

4 TD Algorithme sur les graphes : chemin euleriens (TD 7)

Exercice 1 -

1 - WAW

2 - A -

B -

C -

D -

chemin eulerien : + de deux sommets à nombre impair d'arêtes : impossible
0 possible

cycle eulerien : 0 sommet de degrés pair

Exercice 2 -

1. a) 0 sommet de degrés pair

b) 0 ou 2 sommet de degrés impair

c) oui

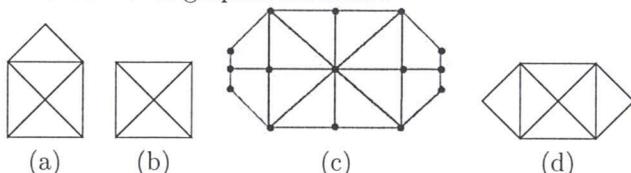
2- Algo de Krager

Un graphe eulérien est, en langage courant, un graphe que l'on peut dessiner dans lever le crayon. Le théorème d'Euler caractérise les graphes eulérien. Il a été publié par Carl Hierholzer en 1873 et on l'appelle aussi le théorème d'Euler-Hierholzer

Un chemin eulérien est un chemin qui passe une fois et une seule par toutes les arêtes d'un graphe.
Un circuit eulérien est un chemin eulérien fermé (qui boucle à son point de départ).

Exercice 1: Echauffement

1. On se donne les graphes suivants :



2. Ces graphes ont-il un circuit eulérien ? Un chemin eulérien ?

Exercice 2: Une condition nécessaire (et suffisante?)

1. Examiner le degré des sommets des graphes de l'exercice précédent.
 - a) Trouver une condition nécessaire pour l'existence d'un circuit eulérien.
 - b) Trouver une condition nécessaire pour l'existence d'un chemin eulérien.
 - c) Cette condition est-elle suffisante ?
2. Écrire dans un langage de haut niveau un algorithme qui construit un circuit eulérien.
 - a) Quelle est sa complexité ?
 - b) Comment l'utiliser pour trouver un chemin eulérien ?

Exercice 3: Implémentation

1. Proposer des structures de données pour représenter le graphe et les données intermédiaires.
2. Écrire l'algorithme correspondant en pseudo-code.

Exercice 4: Chemin hamiltonien

Une chemin hamiltonien est un chemin qui passe une fois et une seule par chaque sommet du graphe.
Comparer ce problème et celui du cycle eulérien.

08/03/2018

ALGO6

TDE Scéance 6 et 7 : enveloppes convexes

Algorithm de Jarvis

complexité en $O(n \times h)$ avec $h = |\text{hull}|$

$$n = |\text{pl}|$$

Solution du prof :

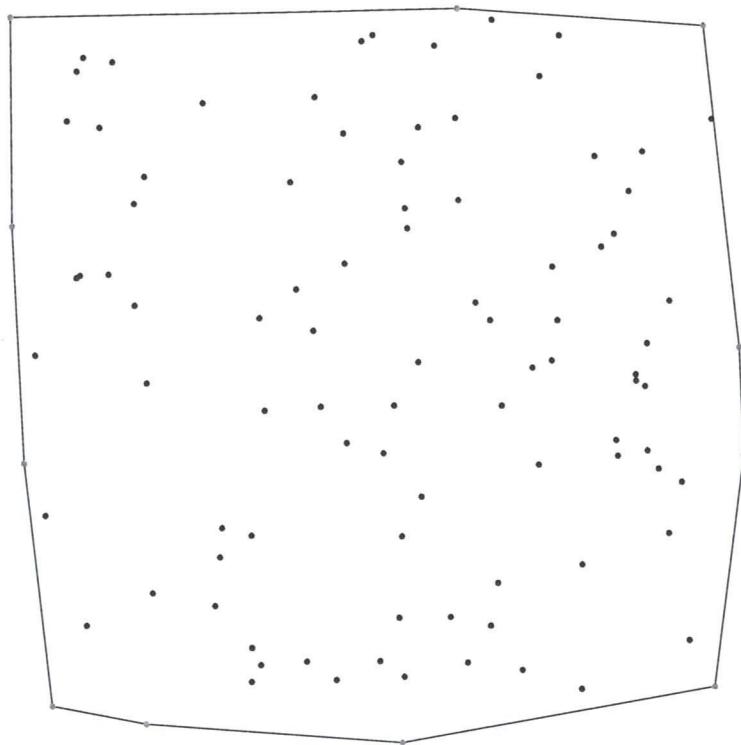
```
Convex-hull (p) {  
    for (i=0; i < p.size(); i++) {  
        for (j=0; j < p.size(); j++) {  
            if (j == i)  
                continue;  
            for (h=0; h < p.size(); h++) {  
                if (h != j && h != i) {  
                    triangle t = new triangle (P[i], P[j], P[h]);  
                    if (t.point_is_inside (P[x])) {  
                        hull.remove (x);  
                    }  
                }  
            }  
        }  
    }  
    return hull;  
}
```

```
class triangle {  
    public P1, P2, P3;  
    public boolean is_point_inside (point P);  
}
```

Exercice 2-

UE ALGO6 — TD2 — Séance 6 et 7 : Enveloppes convexes

Soit un ensemble $P = \{p_1, \dots, p_n\}$ de points du plan, avec $\forall i, p_i = (x_i, y_i)$. L'*enveloppe convexe* de P est le plus petit polygone convexe contenant tous les points de P .



Exercice 1. Identifier les propriétés des points de l'enveloppe convexe. Écrire un algorithme calculant l'enveloppe convexe d'un ensemble de points du plan à partir de ces propriétés.

Exercice 2. Écrire un algorithme de type «diviser pour régner» permettant de calculer l'enveloppe convexe d'un ensemble de points du plan.

CheminsProblèmeGraphes orientés

sommets (nœuds)

arcs

Trouver des chemins dans le graphe
(construit ?, existence ?)Application

→ des couches.

{0, ..., 5} et {0, ..., 3}

Les configurations possibles sont le produit cartésien des ensembles.

Modèle

configuration → sommet graphe

opération → arc

type opération → couleur de l'arc

coût en eau → valuation de l'arc

Dynamique

1 opération → 1 étape

exécution → chemin

Problème 1 :

Existence d'un chemin du sommet 0 au sommet d.

Problème 2 :

Construire un chemin de 0 à d. (représenter le chemin).

Problème 2' : Construire un chemin de valuation minimale.

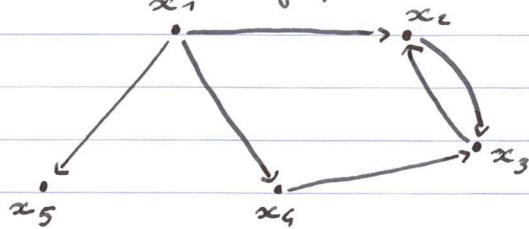
→ réseau : tables de routage.

L'informatique.

→ réseau de transport : GPS.

L'marchandise : minimiser le coût du transport.

chemin dans un graphe orienté d'origine donnée

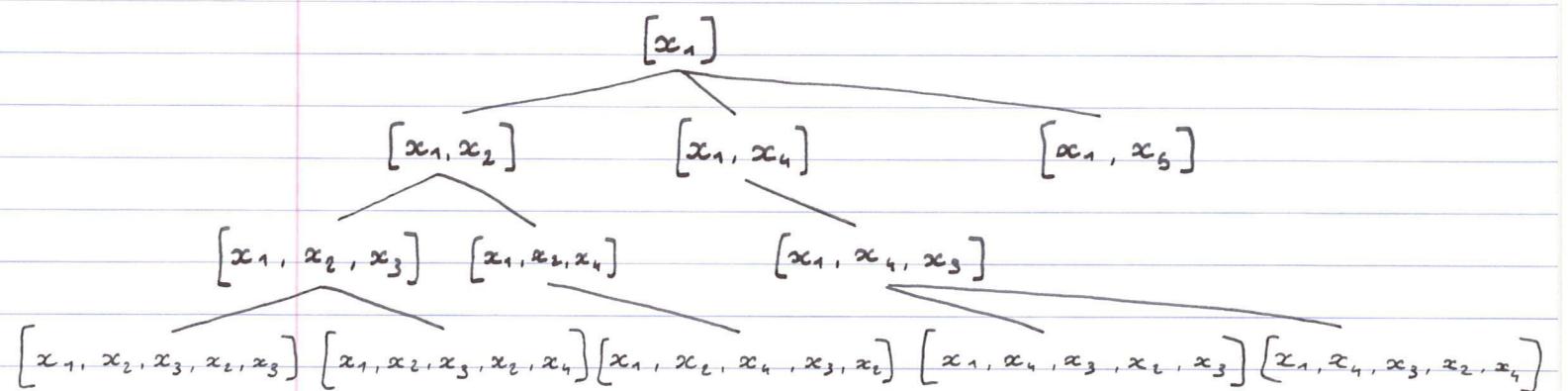


Parcourir l'ensemble des chemins d'origines x_1 .

notation $[x_{i_1}, \dots, x_{i_k}]$ pour tout ℓ .

chemin de longueur $k-1$.

=> Arbre d'énumération des chemins d'origine x_1 .



Algorithm de parcours.

$$\mathcal{E} = \{[x_1]\}$$

Tant que $\mathcal{E} \neq \emptyset$ faire

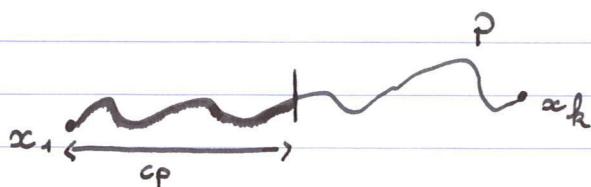
$c = \text{extremité}(\mathcal{E})$; // c est enlevé de \mathcal{E}
 visiter(c);
 pour tout voisin de c d'extremité (c')
 insérer($\mathcal{E}, c.c'$);

} l'algo ne termine pas.

Invariant.

\mathcal{E} contient des chemins d'origine x_1

Pour tout chemin c non visité, il existe un préfixe de c dans \mathcal{E} .



raisonnement par induction

il existe un préfixe de P dans \mathcal{E}

2

- ^{cm}
- soit le chemin choisi n'est pas cp , cp est dans \mathcal{E} la propriété est vérifiée en fin d'itération.
 - soit le chemin choisi est cp
 - si $c_p = c$ alors on visite c et \mathcal{I} est vérifié
 - sinon il existe un voisin x de l'extrémité de c_p sur le chemin C .
 - on insère $C.x$ qui est un préfixe de c .

Structure de \mathcal{E} : pile \rightarrow profondeur d'abord.

file \rightarrow largeur d'abord.

file à priorité \rightarrow poids minimal.

Dijkstra

$D = [\dots, \dots, \dots]$ } sommets.

$D[i] \Rightarrow$ distance entre α_i et α ;

reprendre l'algo de parcours (page précédente) et remplacer "insérer($\mathcal{E}, c.x$)" par :

si $(D[\text{extrémité}(c)] + \text{poids}(\text{extrémité}(c), x)) < D[x]$ alors

$$D[x] = D[\text{ext}(i)] + P(\text{ext}(i), x)$$

insérer ($\mathcal{E}, c.x$)

01/03/2018

ALGO6

④ TD 2 - Séance 5 : Couplage et enumeration des parties.

1. Une histoire d'emploi du temps

Ensemble d'enseignement E

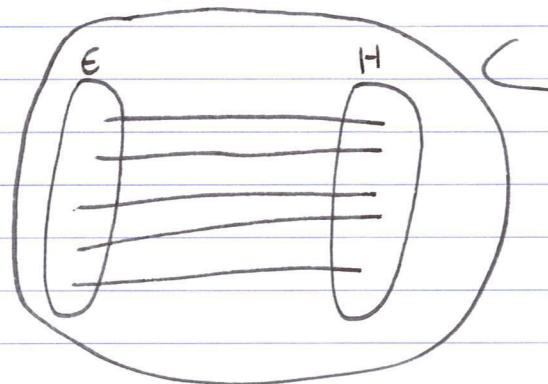
Ensemble de créneau horaire H

Ensemble de contraintes $C \subseteq E \times H$

On prend un enseignement, que l'on place à la première horaire possible en vérifiant la liste des contraintes, puis d'ajouter cette horaire à la liste des contraintes des autres enseignements.

Exercice 1 -

un enseignement est lié à une horaire



3. Couplage parfait

Exercice 2 - NON

Exercice 3 -

Couplage (L, R, A, M)

Si $L = \emptyset = R$ return M

Sinon si $A = \emptyset$ return "NO"

Sinon soit $(x, y) \in A$

$I = \text{couplage } (L \setminus \{x\}, R \setminus \{y\}, A \setminus \{x\}, M)$

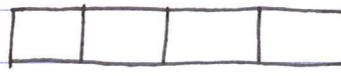
si $I \neq \emptyset$

$I = \text{couplage } (L \setminus \{x\}, R \setminus \{y\}, A \setminus \{x\}, M \cup \{z\})$

retourner I .

④

cm Programmation dynamique
Exemple



: barre de longueur m .

On peut couper la barre en morceaux.
 tableau p des prix :

<u>longueur morceau</u>	1	2	3	4	5	...
prix	2	5	7	9	11	...

Comment découper la barre de manière à ce que la somme des prix des morceaux soit maximale ?

Position des découpes.

-> choix binaire sur chacune des positions
 algorithme d'énumération : 2^{n-1} opérations.

Expression recursive.

$$M(m) = \begin{cases} p(m) & \text{pas de découpe.} \\ M(i) + M(m-i) & \text{si découpe en position } i. \end{cases}$$

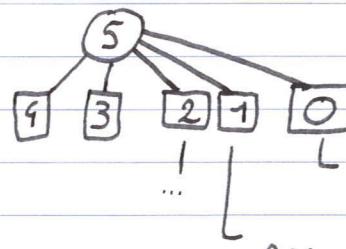
L'prix optimal

-> principe d'optimalité.

$$M(m) = \max \left\{ p(m), \max_{1 \leq i \leq m-1} \left\{ p(i) + M(m-i) \right\} \right\}$$

$$= \max_{1 \leq i \leq m} \left\{ p(i) + M(m-i) \right\} \text{ avec } M(0) = 0$$

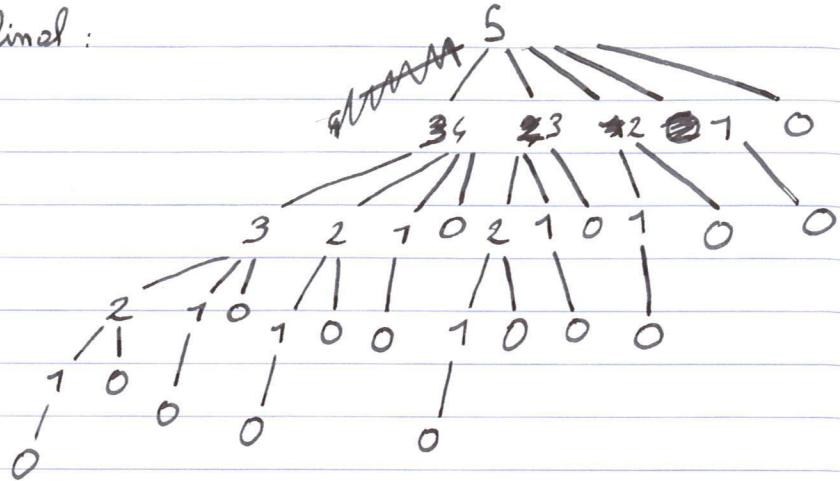
Exemple $m = 5$



! ... L terminal (premier morceau de taille 5)

premier morceau de taille 4, reste à calculer $m=1$.

soit au final :



→ parcours de l'arbre efficace en stockant les valeurs dues au jeu et à mesure. (memorisation)

Équation de Bellman.

graphe de dépendance.



: graphe orienté sans circuit (enlever le 0)

On calcule l'optimum en parcourant le graphe de bas en haut.

M	1	2	3	4	5	6
2	5	7	10	12	15	

L tableau début L $t(2) + t(1)$ L $t(4) + t(1)$ L $t(4) + t(2)$
 $t(2) + t(2)$

Remarque : Ce n'est pas la peine de regarder $i = m/2$ à m .

On regarde le 1^{er} morceau i grande ou à droite et l'un des deux est de taille $m/2$.

Cout de l'algorithme : $O(m^2)$
 (tri topologique)

2

cn

Approche programmation dynamique.

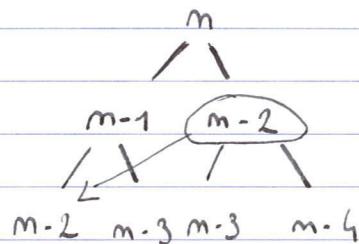
- ↳ identifie les sous problèmes de même nature (calcul redondant)
- ↳ dépendance entre les sous problèmes.
- ↳ graphe sans circuit

→ On rajoute une ligne au tableau précédent qui indique comment on a obtenu le maximum.

	0	1	2	3	4	5	6
M	0	2	5	7	10	12	15
π	1	0	0	2	2	4	..

Fibonacci:

$$\begin{cases} F(0) = F(1) = 1 \\ F(n) = F(n-1) + F(n-2) \end{cases}$$



⇒ même schéma, il y a des répétitions. On parcourt donc le graphe en stockant les valeurs lues.

mémoisation

$$\begin{pmatrix} m \\ m-1 \\ m-2 \\ m-3 \end{pmatrix}$$

Coefficient binomial

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

$\frac{n}{k}$	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1					

triangle de Pascal

