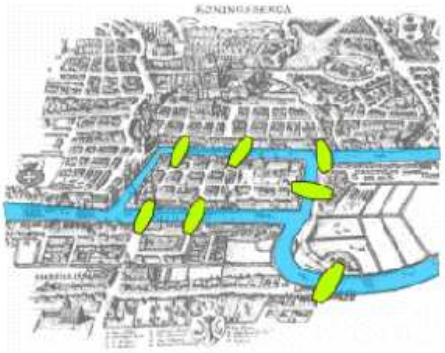
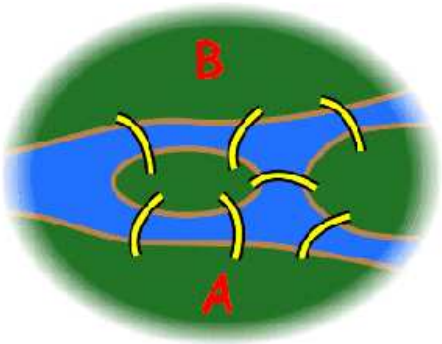
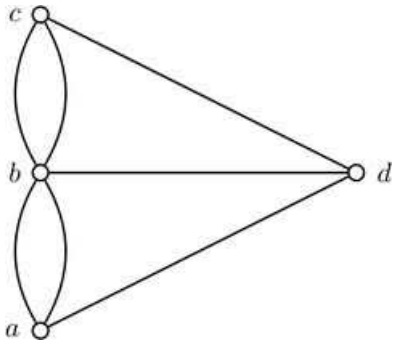
Un peu d'histoire

- Communication d'Euler (1736) sur le problème des ponts de Konigsberg :





- La ville de Konisberg est reliée par 7 ponts. Trouver une façon de partir d'un rivage quelconque et y revenir en passant une et une seule fois par tous les ponts.
- Le problème n'a pas de solution !
- Le problème peut être vu à l'aide du dessin suivant:



- Pause jusqu'à 1847 Kirchoff : théorie des arbres pour circuits électriques
- Cayley 1879 : théorie des arbres (prob d'énumération des molécules chimiques)
- Fin XIX$^{\text{ème}}$: départ des grandes conjectures

Premières conjectures et problèmes

La conjecture des quatre couleurs :

Conj: Quatre couleurs suffisent pour colorier une carte plane de façon que deux pays qui partagent une frontière ne soient pas de la même couleur (Moebius 1840 et après Cayley).

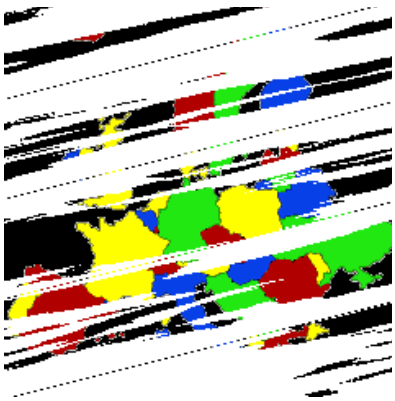


Figure. Europe en 4 couleurs

Solution: Vrai (1976 Appel et Haken en faisant des nombreux cas dénombrés par ordinateur).

Le chemin Hamiltonien :

Sir Hamilton (1859) inventa le casse-tête suivant : On prend un do-décaèdre régulier en bois, un clou est fiché sur chaque sommet marqué du nom de vingt grandes villes mondiales.

Problème: Passez une ficelle une seule fois par chacune des villes (sommets) [Problème ouvert pour conditions nécessaires et suffisantes d'existence d'un tel chemin dans un graphe quelconque]

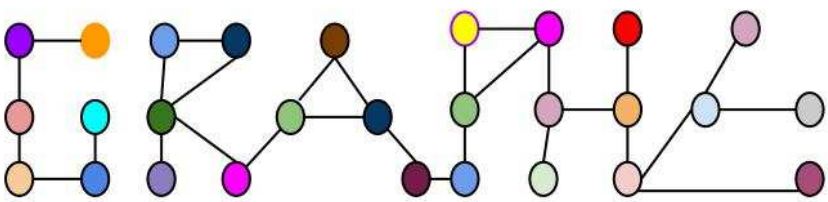
Histoire moderne

1936 : Premier ouvrage sur la théorie des graphes, avec définitions (König)

1946 : L'histoire de la recherche en théorie des graphes commence, motivée par la résolution de problèmes concrets (Kuhn, Ford et Fukerson et Roy [années 1955-1960])

1960 : La théorie des graphes est unifiée au sein d'un ensemble plus vaste d'outils sous l'appellation de recherche opérationnelle et mathématiques discrètes

Introduction:



Définition [graphe non-orienté]

On appelle G un graphe $G = (X, A)$ le couple formé par l'ensemble X (les sommets) et A des relations sur X (les arêtes). Une relation sera une arête, donc tout élément a de A est de la forme

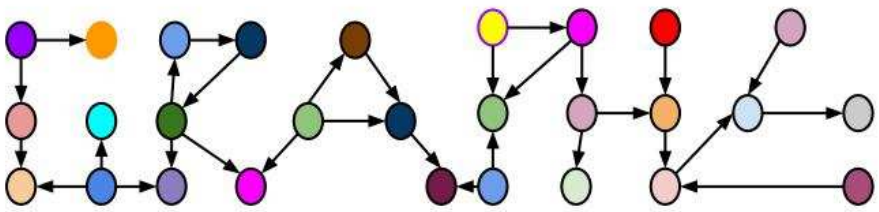
$a = (x, y)$. Cela signifie que x, y sont reliés par une arête.

- [*graphe pas orienté*] Si $(x, y) \in A$ alors $(y, x) \in A$ implicitement (même élément)
- Pour l'arête $a = (x, y)$ on appellera
  - x et y *extrémités* de l'arête a
  - a est incidente en x et y (ou x et y sont adjacents)
  - x et y voisins (y successeur et prédécesseur de x et vice-versa)
- [graphe sans boucle] Un graphe G est sans boucle si A ne contient pas d'arête de la forme (x, x)
- [ordre du graphe] Le nombre de sommets de G est appelé ordre du graphe
- [ordre d'un sommet] Le nombre d'arêtes qui impliquent le sommet x est appelé le dégré de x
- [graphe simple] Un graphe sans boucles ni arêtes « doubles » est appelé graphe simple (où 1-graphe)

Avec l'Ipad : graphes planaires. Un graphe planaire est un graphe que l'on peut dessiner correctement en deux dimensions.

Définition [graphe orienté]

On appelle G un di-graphe $G = (X, A)$ ou graphe orienté, un graphe pour lequel la relation $(x, y) \in A$ est différent de (y, x). On dira que x est l'origine et y l'extrémité de $a = (x, y)$. On dira donc que y est un successeur de x et que x est un prédécesseur de y.

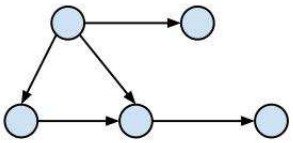


Dictionnaire des graphes

On dénote $\Gamma : X \longrightarrow \mathcal{P}(X)$ (l'ensemble des parties de X) l'application qui fait correspondre à x in X l'ensemble de ses successeurs. L'ensemble des prédécesseurs de x est donc Γ^{-1} (convention d' écriture).

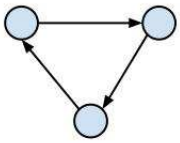
Quelques définitions dans le cas des graphes orientés (cas non orienté facile):

- Degré entrant d'un sommet x : le nombre d'arcs de la forme $a=(y,x)$ avec $y\neq x$. C'est à dire le nombre d'éléments de $\Gamma^{-1}(x)\backslash\{x\}$
- Degré sortant d'un sommet x : le nombre d'arcs de la forme $a=(x,y)$ avec $y\neq x$. C'est à dire le nombre d'éléments de $\Gamma(x)\backslash\{x\}$
- Degré d'un sommet x : la somme du dégrée entrant et du degré sortant
- Puit et source : Un sommet de degré entrant non nul et de degré sortant nul est appelé *puit* et un sommet de degré entrant nul et de degré sortant non nul est appelé *source*
- Un sommet de degrés entrant et sortant nuls est appelé *isolé*
- Graphe symétrique : Un graphe est symétrique si, pour tout arc $a_1=(x,y)\in A$, l'arc $a_2=(y,x)\in A$
- Graphe asymétrique : Un graphe est symétrique si, pour tout arc $a_1=(x,y)\in A$, l'arc $a_2=(y,x)\notin A$
- Graphe transitif : Un graphe est transitif si, pour deux arcs adjacents $a_1=(x,y)$ et $a_2=(y,z)$, alors l'arc $a_3=(x,z)\in A$ (cas pas orienté en TD)



Graphe 1

Graphe 2



Graphe 3

Observation: Le concept de graphe symétrique est très proche de celui des graphes non orientés. En fait, à tout graphe symétrique, on peut associer un graphe non orienté en substituant aux arcs $a_1=(x,y)$ et $a_{2=}(y,x)$ une arête $a=(x,y)$.

Définition [graphe complet]

Un graphe $G=(X,A)$ est dit *complet* si, pour toute paire de sommets (x,y) il existe au moins un arc de la forme (x,y) ou (y,x).

Un graphe simple complet d'ordre n est noté K_n.

Un sous-ensemble de sommets $C\subset X$ tel que deux sommets de C sont reliés par une arête est appelé une *clique*.

Définition [sous-graphe]

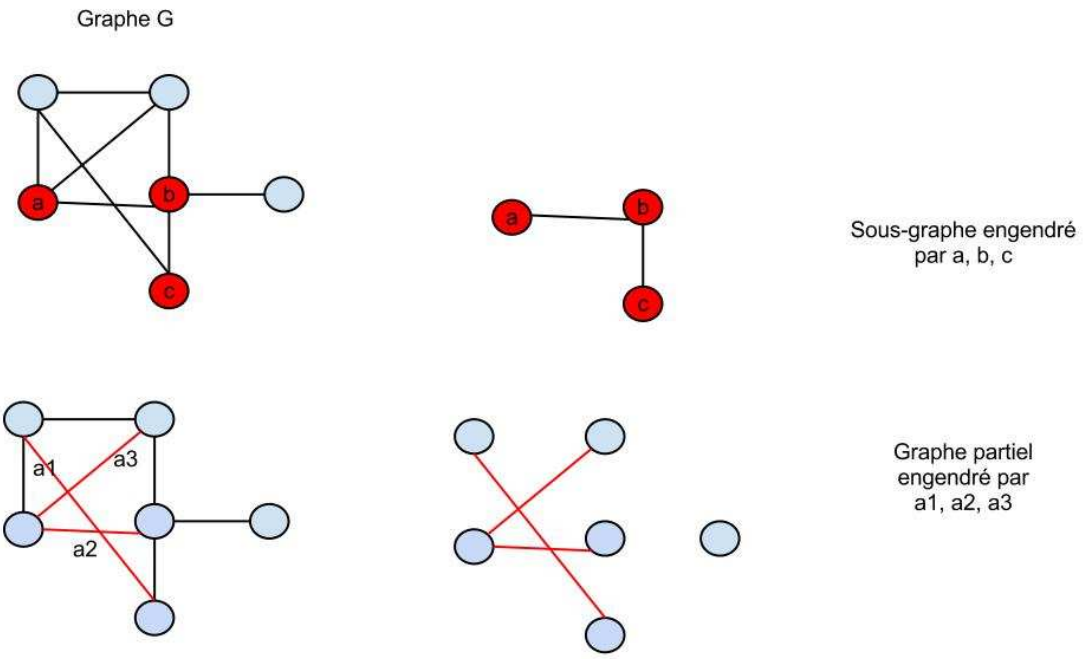
Soit un graphe $G=(X,A)$ et $X'\subset X$. Le sous-graphe engendré par X' est $G'=(X',A')$, où A' est formé par les arêtes d'extrémités dans X'.

Définition [graphe partiel]

Soit un graphe $G=(X,A)$ et $A_1\subset A$, le graphe partiel engendré par A_1 est $G_1=(X,A_1)$. On peut parfois supprimer les sommets de X qui restent isolés.

Observation : une clique d'un graphe G est un sous-graphe complet de G.

Exemple quelconque :



Modes de représentation d'un graphe

L'histoire de la théorie des graphes est fortement relié à l'avènement de puissants calculateurs. Il est donc légitime de s'intéresser à la manière de représenter un graphe en machine. Plusieurs mode sont envisageables selon la nature des traitements que l'on veut appliquer au graphe considéré.

Listes de succession (dictionnaire)

Le graphe est représenté à l'aide d'un dictionnaire : un tableau où chaque ligne correspond à un sommet et comporte la liste des successeurs ou des prédécesseurs de ce sommet.

Matrices d'adjacence

L'idée consiste à coder une matrice où chaque élément est une arête possible. Plus précisément:

Soit un graphe $G = (X, A)$ à n sommets. La matrice d'adjacence de G est égale à la matrice $U = (u_{i,j})$ de dimensions $n \times n$ telle que :

$$\begin{cases} 1 \text{ si } (i,j) \in A \\ 0 \text{ sinon} \end{cases}$$

Observation : un graphe quelconque a une matrice d'adjacence quelconque, alors qu'un graphe non orienté possède une matrice d'adjacence symétrique. L'absence de boucle se traduit par une diagonale nulle.

Ce genre d'encodage engendre des matrices très creuses, cependant la recherche de chemins ou de chaînes s'effectue aisément.

Propriétés de la matrices d'adjacence :

Soit $G=(X,A)$ un graphe à n sommets et $U=(u_{i,j})$ sa matrice d'adjacence, on a :

- La somme des éléments de la $i-$ème ligne de U est égale au degré sortant $d_s(x_i)$ du sommet x_i de G
- La somme des éléments de la $j-$ème colonne de U est égale au degré entrant $d_e(x_j)$ du sommet x_j de G
- U est symétrique si et seulement si le graphe G est symétrique

Matrices d'incidence

Une autre idée de représentation matricielle d'un graphe exploite la relation d'incidence entre arêtes et sommets.

Soit $G=(X,A)$ graphe à n sommets sans boucles. On appelle matrice d'incidence (aux arcs) de G la matrice $M=(m_{i,j})$ de dimension $n\times m$ telle que:

$$m_{i,j}=\begin{cases}1 \text{ si } x_i \text{ est l'extrémité initiale de } a_j\\ -1 \text{ si } x_i \text{ est l'extrémité finale de } a_j\\ 0 \text{ si } x_i \text{ n'est pas extrémité de } a_j\end{cases}$$

Pour un graphe non orienté sans boucles, on a plus simplement:

$$m_{i,j}=\begin{cases}1 \text{ si } x_i \text{ est extrémité de } a_j\\ 0 \text{ sinon}\end{cases}$$

Étude de la connexité

Chaînes et cycles, élémentaires et simples

Une *chaîne* est une séquence finie et alternée de sommets et d'arêtes, débutant et finissant par des sommets, de façon que chaque arête est incidente avec les sommets qui l'encadre dans la séquence

Le premier et dernier sommet sont appelés *extrémités* de la chaîne.

La longueur de la chaîne est égale au nombre d'arêtes qui la composent. Si aucun des sommets de la séquence n'apparaît plus d'une fois, la chaîne et dite *élémentaire*. Si aucune des arêtes de la séquence n'apparaît plus d'une fois, la chaîne et dite *simple*.

Définition [cycle] :

Un cycle est une chaîne dont les extrémités coïncident. Un cycle élémentaire est un cycle minimal pour l'inclusion, c'est à dire qui ne contient strictement aucun autre cycle (On rencontre pas deux fois le même sommet).

Chemins et circuits

Définition [chemin] :

Un chemin est une séquence finie et alternée de sommets et d'arcs, débutant et finissant par des sommets, telle que chaque arc est sortant d'un sommet et incident au sommet suivant dans la séquence.

Si aucun sommet de la séquence n'apparaît plus d'une fois, le chemin est dit *élémentaire* (cf. *simple*).

Un *circuit* est un chemin dont les extrémités coïncident.

En parcourant un *circuit élémentaire* on ne rencontre pas deux fois le même sommet.

Graphes et sous-graphes connexes et cycles

Un graphe est connexe si l'on peut atteindre n'importe quel sommet à partir d'un sommet quelconque en parcourant différentes arêtes. Plus formellement:

Définition [graphe connexe] :

Un graphe G est *connexe* s'il existe au moins une chaîne entre une paire quelconque de sommets de G. La relation:

$$x_i \, R x_j \Leftrightarrow \begin{cases} \text{soit } x_i = x_j \\ \\ \text{soit } \exists \text{ une chaîne joignant } x_i \text{ à } x_j \end{cases}$$

est une relation d'équivalence (réflexivité, symétrie et transitivité). Les classes d'équivalence induites sur X par cette relation forment une partition de X en $X_1, ..., X_p$.

Le nombre p de classes d'équivalence distinctes est appelé *nombre de connexité* du graphe. Un graphe est connexe si et seulement si ce nombre est égal à 1.

Définition [composantes connexes] :

Les sous-graphes G_1, ..., G_p engendrés par les sous-ensembles X_1, ..., X_p sont appelés les *composantes connexes* du graphe.

La vérification de la connexité est un des premiers problèmes de la théorie des graphes (algorithme pour vérifier la connexité).

Graphes et sous-graphes fortement connexes

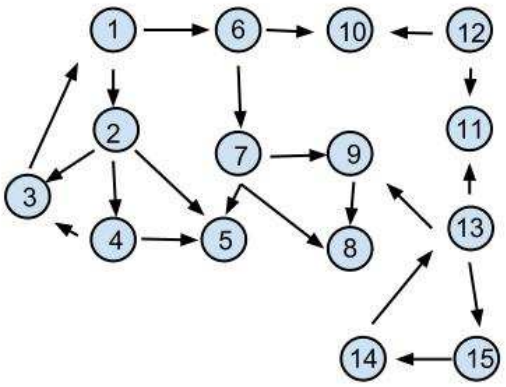
Définition [graphe fortement connexe] :

Un graphe orienté est dit *fortement connexe* s'il existe un chemin joignant deux sommets quelconques. La relation suivante

$$x_i R x_j \Leftrightarrow \begin{cases} \text{soit } x_i = x_j \\ \\ \text{soit il existe un chemin joignant } x_i \text{ à } x_j \text{ et un chemin joignant } x_j \text{ à } x_i \end{cases}$$

est une relation d'équivalence et les classes d'équivalence induites sur X par cette relation forment une partition de X en $X_1, ..., X_q$.

Les sous-graphes $G_1, ..., G_q$ engendrés par les sous-ensembles $X_1, ..., X_q$ sont appelés les *composantes fortement connexes* du graphe.



Définition [graphe réduit] :

On appelle graphe réduit G_r le quotient du graphe G par la relation de forte connexité $G_r = G\backslash R$; les sommets de G_r sont donc les composantes fortement connexes et il existe un arc entre C_i et C_j si et seulement s'il existe au moins un arc entre un sommet de C_i et un sommet de C_j dans G.

Exercice : Le graphe G_r est sans circuit.

La recherche des composantes fortement connexes et la détermination du graphe réduit revêtent une grande importance dans la théorie des graphes

Un graphe connexe à n sommets possède au moins n-1 arêtes. S'il en a exactement n-1, c'est un arbre.

Plus généralement, un graphe à n sommets avec k composantes connexes possède au moins n-k arêtes.

Parcours eulériens et hamiltoniens

L'étude de ces types de problèmes remonte aux origines de la théorie des graphes.

Applications : tournées de distribution, tracé automatique sur ordinateur, problèmes d'ordonnance d'atelier, etc.

Chaînes et Cycles eulériens

Il s'agit d'une généralisation du jeu bien connu consistant à dessiner toutes les arêtes d'un graphe avec un crayon sans jamais le soulever, ni passer 2 fois par la même arête.

Définition [chaîne eulérienne] :

Soit $G = (X, A)$ un graphe orienté.

Une *chaîne eulérienne* est une chaîne rempruntant une fois et une fois seulement chaque arête de G.

Un *cycle eulérien* est une chaîne eulérienne dont les extrémités coïncident. Un graphe possédant un cycle eulérien est appelé *graphe eulérien*.

Un peu d'histoire

Le problème de l'existence et de la détermination d'un cycle eulérien dans un graphe non orienté a été posé et résolu par Euler (1736) à propos du problème des ponts de Königsberg. Le théorème est le suivant:

Théorème [chaîne eulérienne] :
Un graphe non orienté connexe possède une chaîne eulérienne si et seulement si le nombre de sommets de degré impair est égal à 0 ou 2. Il admet un cycle eulérien si et seulement si tous ses sommets ont un degré pair.

Preuve :
Condition nécessaire. Si le cycle eulérien existe, on peut l'orienter de manière arbitraire. En chaque sommet, le nombre d'arcs incidents doit être égal au nombre d'arcs sortants, les sommets doivent donc être de degré pair. Dans le cas d'une chaîne, les deux extrémités font exception, on part (on arrive) une fois de plus, d'où un degré impair pour ces deux sommets.
Condition suffisante. Par récurrence, soit le théorème vérifié pour des graphes connexes à moins de m arêtes. Soit $G = (X, A)$ un graphe de m arêtes vérifiant la condition du théorème. Si $m = 1$, il est vrai. Si G possède deux sommets de degré impair, soient a et b ces sommets. Soit L la chaîne parcourue par un voyageur partant de a dans une direction quelconque qui ne peut pas emprunter la même arête 2 fois. Si à un instant donné, il arrive en un sommet $x \neq b$, il aura utilisé un nombre impair d'arêtes incidents à x et il pourra donc repartir par une arête non déjà utilisée. Quand il ne peut plus bouger, on est donc arrivés en b. Si toutes les arêtes ont été utilisé, cela signifie que L est une chaîne eulérienne.
Sinon, on considère le graphe partiel des arêtes non utilisées. Ce graphe a tous ses sommets de degré pair.
Soient $G'_1, G'_2, ..., G'_p$ les composantes connexes de G qui comportent au moins une arête.
Chacun des sous-graphes G'_i possède moins de m arête et donc un cycle eulérien μ_i (par hypothèse de récurrence). Comme G est connexe, L rencontre successivement $G'_1, G'_2, ..., G'_p$ en les sommets $x_1, ..., x_p$. le parcours alors constitué par :

- La chaîne L entre a et x_1,
- le cycle μ_1 entre x_1 et x_1,
- la chaîne L entre x_1 et x_2,
- le cycle μ_2 entre x_2 et x_2,
- ..
- la chaîne L entre x_p et b.

est une chaîne eulérienne entre a et b dans G. Le théorème est vrai à l'ordre m.

Définition [eulérien cas orienté] :

Un chemin dans un graphe orienté est dit eulérien s'il passe exactement une fois par chaque arête. Un graphe orienté est dit eulérien s'il admet un circuit eulérien. (Voir preuve précédente)

Théorème [chaîne eulérienne] :

Un graphe orienté admet un chemin eulérien (mais pas de circuit eulérien) si et seulement si, pour tout sommet sauf deux (disons a et b), le degré entrant est égal au degré sortant et
$d_e(a) = d_s(a) - 1$ et $d_e(b) = d_s(b) + 1$
Un graphe orienté connexe admet un circuit eulérien si, et seulement si, pour tout sommet c on a $d_e(c) = d_s(c)$.

Problèmes classiques :

- Problème du postier chinois (non orienté) : le graphe n'étant pas forcement eulérien on cherche à minimiser la longueur totale du parcours
- Problème d'organisation de ramassage ordures, inspection de réseaux.
- Dans le cas orienté le problème se ramène à la recherche d'un flot à coût minimum (qu'on verra plus tard)

Quelque résultat (sans preuve):

- Un graphe non orienté admet un cycle chinois si et seulement si il est connexe
- Un graphe orienté admet un circuit chinois si et seulement si il est fortement connexe

Chaînes et Cycles hamiltoniens

Soit $G = (X, A)$ un graphe connexe d'ordre n.

Définition [chemin hamiltonien]
On appelle *chemin (chaîne) hamiltonien* un chemin passant une fois et une seule fois par chacun des sommets de G. Un chemin hamiltonien est donc un chemin élémentaire de longueur $n - 1$.
Un circuit (cycle) hamiltonien est un circuit qui passe une fois, et une seule fois, par chacun des sommets de G.
On dit que G est hamiltonien s'il contient un cycle hamiltonien (cas non orienté) ou un circuit hamiltonien (cas orienté).

Problèmes concrets :

Le problème du voyageur de commerce : visiter n clients $x_1, ..., x_n$ en partant de x_0 et revenir à son point de départ. Il connaît les distances $d_{i,j}$ entre x_i et x_j.

Dans quel ordre doit rendre visite pour que la distance totale soit minimale?

Solution: Chercher un cycle hamiltonien de longueur totale minimale dans le graphe complet G construit sur $X = \{x_1, ..., x_n\}$

Le problème de l'ordonnance des tâches : on cherche un ordre dans lequel effectuer les n tâches données (pas simultanément) tout en respectant des contraintes d'antériorité.

Solution: On construit le graphe G dont l'ensemble des sommets est l'ensemble des tâches et où il existe l'arc (i, j) si la tâche i peut se faire avant la j. Le problème revient à trouver un chemin hamiltonien de G

Notons que l'on ne connaît pas de condition nécessaire et suffisante d'existence de cycles ou de circuits hamiltoniens.

Parcours sur les graphes

- Parcours en largeur
- Parcours en profondeur (TD)
- Tri topologique (TD)
- Arbres couvrant minimale (Kruskall et Prim)

Parcours en largeur d'un graphe (Algorithme BFS)

Le parcours en largeur est un simple algorithme à la base de nombreuses techniques d'algorithmes sur les graphes (ex. Dijkstra avec le plus court chemin et Prim pour l'arbre couvrant minimal).

Idée de l'algorithme:

- Étant donné $G = (X, A)$ et un sommet origine s on empreinte les arcs de G pour découvrir tous les sommets accessibles depuis s
- Il calcule la distance entre s et tout sommet accessible
- Il calcule une *arborescence* de racine s
- Il découvre d'abord tout sommet à distance k de s et après ceux qui sont à distance $k + 1$
- C'est un algorithme de coloriage
- Pour tout v accessible depuis s le chemin de s à v dans l'arborescence correspond à un plus court chemin entre s et v
- Algorithme pour cas orienté et pas orienté

Algorithme de coloriage :

Le parcours en largeur est un algorithme de coloriage. Cela signifie que au départ tout sommet est blanc et l'algorithme se termine quand tout sommet est coloré.

A chaque étape de l'algorithme on colorie un sommet (ou plusieurs) .

- Un sommet est colorié la première fois qu'il est rencontré au cours de la recherche

- Comment colorier : On colorie en gris les sommets accessible par u et on colore u en noir quand on explore la liste d'adjacence d'un de ses fils

Algorithme 1

Input: G donné par listes d'adjacence
Pour chaque $u \in X - \{s\}$ faire
    couleur[u] $\leftarrow$ blanc
    d[u] $\leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
couleur[s] $\leftarrow$ gris
d[s] $\leftarrow 0$
$\pi[s] \leftarrow$ NIL
F$\leftarrow\{s\}$
tant que $F \neq \emptyset$ faire
    $u \leftarrow$ tête $[F]$
        pour chaque $v \in$ Adj[u] faire
            si couleur[v]$\leftarrow$ BLANC alors
                couleur[v]$\leftarrow$ GRIS
                d[v]$\leftarrow$ d[u]+1
                $\pi[v] \leftarrow u$
                Enfile(F,v)
        Défile(F)
        couleur[u]$\leftarrow$ NOIR

Analyse de complexité :

- Au départ tout sommet est blanc et chaque sommet est colorié au plus une fois
- Les opérations d'enfilement se font en temps $O(1)$ (temps pour toutes les sommets est $O(X)$)
- La liste d'adjacence d'un sommet n'est balayée qu'au moment ou le sommet est défilé (temps pour toutes les listes est $O(A)$)
- Coût total de BFS : $O(X + A)$ temps linéaire par rapport à la taille de la liste d'adjacence

Validité et preuve de l'algorithme (partie un peu longue...)

Définition[distance]
Étant donné un graphe $G = (X, A)$ la distance du plus court chemin $\delta(s, v)$ entre s et v est le nombre minimum d'arcs dans un chemin quelconque reliant s à v (ou ∞ s'il n'existe aucun chemin).
Un chemin de longueur $\delta(s, v)$ est le plus court chemin de s à v.
On va montrer que le parcours en largeur calcule les distances de plus court chemin (par étapes).

Lemme 1
Soit $G(X, A)$ un graphe et soit $s \in S$ un sommet arbitraire. Alors, pour tout arc $(u, v) \in A$ on a $\delta(s, v) \leq \delta(s, u) + 1$

Si u est accessible depuis s, alors v l'est aussi. Dans ce cas, le plus court chemin de s à v ne peut pas être plus long que le plus court chemin de s à u prolongé avec (u,v). Si u ne peut pas être atteint à partir de s, alors $\delta(s,u)=\infty$ et l'égalité vaut.

On veut montrer que BFS calcule $d[v]=\delta(s,v)$ pour tout $v\in S$.

Lemme 2
Étant donné un graphe $G=(X,A)$ (orienté ou non), on exécute BFS à partir d'un sommet s. Alors, pour tout $v\in X$ la valeur $d[v]$ calculée par BFS vérifie $d[v]\geq\delta(s,v)$.

Preuve : Par récurrence sur le nombre total des sommets insérés dans F. Quand s vient d'être inséré dans F la base de l'induction est vérifié ($d[s]=0=\delta(s,s)$) et $d[v]\geq\infty\geq\delta(s,v)$ pour tout $v\in S-\{s\}$.
On considère donc un sommet v blanc, d écouvert pendant le balayage des successeurs d'un sommet u. Pour induction on a $d[u]\geq\delta(s,u)$. Suite à l'algorithme on aura:

$$\begin{aligned} d[v] &= d[u]+1 \\ &\geq \delta(d,u)+1 \\ &\geq \delta(s,v) \end{aligned}$$

Une fois v inséré dans F, il vient coloré en GRIS et donc sa valeur $d[v]$ ne sera plus modifiée. L'hypothèse d'induction est donc maintenue.

Le Lemme suivante (ans démonstration) dit que à tout moment, il existe au plus deux valeurs d distinctes dans la file.

Lemme 3
Étant donné un graphe $G=(X,A)$, on exécute BFS à partir d'un sommet s. La file F contient les sommets $\{v_1,\ldots,v_r\}$ où v_1 est la tête de F et v_r la queue.
Alors $d[v_r]\leq d[v_1]+1$ et $d[v_i]\leq d[v_{i+1}]$ pour $i=1,\ldots,r-1$.
On peut maintenant prouver que le parcours en largeur trouve correctement les distances de plus courts chemin

Validité BFS :

BFS exécuté sur $G=(X,A)$ à partir de s, découvre chaque sommet $v\in X$ accessible à partir de s et à la fin $d[v]=\delta(s,v)$ pour tout $v\in X$. Par ailleurs, pour tout $v\neq s$ accessible à partir de s, l'un de plus court chemin de s à v et le plus court chemin de s à $\pi[v]$ complété par $(\pi[v],v)$.
Preuve Prémier cas : v n'est pas accessible depuis s. Pour le Lemme 3 $d[v]\geq\delta(s,v)=\infty$, $d[v]$ ne peut pas être initialisé à une valeur finie à la ligne 14. Par induction, il ne peux pas exister de premier sommet dont la valeur d est mise à ∞ par la ligne. Donc cela est exécuté seulement pour sommets à valeurs $d$finies. Donc v n'est jamais découvert s'il n'est pas accessible.

Deuxième cas : v est accessible depuis s. Soit S_k l'ensemble des sommets situés à une distance k de s. On procède par induction sur k. L'hypothèse est la suivante:

Pour tout sommet $v \in S_k$ il existe exactement un instant où :

- v est coloré en GRIS
- $d[v]$ a la valeur k
- Si $v \neq s$, alors $\pi[v]$ a la valeur u pour un certain $u \in S_{k-1}$ e
- v est inséré dans la file F

Notez que cette situation se produit au plus une fois.

Si $k = 0$ (base) on a $S_0 = \{s\}$, puisque s est le seul sommet à distance 0 de s. Pendant l'initialisation, s est colorié en gris, $d[s]$ est mis à 0 et s est placé dans F (ce qui vérifie l'hypothèse d'induction).

Pour l'étape inductive, on remarque que F n'est jamais vide avant la fin de l'algorithme et que, une fois qu'un sommet u est inséré dans F, ni $d[u]$ ni $\pi[u]$ ne sont modifiés.

Après le Lemme 3, si les sommets sont dans F dans un ordre $v_1, ..., v_r$, la séquence des distances est monotone croissante: $d[v_i] \leq d[v_{i+1}]$ pour $i = 1, ..., r-1$.

Considérons $v \in S_k$ où $k \geq 1$. La propriété de monotonie et le Lemme 3 ($d[v] \geq k$) implique que si v doit être découvert, il l'est après que tout sommet de S_{k-1} est infilé.

Comme $\delta(s, v) = k$, il existe un chemin de k arcs de s à v, donc il existe un sommet $u \in S_{k-1}$ tel que $(u, v) \in A$. Soit u le premier sommet de ce type colorié en GRIS.

Au moment de la coloration, sa liste d'adjacence est parcourue et v est découverte. Après, on colore v en GRIS et on fixe $d[v] = d[u] + 1 = k$, $\pi[v] = u$ et on insère v dans la file.

Comme v est quelconque, l'hypothèse est montré. On observe ensuite que $\pi[v] \in S_{k-1}$. On peut donc obtenir un plus court chemin de s à v en prenant un plus court chemin de s à $\pi[v]$ et en le prolongeant par $(\pi[v], v)$.

Parcours en profondeur d'un graphe en TD et tri topologique

Recherche de chemins faire en TD

Arbres et arborescence

Définition [arbre]

Un arbre $\mathcal{T} = (X, T)$ est un graphe connexe sans cycles.

Un graphe sans cycles qui n'est pas connexe est appelé une forêt (chaque composante connexe est un arbre).

- Un arbre est un graphe simple
- $\mathcal{T}$ est un arbre si et seulement s'il existe une chaîne et une seule entre deux sommets quelconques

Arbres et graphes : Étant donné $G = (X, A)$, un arbre de G est un graphe partiel connexe et sans cycles.

- Si cet arbre contient tous les sommets de G on l'appellera *arbre couvrante* (ou *arbre maximum*)
- Une forêt de G est un graphe partiel sans cycle de G
- Une forêt maximale de G est une forêt de G maximale pour l'inclusion (si on rajoute une arête on a un cycle)

Recherche de forêt maximale (Algorithme de coloriage rouge et vert)

Considérons $G = (X, A)$ à n sommets, m arcs et p composantes connexes; l'algorithme suivant permet de trouver une forêt maximale de G.

- Initialement, tous les arc sont pas colorés. On examine tous les arc pour les colorier soit en rouge soit en vert
- A une étape quelconque, G_c est le graphe engendré par les arcs colorés et G_r le graphe engendré par les arcs rouges
- Chaque fois qu'un arc a pas coloré est examiné :
  - soit il passe par a un cycle élémentaire μ dont tous les arcs sauf a sont rouges. Dans ce cas on colore l'arc en vert. Le nombre de connexité de $G_c$et G_r reste constante
  - soit un tel cycle n'existe pas; a peut connecter deux sommets qui n'étaient pas connecté encore dans G_c. On colore a en rouge et le nombre de connexité de G_c et G_r décroît de 1.

Exercice : Prouver que le G_r obtenu est bien une forêt maximale de G.

Propriétés :

- Si G possède n sommets et p composantes connexes, une forêt maximale de G à $n - p$ arcs exactement.
- Soit $\mathcal{T} = (X, T)$ une forêt maximale de G, alors $\mathcal{T}$ et G ont le même nombre de connexité.

Arbres couvrants de poids minimum

Considérons le problème qui consiste à relier n villes par un réseau câblé de la manière la plus économique possible. On suppose connaître la longueur $l_{ij} = l(a_{ij})$ du câble pour relier la ville i à j. Le réseau doit être:

- connexe
- sans cycle (pour coût minimal)

C'est donc l'arbre maximum le plus économique.

Exemple pratique: Imaginez que l'on souhaite concevoir un circuit électrique. On veut donc interconnecter un ensemble de n broches avec $n-1$ câbles, chacun reliant deux broches. Parmi tous les arrangements possibles, celui qui utilise une longueur de câble minimale est optimale.

Réformulation du problème :

Soit G un graphe connexe pondéré par une fonction positive l des arêtes. Soit un arbre couvrant $\mathcal{T} = (X, B)$ défini comme graphe partiel de G avec un ensemble d'arêtes B. Son coût total est:

$$l(\mathcal{T}) = \sum_{a \in B} l(a)$$

On dit que $\mathcal{T}$ est un arbre couvrant de poids minimal de G si $l(\mathcal{T})$ est minimal parmi les poids de tous les arbres couvrants possibles de G.

Exercice: Si tous les poids sont différents l'arbre couvrant de poids minimal est unique.

Plusieurs algorithme sont disponibles pour résoudre ce problème. Les plus simples sont les algorithmes de **Prim** et de **Kruskal**. Les deux se basent sur le concept d'**arête sûre**.

ACM générique et arête sûre

Soit $G = (X, A)$ un graphe non orienté, connexe avec une fonction de pondération $l : A \to \mathbb{R}$. On souhaite trouver un arbre couvrant minimal pour G. La stratégie utilisée par Prim et Kruskal est dite *gloutonne* :

- A chaque étape on a l'invariant suivant : le nombre d'arêtes géré par l'algorithme (on va l'appeler E) fait parti d'un arbre couvrant minimal.
- Si à une étape donnée E fait parti d'un arbre couvrant minimal et on rajoute l'arête (u, v), alors $E \cup (u, v)$ fera parti d'un arbre couvrant minimal.
- L'arête rajouté est appelé sûre car elle garantie l'invariant de l'algorithme.

Algorithme 2

E$\leftarrow\emptyset$
tant que (E n'est pas un AC) faire :
    trouver une arête sûre pour E
    $E \leftarrow E \cup \{(u, v)\}$
retourner E

Questions ?

On a vu que le problème de l'algorithme ACM est de trouver des arêtes sûres. Prim et Kruskall ont recours à cette règle pour trouver efficacement des arêtes sûres. On va donner un théorème qui garantie que une arête est sûre. Pour cela on a besoin de quelque définition.

Définition [coupure d'un graphe]

La coupure $(P, X - P)$ du graphe orienté $G = (X, A)$ est une partition de X. On dit que

- $(u, v) \in A$ traverse la coupure $(P, X - P)$ si l'une de ses extrémités est un sommet de P et l'autre un sommet de $X - P$
- Une coupure respecte un ensemble E d'arêtes si aucune arête de E traverse la coupure.
- Une arête est une arête minimale pour la traversée de la coupure si son poids est minimal parmi toutes les arêtes qui traversent la coupure.

Théorème de l'arête sure

Soit $G = (X, A)$ un graphe non orienté connexe, avec w fonction de pondération à valeurs réelles. Soit $E \subset A$ inclus dans un ACM de G et $(P, X - P)$ une coupure quelconque de G qui respecte E.

Si (u, v) est une arête minimale traversant $(P, X - P)$ alors (u, v) est sûre pour E.

Preuve :

Soit $\mathcal{T}$ un ACM qui inclut E et supposons que $\mathcal{T}$ ne contient pas l'arête minimale (u, v) (sinon ok) :

- On va construire un ACM $\mathcal{T}'$ qui inclut $E \cup \{(u, v)\}$ par une technique de copier - coller en montrant que (u, v) est une arête sûre pour E
- L'arête (u, v) est dans la coupure, donc il existe une autre arête de $\mathcal{T}$ qui fait partie de la chaîne et qui traverse la coupure (x, y)
- L'arête (x, y) n'est pas dans E car elle respecte pas la coupure
- Comme le chemin de u à v est unique, la suppression de (x, y) sépare $\mathcal{T}$ en deux
- On reconnecte $\mathcal{T} \setminus \{(x, y)\}$ en rajoutant (u, v). On appelle ce nouveau arbre $\mathcal{T}'$

On montre que $\mathcal{T}'$ est un ACM. Puisque (u, v) est une arête minimale traversant la coupure et que (x, y) traverse aussi :

$$w(\mathcal{T}') = w(\mathcal{T}) - w(x, y) + w(u, v) \leq w(\mathcal{T})$$

Cela implique que $w(\mathcal{T}) = w(\mathcal{T}')$ et donc $\mathcal{T}'$ est minimale aussi.

Est-ce qu'on a fini ? Non, il reste à montrer que (u, v) est une arête sûre pour E.

On a $E \subset \mathcal{T}'$, car $E \subset \mathcal{T}$ et $(x, y) \notin E$. Donc $E \cup \{(u, v)\} \subset \mathcal{T}'$.

Comme $\mathcal{T}'$ est ACM alors (u, v) est une arête sûre pour E.

Exercice :

Montrez que si $G = (X, A)$ graphe non orienté connexe, avec fonction de pondération w à valeurs réelles définie sur A. Soit $E \subset A$ inclus dans un ACM de G, et soit

C une composante connexe (arbre) de la forêt $G_E = (X, A)$. Si (u, v) est arête minimale reliant C à une autre composante de G_E alors (u, v) est une arête sûre pour E.

(La coupure $(C, S - C)$ respecte E et (u, v) est donc arête minimale pour cette coupure)

Les deux algorithmes de Kruskall et Prim utilisent le théorème de l'arête sûre, qui trouvent de deux façon différentes et qui appliquent à l'algorithme ACM générique. Pour l'algorithme de Kruskal, l'ensemble E est une forêt. Dans l'algorithme de Prim E est un arbre unique.

Algorithme de Kruskall

Idée : L'algorithme de Kruskall suit directement de ACM générique. Il trouve une arête sûre à ajouter à la forêt, en cherchant une arête (u, v) de poids minimal parmi toutes les arêtes reliant deux arbres

L'exercice précédent garantie la validité de l'algorithme (ex.)

- L'algorithme de Kruskal est un algorithme glouton, car il ajoute une arête de poids minimal
- Il ressemble à l'algorithme pour calculer les composantes connexes d'un graphe

Algorithme 3

ACM-KRUSKAL(G,w)
$E \leftarrow \emptyset$
pour chaque $v \in X$ faire
    créer-ensemble(v)
trier les arêtes par ordre croissante de pondération w
pour chaque (u, v) par ordre
    faire si Trouver-ensemble(u) $\neq$ Trouver-ensemble(v)
        alors $E \leftarrow E \cup \{(u, v)\}$
            Union(u, v)

où : Trouver-ensemble retourne le représentant de l'ensemble contentant u
    Union retourne la combinaison de deux arbres

Temps d' exécution :

Le temps d'exécution de l'algorithme dépend de l'implémentions de la structure de données d'ensembles disjoints. Une des versions plus efficaces requiert en temps $O(X)$ pour l'initialisation, $O(\text{Alog}(A))$ pour le tri des arêtes et encore $O(\text{Alog}(A))$ pour les opérations sur la forêt d'ensembles disjoints.

#Exemple d'exécution avec l'Ipad#

Algorithme de Prim : détails en TD

Comme l'algorithme de Kruskal, l'algorithme de Prim est un cas particulier de ACM générique. Cet algorithme ressemble à l'algorithme de recherche de plus court chemin dans les graphes. La stratégie est gloutonne et consiste en rajouter une arête de poids moindre à un arbre unique (de façon que le graphe résultant soit encore un arbre).

Le problème du plus court chemin

Étant donné $G=(X,A)$ on associe à chaque arc $a\in A$ un nombre $l(a)\in\mathbb{R}$ appelé longueur de l'arc. Si $a=(i,j)$ on utilise l_{ij} pour la longueur de l'arc a.

Le problème du plus court chemin entre deux sommets i et j sera de trouver un chemin $\mu(i,j)$ dont la longueur totale $l(\mu)=\sum_{a\in\mu(i,j)} l(a)$ soit minimum.

Le longueur $l(a)$ peut s'interpréter aussi bien comme coût de transport sur l'arc a, ou bien comme le temps de parcourt ou comme dépense de construction de l'arc a.

Plusieurs algorithmes sont possibles, selon la structure du graphe en question.

Algorithme 1 : (Dijkstra-Moore)

Cet algorithme est efficace quand les longueurs sont positives. Posons:

- $X=\{1,2,...,n\}$ et l_{ij} la longueur de l'arc (i,j) de A. On part du sommet 1.
- On définit $\pi^\star(i)$ comme la longueur minimum des chemins du sommet 1 au sommet i. Notons $\pi^\star(1)=0$
- L'algorithme procède en $n-1$ itérations : au début de chaque itération, les sommets sont partagés en S et $\bar{S}=X\setminus S$
- S contient les sommets « marqués »: les sommets pour lesquels $\pi(i)$ représente la longueur du plus court chemin entre 1 et i
- $\bar{S}$ contient les sommets ayant une marque provisoire: $\forall\, k\in\bar{S}:\pi(k)=\min_{i\in S\cap\Gamma_k^1}(\pi(i)+l_{ik})$
- A une étape quelconque si j est le sommet de marque provisoire $\pi(j)=\min_{k\in\bar{S}}(\pi(k))$ alors $\pi(j)=\pi^\star(j)$. C'est à dire que j peut être inclus dans S.

- On met à jour S incluant j et les marques provisoires de $k \in \bar{S}$ reliés par (j,k): on remplace donc $\pi(k)$ par $\pi(j)+l_{jk}$ chaque fois que $\pi(j)+l_{jk} < \pi(k)$
- Si l'on veut tracer le plus court chemin (pas seulement la longueur) on se rappelle dans un tableau les prédécesseurs selon le chemin.

Algorithme 4

(a)Initialisation
$\bar{S} = \{2,...,n\}$
$\pi(1) = 0$
$\pi(i) = \begin{cases} l_{ij} \text{ si } i \in \Gamma(1) \\ +\infty \text{ sinon} \end{cases}$

(b)Sélectionner j tel que
$\pi(j) = \min_{k \in \bar{S}} (\pi(k))$

Faire $\bar{S} \leftarrow \overline{S} \backslash \{j\}$
Si $\bar{S} = 0$ alors FIN
(c)Faire pour tout $i \in \bar{S} \cap \Gamma(j)$
$\pi(i) \leftarrow \min(\pi(i), \pi(j) + l_{ji})$
retourner en (b)

#Exemple#

Algorithme 2 : (Floyd)

On pourrait s'intéresser à la recherche de chemin entre deux sommets quelconque. Dans ce cas, l'algorithme de Dijkstra Moore n'est pas efficace. On préfère un algorithme se rattachant aux méthodes matricielles : l'algorithme de Floyd. Cet algorithme est particulièrement simple dans la mise en oeuvre.

- Notons $L = (l_{ij})$ la matrice $n \times n$ dont l_{ij} est la longueur de l'arc (i,j) s'il existe et ∞ sinon. Pour la diagonale on pose 0.
- Pour $1 \leq k \leq n$, notons $L^{(k)} = (l_{ij})$ la matrice $n \times n$ dont $l_{ij}^{(k)}$ est la longueur minimale d'un chemin de i à j dont les sommets intermédiaires sont dans $\{1,...,k\}$.
- Pour $k = 0$, on a $L^{(0)} = L$
- Relation entre les matrices : $l_{ij}^{(k)} = \min\left(l_{ij}^{(k-1)}, l_{ik}^{(k-1)} + l_{kj}^{(k-1)}\right)$

On va prouver cette dernière relation. On a deux cas à selon si le plus court chemin entre i et j emprunte k ou pas (prouver cela).

La matrice $L^{(n)}$, donnant l'ensemble des valeurs de plus courts chemins dans le graphe pourra donc être déterminée en n étapes de récurrence à partir de la relation. L'algorithme de Floyd recherche la matrice des plus courts chemins dans un graphe à valeurs positives.

Algorithme 5

Pour k de 1 à n
  Pour tout i et j de 1 à n faire
    $l_{ij} = \min(l_{ij}, l_{ik} + l_{kj})$

#exemple avec un graphe #

Réseaux, réseaux de transport et problèmes de flots

Définition [réseau]

Un graphe fortement connexe, sans boucle et ayant plus d'un sommet est appelé un *réseau*. On appelle *noeud* d'un réseau, un sommet qui a plus de deux arcs incidents.

- On appelle *noeud* d'un réseau, un sommet qui a plus de deux arcs incidents
- Les autres sommets sont appelés *antinoeuds*
- On appelle *branche* tout chemin pour lequel seuls les premiers et derniers sommets sont des noeuds

Définition [flot]

Soit $G = (V, A)$ graphe orienté, un *flot* est l'affectation d'une valeur réelle à chaque arc de G telle que:

- Cette valeur représente la quantité transportée sur cet arc
- En chaque sommet, la somme des flots entrants est égale à la somme des flots sortants (loi Kirchoff)

Problèmes classique : Flot maximale

- Chaque arc à une capacité maximale (borne supérieure du flot autorisé sur cet arc)
- Le problème du flot maximal consiste à déterminer un flot dont la valeur en un certain lieu est maximale
- On peut se donner un coût de transport d'une unité de flot sur chaque arc et chercher le flot maximal de coût minimal

Définition [réseau de transport]

On appelle un réseau de transport un graphe orienté antisymétrique évalue $G = (X, A, C)$, sans boucles et dans le quel il existe :

- un sommet x_1 sans prédécesseur ($\Gamma^{-1}(x_1) = \emptyset$) nommé *source*
- un sommet x_n sans successeur ($\Gamma(x_n) = \emptyset$) nommé *puits* du réseau

- un chemin (au moins) entre x_1 et x_n

Donnée la fonction de pondération C (supposée positive), $C(a)$ sera la *capacité* de l'arc a.

Définition [flot pour le réseau]

On désigne

- A_x^- : ensemble des arcs sortants de x
- A_x^+ : ensemble des arcs entrants de x

On dit que $\varphi(a) : A \to \mathbb{R}$ est un *flot pour le réseau* si :

- il est positif : $\varphi(a) \geq 0 \forall a \in A$
- il vérifie la loi de Kirchoff : $\sum_{a \in A_x^-} \varphi(a) - \sum_{a \in A_x^+} \varphi(a) = 0$ pour tout $x \neq x_1$ et $x \neq x_n$
- il ne dépasse pas la capacité des arcs : $\varphi(a) \leq C(a), \quad \forall a \in A$

Si x n'est ni x_1 ni x_n, la quantité entrante en x doit être égale à la quantité sortante. On désigne par φ_x telle quantité:
$\varphi_x = \sum_{a \in A_x^-} \varphi(a) = \sum_{a \in A_x^+} \varphi(a)$

Définition [valeur du flot pour le réseau]

Si φ est un flot sur un réseau G, alors on a $\varphi_{x_1} = \varphi_{x_n}$. Cette quantité s'appelle *valeur du flot*.

Problèmes classique : Recherche d'un flot complet

Définition [arc saturé -flot complet]

Pour un flot φ dans un réseau de transport $G = (X, A, C)$, on dit que un arc a est *saturé* si on a $\varphi(a) = C(a)$
Le flot est dit *complet* si tout chemin de x_1 à x_n contient **au moins** un arc saturé.

Augmentation du flot:

- Le graphe partiel engendré par les arcs non saturés par le flot (si le flot n'est pas complet) contient un chemin μ allant de l'entrée à la sortie
- Donc, on peut définir un nouveau flot pour le réseau en augmentant de 1 le flot de chacun des arcs constituant μ
- La value du flot est augmentée de 1
- On peut augmenter la value d'un flot incomplet jusqu'à ce qu'il soit complet
- Cependant, le flot obtenu n'est pas forcement le flot maximal (##exemple##)

Amélioration du flot

Soit φ un flot complet. On va utiliser une procédure itérative pour marquer tous les sommets du graphe où il est possible de faire transiter une unité de flot supplémentaire.

- On définit un processus d 'étiquetage. En x_1 on place une étiquette $+$. Soit x un sommet déjà marqué :
- On marque $+x$ tout successeur y non marqué de x pour lequel le flot n'est pas max $(\varphi(x,y) < (x,y))$.
- On marque avec une étiquette -x tout prédécesseur y non marqué de $x$pou lequel le flot n'est pas nul $\varphi(y,x) > 0$

L'étiquette a comme rôle d'indiquer si le flot peut être augmenté dans le sens du parcours $(+)$ ou diminué s'il est dans le sens contraire

- Si on arrive à marquer x_n, il existe un chemin μ dont tous les sommets sont marqués avec l'indice du sommet précédent au signe près.
- On est pas obligé de marquer tous les sommets pour avoir une augmentation

Soit alors :

$$\begin{cases} \varphi'(a) = \varphi(a) \text{ pour } a \notin \mu \\ \varphi'(a) = \varphi(a) + 1 \text{ pour } a \in \mu \text{ et } a \text{ dans le sens de } l'\text{orientation} \\ \varphi'(a) = \varphi(a) - 1 \text{ pour } a \in \mu \text{ et } a \text{ dans le sens opposé de } l'\text{orientation} \end{cases}$$

- On vérifie que φ' est encore un flot
- Comme $\varphi'_{x_n} = \varphi_{x_n} + 1$ la valeur du flot φ est supérieure, donc meilleure
- On systématise l'idée avec la notion de *réseau résiduel*.

Définition [réseau résiduel]

soit un réseau de transport $G = (X, A, C)$ possédant un flot complet φ. On appelle graphe résiduel, le réseau $\bar{G}(\varphi) = (X, \bar{A}, \bar{C})$ tel que :

- si $a \in A$ et $\varphi(a) < C(a)$ alors $a \in A$ et $\bar{C}(a) = C(a) - \varphi(a)$
- si $a = (x,y) \in A$ et $\varphi(a) > 0$ alors $a^{-1} = (x,y) \in \bar{A}$ et $\bar{C}(a^{-1}) = \varphi(a)$
- Le réseau résiduel indique le long de quels arcs on peut augmenter ou diminuer le flot.

Théorème [flot maximal et graphe résiduel]

Soit φ flot de G (de x_1 à x_n) et $\bar{G}(\varphi)$ le réseau résiduel associé à φ. Une condition nécessaire et suffisante pour que φ soit maximal est qu'il
n'existe pas de chemin de x_1 à x_n dans $\bar{G}(\varphi)$.
##Preuve ?##
1,5

Recherche d'un flot maximal : méthode de Ford-Fulkerson

Algorithme 6

RECHERCHE D'UN FLOT MAXIMUM

(a) Itération $k=0$

On part d'un flot $\varphi^0=(0,\cdots,0)$

(b) A l'itération $k>0$, soit φ^k le flot courant.

Rechercher un chemin μ^k de x_1 à x_n dans le réseau résiduel $\bar{G}(\varphi^k)$. S'il n'existe pas, FIN: le flot est maximal.

(c) Soit ε^k la capacité résiduelle de μ^k (min capacités résiduelles des arcs du chemin). On définit φ^{k+1} par :

$$\begin{cases} \varphi_a^{k+1}=\varphi_a^k+\varepsilon^k \\ \varphi_a^{k+1}=\varphi_a^k-\varepsilon^k \end{cases}$$

à selon de l'orientation de a par rapport au sens μ

Faire $k\leftarrow k+1$ et retourner en (b).

- Par définition de réseau résiduel, les flots φ^k sont compatibles avec les capacités
- Par ailleurs ε^k est positif à chaque itération et donc la suite des flots est **strictement** croissante
- A chaque étape, on cherche une chaîne joignant x_1 à x_n, mais telle chaîne n'est pas **unique**

###Exemples ###

- L'algorithme peut être adapté au cas où les capacités des arcs ont des bornes inf non nulles (positives et négatives)
- En pratique, on associe à la plupart des réseaux de transport une notion de coût unitaire du transport:
- Pour chaque arc a on associe un nombre réel $w(a)$ représentant ce coût unitaire
- Le coût total d'un flot φ sera la somme des flots élémentaires

$W(\varphi)=\sum_{a\in A} w(a)\varphi(a)$

- On peut rechercher, parmi les flots à valeur maximale, celui à coût minimal (FF adapté algorithme de Busacker-Gowen)

Couplages

Les problèmes de couplages interviennent naturellement dans les applications de la théorie de graphes:

- Problèmes de tournées

- Détermination plus court chaînes dans un graphe non orienté
- Une classe de problèmes en programmation linéaire en nombre entiers

Définition [couplage]

Soit $G = (X, A)$ un graphe orienté, un *couplage* est un sous-ensemble d'arêtes $K \subset A$ tel que deux arêtes quelconque ne sont pas adjacentes.

Un sommet i est dit *saturé* par le couplage s'il existe une arête de K incidente à i. Un couplage qui sature tous les sommets de G est dit *parfait*.

###Exemple###

Il n'est pas toujours possible de trouver un couplage parfait. On appellera *couplage maximal* un couplage de cardinale maximale pour le graphe.

Le problème du couplage maximal

Définition[chaîne alternée]

Soit K un couplage de $G = (X, A)$:

- une *chaîne alternée* (relativement à K) est une chaîne élémentaire de G dont les arêtes appartiennent alternativement à K et $\bar{K}$
- une *chaîne alternée augmentante* (améliorante) est une chaîne alternée joignant deux sommets insaturés

Une chaîne alternée est une chaîne ou les traits sont fins et épais. #exemple#

Opération de transfert le long d'une chaîne alternée :

Étant donné $K \subset A$ couplage, considérons la chaîne alternée L dont chaque extrémité est un sommet insaturé (ou est telle que l'unique arête de K qui lui est incidente soit dans L) . On échange le rôle des arêtes fines et épaisses le long de L et on obtient un nouveau couplage K'.

Observation: *Une opération de transfert long une chaîne augmentante donne un couplage de cardinale augmentée de 1.*

#Exemple#

Théorème :

Un couplage K est maximal si et seulement s'il n'existe pas de chaîne alternée augmentante relativement à K.

Ce théorème montre que pour rechercher un couplage maximal, il suffit de savoir trouver une chaîne alternée augmentante relativement à un couplage K.

#dessin#

la procédure pour construire un couplage maximal K pour G est la suivante :

- Initialiser $K = 0$
- Trouver une chaîne améliorante K_a de K et effectuer le transfert le long de cette chaîne et remplacer K avec le nouveau couplage

- Répéter le point précédent

#construire une chaîne alternée (en fait arbre alterné)#

Couplage maximal et flot maximal

Soit $G = (X, A, \Gamma)$ un graphe biparti, on veut déterminer un couplage maximal.

- On construit un réseau de transport en ajoutant une source S liée à tous les sommets de X et une destination P liée à tous les sommets de Y
- On fixe à 1 toutes les capacités des arcs du réseau

#figure#

- On fait passer une unité de flot par ces arêtes et sur les arêtes vers S et P . On obtient bien un flot sur le réseau
- La loi de Kirchoff en un sommet x exprime le fait que il y a au plus une arête du couplage passant pour x
- Réciproquement, un flot sur ce réseau ne peut transporter plus qu'une unité de flot sur chaque arête
- Donc l'ensemble des arêtes transportant une unité de flot entre X et Y détermine un couplage sur G
- La valeur du flot est donc égale au nombre d'arêtes du couplage
- Le problème du couplage se ramène à la recherche du flot maximal (FF)

#Flot complet et maximal exemple#

1,5