Algorithmique
TD n° 9 : Algorithmes gloutons

Exercice 1 :

Vous devez ranger tous les n livres d'une bibliothèque dans des étagères. Les étagères sont composées de tablettes qui peuvent contenir des livres jusqu'à une certaine hauteur. L'ordre des livres est fixé et ne peut pas être changé. Chaque tablette doit contenir un intervalle contigu de la séquence des livres. Il y a deux tableaux `H[1..n]` et `E[1..n]` qui contiennent les hauteurs et les épaisseurs des livres dans l'ordre. Toutes les tablettes ont la même longueur L. L'épaisseur totale de tous les livres sur une tablette ne peut pas excéder L.

1. Supposons que tous les livres ont la même hauteur h et que toutes les tablettes peuvent stocker des livres de cette hauteur. Donner un algorithme glouton pour stocker les livres en utilisant le moins de tablettes possibles. Montrer qu'il est correct.

2. Supposons maintenant que les livres peuvent avoir des hauteurs différentes et qu'on peut ajuster chaque tablette pour accueillir le livre le plus haut sur cette tablette. En particulier, la hauteur de chaque tablette vide peut être 0. Le problème est de stocker les livres de sorte que la somme totale des hauteurs des tablettes (appelé coût) soit le plus petit possible. Montrer que l'algorithme glouton précédent ne donne pas toujours la meilleure solution.

3. Donner un algorithme pour trouver le meilleur coût.

Exercice 2 :

On considère deux tableaux d'entier positifs ou nuls `L[1..n]` et `C[1..m]`. Une matrice M d'entiers positifs ou nuls de dimension $n \times m$ est appelée *compatible* avec L et C, si la somme des éléments de chaque ligne i est égale à $L[i]$ et que la somme des éléments de chaque colonne j est égale à $C[j]$. Un exemple d'une matrice compatible avec L et C est donné dans la figure 1.

L : \ C :	27	94	42	50
57	12	42	3	0
47	3	10	31	3
71	2	31	4	34
38	10	11	4	13

L : \ C :	20	15	30	5
22	?	?	?	?
13	?	?	?	?
4	?	?	?	?
31	?	?	?	?

FIGURE 1 – Une matrice compatible et un exemple à résoudre

1. Montrer que si M est compatible avec L et C, alors L et C doivent vérifier la propriété :

$$\sum_{i=1}^{n} L[i] = \sum_{j=1}^{m} C[j] \tag{1}$$

2. Montrer que si L et C satisfont la propriété (1), alors il existe une matrice M compatible avec L et C. Pour cela donner un algorithme glouton qui construit une matrice M compatible avec L et C. Appliquer l'algorithme sur l'exemple de la figure 1.

Exercice 3 :

On considère deux tableaux d'entier positifs ou nuls `L[1..n]` et `C[1..m]` comme dans l'exercice précédent. Ici, on cherche une matrice *binaire* (c'est-à-dire composée de 0 et de 1) compatible avec L et C.

1. La propriété (1) de l'exercice précédent est toujours vraie pour chaque matrice binaire compatible. Par contre, même si on exige que $L[i]$ (resp. $C[j]$) soit toujours inférieur ou égal à n (resp. m), il y a des L et C vérifiant la propriété pour lesquels il n'y a pas de matrice binaire compatible. Donner un exemple.

2. Proposer une propriété sur L et C plus forte de sorte qu'on soit sûr de pouvoir trouver une matrice binaire compatible. Pour cela donner un algorithme glouton.

Exercice 4 : Note d'examen (le retour)

Un sujet d'examen est composé de n questions ayant chacune un barème (c'est-à-dire une note max.) b_1,...,b_n. La somme des b_i est supérieure (ou égale) à 20. Une fois corrigée, la copie d'un étudiant-e est représentée par une note v_1,..., v_n pour les différents exercices. On souhaite désormais calculer la note de l'étudiant-e sur 20. On a déjà vu deux manière de faire, en voici une troisième : on choisit les UEs de manière à arriver à un barème de 20 (exactement) mais en s'autorisant à revoir à la baisse le nombre de points d'un exercice (en ajustant la note de manière proportionnelle). Par exemple, une note de 4 sur 6 pour un exercice, peut être utilisée comme un 2 sur 3 (ou un 1 sur 3/2, etc.).

- Proposer un algorithme pour calculer la note d'un-e étudiant-e.
- Laquelle des trois méthodes donne la meilleure note à une copie ?
- Laquelle est utilisée par le serveur pédagogique pour calculer les notes de semestre ?

Exercice 5 : Gloutonner pour approximer

Une instance du problème BIN-PACKING est donnée par n rationnels $x_1, \ldots, x_n$ avec $\forall i, 0 < x_i \leqslant 1$. On cherche le plus petit entier m tel que l'on peut ranger tous les éléments en m sacs de capacité 1, c'est-à-dire le plus petit entier m tel qu'il existe une fonction $f : [1, n] \to [1, m]$ avec $\forall b \in [1, m] \sum_{i \in f^{-1}(b)} x_i \leqslant 1$. Ce problème est NP-dur.
On considère l'algorithme itératif (glouton) plaçant chaque élément dans le premier sac disponible dans lequel il peut rentrer (l'algorithme "ouvre" un nouveau sac vide si l'élément ne peut rentrer nulle part). On note m le résultat et f l'assignation correspondante fournie par cet algorithme.

1. Montrer qu'au moins $m-1$ paquets sont remplis (strictement) à plus de la moitié de leur capacité (c'est-à-dire qu'il existe $m-1$ valeurs de b telles que $\sum_{i \in f^{-1}(b)} x_i > 1/2$).
2. Montrer que toute solution au problème du bin packing est au moins égale à S = $\sum_{i=1}^{n} x_i$.
3. En déduire que l'algorithme glouton est un algorithme de 2-approximation.

Note. L'algorithme qui trie d'abord les éléments en ordre décroissant et ensuite applique la stratégie de l'algorithme précédent ne fournit pas non plus la solution optimale, mais on peut montrer qu'il est un algorithme de 3/2-approximation. On peut aussi montrer qu'aucun algorithme avec un facteur d'approximation plus bas que 3/2 ne peut exister (à moins que $P = NP$).

Exercice 6 : MAX3CNF

Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de variables propositionnelles. Une formule de type « 3CNF » [1] est une conjonction de m clauses, chaque clause est une disjonction de 3 littéraux, et chaque littéral est une variable x_i ou sa négation $\neg x_i$. Par exemple, la formule suivante est une formule 3CNF qui contient 4 clauses :

$$(x_2 \vee \neg x_4 \vee \neg x_1) \wedge (x_1 \vee x_4 \vee x_3) \wedge (\neg x_4 \vee \neg x_2 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

Dans cet exercice, on suppose en plus que **chaque clause contient toujours trois variables distinctes** (on interdit donc les clauses de la forme $x_2 \vee x_2 \vee x_1$ ou $x_1 \vee \neg x_1 \vee x_4$). Le problème SAT consiste à chercher une valuation pour les variables de X qui permet de satisfaire la formule (donc toutes les clauses). Dans la formule, ci-dessus il suffit de prendre la valuation $\{x_1 : \top, x_2 : \top, x_3 : \bot, x_4 : \bot\}$.
Ici on s'intéresse au problème MAX3CNF : étant donnée une formule 3CNF, on cherche le nombre maximal de clauses que l'on peut satisfaire avec une valuation.

1. Résoudre le problème MAX3CNF pour la formule ci-dessous :

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

2. Expliquer pourquoi la résolution du problème MAX3CNF est difficile ?
3. On considère l'algorithme probabiliste ci-dessous :

Procédure Algo1 (ϕ)
begin
  pour $i = 1 \ldots n$ **faire**
    Choisir aléatoirement une valeur $v_i \in \{\bot, \top\}$ pour x_i
  retourner le nombre de clauses de ϕ satisfaites par $\{v_1, \ldots, v_n\}$

   (a) Quelle est la complexité de l'algorithme ?
   (b) Quelle est la probabilité qu'une clause d'une formule 3CNF ne soit pas satisfaite par la valuation construite par l'algorithme ? En déduire la probabilité qu'une clause soit satisfaite.
   (c) Quelle est l'espérance du nombre de clauses satisfaites par une valuation aléatoire pour X ?
   (d) En déduire que toute formule 3CNF avec au plus 7 clauses est satisfaisable.
   (e) Montrer que l'espérance de la valeur retournée par l'algorithme est supérieure à $\frac{7}{8}K$ où K est le résultat optimal.

4. A partir des résultats précédents, proposer un algorithme glouton efficace pour MAX3CNF qui renvoie toujours un résultat supérieur à $\frac{7}{8}K$. Donner sa complexité.

1. CNF signifie *Conjunctive Normal Form*.