Conventions : u, v, v_n sont des sommets d'un graphe
$G = (V,E), G_r$ sont des graphes orientés
$R = (V,E,C)$ est un réseau d'un graphe $G(V,E)$ dans lequel chaque arête à une pondération $c(e)$.

Définitions

Un graphe **non-orienté** est formé de **sommets** et d'**arêtes**, tandis qu'un graphe **orienté** est formé de **sommets** et d'**arcs** (extrémités *initiales* et *finales*).

Un **graphe simple** est un graphe sans boucles ni arcs (ou arêtes) multiples.

Un **graphe sous-jacent** est un graphe formé en remplaçant les arcs d'un graphe orienté par des arêtes.

Degré : nombre d'arêtes d'un sommet ($\sum d = 2|E|$), ou degré entrant (intérieur) et degré sortant (extérieur) d'un sommet pour un graphe orienté ($\sum d_+ = \sum d_- = |E|$).

Un **graphe partiel** est un graphe dont on ne conserve qu'un sous-ensemble d'arêtes (ou d'arcs), mais tous les sommets.

Un **sous-graphe** est un graphe dont on ne conserve qu'un sous-ensemble des sommets ainsi que tous les arcs entre le sous-ensemble des sommets.

Sous-graphe partiel : réunion des 2 définitions précédentes.

Un graphe non-orienté comporte une **chaîne** (suite formée de sommets et d'arêtes, qui est dite **simple** si chaque arête y apparait au plus une fois, et **élémentaire** si chaque sommet y apparait au plus une fois.) La **longueur** d'une chaîne correspond au nombre d'arêtes. Un **cycle** est une chaîne fermée dont les 2 extrémités sont confondues.

Pour un graphe orienté on parle de **chemin** et de **circuit**.

Un graphe est **r-régulier** si tous ses sommets sont de degré r.

Un graphe non-orienté est **connexe** s'il existe une chaîne entre chaque paire de sommets. Une **composante connexe** d'un graphe G est un sous-ensemble maximal de sommets étant tous connexe entre eux.

Un graphe orienté est **fortement connexe** si pour chaque paire de sommets $\{u,v\}$ il existe un arc $u \to v$ et un arc $v \to u$. Une **composante fortement connexe** d'un graphe orienté G est un sous-ensemble maximal de sommets étant tous fortement connexe entre eux. Un graphe orienté est **semi-connexe** si, pour chaque paire de sommets $\{u,v\}$ il existe un arc $u \to v$ ou un arc $v \to u$. Un **graphe orienté connexe** est dit d'un graphe orienté dont le graphe sous-jacent est connexe.

Un **graphe réduit** est un graphe orienté G_r dont les sommets correspondent aux composantes fortement connexes de G et dont les arcs représentent les liens entre les composantes connexes.

Une **forêt** est un graphe sans cycle. Un graphe connexe et sans cycle est un **arbre**. Si G est un graphe non orienté connexe, on appelle **arbre recouvrant** un graphe partiel de G qui est un arbre.

Une **arborescence** de **racine** r est un arbre orienté tel qu'il existe un chemin du sommet r vers tous les autres (le degré entrant de chaque sommet, à l'exception de la racine, vaut 1). Losrqu'on inverse le sens de tous les arcs d'une arborescence, on parle alors d'**anti-arborescence** et d'**anti-racine** (le degré sortant de chaque sommet, à l'exception de la racine, vaut 1).

Représentation des graphes *(* : permet de représenter n'importe quel graphe)*

Matrice d'adjacence (*) (sommets-sommets) : l'élément a_{ij} comporte le nombre d'arêtes entre v_i et v_j ou le nombre d'arcs dont v_j et l'extrémité terminale et v_i l'extrémité initiale. *Propriétés :* matrice symétrique.
Matrice d'incidence (sommets-arcs) : l'élément a_{ik} vaut 1 si le sommet v_i est une extrémité de l'arête e_k. Cet élément vaut $-1/1$ si le sommet v_i est l'extrémité terminale/initiale de l'arc e_k *(problème de représentation des boucles)*.
Liste d'adjacence (*) : n listes contenant les sommets adjacents à v_n.
Liste de successeurs et de prédécesseurs (*) : $2 \cdot n$ listes contenant les successeurs et prédécesseurs à v_n (cas orienté).
Liste d'incidence : ne permet pas de représenter les sommets isolés.
Note : l'utilisation de matrices est privilégiée lorsque le graphe est dense, sinon l'utilisation de listes permet d'économiser la mémoire.

Exploration d'un graphe

Largeur (BFS) $O(n+m)$: stocke les voisins dans une queue, fourni une chaîne unique de longueur minimale entre u et v.
Profondeur (DFS) $O(n+m)$: récursif, avance le plus "loin" possible et numérote les sommets dans l'ordre de traitement. Peut être dérécursifié en utilisant une pile.

Composantes connexes $O(n+m)$: basé sur l'exploration en largeur, retourne pour chaque sommet son numéro de composante connexe.
Composantes fortement connexes $O(n^2 + n*m)$: parcours la liste des successeurs et note $+$ les sommets accessibles ; parcours la liste des prédécesseurs et note $-$ les sommets accessibles. Les sommets notés $+$ et $-$ forment une composante fortement connexe.
Tarjan (composantes fortement connexes) $O(n+m)$: basé sur l'algorithme *DFS*, calcule en plus pour chaque sommet le numéro $low[u]$ du sommet situé le plus "haut" au-dessus de lui dans l'arbre DFS et accessible depuis u. Si à la fin du traitement $low[u] = dfsnum[u]$ alors tous les sommets situés en dessous de u forment une composante fortement connexe.

Arbres et arborescences

Remarques :

- Un arbre ou une forêt est un graphe simple : car arêtes multiples ou boucles forment des cycles.
- Chaque composante connexe d'une forêt est un arbre.
- Un arbre est une forêt connexe.
- Tout arbre sur 2 sommets ou plus possède au moins deux sommets pendants ou *feuilles*.

Théorème : Si G est un graphe simple sur n sommets alors toutes ces

propriétés sont équivalentes :
- G est un arbre ;
- G est connexe et sans cycle ;
- G est sans cycle et comporte $n-1$ arêtes ;
- G est connexe et comporte $n-1$ arêtes ;
- Chaque paire de sommets distincts de G est reliée par une chaîne unique ;
- G est connexe et minimal pour cette propriété (tout graphe partiel $G' \subset G$ n'est pas connexe) ;
- G est sans cycle et maximal pour cette propriété (l'ajout d'une arête à G crée un – unique – cycle) ;

Propriété : Un graphe est connexe si et seulement s'il admet un arbre recouvrant.

Arbre recouvrant de poids minimum/maximum : Étant donné un réseau R non orienté, on cherche un arbre recouvrant $T = (V, E_T)$ dans R de poids total minimum/maximum. 2 algorithmes :
- **Algorithme de Kruskal** $O(m^2)$ améliorable en $O(m \cdot log(m))$: numéroter les arêtes selon leur poids du plus petit au plus grand, choisir les arêtes qui ne forment pas de cycle.
- **Algorithme de Prim** $O(n^2)$ améliorable en $O(m+n \cdot log(m))$: insérer le sommet de départ dans l'arbre, puis à chaque itération ajouter à l'arbre l'arête de plus petit poids reliant un sommet de l'arbre à un sommet n'étant pas encore dans l'arbre. Afin d'accélérer le processus on stocke pour chaque sommet son arête de plus petit poids le reliant à l'arbre, de même que l'autre extrémité de l'arête (le sommet étant déjà dans l'arbre).

Chaîne de section maximale :
- **Problème** : connaître le débit maximum que l'on pourra avoir sur une chaîne reliant n'importe quelle paire de noeuds ; le débit de la chaîne étant donné par sa *section inférieure* (poids minimum d'une de ses arêtes).
- **Théorème** : L'arbre recouvrant T_{max} de poids maximal dans R contient des chaînes de section maximale entre toute paire de sommets $\Rightarrow$ construire l'arbre de poids maximal puis prendre le poids le plus petit parmi les arêtes de l'arbre.

Plus court chemin dans les réseaux

Plus court chemin depuis une source unique

Bellman-Ford $O(n*m)$: à chaque itération passer en revue tous les arcs et si un chemin plus court passant par l'arc testé est trouvé mettre la variable continuer à vrai. Arrêt lorsque aucune modification des chemins n'est effectuée (variable continuer = faux) ou si le nombre d'itération est égal au nombre de sommets (si la variable continuer = vrai alors un circuit de longueur négative existe).

Exemple d'application :

Itération	Arc	Couple $\lambda_j, p[j]$				Continuer
départ		(0,-)	(∞, -)	(∞, -)	(∞, -)	faux
...						
n	(1,2)	-	(1,1)	(2,3)	(4,1)	vrai

Dijkstra $O(m+n*log(n))$ *(fonctionne uniquement avec des arcs de poids non-négatifs)* : créer un ensemble L des sommets dont on ne connait pas les plus court chemins et un ensemble $S = V \backslash L$; à chaque ajout d'un sommet dans S calculer le plus court chemin pour ces voisins en se basant sur le poids du chemin du sommet courant, puis ajouter à S le sommet ayant le cout le plus faible. L'algorithme s'arrête lorsque l'ensemble L est vide.

Exemple d'application :

Itération	Couple $\lambda_j, p[j]$				Liste
0	**(0,-)**	(∞, -)	(∞, -)	(∞, -)	$\{1,2,3,4\}$
1	-	(3,1)	**(1,1)**	(∞, -)	$\{2,3,4\}$
2	-	**(2,3)**	-	(4,3)	$\{2,4\}$
3	-	-	-	**(4,3)**	$\{4\}$
4	-	-	-	-	$\varnothing$

Plus court chemin entre tous les couples de sommets

Floyd-Warshall $O(n^3)$: calcule itérativement ($k = 1..n$) les plus courts chemins n'utilisant que les sommets 1 à k comme intermédiaires. La matrice se calcule en prenant les lignes et colonnes k et en regardant si la somme des éléments $c_{ik} + c_{kj} <= c_{ij}$, auquel cas on remplace cette valeur et on copie le prédécesseur p_{kj} dans la matrice des prédécesseurs. Si les valeurs dans la diagonale sont plus petites que 0 alors il existe un circuit à cout négatif.
Exemple d'application ($k = 0$ et 1) :

$$(w_{ij}) = \begin{bmatrix} 0 & 3 & \mathbf{8} & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ \mathbf{4} & \infty & \infty & 0 \end{bmatrix} \begin{bmatrix} 0 & 3 & 8 & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ 4 & 7 & \mathbf{12} & 0 \end{bmatrix}$$

$$(p_{ij}) = \begin{bmatrix} - & 1 & \mathbf{1} & - \\ - & - & 2 & 2 \\ - & - & - & 3 \\ 4 & - & - & - \end{bmatrix} \begin{bmatrix} - & 1 & 1 & - \\ - & - & 2 & 2 \\ - & - & - & 3 \\ 4 & 1 & \mathbf{1} & - \end{bmatrix}$$

Dantzig $O(n^3)$: à chaque itération ajouter à la matrice D le sommet k et en utilisant l'ensemble des sommets ajoutés calculer l'ensemble des plus courts chemins entre tous les couples de sommets de 1 à k. Si la diagonale contient des valeurs plus petites que zéro alors il existe un circuit à coût négatif.

Graphes sans circuit

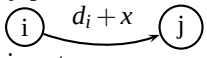
Une **fonction de rang** est une fonction affectant à chaque sommet v d'un graphe orienté un numéro appelé **rang** (noté $rang(v)$) tel que, pour tout arc (u,v) on ait $rang(u) < rang(v)$.
Tri topologique $O(n+m)$: calcule le degré entrant de chaque sommet puis retire et numérote les sommets ayant un degré nul en prenant soin de décrémenter le degré de leurs successeurs.
Plus courts chemins dans un réseau sans circuit : traiter les sommets dans l'ordre du tri topologique, puis pour chaque sommet prendre le chemin le plus court depuis ces prédécesseurs en comptant le poids de l'arc jusqu'au sommet courant.

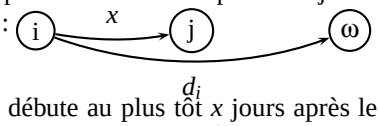
Application à la gestion de projets

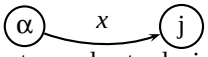
Graphes potentiels-tâches : chaque sommet correspond à une tâche, un arc de poids d_i (durée de la tâche i) relie le sommet i au sommet j si l'exécution de la tâche i dont précéder celle de la tâche j. Si le projet est réalisable le graphe ne doit pas comporter de circuit.

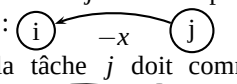
Le **chemin critique** est donné en calculant pour chaque sommet les dates de début *au plus tôt* (maximum des chemins des prédécesseurs) et *au plus tard* (minimum des successeurs). Si ces deux dates sont identiques alors la tâche est **critique**.

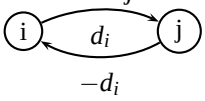
contraintes : les contraintes peuvent être écrites sous forme mathématique : $t_j \geqslant t_i + d_i$, ou t_i et t_j représentent les dates de début des tâches i et j, tandis que d_i correspond au délai minimal entre ces deux événements.

- "la tâche j peut débuter au plus tôt x jours après la fin de la tâche i" : 

- "la tâche j peut commencer au plus tôt x jours après le début de la tâche i" : 

- "la tâche i débute au plus tôt x jours après le commencement des travaux" : 

- "la tâche j débute au plus tard x jours après le début de la tâche j : 

- "la tâche j doit commencer sitôt la tâche i terminée" : 

Les familles et problèmes classiques de théorie des graphes

Graphe complet, clique : graphe simple G non orienté avec n sommets dont toutes les paires de sommets sont reliées par une arête, notation : K_n.

Propriété : le graphe complet K_n possède $\frac{n(n-1)}{2}$ arêtes.

Tournoi : graphe orienté dont le graphe sous-jacent (non orienté) est un graphe complet.

Propriété : un tournoi possède au plus un sommet sans successeur et au plus un sommet sans prédécesseur.

Graphe complémentaire : graphe ne possédant que les arêtes que le graphe simple non orienté $G = (V,E)$ ne possède pas.

Graphe biparti : un graphe est *biparti* si l'ensemble des sommets peut être partitionné en deux sous-ensembles distincts X et Y de sorte que tout arc ait une extrémité dans X et l'autre dans Y.

Graphe biparti complet (noté $K_{m,n}$) : graphe simple, avec $|X| = m$ et $|Y| = n$ dont tous les sommets de X sont reliés directement à tous les sommets de Y.

Couplage : dans un graphe non orienté, sous-ensemble d'arêtes tel que deux arêtes quelconques de ce sous-ensemble n'ont pas d'extrêmité commune.

Couplage parfait : couplage dont les arêtes sont incidentes à tous les sommets.

Couplage maximum : couplage de cardinalité maximum.

Coloration des sommets : attribution d'une couleur à chaque sommet d'un graphe de sorte que les sommets aient une couleur différente à chaque extrémité d'une arêtes.

k-coloration : coloration en k couleurs des sommets d'un graphe. Un graphe admettant une k-coloration est dit *k-coloriable* ou *k-chromatique*.

Nombre chromatique ($\chi(G)$) : le plus petit k pour lequel un graphe G admet une k-coloration.

Graphe planaire (qui peuvent être représentés sur le plan sans croisement d'arêtes) : il existe toujours une coloration en 4 couleurs.

Propriétés :

- Si G est un graphe simple et H un sous-graphe de G alors $\chi(H) \leq \chi(G) \Rightarrow$ toute coloration de G est également une coloration de H.
- Le nombre chromatique du graphe complet (clique) K_n est $\chi(K_n) = n$.
- Soit $\omega(G)$, la taille de la plus grande clique contenue dans le graphe simple G. Alors $\chi(G) \geq \omega(G)$.
- Soit $\Delta(G)$, le degré le plus grand d'un sommet du graphe simple G. Alors $\chi(G) \leq \Delta(G) + 1$. Si G est connexe, il y a égalité uniquement pour $G = K_n$ ou pour G = cycle de longueur impaire.

Ensemble stable : sous-ensemble de sommets d'un graphe non adjacents deux à deux.

Nombre de stabilité ($\alpha(G)$) : nombre maximum de sommets d'un ensemble stable.

Propriétés :

- Dans une coloration, les sommets ayant reçu la même couleur forment un ensemble stable.
- Une clique dans l'ensemble complémentaire $\bar{G}$ correspond à un stabble dans G et vice-versa.

Coloration des arêtes : attribution d'une couleur à chaque arête de G de sorte que les arêtes incidentes à chaque sommet reçoivent toutes des couleurs différentes.

Indice chromatique ($q(G)$) : nombre minimal de couleur nécessaire à la coloration des arêtes de G.

Propriétés : $q(K_{2n+1}) = 2n+1$ et $q(K_{2n}) = 2n-1$

Graphe sur les arêtes ($L(G)$) : construit en attribuant à chaque arête de G un sommet de $L(G)$ et deux sommets distincts de $L(G)$ seront reliés par une arête si les deux arêtes qu'ils représentent on au moins une extrémité commune dans G.

Propriété : la coloration des arêtes de G peut se ramener à une coloration des sommets de $L(G)$ ($q(G) = \chi(L(G))$).

Théorème de Vizing : pour tout graphe simple G, on a $\Delta(G) \leq q(G) \leq \Delta(G) + 1$.

Théorème de König : pour tout graphe biparti G, on a $q(G) = \Delta(G)$.

Un **recouvrement** (des sommets par les arêtes) d'un graphe simple non orienté est un sous-ensemble d'arêtes tel que chaque sommet est l'extrémité d'au moins une arête choisie.

Un **transversal** d'un graphe simple non orienté est un sous-ensemble des sommets tel que chaque arête du graphe est incidente avec au moins un sommet choisi.

Graphes eulériens

Un graphe G est **eulérien** s'il possède une chaîne ou un cycle passant une et une seul fois par chacune de ses arêtes (*cas non orienté*), ou un chemin ou un circuit passant une et une seule fois par chacun de ses arcs (*cas orienté*).

Propriétés :

- Un graphe non orienté connexe est **eulérien** s'il possède 0 ou 2 sommets de degré impair.
- De plus si tous les sommets sont de degré pair le graphe possède un **cycle eulérien** tandis qu'avec 2 sommets de degré impair il possède une **chaîne eulérienne**.
- Un graphe orienté connexe $G = (V,E)$ avec $deg_+(v) = deg_-(v), \forall v \in V$ contient un **circuit eulérien**. Un tel graphe

contient un **chemin eulérien** s'il existe une différence de maximum 1 entre les degrés de 2 sommets.

Algorithme de recherche d'un cycle eulérien : pour un graphe $G = (V,E)$ dont tous les sommets sont de degré pair, permet d'obtenir l'ordre de visite des sommets du parcours. Commencer par orienter les arêtes du graphe de façon arbitraire mais de telle sorte que $deg_+(v) = deg_-(v)$ pour tous les sommets du graphes (si le graphe est orienté et que $deg_+(v) = deg_-(v)$, sauter cette étape). Construire une anti-arborescence en partant d'un sommet choisi r. Parcourir les arcs en partant de la racine et numéroter les arcs en passant par l'anti-arborescence uniquement s'il s'agit du dernier arc non encore utilisé quittant le sommet.

Remarque : si le graphe comporte exactement 2 sommets, u et v, de degré impair, il existe un parcours eulérien. Pour le trouver, ajouter une arête entre u et v, l'orienter $u \to v$, choisir $r = v$ et ignorer le dernier trajet vers r.

Problème du postier chinois : il s'agit de trouver dans un réseau R une chaîne (ou un chemin dans le cas orienté) passant au moins une fois par chacune des arêtes (arcs) et qui soit de poids minimum.

Résolution : le problème revient à déterminer les arcs à dédoubler afin de n'avoir que des sommets de degré pair. Dans le cas d'un réseau orienté il faut utiliser deux ensembles de sommets de degré impair, ceux ayant plus d'arcs sortants que d'entrants et ceux en ayant moins. Equilibrer ensuite les degrés en cherchant les plus courts chemins du deuxième ensemble vers le premier.
Commencer par orienter arbitrairement les arêtes du graphe.
Relier les sommets de degré impair ensemble en dédoublant les arcs dont $\sum(poids) = min$.
Construire un cycle eulérien.

Un graphe non orienté G est dit **hamiltonien** s'il possède un cycle passant une et une seule fois par chacun de ses sommets, et un cycle passant une et une seule fois pas chaque sommet d'un graphe est dit **hamiltonien**.

Heuristique

Problème du voyageur de commerce : trouver le circuit le plus court passant exactement une fois par chaque sommet. Si on énumère chacune des solutions possibles on arrive à $(n-1)!$ tournées différentes ($/2$ si les distances sont symétriques).

Heuristique du plus proche voisin $O(n^2)$: partir du sommet initial et choisir le sommet se trouvant à la plus petite distance du sommet initial. Continuer avec le sommet suivant jusqu'à ce tous les sommets soient visités.

Heuristique de l'insertion la plus coûteuse $O(n^3)$: relier le premier sommet avec le sommet le plus éloigné. Pour chaque sommet ne faisant pas encore partie de la tournée, calculer le détour par rapport à toutes les arêtes de la tournée et stocker le détour le plus petit. Dans la liste des détours minimums de chaque sommet, sélectionner le plus grand des détours $\Rightarrow$ insérer ce sommet. Continuer tant qu'il reste des villes à insérer.

Heuristique de l'insertion la moins coûteuse : même chose que l'heuristique précédente, mais en sélectionnant le plus petit des détours au lieu du plus grand.

Heuristique de la ville la plus éloignée : même chose que l'heuristique précédente, mais en choisissant la ville la plus éloignée d'une des ville de la tournée en construction.

Heuristique d'amélioration – 2-opt : tester toutes les permutations de sommets et si la permutation réduit la longueur du parcours elle est conservée. Suppposons que les deux arêtes (x_1,y_1) et (x_2,y_2) soient deux arêtes non consécutives du circuit. Un nouveau circuit peut être obtenu en remplaçant ces deux arêtes par les deux arêtes (x_1,y_2) et (x_2,y_1). Si le poids du nouveau circuit est inférieur, on procède à l'échange. Sinon, on garde les deux arêtes (x_1,y_1) et (x_2,y_2). Attention au sens du parcours !

Heuristique d'amélioration – 3-opt : au lieu de considérer des paires d'arêtes, on considère des triplets d'arêtes deux à deux non consécutives du circuit. Si ces 3 arêtes sont supprimées, le circuit se coupe en 3 chemins disjoints. On peut alors recombiner ces chemins en utilisant d'autres arêtes entre les sommets au bout de ces chemins.

Flots dans les réseaux

Flot : un *flot* de s à t dans un réseau $R = (V,E,C)$ est une attribution de valeur x_{ij} à chaque arc $(i,j) \in E$ telle que $\sum_{j \in Succ(i)} x_{ij} = \sum_{K \in Pred(i)} x_{ki} \forall i \in v, i \neq s,t$.

Valeur du flot : de s à t est $\sum_{j \in Succ(s)} x_{sj} = \sum_{K \in Pred(t)} x_{kt}$.

Circulation : même définition que le flot sauf que cela n'est pas $\forall i \in v, i \neq s,t$.

Compatible : un flot (ou une circulation) de s à t est *compatible* si $0 \leq x_{ij} \leq c_{ij}$ (c_{ij} représentant la capacité de l'arc).

Flot de valeur maximale : il s'agit de trouver le flot maximum que l'on peut faire transiter du sommet-source s au sommet-puit t.

Algorithme de Floyd & Fulkerson : permet de trouver un flot de valeur maximum de s à t.

1. Construire le réseau d'augmentation R^*.
2. Chercher un chemin de s à t dans le réseau d'augmentation R^*, stop si plus aucun chemin, sinon augmenter le flot au maximum le long du chemin (diminuer le flot dans les arcs de R) et reprendre en 1.
   Pour trouver la coupe minimale : partir du sommet initial et prendre tous les sommets encore atteignables.
3. Fin : vérifier que l'ensemble des sommets marqués dans R^* à la dernière étape forme une coupe séparant s de t telle que sa capacité est égale au flot que l'on a trouvé.

Coupe : ensemble des arcs sortants d'un sous-ensemble de sommets.

Capacité de la coupe : somme des capacités des arcs de la coupe.

Théorème du flot maximum et de la coupe minimum (Ford & Fulkerson) : la valeur d'un flot maximum de s à t est égale à la capacité minimale d'une coupe séparant s de t.