Algorithmes gloutons

Exercice 1 Comment rendre la monnaie.

Nous considérons des pièces de monnaie de 1, 2, et 5 centimes. Notons $N(x)$ le nombre minimun de pièces pour obtenir x centimes.

Question 1.1 Quelle est la valeur de $N(0)$, $N(1)$, $N(2)$, $N(3)$, $N(4)$, $N(5)$?

Correction
$N(0) = 0$, $N(1) = 1$, $N(2) = 2$, $N(3) = 2$, $N(4) = 2$, $N(5) = 1$ □

Question 1.2 Donner un algorithme qui calcule $N(x)$ et sa complexité en terme d'opérations.

Correction
Algorithme Glouton :
— Trier les types de pièces par valeur décroissante.
— Pour chaque valeur de pièce, maximiser le nombre de pièces choisies.
Plus formellement, soit R la somme restante, initialisée à S. Pour chaque valeur v_i, prendre $c_i = \lfloor \frac{R}{v_i} \rfloor$ pièces, et poser $R \leftarrow R - c_i * v_i$.
Pour prouver l'optimalité de l'algorithme glouton avec les valeurs 5, 2 et 1 :
— Au plus une pièce de 1 (sinon une de 2)
— Au plus deux pièces de 2 (sinon une de 5 et une de 1)
— Le nombre de pièces de 5 est donc $\lfloor \frac{x}{5} \rfloor$
— Conclure avec ce qui précède
□

Question 1.3 Appliquer cet algorithme qui calcule $N(14)$ en considérant maintenant qu'on dispose des pièces de monaies de 1, 7 et 10 centimes. Que constatez-vous ?

Correction
Si on utilise l'algorithme précédent, alors on utilise une pièce de 10 et quatre pièces de 1. □

Exercice 2 Comment gérer un cinéma

Un cinéma possède s salles de cinéma. Chaque semaine, le cinéma propose une liste de films à voir. Chaque film correspond à un nombre fini de séances. Chaque séance se déroule selon un horaire (intervalle de temps) précis. Les films n'ont pas tous la même durée. On précise qu'il est impossible de changer les horaires des séances.

Question 2.1 Un étudiant qui a une journée de libre veut voir le maximum de films pendant cette journée (il peut voir un même film plusieurs fois). Ce problème peut être résolu par un algorithme glouton :

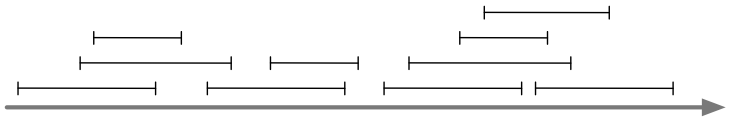


FIGURE 1 – Planning des séances de cinéma

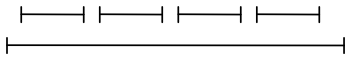
Entrée : Un ensemble $\mathcal{I}$ des séances : chaque séance i est caractérisée par l'intervalle (d_i, f_i).
Sortie Un ensemble S des séances.

1. Classer les intervalles par valuations v croissantes : $v(i_1) \leq v(i_2) \leq \cdots \leq v(i_n)$
2. Initialiser la recherche avec $S = \emptyset$;
3. Tant que $\mathcal{I}$ n'est pas vide faire
   (a) Choisir $i \in \mathcal{I}$ qui a la plus petite valuation.
   (b) Ajouter i dans S
   (c) Supprimer toutes les séances de $\mathcal{I}$ qui ne sont pas compatible avec i
4. Retourner S

Maintenant il reste à déterminer la valuation. Donner un exemple où l'algorithme ne retourne pas la solution optimale :

1. si la valuation v d'un intervalle est sa date de début (d_i).

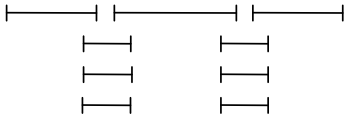
Correction



□

2. si la valuation v d'un intervalle est sa durée $(f_i - d_i)$.

Correction



□

Montrer que l'algorithme est optimal si la valuation v d'un intervalle est sa date de fin (f_i).

Correction

Soit S la solution retournée par le glouton. Soit S^* une solution optimale.

Notons j le premier élément de S inséré par l'algorithme glouton. Nous allons prouver qu'il existe une solution optimale qui contient élément j. Si S^* contient élément j, alors il n'y a rien à prouver.

Sinon, notons i l' élément de S^* tel que i est l'élément de S^* ayant la valeur f_i la plus petite.

Considérons l'ensemble $S' = S^* - \{i\} \cup \{j\}$. Tout d'abord, nous pouvons remarquer que $f_j \leq f_i$: j se termine avant i. Si ce n'était pas le cas, l'algorithme glouton aura choisit la séance i. De plus, cela signifie aussi que la séance j n'intersecte pas avec les séances de $S^* - \{i\}$ car toutes les autres séances de $S^* - \{i\}$ commence après la date f_i.

Comme $|S'| = |S^*|$, S' est une solution optimale qui contient j. Pour prouver que S est une solution optimale, on se restreint aux solutions optimales contenant j et on raisonne par récurrence.

Les ensembles S^* et S ont les k premiers éléments identiques (mais ils différent aux $k+1$ élément).

Soit i le $k+1$ élément de S^* (il n'est pas dans S)

Soit j le $k+1$ élément de S (il n'est pas dans S^*)

Nous allons prouver que i peut être remplacé par le premier élément j de S qui n'est pas dans S^*.

En effet il suffit de remarquer que $f_j \leq f_i$: j se termine avant i. Si ce n'était pas le cas, l'algorithme glouton aura choisit la séance j.

On obtient une autre solution $S' = S^* - \{i\} \cup \{j\}$ telle que $|S'| = |S^*|$ et donc toujours optimale.

□

Question 2.2 La personne qui gère l'affectation des salles doit affecter à chaque séance une salle. Son problème est de comprendre si le nombre de salles est suffisant.

Proposer un algorithme polynomial. Justifier. *Indication : on pourra considérer les séances dans l'ordre de leur date de début.*

Correction

Voici en effet un algorithme :

1. on trie les séances par date de début ;
2. on affecte les salles selon l'ordre obtenu par ce tri (bien entendu, une salle est libérée dès qu'une séance est terminée) ;
3. on calcule le nombre de salles utilisées par cette affectation ;
4. si ce nombre de salles est inférieur à s, alors on retourne vrai, sinon on retourne faux.

Cet algorithme est correct. En effet, la remarque clé est la suivante : l'algorithme considère bien une affectation optimale en nombre de salles. En effet, supposons qu'une séance v soit affectée dans une salle k. Puisque cette séance v n'a pas été affectée avant par l'algorithme c'est que la date de début a de cette séance tombe sur un moment où les salles 1 à $k-1$ sont utilisées. Les séances à ce moment là ont toute la date a en commun dans leur intervalle (par définition de la façon dont fonctionne l'algorithme). Il faut donc au moins k salles à ce moment là, et quoi que l'on fasse, dans toute affectation à ce moment là, le nombre de salles est au moins k. L'algorithme produit donc une solution nécessairement optimale.

L'algorithme est bien polynomial. □

Exercice 3 La théorie des matroïdes permet de comprendre si un algorithme glouton est optimal pour un problème. Voici la définition d'un matroïde.

— $(\mathcal{X}, \mathcal{F})$ est un matroïde si $\mathcal{X}$ est un ensemble de n éléments, $\mathcal{F}$ est une famille de parties de $\mathcal{X}$ ($\mathcal{F} \subset \mathcal{P}(\mathcal{X})$) vérifiant

1. « **Hérédité** » : $X \in \mathcal{F}$ implique $\forall Y \subset X$, $Y \subset \mathcal{F}$
2. « **Echange** » : $(A, B \in \mathcal{F}, |A| < |B|)$ implique $\exists x \in B - A$ tel que $A \cup \{x\} \in \mathcal{F}$.

— Les éléments de $\mathcal{F}$ sont appelés « *indépendants* ».

Question 3.1 Montrer que les forêts d'un graphe correspondent à un matroïde. *(indication : une forêt qui contient x arbres possèdent $n - x$ arêtes.)*

Correction

Soit $G = (V, E)$ un graphe , $|V| = n$, on définit alors $\mathcal{X} = E$ et $\mathcal{F} = \{A \subset E / A \text{ sans cycles}\}$, c.a.d. qu'un ensemble d'arêtes est un indépendant ssi c'est une forêt.

L'hérédité est évidente, car un sous-ensemble d'une forêt est une forêt (en enlevant des arêtes à une forêt on ne crée pas de cycles).

L'échange : Soient A et B des forêts de G tq $|A| < |B|$. Alors A (resp. B) contient $n - |A|$ (resp. $n - |B|$) arbres (avec chaque arête ajoutée à A, on relie deux arbres, donc on décrémente le nombre d'arbres de $|A|$).

Donc B contient moins d'arbres que A, il existe alors un arbre T de B qui n'est pas inclus dans un arbre de A, c.a.d. qu'il existe deux sommets u et v de T qui n'appartiennent pas au même arbre de A. sur le chemin de u à v dans T, il existe une paire de sommets consécutifs qui ne sont pas dans le même arbre de A (ils sont de chaque côté du *pont* qui traverse d'un arbre à l'autre). Soit (x, y) l'arête en question. d'ou $A \cup \{(x, y)\}$ est sans cycle, i.e. $A \cup \{(x, y)\} \in \mathcal{F}$

□

Un indépendant est dit « *maximal* » s'il est maximal au sens de l'inclusion.

Question 3.2 Montrer que tous les indépendants maximaux ont le même cardinal.

Correction

si ce n'était pas le cas, on pourrait les étendre par la propriété d'échange. □

Considérons les **matroïdes pondérés**

— On ajoute une fonction de poids $w : x \in \mathcal{X} \mapsto w(x) \in \mathbb{R}^{\geq 0}$.

— On pose

$$w(X) = \sum_{x \in X} w(x).$$

Question 3.3 Etant donné un matroïde pondéré, considérons l'algorithme glouton qui construit un indépendant de poids maximal (vu en cours)

1. Trier les éléments de $\mathcal{X}$ par poids décroissants : $w(s_1) \geq w(s_2) \geq \cdots \geq w(s_n)$.
2. $A := \emptyset$.
3. Pour i=1 à n faire
   (a) Si $A \cup \{s_i\} \in \mathcal{F}$ alors $A := A \cup \{s_i\}$
4. retourner A

Prouver la validité de votre algorithme.

Correction

Cet algorithme retourne une solution optimale.

— $\emptyset$ est un independant, par hérédité.

— Soit s_k le premier élément indépendant de $\mathcal{X}$ (= le premier indice i de l'algorithme précédent tel que $\{s_i\} \in \mathcal{F}$.
— Lemme : Il existe une solution optimale qui contient s_k.
— Soit B un indépendant de poids maximal, et A la solution retournée par l'algorithme. On a $s_k \in A$.
— Supposons $s_k \notin B$ (sinon, rien à prouver) : on applique $|B| - 1$ fois la propriété d'échange pour compléter $\{s_k\}$ par des éléments de B : on obtient un indépendant A' qui contient tous les éléments de B sauf un seul que l'on peut appeler b_i.
— On a $w(A') = w(B) - w(b_i) + w(s_k)$, or $w(s_k) \geq w(b_i)$, car $\{b_i\} \in \mathcal{F}$ (hérédité) et b_i a été considéré après s_k par ordre décroissant.
— Donc $w(A') \geq w(B)$, donc $w(A') = w(B)$, et A' est une solution optimale qui contient s_k.
— Puis par récurrence, on obtient que glouton donne la solution optimale : on se restreint à une solution contenant $\{s_k\}$, et on recommence avec

$$\mathcal{X}' = \mathcal{X} - \{s_k\}$$

et

$$\mathcal{F}' = \{X \subset \mathcal{X}' | X \cup \{s_k\} \in \mathcal{F}\}.$$

□

Exercice 4 Ordonnancement de tâches

Le problème de l'ordonnancement de tâches sur un seul processeur se compose
— d'un ensemble de tâches $\mathcal{X} = \{1, \ldots, n\}$ de durée 1,
— avec des dates limites $d_1, \ldots, d_n$: la tâche i doit finir avant la date d_i, sinon on doit payer la pénalité w_i.
L'objectif est de trouver un ordonnancement des tâches pour minimiser la somme des pénalités.

Afin de résoudre ce problème, on utilise les définitions suivantes :
— Un ensemble A de tâches est dit *indépendant* s'il est possible de les ordonnancer de sorte que personne ne soit en retard.
— On note $N_t(A)$ le nombre de tâches de l'ensemble A dont la date limite est $\leq t$.

Question 4.1 Montrer que les 3 propriétés suivantes sont équivalentes

1. A est indépendant ;
2. Pour $t = 1, \ldots, n$, on a $N_t(A) \leq t$;
3. Si les tâches de A sont ordonnancées dans l'ordre croissant de leur dates limites, alors aucune tâche n'est en retard.

Correction

Preuve :
— $1 \to 2$: supposons $N_t(A) > t$, alors il y a au moins une tâche qui se déroulera après le temps t, et donc A ne peut pas être indépendant.
— $2 \to 3$: trivial.
— $3 \to 1$: par définition.

□

Question 4.2 Montrer que $(\mathcal{X}, \mathcal{F})$, où $\mathcal{X}$ est l'ensemble des tâches et $\mathcal{F}$ la famille des sous-ensembles de tâches indépendantes, vérifie les propriétés suivantes :

1. *Hérédité* : $X \in \mathcal{F}$ implique $\forall Y \subset X, Y \subset \mathcal{F}$
2. *Échange* : $(A, B \in \mathcal{F}, |A| < |B|)$ implique $\exists x \in B - A$ tel que $A \cup \{x\} \in \mathcal{F}$.

Correction

Remarque

— Minimiser les pénalités des tâches en retard = Maximaliser les pénalités des tâches qui ne sont pas en retard.

— Théorème : $(\mathcal{X}, \mathcal{F})$, où $\mathcal{X}$ est l'ensemble des tâches, $\mathcal{F}$ la famille des sous-ensembles de tâches indépendantes est un matroïde.

Preuve

— Hérédité : triviale.

— échange : Soit A, B indépendants avec $|A| < |B|$. Il faut montrer qu'il existe $x \in B$ tel que $A \cup \{x\}$ soit indépendant.
Idée : comparer $N_t(A)$ et $N_t(B)$ pour $t = 1, 2, \ldots, n$.
On cherche le plus grand $t \leq n$ tel que $N_t(A) \geq N_t(B)$. On sait que $t < n$ donc $N_t(A) \geq N_t(B)$, $N_{t+1}(A) < N_{t+1}(B)$, …, $N_n(A) < N_n(B)$.
Dans B il y a plus de tâches de date limite $t+1$ que dans A : on en prend une x qui n'est pas déjà dans A : $A \cup \{x\}$ est indépendant.

□

Question 4.3 Déterminer un algorithme glouton qui fournit un ensemble indépendant A optimal.

Correction

On peut constater que les caractéristiques utiles à l'algorithme sont celles-là mêmes de *matroïde*. Cet algorithme peut donc être utilisé de manière générique sur tout problème qui présente ces caractéristiques.

1. Trier les éléments de $\mathcal{X}$ par poids décroissants
$$w(s_1) \geq w(s_2) \geq \cdots \geq w(s_n).$$
2. $A := \emptyset$.
3. Pour $i := 1$ à $n = |\mathcal{X}|$ faire
   (a) Si $A \cup \{s_i\} \in \mathcal{F}$ alors $A := A \cup \{s_i\}$
4. retourner A

□

Question 4.4 Appliquer cet algoritme à l'exemple suivant composé de 7 tâches :

i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	7	6	5	4	3	2	1

Correction

Itérations :

— $A = \{1\}$;

— $A = \{2, 1\}$;

— $A = \{2, 1, 3\}$

— $A = \{2, 4, 1, 3\}$;
— $A = \{2, 4, 1, 3, 7\}$.
— Résultat : $\{2, 4, 1, 3, 7\}$ et la pénalité maximale est 5.

□

Exercice 5 Coloriage des graphes planaires

Un graphe est *planaire* si on peut le dessiner dans le plan sans que les arêtes ne se croisent.

Soit $G = (V, E)$ un graphe planaire non vide *connexe*. Notons $n = |V|$, $a = |E|$. Considérons un dessin de G dans le plan (sans que les arêtes ne se croisent). Soit f le nombre de régions du plan délimitées par les arêtes de G.

Question 5.1 Montrer que les graphes planaires respectent la relation d'Euler $n - a + f = 2$.

(*Indication : raisonner par récurrence sur le nombre de faces*)

Correction

On procède par récurrence sur le nombre f de faces. Si $f = 1$, le graphe possède uniquement une unique face. Par conséquent, le graphe connexe ne possède aucun cycle et est un arbre. Ainsi, $n - a + f = n - (n - 1) + 1 = 2$ et la formule est vérifiée.

Supposons que la formule d'Euler satisfaite pour les valeurs inférieur à f.

Soit e une arête d'un cycle du graphe. Par définition, l'arête appartient sépare deux faces A et B. En supprimant cette arête, le nouveau graphe obtenu possède le même nombre de sommets (n), $a - 1$ arêtes. De plus ce graphe a $f - 1$ faces puisque A et B forment une seule face de ce nouveau graphe. En appliquant l' hypothèse de récurrence, on a

$$n - (a - 1) + f - 1 = 2$$

En simplifiant on obtient $n - a + f = 2$ et le graphe respecte bien la formule d'Euler.
Donc tout graphe planaire satisfait la formule d'Euler. □

Question 5.2 On suppose $n \geqslant 3$. Montrer que $3f \leq 2a$.

Correction

Soit $\mathcal{F}$ l'ensemble de faces. Notons $nb(F)$ le nombre d'arêtes qui délimitent la face F
Chaque face F est délimitée par au moins 3 arêtes.

$$nb(F) \geq 3$$

Donc $\sum_{i \in \mathcal{F}} nb(F) \geq 3 \cdot f$

Chaque arête est une frontière de deux faces et donc on peut en déduire que

$$\sum_{i \in \mathcal{F}} nb(F) = 2a$$

Donc en combinant les deux équations, on obtient $3f \leq 2a$. □

Question 5.3 Montrer qu'il existe un sommet de degré au plus 5 dans le graphe planaire G.

Correction

Si G possède des sommets de degré 1, alors il existe bien un sommet de degré au plus

5 dans G.

Supposons que G ne possède pas des sommets de degré 1. Raisonnons par l'absurde et supposons que pour tout sommet v de G, $d(v) \geq 6$. On a $6n \leq 2a$ car la somme de tous les sommets est égale à 2 fois le nombre de arêtes.

En appliquant la formule d'Euler ($n - a + f = 2$), on a $f = 2 + a - n$.

$$\begin{aligned} 3f &\leq 2a \\ 3(2 + a - n) &\leq 2a \\ a &\leq 3n - 6 \\ 2a &\leq 6n - 12 \end{aligned}$$

Ce qui est en contradiction avec $6n \leq 2a$. Donc il existe un sommet de degré au plus 5 dans G. □

Question 5.4 Concevoir un algorithme de 6-coloration des graphes planaires en temps polynomial.

Correction

Notons $\mathcal{D}(G)$ l'ensemble des sommets de G tel que leur degrés sont inférieurs ou égale à 5. Nous allons contruire une suite de graphes $G_1, \ldots, G_n$ de la façon suivante

1. $G_1 \leftarrow G$
2. pour i allant de 2 à n faire
   (a) calculer $\mathcal{D}(G_i)$;
   (b) extraire un sommet v_i tel que $v_i \in \mathcal{D}(G_i)$;
   (c) construire $G_{i+1} = (V_{i+1}, E_{i+1})$ tel que
   $V_{i+1} = V_i \backslash \{v_i\}$, et $E_{i+1} = E_i \backslash \{e : e \text{ est adjacent à } v_i \text{ dans } G_i\}$.

Tout d'abord remarquons que ces graphes sont des graphes planaires (puisqu'on enlève simplement un sommet entre deux graphes G_i et G_{i-1}.)

A partir de cette suite, nous pouvons construire une coloration $c : V \to \{1, \ldots, 6\}$ de la façon suivante :

1. pour i allant de n à 1 faire
   (a) colorier v_i avec la plus petite couleur qui n'appartient aux couleurs de son voisinnage dans G_i ;

Nous construisons une coloration utilisant 6 couleurs : à chaque itération i, le sommet a au plus 5 voisins et donc il peut être voisin de sommets ayant au plus 5 couleurs différentes. □

Remarque : En fait, tout graphe planaire peut être colorer avec 4 couleurs. Ce résultat est connu sous le nom de théorème des quatre couleurs. Il a été démontré en 1976 par Appel et Haken.