

Présentation

lundi 30 août 2021 23:34

Roland GRAPPE grappe@lipn.fr

Algorithmique des graphes

Plan:

- Arbre couvrant de poids minimal
 - ↳ en profondeur
- Parcours
 - ↳ en largeur
 - ↳ Kosaraju-Shamir
 - ↳ composantes fortement connexes
- Plus court chemin
 - ↳ Bellman-Ford (tri topologique)
 - ↳ Dijkstra
 - ↳ Ford
- Flot Max - Coupe minimum
- Couplages
 - ↳ dans un graphe biparti
 - ↳ en général
- Mariages stables
- Voyageur de commerce
 - ↳ $\frac{3}{2}$ approximation de Christofides
 - ↳ NP-Completeness

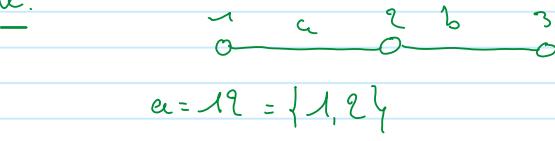
Chp 1 : Arbres couvrants

vendredi 3 septembre 2021 08:43

I - Définitions

Un graphhe non orienté $G = (V, E)$ est composé d'un ensemble V de sommets et d'un ensemble E d'arêtes, où une arête est une paire de sommets.

Exemple:



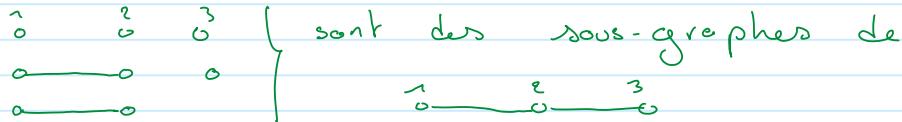
$$\begin{aligned}d(1) &= 1 \\d(2) &= 2 \\d(3) &= 1\end{aligned}$$

arêtes parallèles

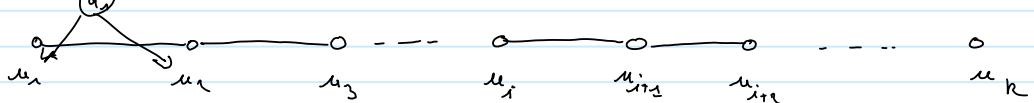
Un graphhe simple est un graphhe sans arêtes multiples. Le degré d'un sommet $d(v)$ est le nombre d'arêtes qui lui sont incidentes, $d(v)$.

Un sous-graphhe $H = (U, F)$ de G est un graphhe avec $U \subseteq V$, $F \subseteq E$ où les arêtes de F sont des sommets de U .

Exemple:

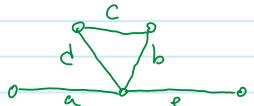


Une chaîne est une séquence d'arêtes a_1, \dots, a_k avec $a_i = u_i, u_{i+1}$.



Une chaîne est élémentaire si elle touche au plus une fois chaque sommet.

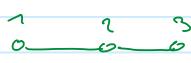
Exemple:



abcde n'est pas une chaîne élémentaire.

Un graphhe est connexe si toute paire de sommets est reliée par une chaîne.

Exemple:



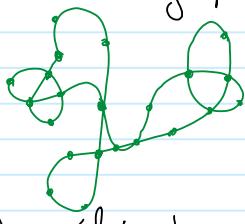
connexe



non connexe

Un cycle est un graphhe connexe où tous les degrés sont paires.

Exemple:

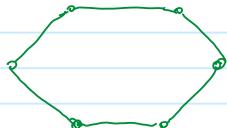


Un cycle est élémentaire si tous les degrés sont 2.

Exemple:

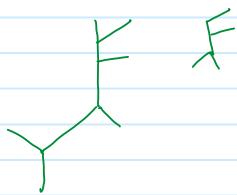


Exemple:



Une forêt est un graphe sans cycle

Exemple:



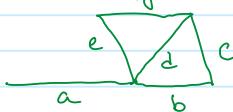
Un arbre est un graphe connexe sans cycle

Exemple:



Un arbre couvrant d'un graphe $G = (V, E)$ est un arbre contenant tous les sommets de G .

Exemple:



$\{c\}$ $\{a, e, b\}$ $\{$ arbres non couvrants

$\{a, c, d, f\}$ est un arbre couvrant

II- Arbre couvrant de poids minimum (Minimum Spanning Tree)

Entrée: Un graphe connexe non orienté $G = (V, E)$ avec un poids c_e sur chaque arête $e \in E$

Sortie: Un arbre couvrant de G , de poids minimum (le poids de l'arbre étant la somme des poids de ses arêtes qui le composent)

Kruskal:

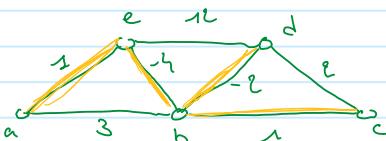
Etape 1. Faire la liste des arêtes de G dans l'ordre croissant de leur poids; a_1, \dots, a_m ($c_{a_1} \leq \dots \leq c_{a_m}$)

Etape 2. Soit $H = (\emptyset, \emptyset)$

Pour $i = 1, \dots, m$ -

| Si a_i à H ne crée pas de cycle: on ajoute a_i à H .
| Sinon, on jette a_i .

Exemple:



- ① be, bd, ae, bc, cd, ab, de
② ✓ ✓ ✓ ✓ ✗ ✗ ✗

{be, bd, ae, bc} forme un arbre couvrant de poids min de G , son poids est 4.

Remarque: Combien d'arête comporte un arbre couvrant? $|V|-1$

Remarque: C'est un algo glouton qui fournit toujours une solution optimale, ce

Remarque: C'est un algorithme glouton qui fournit toujours une solution optimale, ce qui n'est pas le cas de tous les algorithmes.

Le cadre général dans lequel un algorithme glouton marche toujours est quand les objets sur lesquels on optimise (ici les arbres courrants) sont des indépendants d'un matroïde.

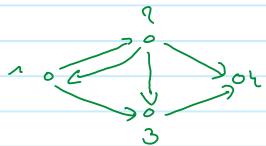
Chp 2 : Parcours

vendredi 3 septembre 2021 10:46

I - Définitions

Un graphe orienté $\mathbb{D} = (\mathcal{V}, \mathcal{A})$ est composé de sommets et d'arcs, un arc étant un couple de sommets.

Exemple:



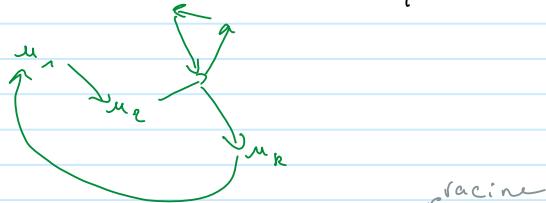
$$\mathcal{A} = \{2, 3\} = 83$$

Un chemin est une séquence d'arcs a_1, \dots, a_k avec $a_i = u_i, u_{i+1}$. Il est élémentaire s'il touche chaque sommet au plus 1 fois.

Exemple:
 a_1, a_2, a_3, a_4, a_5 est élémentaire.

Un circuit est un chemin a_1, \dots, a_k tel que $a_1 = u_1, u_2$ et $a_{k+1} = a_1$. Il est élémentaire s'il touche chaque sommet au plus 1 fois.

Exemple:



Une arborescence ressemble à , c'est-à-dire que sans les orientations, c'est un arbre, contenant un sommet racine à partir duquel sont orientées tous les arcs.

II - Parcours en profondeur d'abord (DFS : Depth Search First)

Routine explore:

Entrée: Un graphe orienté $\mathbb{D} = (\mathcal{V}, \mathcal{A})$ et un sommet $w \in \mathcal{V}$

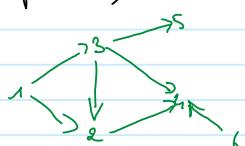
Sortie: Une arborescence des sommets w tel qu'il existe un chemin de v à w

Explore (\mathbb{D}, v)

Tant qu'il existe un successeur w de v non marqué

|
Flagger w
Explore (\mathbb{D}, w)

Exemple:



Explore ($\mathbb{D}, 1$)

Sommet marqué: 1, 2, 4, 3, 5

} on dessine les sommets visités de gauche à droite

Remarque: Souvent, une règle de priorité est spécifiée pour les cas d'égalités. Par exemple, s'il y a le choix, prendre le plus petit.

Remarque: Souvent, une règle de priorité est spécifiée pour les cas d'égalités.
Par exemple, si l'y a le choix, prendre le plus petit.

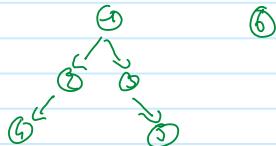
DFS (D):

Entrée: Un graphe orienté $D = (V, A)$

Sortie: Tant qu'il existe un sommet non marqué v :

Marquer v
Explorer (D, v)

Exemple: DFS (D)



A quoi ça sert?

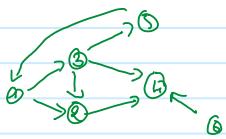
① Déterminer les sommets "atteignables" à partir d'un sommet v donné, ainsi qu'un chemin reliant v à chacun de ces sommets.

② A détecter la présence de circuits.

Etant donné une solution issue d'un DFS de D (c'est-à-dire un ensemble d'arbres disjoints). On classe les arcs de D comme suit:

- Les arcs d'arborescence (ceux qui sont dans la solution)
- Les arcs sortant v est un arc sortant si: il existe un chemin de u à v dans l'arborescence.
- Les arcs arrivant v est un arc arrivant si: il existe un chemin de v à u dans l'arborescence.
- Les arcs croisés sont les autres.

Exemple:



Type d'arcs:

- arcs d'arborescence : 12, 24, 13, 35
- — sortant : 14
- — arrivant : 51
- — croisés : 32, 34, 64

Remarque: Il existe un arc arrivant si le graphe contient un circuit.

Complexité: $O(|V| + |A|)$

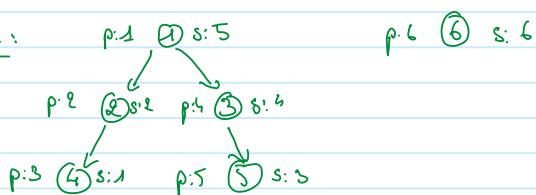
III - Kosaraju-Shamir

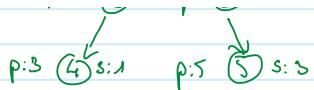
Etant donné un parcours de D :

- L'indice préfixe d'un sommet lui est attribué la première fois qu'on le rencontre lors du parcours, c'est-à-dire que l'on numérote les indices par ordre d'apparition.
- L'indice suffixe d'un sommet lui est attribué lorsque l'on a fini de l'explorer, c'est-à-dire qu'on numérote les sommets par ordre de disparition.

Ces indices se retrouvent de manière graphique en faisant le tour du parcours de gauche à droite

Exemple:





Une composante fortement connexe (c.f.c) d'un graphe orienté est un sous-graphe maximal dans lequel, pour toute paire de sommets u et v , il y a un chemin de u à v et un chemin de v à u .

Lorsque D est lui-même une c.f.c, on dit que D est fortement connexe.

Exemple: fortement connexe

non, ses c.f.c sont {1,4} et {2,3} du graphe réduit

fortement connexe

Le graphe réduit est celui obtenu en contractant chaque cfc.

Kosaraju - Shämír

Entrée: D graphe orienté

Sortie: les cfc de D

Algo:

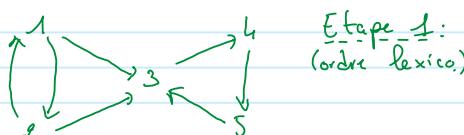
Etape 1: Faire un parcours en profondeur de D (retenir les indices suffixes de chaque sommet)

Etape 2: Construire le graphe D' obtenu à partir de D en inversant chaque arc.

Etape 3: faire un parcours en profondeur de D' en prenant toujours (quand on a le choix) le sommet de plus grand indice suffixe

Les arborescences obtenues après l'étape 3 sont les c.f.c. de D .

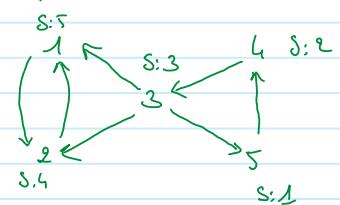
Exemple:



Etape 1:
(ordre lexico.)

- ① s:5
- ② s:4
- ③ s:3
- ④ s:2
- ⑤ s:1

Etape 2: D'



Etape 3:



Les cfc de D sont {1,2} et {3,4,5}

Son graphe réduit est:



Chp 3 : Plus courts chemins

mardi 14 septembre 2021 08:43

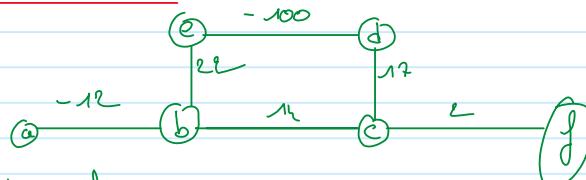
Etant donné un graphe orienté $D = (V, A)$, mun. de poids sur chaque arc. Un plus court chemin entre u et v est un chemin de u à v minimisant

$$\min \left\{ \sum_{a \in P} c_a : P \text{ chemin de } u \text{ à } v \right\}$$

Poids du chemin P

I - Condition d'existence

Exemple :



Plus court chemin de a à f ?
Il n'en existe pas, car $b \rightarrow c \rightarrow d \rightarrow e \rightarrow a$ est un circuit de poids strictement négatif.

On l'appelle circuit absorbant.

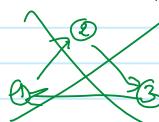
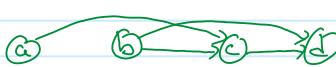
On peut définir les plus courts chemins à condition qu'il n'existe pas de circuit absorbant.

II - Cas sans circuit

① Tri topologique

Un tri topologique d'un graphe orienté $D = (V, A)$ est un ordre sur les sommets v_1, \dots, v_n tel que si (v_i, v_j) est un arc de A , alors $i < j$.

Exemple



Théorème: Un graphe orienté admet un tri topologique si et seulement s'il ne contient pas de circuit.

Remarque: Dans un graphe sans circuit, il existe au moins un sommet sans préédécesseur.

Preuve: Si tout le monde avait un préédécesseur, alors on trouverait un circuit en les remontant (car $|V| < \infty$)

Algorithm: Tri Topo

Entrée: Un graphe orienté $D = (V, A)$

Sortie: Un tri topologique de D si l'en existe, ou bien un certificat d'absence de circuit.

Initialisation: $D = (V, E)$, $T = \emptyset$

Tant qu'il existe un sommet sans préédécesseur dans $D \setminus T$:
on ajoute u à T

Termination: Si T contient tous les sommets de D à la fin, T est un tri Topo. Sinon, il existe un circuit.

Terminaison: Si T couvre tous les sommets de D à la fin, T est un tr \acute{e} Topo. Sinon, il existe un circuit.

Complexité: $O(|V| + |A|)$

② Algorithme de Bellman

Entrée: Un graphe orienté $D = (V, A)$, un sommet u , des poids sur les arcs $c: A \rightarrow \mathbb{R}$

Sortie: les plus courts chemins de u vers chacun des autres sommets ou l'existence d'un circuit.

Etape 1: Faire un tr \acute{e} Topo

S'il existe un circuit, on s'arrête.

Sinon: soit v_1, \dots, v_n le tr \acute{e} topo (on peut supposer $u = v_1$, quitte à supprimer les sommets avant u dans le tr \acute{e} topo).

Etape 2: Initialisation: $\lambda(v_i) = \infty$

Itération: Pour $i=2 \dots n$

$$\lambda(v_i) = \min_{\substack{v_j \text{ précédent} \\ \text{de } v_i}} \{ \lambda(v_j) + c(v_j, v_i) \}$$

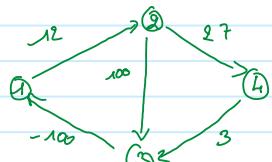
Terminaison: à la fin, $\lambda(v_i)$ contient la valeur d'un plus court chemin de u à v_i .

Remarque: On a gardé que les valeurs. Si on veut aussi la liste des arcs utilisés par les plus courts chemins, il suffit de retenir à chaque étape i l'indice k tel que:

$$\lambda(v_k) + c(v_k, v_i) = \min_{v_j \text{ précédent de } v_i} \{ \lambda(v_j) + c(v_j, v_i) \}$$

Le graphe construit de ces arcs est l'arborescence des plus courts chemins à partir de u .

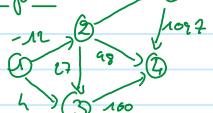
Exemple:



Les plus courts chemins à partir de ①:

Chaque sommet a un précurseur; il y a 1 circuit.
L'algo de Bellman ne s'applique pas.

Exemple: PCC à partir de ① :



Etape 1: Tr \acute{e} Topo 1 2 3 5 4

Etape 2	Sommet	1	2	3	5	4
	λ	0	-12	4	27	-915
	pred	/	1	1	2	5

Arborescence des plus courts obtenue:



Complexité: $O(|V|^2)$ car chaque sommet a au plus $|V|$ précurseurs et qu'on fait $|V|$ étapes.

$O(|V| + |A|)$ après une analyse plus fine? Selon la représentation

III - Poids positifs ou nuls

Algorithme de Dijkstra

III - 'Tôds positifs ou nuls

Algorithme de Dijkstra

Entrée: Un graphe orienté $\mathcal{D} = (\mathcal{V}, \mathcal{A})$, un sommet u dans \mathcal{V} , et un poids sur chaque arc, positif ou nul $c: \mathcal{A} \rightarrow \mathbb{R}_+$

Sortie: Les plus courts chemins de u vers chaque autre sommet.

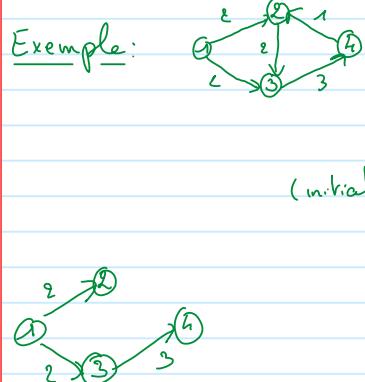
Initialisation: $\lambda(u) = 0$; $\lambda(v) = +\infty$ pour tout $v \neq u$

Itération: Tant qu'il existe un sommet non marqué.

Soit v non marqué ayant la plus petite valeur $\lambda(v)$ parmi les sommets non-marqués.

- Marquer v
- Pour tout successeur non marqué w de v :
 $\lambda(w) = \min\{\lambda(w), \lambda(v) + c(v,w)\}$

Exemple:



PCC à partir de ①

Sommets marqués	Sommets courants	1	2	3	4
(initialisation)	①				
①	②		2	2	$+\infty$
①, ②	③		2	2	5
①, ②, ③	④				5

Complexité: $\mathcal{O}(N^2)$

IV - Cas général

Algorithme de Ford-Bellman

Entrée: Un graphe orienté $\mathcal{D} = (\mathcal{V}, \mathcal{A})$, $u \in \mathcal{V}$, $c: \mathcal{A} \rightarrow \mathbb{R}$ des coûts quelconques

Sortie: Les plus courts chemins de u vers chaque autre sommet, ou bien l'existence d'un circuit absorbant.

Remarque: Si on a calculé les plus courts chemins de u vers les autres sommets, ayant au plus k arcs, un chemin d'au plus $k+1$ arcs = un chemin d'au plus $k+1$ arcs + éventuellement 1 arc

Initialisation: $\lambda(u) = 0$, $\lambda(v) = +\infty \forall v \neq u$

Itération: Pour i de 1 à $|N|$:

Soit S l'ensemble des sommets dont la valeur a changé de sommet à l'étape $(i-1)$. (Seul u a changé de valeur à l'étape 0)

► On met à jour les valeurs des successeurs des sommets de S :

Si w est successeur de s :

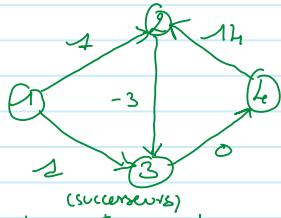
$$\lambda(w) = \min\{\lambda(w); \lambda(s) + c(s,w)\}; s \text{ précurseur de } w \text{ dans } S\}$$

Termination: Si les valeurs à l'itération n sont les mêmes qu'à l'itération $n-1$, alors ce sont les valeurs des plus courts chemins. Sinon, il existe un

Terminaison: Si les valeurs à l'itération n sont les mêmes qu'à l'itération $n-1$, alors ce sont les valeurs des plus courts chemins. Sinon, il existe un circuit absorbant.

Rémarque: Si en cours d'algo, la ligne i et la ligne $i+1$ sont identiques, on peut s'arrêter, ce sont les valeurs des plus courts chemins.

Exemple:

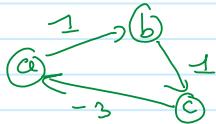


(init)	i	S	$\Gamma^+(S)$	1	2	3	4
	0	/	/	0	$+\infty$	$+\infty$	$+\infty$
étape 1	1	2, 3	2, 3	0	1	1	$+\infty$
étape 2	2, 3	3, 4	3, 4	0	1	-2	1
étape 3	3, 4	2, 4	2, 4	0	-1	-2	-2
étape 4	4	2	2	0	1	-2	-2

des valeurs à l'itération 4 sont les mêmes qu'à l'itération 3. Ce sont les valeurs des plus courts chemins.

Exemple: PCC (a)

i	S	$\Gamma^+(S)$	a	b	c
0	/	/	0	$+\infty$	$+\infty$
1	a	b	0	1	$+\infty$
2	b	c	0	1	2
3	c	a	-1	1	0



les 2 lignes sont égales \Rightarrow il y a 1 circuit absorbant.

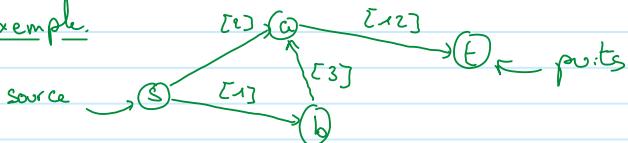
Chp 4 : Flots

mardi 21 septembre 2021 09:08

I - Definitions

Un réseau de transport est un graphe orienté $D = (V, A)$ où les arcs sont munis d'une capacité $c: A \rightarrow \mathbb{R}_+$, avec une source $s \in V$ (sans prédecesseurs) et un puits $t \in V$ (sommet sans successeurs)

Exemple.



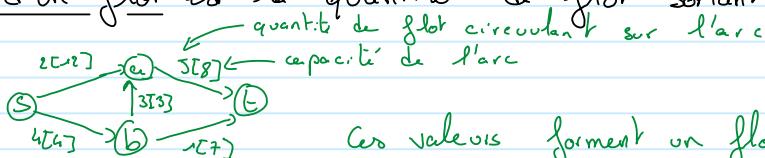
Notation: Etant donné un ensemble de sommets $X \subseteq V$.
 $S^+(X)$ est l'ensemble des arcs "sortants" de X .
 $S^-(X)$ est l'ensemble des arcs "entrants" dans X .

Exemple: $S^+(s, a) = \{ sb, at \}$ $S^-(s, a) = \{ ba \}$

Un flux f est la donnée d'une valeur f_a sur chaque arc dans A telle que:
 $f_a \geq 0$ pour tout $a \in A$ (non négativité)
 $f_a \leq c_a$ pour tout $a \in A$ (respect des capacités)
 $\sum_{a \in S^+(s)} f_a = \sum_{a \in S^-(t)} f_a, \forall u \neq s, t$ (loi des noeuds de Kirchhoff, conservation du flux)

La valeur d'un flux est la quantité de flux sortant de la source.

Exemple:



Ces valeurs forment un flux de valeur 6

Remarque: La valeur du flux est égale à la quantité de flux entrant dans le puits.

Problématique: Déterminer un flux de valeur maximum. (On l'appellera flux maximum)

II - Comment trouver de bonnes bornes supérieures ?

Pour un ensemble de sommets contenant la source s mais pas le puits t , $S^+(X)$ est une st-coupe de capacité la somme des capacités des arcs qui la composent.

Exemple: $S^+(s)$ est une st-coupe de capacité 16
 $S^+(s, a)$ est une st-coupe de capacité 12.

Remarque. Si l'on supprime les arcs d'une st-coupe, il n'y a plus de chemins de s à t .

Consequence: La valeur d'un flux est toujours inférieure à la capacité de toute st-coupe.

Consequence: Si un flux a pour valeur la capacité d'une st-coupe, alors il est maximum.

Combien y a-t-il de st-coupes ?

est maximum.

Combien y a-t-il de st-coupes?

Auivant que de sous-ensembles de $V \setminus \{s, t\}$: 2^{N-2} .

III - Flot Flav - Coupe min

1) Chaîne augmentante

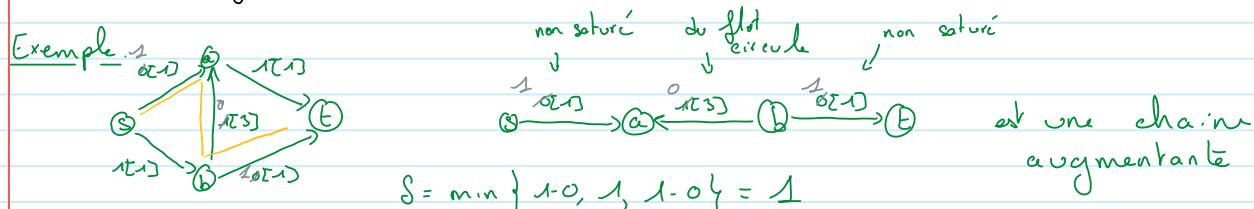
arc saturé; la valeur du flot circulant sur l'arc est égale à la capacité de l'arc.

Une chaîne de s à t est augmentante si, pour un flot donné f :
(non saturé) $f_{st} < c_{st}$ pour tout arc parcouru dans le bon sens de s à t
(du flot circule sur l'arc) $f_{st} > 0$ mauvais

Etant donné une chaîne augmentante pour un flot f , soit

$$S = \min \begin{cases} c_{st} - f_{st} & \text{pour les arcs dans le bon sens} \\ f_{st} & \text{mauvais} \end{cases}$$

On peut augmenter la valeur de f de S en faisant:
 $f_{st} + S$ sur les arcs dans le bon sens
 $f_{st} - S$ mauvais



O(iv) Algorithme: MARQUAGE

Entrée: Un réseau de transport $D = (V, A)$, $c: A \rightarrow \mathbb{R}_+$, et éventuellement un flot f .

Sortie: Une chaîne augmentante si il en existe et une st-coupe minimum

Initialisation: Marquer s du label +

Itération: Tant qu'il existe un sommet u marqué non traité

■ Traiter u :

- Pour tout successeur v de u non marqué avec (u, v) non saturé:
Marquer v du label $+u$ ($f_{uv} < c_{uv}$)
- Pour tout prédécesseur w de u non marqué avec (w, u) transportant du flot ($f_{wu} > 0$):
Marquer w du label $-u$

Si le puits est marqué lors du MARQUAGE, alors il existe une chaîne augmentante. Pour la trouver, il suffit de remonter les labels à partir du puits. Sinon, le flot est maximum, et l'ensemble des sommets marqués forme une st-coupe de capacité minimum.

FORD-FULKERSON (1956)

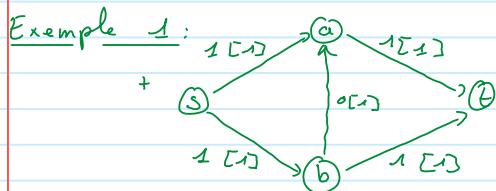
$O(F \times |V|)$

Entrée: Un réseau de transport et un flot

Sortie: Un flot maximum et une st-coupe de capacité minimum

Sortie: Un flot maximum et une st-coupe de capacité minimum

Tant que le puits est marqué lors du MARQUAGE
Augmenter la valeur du flot grâce à la chaîne correspondante



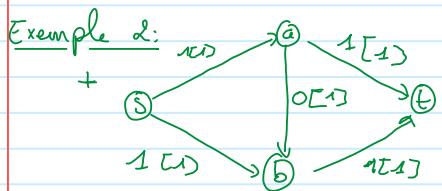
Itération 1: $\begin{matrix} \textcircled{S} & \xrightarrow{0[1]} & \textcircled{C} & \xrightarrow{2[2]} & \textcircled{T} \end{matrix}$

$$S = \min\{1-0, 1-0\} = 1$$

Itération 2: $\begin{matrix} \textcircled{S} & \xrightarrow{0[1]} & \textcircled{B} & \xrightarrow{0[1]} & \textcircled{T} \end{matrix}$

$$S = \min\{1-0, 1-0\} = 1$$

Itération 3: le flot est maximum et $S^*(\{s\}) = \{s, b\}$ est une st-coupe de capacité minimum égale à 1



Itération 1: $\begin{matrix} \textcircled{S} & \xrightarrow{0[1]} & \textcircled{B} & \xrightarrow{1[1]} & \textcircled{A} & \xrightarrow{0[1]} & \textcircled{T} \end{matrix}$

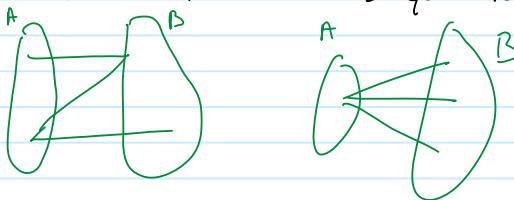
$$S = \min\{1-0, 1-0\} = 1$$

Itération 2: le flot est maximum et $S^*(\{s\}) = \{s, b\}$ est une st-coupe de capacité minimum égale à 1

IV - Application - Couplage

Un graphe non orienté $G = (V, E)$ est dit biparti si ses sommets peuvent être partagés en A et B tels que toutes les arêtes sont entre A et B.

Exemple:



Un couplage est un ensemble d'arêtes qui couvre au plus une fois chaque sommet

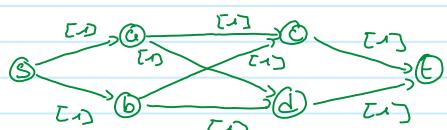
Exemple: a c $\emptyset, \{bc\}, \{bc, ad\}$
b d

Problème: Trouver un couplage (de cardinalité) maximale.

On modélise ce problème comme un problème de flot max.

On oriente tous les arcs de A vers B, on ajoute :

- une source s et tous les arcs $sa, a \in A$
- un puits t ————— bt, $b \in B$



les arcs ab, $a \in A, b \in B$ ayant la valeur 1 dans un flot max correspondent à un couplage dans le graphe de départ.

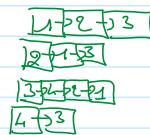
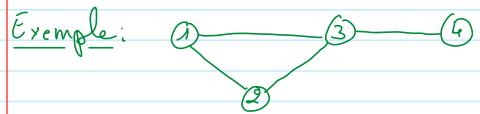
Chp 5 : Représentation des graphes

vendredi 24 septembre 2021 08:41

I - Graphes non-orientés

1) Liste d'adjacence

Le graphe $G = (V, E)$ est représenté par $|V|$ liste chaînes : la liste v ($v \in V$) contient les voisins de v .



Taille : $O(|V| + |E|)$

2) Matrice d'adjacence

La matrice d'adjacence d'un graphe $G = (V, E)$ est la matrice A de $\{0, 1\}^{|V| \times |V|}$ telle que :

$$A_{v,w} = 1 \Leftrightarrow vw \in E$$

Exemple :

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 1 & 0 \end{matrix}$$

Remarque : Cette matrice est symétrique ($A = A^T$)

Taille : $|V|^2$

3) Matrice d'incidence

La matrice d'incidence d'un graphe $G = (V, E)$ est la matrice M de $\{0, 1\}^{|V| \times |E|}$ telle que :

$$M_{v,e} = 1 \Leftrightarrow e \text{ est incidente à } v$$

Exemple :

$$\begin{matrix} & 1_2 & 1_3 & 2_3 & 3_4 \\ 1 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 1 & 1 \\ 4 & 0 & 0 & 0 & 1 \end{matrix}$$

Taille : $|V| \times |E|$

Remarque : Cette représentation est moins utilisée algorithmiquement.

Exercice. M^T est dans $\mathbb{R}^{|V| \times |V|}$

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ 1 & 2 & 1 & 1 & 0 \\ 2 & 1 & 2 & 1 & 0 \\ 3 & 1 & 1 & 3 & 1 \\ 4 & 0 & 0 & 1 & 1 \end{matrix}$$

Remarque : Sur la diagonale \rightarrow les degrés

$$M^T M = \sum_{k=1}^n M_{i,k} M_{k,j} = \sum_{e \in E} M_{i,e} M_{j,e}$$

$$M^T M = A + D$$

Remarque. $\{x \in \{0, 1\}^N : M^T x \leq 1\}$ est l'ensemble des vecteurs d'incidences des stables de G
ensemble des sommets deux à deux non adjacents

$$M^T x \leq 1 \Leftrightarrow x_i + x_j \leq 1 \quad \forall uv \in E$$

II - Graphes orientés

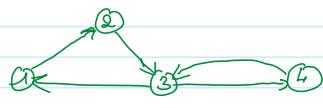
$$1 \geq x_i < 1 \Rightarrow x_{in} + x_{out} \leq 1 \text{ pour } e$$

II - Graphes orientés

1) Listes d'adjacence

Un graphe orienté $D = (V, E)$ est représenté par $|V|$ listes chaînées où la liste chaînée associée à un sommet v est la liste de ses successeurs.

Exemple.



Variante: ① liste des prédécesseurs à la place des successeurs

② Chaque sommet a deux listes: ses successeurs et ses prédécesseurs

Remarque: représentation adaptée pour Bellman

2) Matrice d'adjacence

La matrice d'adjacence d'un graphe orienté $D = (V, A)$ est la matrice B de $10, 1 \times |V| \times |V|$ avec $B_{uv} = 1 \Leftrightarrow (u, v) \in A$

Exemple.

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 & 0 \end{matrix}$$

Remarque: B est symétrique si et seulement si $(u, v) \in A \Leftrightarrow (v, u) \in A$

Taille: $|V|^2$

Exercice: Que contient B^2 ? Dans l'exemple:

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 \\ 4 & 0 & 0 & 0 & 1 \end{matrix}$$

$$B^2_{uv} = \sum_{w \in V} B_{uw} B_{vw}$$



$(B^2)_{uv} \neq 0 \Leftrightarrow$ il existe un chemin de longueur 2 de u à v .

3) Matrice d'incidence

La matrice d'incidence d'un graphe orienté $D = (V, A)$ est la matrice M de $10, 1 \times |A|$ telle que:

$$M_{u,a} = \begin{cases} 1 & \text{si } \xrightarrow{a} \text{l'arc sort de } u \\ -1 & \text{si } \xrightarrow{a} \text{l'arc entre en } u \\ 0 & \text{si } \xrightarrow{a} \text{l'arc ne touche pas } u \end{cases}$$

Exemple:

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 2 & -1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & -1 & 1 & 1 & -1 & 0 \\ 4 & 0 & 0 & 0 & -1 & 1 & 0 \end{matrix}$$

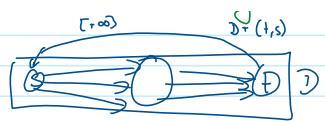
Remarque: Soit $x \in \mathbb{R}^{|A|}$ tel que $Mx = 0$

$$\begin{cases} x_{12} = x_{32} \\ x_{21} = x_{23} \\ x_{32} + x_{34} = x_{23} + x_{43} \\ x_{34} = x_{43} \end{cases}$$

Conservation du flux!



flux de $s \rightarrow t \Leftrightarrow f \geq 0$ dans $D + (+, s)$ tel que $\begin{cases} f \leq c \\ Mf = 0 \end{cases}$



flot de sat $\Leftrightarrow f \geq 0$ dans $D + (t,s)$ tel que $\begin{cases} f \leq c \\ af = 0 \end{cases}$

Chp 6 : Mariages stables

vendredi 24 septembre 2021 13:07

Problème : affectation étudiants - hôpitaux
APB, Parcoursup

Prix Nobel : 2012 Roth & Shapley
Applications → ↗ algorithme avec Gale en 1962

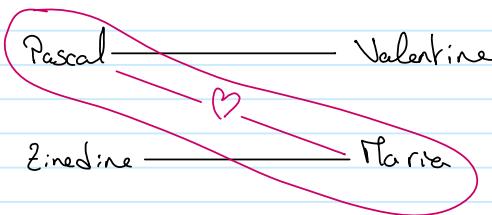


I - Définitions

On a n hommes et n femmes. Chaque personne classe les personnes de l'autre sexe par ordre de préférence.

But: former n couples (homme-femme) de la "meilleure" manière possible.
C'est ce qu'on appelle un mariage

Imaginons que Pascal soit en couple avec Valentine, que Zinedine soit en couple avec Maria mais Pascal préfère Maria à Valentine et Maria préfère Pascal à Zinedine.



Pascal et Maria se préfèrent à leurs conjoints respectifs

La paire Pascal-Maria est dite instable.

But: Trouver un mariage stable (c'est-à-dire sans paire instable)

II - Algorithme de Gale-Shapley (1962)

Entrée: n listes de préférences des hommes
 n listes de préférences des femmes

Sortie: Un mariage stable

Chaque matin, chaque homme propose à l'actuelle favorite de sa liste. L'après-midi, chaque femme répond "peut-être" à sa proposition préférée et NON aux autres. Le soir, chaque homme ayant reçu NON rase le nom de la femme correspondante de sa liste. Et on recommence jusqu'à ce que chaque homme ait reçu "peut-être".

III - Correction de l'algorithme

L'algorithme termine:

Chaque soir, sauf si tous les hommes reçoivent "peut-être", au moins un homme a reçu NON donc au moins un nom a été rayé dans la liste des hommes: il y a n listes de taille n , donc ça termine au pire en n^2 jours.

Remarque: Si Valentine répond "peut-être" à Zinedine, alors pendant toute la suite de l'algo Valentine sera avec un homme qu'elle aime au moins autant que Zinedine. En particulier, elle va finir avec un tel homme.

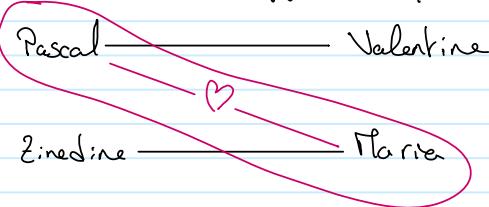
L'algorithme termine avec un mariage.

L'algorithme termine avec un mariage.

Raisonnons par l'absurde et supposons que Pascal ne soit pas en couple une fois l'algorithme terminé : alors il a demandé à toutes les femmes et elles lui ont toutes dit NON. Mais alors d'après la remarque, chaque femme a terminé avec un homme et il y en a un seul : Pascal. C'est absurde.

L'algorithme termine avec un mariage stable.

Par l'absurde supposons que :



Pascal-Maria soit une paire instable. En particulier Pascal préfère Maria à Valentine. Donc Pascal a proposé à Maria avant de proposer à Valentine, et Maria lui a dit NON. Donc Maria, d'après la remarque, a fini avec quelqu'un qu'elle préfère à Pascal, ce qui n'est pas le cas. Absurde.

IV - Asymétrie de l'algorithme

Question : échanger les rôles des hommes et des femmes a-t-il un impact sur la solution ?

Un homme et une femme sont partenaires valides s'il existe un mariage stable où ils sont ensemble.

Un mariage stable est dit homme(femme)-favorisateur si chaque homme (femme) termine avec sa (son) partenaire valide préférée.

Un mariage stable est dit homme(femme)-défavorisateur si chaque homme (femme) termine avec la/le pire de ses partenaires valides.

Théorème : L'algorithme de Gale-Shapley renvoie un mariage stable :

- homme-favorisateur,
- femme-défavorisateur.

Leçon de vie : mieux vaut proposer.

Éléments de preuve :

Supposons que Valentine et Pascal sont les premiers partenaires valides où V. a dit NON à Pascal (Alors V. a fini avec quelqu'un qu'elle préfère à P. c'est Zinedine.)

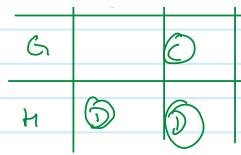
Comme ils sont valides, V et P sont en couple dans un autre mariage stable. Dans ce mariage, Z est en couple avec Maria. Alors Z préfère M. à V. : il lui a donc proposé avant de proposer à V, et elle lui a dit NON ayant que V ne dise NON à P.

Exemple :

Armand	$E > F > G > H$
Bilal	$F > H > G > E$
Chris	$X > G > H > F$
Didier	$H > G > E > F$
Erica	$B > A > C > D$

	jour 1	jour 2
E	(A)	(A)
F	(B)	(B)
G		(C)

Didier H > G > E > F
 Erica B > A > C > D
 Fanny A > B > D > C
 Gertrude D > C > B > A
 Hermione D > B > A > C



Exercice

A: S > T > U > V

B: X > X > X > T

C: X > U > T > V

D: X > V > T > S

S: D > A > B > C

T: A > D > C > B

U: C > D > B > A

V: A > C > D > B

jour	1	2	3	4	5
S	(A) B C	(A)	(A)	(A)	(A)
T					(B)
U	(D)	(C) D	(C)	(B) C	(C)
V		(B)	(B) D	(D)	(D)

Chp 7 : Voyageur de commerce (Traveling Salesman Problem)

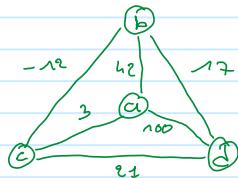
jeudi 30 septembre 2021 08:47

I - Problème et complexité

① Le problème du TSP

Dans un graphe non orienté complet où chaque arête a un poids, on cherche un tour (ou circuit Hamiltonien), qui est un cycle passant exactement une fois par sommet de poids minimum.

Exemple:



tour: ac, bc, bd, da de poids 108.
• ab, bd, dc, ca de poids 83

② Complexité

Le problème de décision associé au TSP est le suivant:

Étant donné $K_n = (V_n, E_n)$ (le graphe complet sur n sommets), $c: E_n \rightarrow \mathbb{R}$ et $k \in \mathbb{R}$, existe-t-il un tour de poids inférieur ou égal à k ?

Un problème de décision est dans la classe P s'il existe un algorithme polynomial qui répond à la question.

Exemple: mariage stable, flot max, plus courts chemins

Un problème de décision est donc de la classe NP s'il existe un certificat testable en temps polynomial que la réponse est oui.

Exemple: Si on nous donne un tour T , on peut en temps polynomial $O(n)$ que son poids est $\leq k$, et que c'est bien un tour (il suffit de parcourir le tour et de s'assurer qu'on croise tous les sommets exactement une fois). Donc TSP est dans NP.

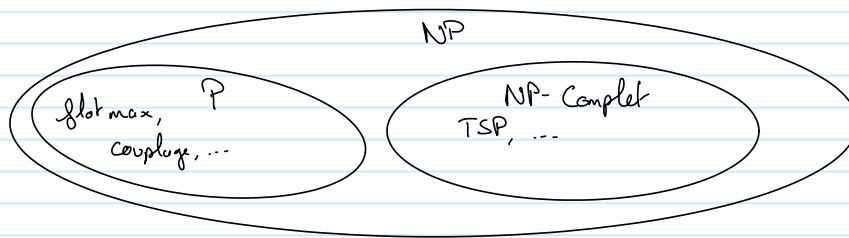
Remarque: $P \subseteq NP$

↳ L'algorithme polynomial de la classe P fournit un certificat polynomial.

Question à MI\$: $P = NP$?

Dans la classe NP, il y a un ensemble de problèmes tous (polynomiallement) équivalents entre eux: on les appelle problèmes NP-complets.

On ne connaît pas d'algorithme polynomial pour ces problèmes.



Le cribo actuel est que $P \neq NP$, ce qui impliquerait qu'il n'existe pas d'algorithme polynomial pour les problèmes NP-complets.

Théorème (Karp 1972):

Déterminer si un graphe donné contient un tour est NP-complet (problème du circuit Hamiltonien).

Consequence: Le TSP est NP-complet

↳ TSP contient circuit Hamiltonien, il suffit dans le graphe complet de mettre des poids 1 sur les arêtes du graphe dans lequel on cherche un circuit Hamiltonien, +∞ sur les autres: il existe un TSP de poids n si et seulement si le graphe de départ contient un tour.

arêtes du grapho dans lequel on cherche un circuit Hamiltonien, + ou sur les autres: il existe un TSP de poids n si et seulement si le grapho de départ contient un tour.

Def: Un problème est α -approximable si pour toute instance, on peut construire (en temps polynomial) une solution S dont le poids est au pire $\alpha \times$ le poids d'une solution optimale.

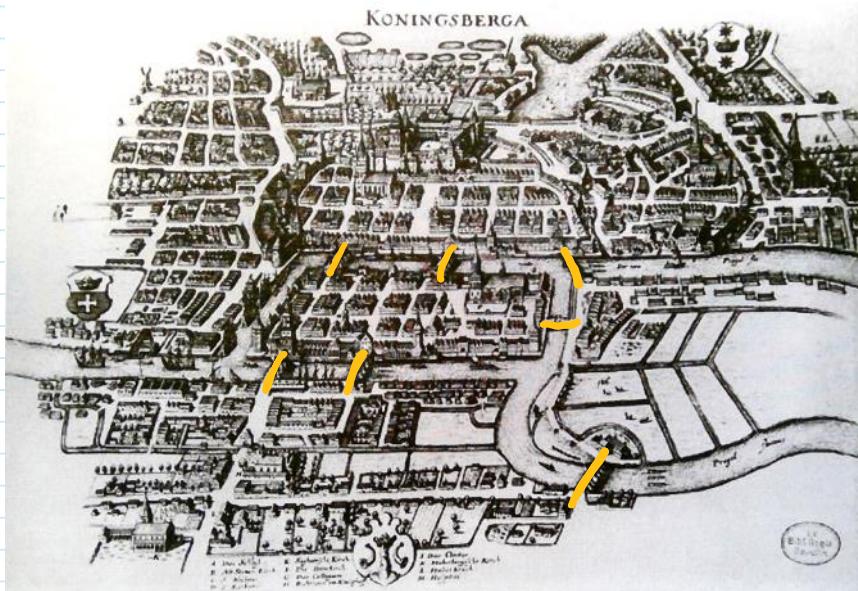
Théorème: Pour tout α , le TSP n'est pas α -approximable.

II - Metric TSP (TSP métrique)

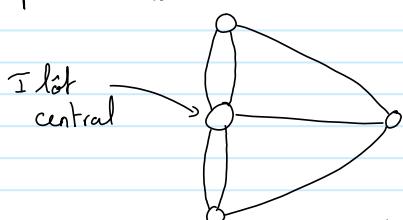
① Graphes euclériens

Enigme des ponts de Königsberg:

Dans les années 1700, les touristes de Königsberg se posaient le défi: de se promener dans la ville en traversant chaque pont exactement une fois en revenant à son point de départ à la fin.



Résolu par Euler en 1735 c'est impossible!
Il a représenté la ville:



et le problème devient: existe-t-il un circuit passant une fois et une seule fois par chaque arête?

Lorsque l'on parcourt un circuit, dès qu'on entre dans un sommet par une arête, on en repart par une autre pas encore utilisée: donc tous les degrés sont pairs ce n'est pas le cas de Königsberg

Def: Un grapho euclérien est un grapho connexe où tous les degrés sont pairs.

Remarque: Un grapho est euclérien si et seulement s'il existe un circuit passant exactement une fois par chaque arête.

Preuve: sens \Leftarrow voir énigme de Königsberg
sens \Rightarrow - - -

Consequence: Décider si il existe un circuit passant exactement une fois par arête est dans P.

Remarque: C'est la version "arête" du circuit Hamiltonien.

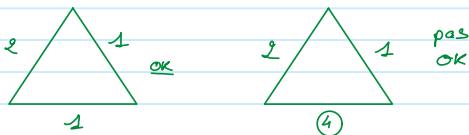
② Remarques préliminaires

Le TSP est dit métrique lorsque la fonction de poids $d: E \rightarrow \mathbb{R}_+$ vérifie l'inégalité triangulaire:

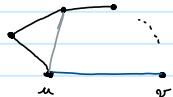
(2) Remarques préliminaires

Le TSP est dit métrique lorsque la fonction de poids $d: E_n \rightarrow \mathbb{R}_+$ vérifie l'inégalité triangulaire:
 $d(u,v) \leq d(u,w) + d(w,v) \quad \forall u,v,w \in V_n$

Exemple:



Consequence: Le plus court chemin entre u et v est toujours l'arête uv .



Remarque: Le TSP métrique est NP-complet.

Lemme: Soit $G=(V,E)$ un graphe eulérien avec $d:E_n \rightarrow \mathbb{R}_+$ une métrique (inégalité Δ satisfait) (possiblement E contient des arêtes multiples). Alors, il existe un tour T de poids inférieur ou égal à $\frac{d(A)}{2}$, le poids du circuit correspondant à G .

Preuve algorithmique:

On parcourt G en suivant un circuit qui passe par chaque arête et on note v_1, \dots, v_n l'ordre dans lequel les sommets de V_n sont rencontrés. Alors $T=v_1 \dots v_n$ est un tour: tous les sommets sont rencontrés car G est connexe. Son poids total est inférieur à celui de G car lorsque l'on prend l'arête $v_k v_{k+1}$, c'est un raccourci par rapport au chemin dans G , d étant une métrique.

Remarque: Soit A un arbre couvrant de poids min de K_n, d . Alors son poids est inférieur au poids d'un tour de poids min.

Preuve: Soit T un tour, alors $T \setminus e$ est un arbre couvrant pour tout $e \in T$, d'où $d(T) \geq d(T \setminus e) \geq d(A)$ (car $d > 0$).



(3) 2-approximation du TSP métrique

Théorème: Le TSP métrique est 2-approximable.

Preuve algorithmique:

Entrée: $K_n = (V_n, E_n)$, d une métrique
Sortie: Un tour T de poids $d(T) \leq 2 \text{ OPT}$

Etape 1: Soit A un arbre couvrant de poids min de K_n, d .

Etape 2: Soit $G = (V_n, E)$ le graphe eulérien obtenu en doublant chaque arête de A .

Etape 3: On applique le lemme et on obtient un tour de T

Analyse: $d(T) \leq d(E)$ (d'après le lemme) et $d(E) = 2d(A)$ par construction or $d(A) \leq d(\text{OPT})$. D'où $d(T) \leq 2 \text{ OPT}$.

(4) $\frac{3}{2}$ -approximation:

Déf. Un couplage est dit parfait lorsque il couvre tous les sommets du graphe.

Culture générale: Trouver un couplage parfait de poids min est dans P.

Théorème: Le TSP métrique est $\frac{3}{2}$ -approximable.

Algorithme de Christofides:

Entrée: $K_n = (V_n, E_n)$, d une métrique
Sortie: Un tour T de poids $d(T) \leq \frac{3}{2} \text{ OPT}$

Etape 1: Soit A un arbre couvrant de poids min de K_n, d .

Etape 2: Soit M un couplage parfait de poids min sur l'ensemble des sommets de degré impair dans A .

Etape 3: On applique le lemme dans $A+M$ qui est un graphe eulérien, et on obtient un tour T .

Etape 2: on va supposer que le poids min sur un arête entre deux sommets est impair dans A.

Etape 3: On applique le lemme dans $A+M$ qui est un graphe eulérien, et on obtient un tour T.

Analyse: $d(T) \leq d(A+M)$ d'après la lemme et $d(A+M) = d(A) + d(M)$.

Or $d(M) \leq \frac{1}{2} OPT$ car un tour optimum contient un couplage C de cardinalité max et de poids $d(C) \leq \frac{1}{2} OPT$ et $d(M) \leq d(C)$ par construction

Comme $d(A) \leq OPT$ on obtient $d(T) \leq \frac{3}{2} OPT$