TD 14 – Révisions

Exercice 1.
On définit le problème de décision 2-Partition ainsi : soit $S = \{a_1, \dots, a_n\}$ des entiers, existe-t-il $I \subset S$ tel que $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

1. Montrer que 2-Partition est NP-complet.

Exercice 2.

1. Étant donnés $2n$ entiers $a_1, a_2, \dots, a_{2n}$, trouver un sous-ensemble $I \subset [1..2n]$ tel que $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$ et $|I| = n$.

Exercice 3.

1. Etant donnés n entiers $a_1, a_2, \dots, a_n$, peut-on trouver un sous-ensemble $I \subset [1..n]$ tel que $|\sum_{i \in I} a_i - \sum_{i \notin I} a_i| \leq 1$

Exercice 4.
On s'intéresse au problème du sac-à-dos :
*I*nstance : Un ensemble fini X d'objets ; pour chaque objet $x_i \in X$, une valeur $p_i \in \mathbb{N}$ et une taille $a_i \in \mathbb{N}$. Une capacité $B \in \mathbb{N}$.
*S*olution : Un sous-ensemble Y de X tel que $\sum_{x_i \in Y} a_i \leq B$
*M*esure : La valeur totale des objets choisis, i.e. $\sum_{x_i \in Y} p_i$.
On notera $m^*(x)$ la mesure optimale (maximale).

1. Formuler le problème de décision associé et montrer qu'il est $\mathcal{NP}$-complet.
2. Mais alors, comment a-t-on pu résoudre ce problème en cours par programmation dynamique ?

Glouton Dans l'algorithme glouton, on trie les éléments par rapports p_i/a_i décroissants, et on choisit chaque élément examiné dans cet ordre s'il y a la place pour lui. Soit $m_g(x)$ la mesure de l'algorithme glouton.

3. Soit K un entier arbitrairement grand. Construire une instance x telle que $m^*(x)/m_g(x) > K$.
4. Soit p_{max} la valeur maximale d'un objet. Montrer que $m^*(x)/\max\{m_g(x), p_{max}\} < 2$.
Indication : soit j l'indice du premier élément que l'algorithme glouton ne prend pas ; montrer que $m^*(x) \leq \sum_{i=1}^{j} p_i$.

Programmation dynamique revisited

5. Pour $1 \leq k \leq n$ et $0 \leq p \leq \sum_{i=1}^{n} p_i$, on cherche, parmi les sous-ensembles de $\{x_1, x_2, \dots, x_k\}$ de valeur totale égale à p et de taille majorée par B, un qui soit de taille minimale. On note $M^*(k, p)$ une solution optimale de ce problème et $S^*(k, p)$ la taille correspondante. Expliquer comment calculer les $M^*(k, p)$ par programmation dynamique. Quelle est la complexité de la résolution du problème du sac-à-dos ?
6. Pour tout rationnel $r > 1$, on considère le schéma d'approximation suivant : soit p_{max} la valeur maximale d'un objet et $t = \lfloor \log(\frac{r-1}{r} \frac{p_{max}}{n}) \rfloor$. On résout par programmation dynamique comme plus haut une instance modifiée du problème original : les tailles des n objets sont toujours a_i mais les valeurs sont $p'_i = \lfloor \frac{p_i}{2^t} \rfloor$. Soit $m_{AS}(x, r)$ la mesure de la solution ainsi obtenue.
   (a) Montrer que la complexité du schéma d'approximation est $O(\frac{r}{r-1} n^3)$.

(b) Montrer que $m^*(x)/m_{AS}(x,r) \leq r$.
Indication : montrer que $\frac{m^*(x)-m_{AS}(x,r)}{m^*(x)} \leq \frac{n2^t}{p_{max}}$.

Exercice 5.
On part d'un graphe qui possède trois sommets a, b, c reliés sous forme d'un triangle (arêtes (a,b), (b,c) et (c,a)). Le graphe n'est pas dirigé, et il est possible d'avoir plusieurs arêtes reliant deux sommets. Deux opérations sont effectuées sur le graphe :
– **lien(i,j)**, où i et j sont deux sommets du graphe : une arête (i,j) est rajoutée au graphe. Le coût de l'opération est 1.
– **coupe(i)**, où i est un sommet du graphe : le sommet i, de degré d, est supprimé du graphe, et remplacé par d sommets $i_1,\ldots,i_d$. Ces nouveaux sommets sont reliés par un cycle, et chaque voisin de i dans le graphe initial est rattaché à exactement l'un des nouveaux sommets $i_1,\ldots,i_d$. Le coût de l'opération est d, le degré de i.

1. Illustrer sur un petit exemple ces deux opérations.
2. Quel est le degré des sommets $i_1,\ldots,i_d$ après une opération de coupe ?
3. Trouver deux constantes α et β les plus petites possibles telles que le coût total d'une séquence de n liens et m coupes soit au plus $\alpha n + \beta m$.