(15)

• Analyse de l'algo :

① Terminaison : OK.

③ Complexité : ARBRE-COUVRANT + le tri : $O(n^2) + O(m \log m)$

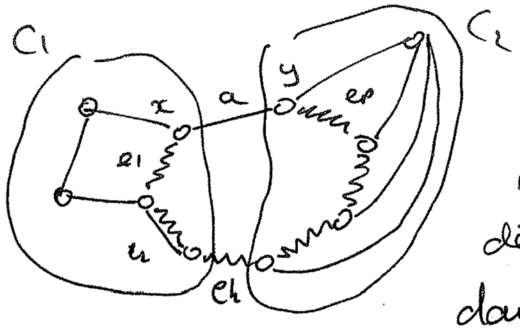
(Rq : $m \leq n^2 \Rightarrow \log m \leq \log(n^2) \Rightarrow \log m \leq 2 \log n$

et $O(m \log m) = O(m \log n)$.)

② Correction : Soit T l'arbre couvrant renvoyé par l'algo sur un graphe G connexe. On considère A un arbre couvrant G de poids minimal et ayant un nombre maximal d'arêtes en commun avec T.

- Si T est un A.C.P.M alors A = T et c'est fini.
- Sinon, on va aboutir à une contradiction :

choisissons $a = xy$ une arête de A qui n'appartient pas à T.



comme a n'a pas été choisie par l'algo, x et y étaient déjà relié par un chemin $e_1, e_2 \dots e_p$ dans T. Donc les e_i ont été traités avant a et : $\forall i = 1 \dots p$ $p(e_i) \leq p(a)$.

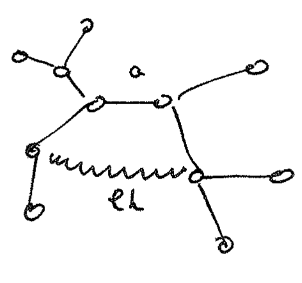
On va modifier A :

technique classique

$A \setminus a$ possède 2 composantes connexes : C_1 et C_2

P qui relie x et y dans T a donc une arête au moins qui va de C_1 à C_2 : e_k.

- Si $p(e_k) < p(a)$:



$(T-a)+e_k$ est un arbre de poids :

$$p(T) - p(a) + p(e_k) < p(T).$$

et $(T-a)+e_k$ serait un arbre couvrant de G.

de poids $< T$ $\longrightarrow$ exclu.

- Donc $p(e_k) = p(a)$ et $(T-a)+e_k$ est aussi un arbre couvrant de poids min, mais il a plus d'arête en commun avec A que n'en avait T, cela contredit le choix de T.

Rq : • meilleur algo : Chazelle 2000 : $O(m\alpha(n,m))$

En randomisé : $O(m)$ Karger, Klein, Tarjan 1995 $\boxed{O(m)?}$

Problème du jour n° 3

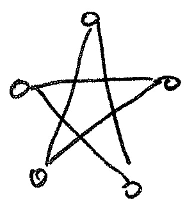
VOYAGEUR DE COMMERCE (ou Postier chinois)

Entrée : Un ensemble de n points dans le plan : $1 \ldots n$

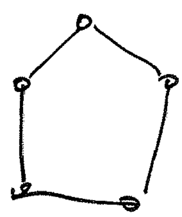
Sortie : une énumération $(v_1 \ldots v_n)$ telle que :

$$d(v_1,v_2) + d(v_2,v_3) + \ldots + d(v_{n-1},v_n) + d(v_n,v_1) \text{ soit min.}$$

Exemple :



Pas bien.



Bien.

C'est un problème NP-difficile.

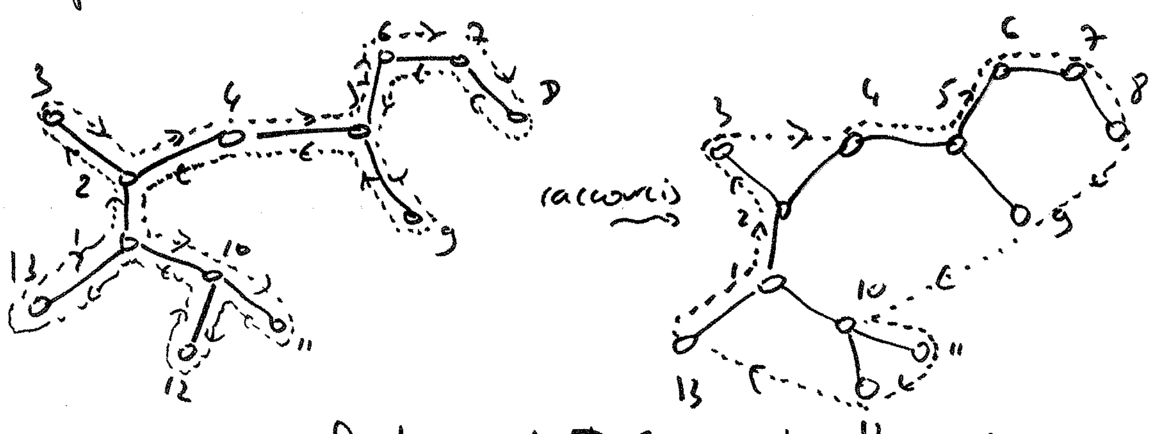
- Rq : tous les problèmes NP-difficiles sont équivalents entre eux : si jamais on trouve un algo polynomial pour résoudre un problème NP-difficile alors on aura un algo polynomial pour chacun d'entre eux.

Dans le cas du VOYAGEUR DE COMMERCE, on a la possibilité d'approcher la solution optimale à un facteur 2, en temps polynomial : on a une 2-approximation :

2 APPROX pour VDC

- Construire K_n sur $\{1, \dots, n\}$ pondéré par $p(ij) = d(i,j)$
- Trouver un arbre couvrant T de poids min pour ce graphe
- Parcourir T de façon à traverser 2 fois exactement chaque arête de T
- Modifier le parcours de façon à ne passer qu'une fois par chaque sommet

Ex :



Rq : on a un parcours en profondeur de T (voir chap. suiv.)

Propriété : L'itinéraire construit I est au pire 2 fois plus long que le meilleur itinéraire possible I_{opt}.

Pr : Si on enlève une arête à un parcours I_{opt}, on obtient un chemin, donc un arbre qui est de poids $\geq T$ (car T est un arbre de poids min). $p(I_{opt}) \geq p(T)$.

Par ailleurs : on parcours 2 fois T (puis on prend les raccourcis)

$\sim$ $p(I) \leq 2p(T)$ d'où : $p(I) \leq p(I_{opt})$. ▨

Rq : N. Christofides : $3/2$-approx, 1976 (pas très dur à comprendre).

Retour sur le tp :

COMPOSANTE-OPTIMISÉE :

L1 • $\forall x$ $\left[\ coup(x) \leftarrow x\right.$

L2 $L(coup(x)) \leftarrow \{x\}$ // Liste des sommets de $L(coup(x))$.

L3 $t(coup(x)) \leftarrow 1$ // taille de $L(coup(x))$.

L4 • Pour $xy \in E$

L5 • Si $coup(x) \neq coup(y)$

L6 • Si $t(\overset{coup(x)}{x}) > t(\overset{coup(y)}{y})$, échanger x et y // on assure $t(x) \leq t(y)$

• $aux = coup(x)$

L7 • Pour $z \in L(\underset{aux}{\cancel{coup(x)}})$ // on "vide" $coup(x)$ dans $coup(y)$.

L8 $\left[\ comp(z) \leftarrow coup(y)\right.$

L9 • $L(coup(y)) \leftarrow L(\underset{aux}{\cancel{coup(x)}}) \cup L(coup(y))$. // fusion.

L10 • $t(coup(y)) \leftarrow t(\underset{aux}{\cancel{coup(x)}}) + t(coup(y))$.

L11 • $L(\underset{aux}{\cancel{coup(x)}}) \leftarrow \emptyset$, $t(\underset{aux}{\cancel{coup(x)}}) \leftarrow 0$ // + propre

Temps : $L_4 \to O(m)$ Pour échanger un sommet entre 2 composantes les L_5 à L_{11} donne un temps $O(1)$. (grâce aux listes !)

lorsqu'un sommet change de composante, la nouvelle composante qui le contient a taille double (au moins) de la précédente (grâce à L_6). Un sommet change donc $\log n$ fois au plus de composante. Donc L_5 à $L_{11} \rightarrow O(n \log n)$.

On a : $\boxed{O(m + n \log n)}$.

Ainsi on peut obtenir Kruskal en $O(m \log n)$ (tri) + $O(m + n \log n)$

soit : $\underline{O(m \log n)}$.

Chap III : Parcours.

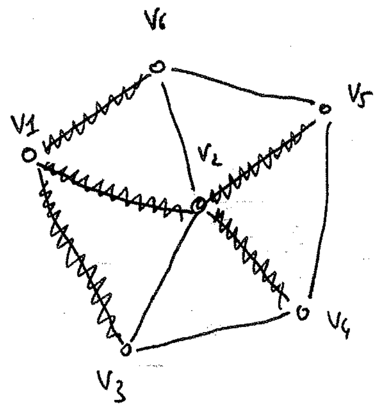
I) Définitions.

. Soit $G = (V,E)$ un graphe connexe. Un parcours de G est donné par :

- Une énumération $v_1, \dots, v_n$ de V c'est l'ordre de parcours.
- Une fonction père : $V \longrightarrow V$ telle que :
  - . $\text{père}(v_1) = v_1$
  - . $\text{père}(v_i) \in \{v_1, \dots, v_{i-1}\}$
  - . $v_i\,\text{père}(v_i)$ est une arête de G $\forall i \geq 2$.

L'ensemble des arêtes $\{v_i\,\text{père}(v_i) : i = 2 \dots n\}$ est l'arbre du parcours.

. Exemple :



	V1	V2	V3	V4	V5	V6
père	V1	V1	V1	V2	V2	V1.

. Rem : . v_1 est la racine du parcours.

. l'ensemble des arêtes $v\,\text{père}(v)$ forme un arbre couvrant de G.

. la restriction à $v_1, \dots, v_i$ est aussi un parcours, dans l'arbre correspondant v_i est une feuille.