice Hall, 1993, p. 15].*



La valeur d'un flot

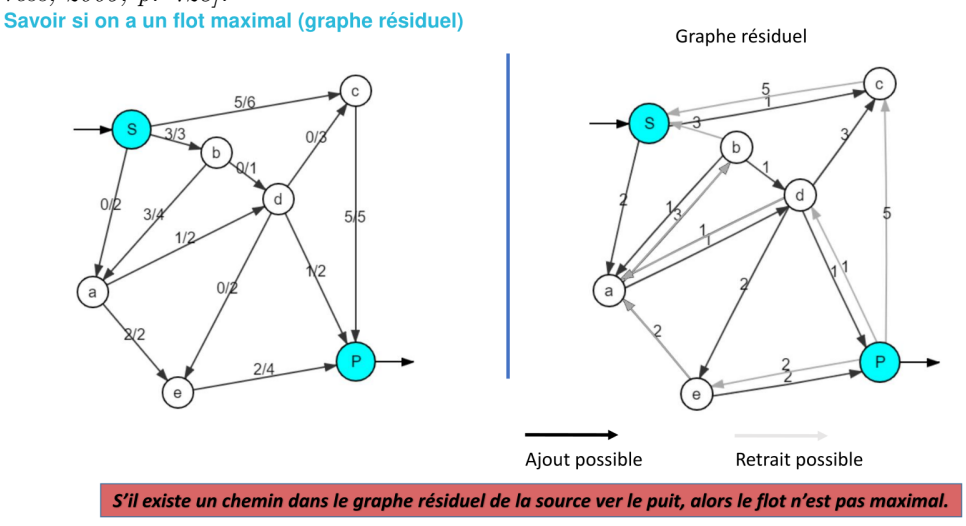
La **valeur d'un flot** est définie comme la quantité totale de flot sortant de la source s (ou entrant dans le puits t). Mathématiquement, la valeur $|f|$ d'un flot f est donnée par :

$$|f| = \sum_{v \in V} f(s, v)$$

Maximiser la valeur d'un flot

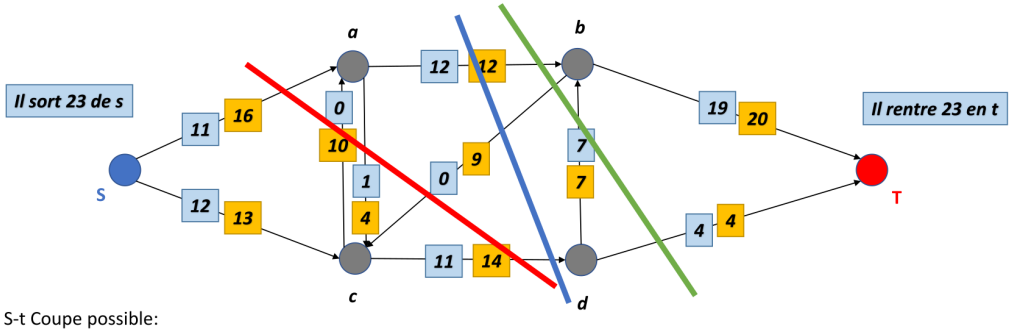
L'objectif principal dans les réseaux de flots est de trouver un **flot maximal**, c'est-à-dire un flot dont la valeur est la plus grande possible. Cet objectif est atteint en utilisant des algorithmes comme celui de **Ford-Fulkerson** ou **Edmonds-Karp**.

Ajout : L'algorithme de Ford-Fulkerson fonctionne en trouvant des **chaînes améliorantes** dans le graphe résiduel. Une chaîne améliorante est un chemin de la source au puits dans le graphe résiduel, où chaque arc a une capacité résiduelle positive. *Source : [Cormen et al., "Introduction to Algorithms", 3rd Edition, MIT Press, 2009, p. 728].*



Savoir si on a un flot maximal

Un flot est maximal si et seulement s'il n'existe plus de chaîne améliorante dans le graphe résiduel. Une autre façon de vérifier la maximalité d'un flot est de calculer la **capacité d'une st-coupe minimale**. Selon le **théorème max-flot min-coupe**, la valeur d'un flot maximal est égale à la capacité d'une st-coupe minimale. **Ajout :** Le théorème max-flot min-coupe stipule que dans un réseau de flot, la valeur maximale d'un flot de s à t est égale à la capacité minimale d'une coupe séparant s de t. *Source : [Ford et Fulkerson, "Maximal Flow Through a Network", Canadian Journal of Mathematics, 1956].*



S-t Coupe possible:

(A1,B1) = ({s,c}, {a,b,d,t}) ; Capacité C (A1,B1) = Csa + Cca + Ccd = 16 + 4 + 14 = 34

(A2,B2) = ({s,a,c}, {b,d,t}) ; Capacité C (A2,B2) = Cab + Ccd = 12 + 14 = 26

(A3,B3) = ({s,a,c,d}, {b,t}) ; Capacité C (A3,B3) = Cab + Cdb + Cdt = 12 + 7 + 4 = 23

Algorithme de Ford-Fulkerson

L'algorithme de Ford-Fulkerson est utilisé pour calculer un flot maximal dans un réseau. Voici les étapes principales :

1. Initialiser le flot à zéro pour tous les arcs.
2. Trouver une chaîne améliorante dans le graphe résiduel.
3. Augmenter le flot le long de cette chaîne par la capacité résiduelle minimale sur la chaîne.
4. Mettre à jour le graphe résiduel.
5. Répéter jusqu'à ce qu'il n'y ait plus de chaîne améliorante.

Ajout : La complexité de l'algorithme de Ford-Fulkerson dépend de la méthode utilisée pour trouver les chaînes améliorantes. Avec une recherche en largeur (algorithme d'Edmonds-Karp), la complexité est $O(VE^2)$, où V est le nombre de sommets et E le nombre d'arcs. *Source : [Cormen et al., "Introduction to Algorithms", 3rd Edition, MIT Press, 2009, p. 734].*

Choix des chaînes améliorantes

Le choix des chaînes améliorantes peut influencer l'efficacité de l'algorithme. L'algorithme d'**Edmonds-Karp** utilise toujours le plus court chemin (en nombre d'arcs) dans le graphe résiduel, ce qui garantit une complexité polynomiale.

Affectation de tâches

L'affectation de tâches est une application classique des réseaux de flots. Elle peut être modélisée comme un problème de flot maximal dans un graphe biparti, où les sommets représentent soit des tâches, soit des ressources.

Ajout : Dans un graphe biparti, les tâches sont reliées à une source et les ressources à un puits. Les capacités des arcs représentent les contraintes d'affectation. *Source : [Ahuja et al., "Network Flows: Theory, Algorithms, and Applications", Prentice Hall, 1993, p. 412].*