TD 9. Algorithmes d'approximation

Exercice 1. *Partition*

Notation. *Pour un ensemble S d'entiers, on note $\Sigma_S = \sum_{s \in S} s$.*

Étant donné un ensemble de n entiers positifs $A = \{a_0, \ldots, a_{n-1}\}$, on cherche une partition $A = X \sqcup Y$ équilibrée, c'est-à-dire telle que $\Sigma_X \simeq \Sigma_Y$. Plus précisément, on cherche à minimiser $\max(\Sigma_X, \Sigma_Y)$. On propose l'algorithme glouton suivant.

```
PARTITION(A) :
(X, Y, Σ_X, Σ_Y) ← (∅, ∅, 0, 0)
Pour i = 0 à n − 1 :
   Si Σ_X < Σ_Y : X ← X ∪ {a_i}, Σ_X ← Σ_X + a_i
   Sinon : Y ← Y ∪ {a_i}, Σ_Y ← Σ_Y + a_i
Renvoyer (X, Y)
```

1. **i.** PARTITION fournit-il une solution optimale sur l'entrée $A = \{4, 2, 3, 2, 7\}$?
   ii. Quelle est sa complexité ?

Pour une entrée A, soit $(X^\star, Y^\star)$ une solution optimale et $\text{OPT} = \max(\Sigma_{X^\star}, \Sigma_{Y^\star})$. Soit (X, Y) la solution renvoyée par PARTITIONGLOUTON, et on suppose sans perte de généralité $\Sigma_X \geq \Sigma_Y$.

2. **i.** Montrer que pour tout i, $\text{OPT} \geq a_i$.
   ii. Montrer que $\text{OPT} \geq \frac{1}{2}\Sigma_A$.
3. On considère le dernier élément a_k ajouté par PARTITION à X.
   i. Montrer que $\Sigma_X - a_k \leq \frac{1}{2}(\Sigma_A - a_k) \leq \text{OPT} - \frac{1}{2}a_k$.
   ii. En déduire que PARTITION est une $\frac{3}{2}$-approximation pour le problème.
   iii. Construire un exemple pour lequel PARTITION fournit une solution égale exactement à $\frac{3}{2}\text{OPT}$.
4. *(bonus)* On modifie très légèrement PARTITION en triant les a_i en ordre décroissant : $a_0 \geq a_1 \geq \cdots \geq a_{n-1}$. On garde les mêmes notations que précédemment.
   i. Montrer que sur l'entrée $\{10, 10, 9, 9, 2\}$, l'algorithme n'est pas optimal.
   ii. Montrer que si $\#X = 1$, alors la solution renvoyée est optimale.
   iii. On suppose que $\#X \geq 2$. Montrer que le dernier élément a_k ajouté par PARTITION à X vérifie $a_k \leq \frac{2}{3}\text{OPT}$.
   iv. En déduire que PARTITION avec tri est une $\frac{4}{3}$-approximation.
   v. Construire un exemple pour lequel PARTITION avec tri fournir une solution exactement égale à $\frac{7}{6}\text{OPT}$.[1]

1. *Remarque. On peut en fait montrer (mais c'est difficile) que l'algorithme glouton avec tri renvoie toujours une solution* $\leq \frac{7}{6}\text{OPT}$.

Exercice 2. *Coupe maximale*

Soit $G=(S,A)$ un graphe. Une *coupe* de G est une partition des sommets $S=X\sqcup Y$ en deux sous-ensembles disjoints non vides. La *taille* d'une coupe $X\sqcup Y$ est le nombre d'arêtes dont une extrémité est dans X et l'autre dans Y. On cherche à calculer une coupe $X\sqcup Y$ de taille maximale.
On utilise l'algorithme probabiliste simpliste suivant : chaque sommet $s\in S$ est affecté indépendamment à X avec probabilité $\frac{1}{2}$ et à Y avec la même probabilité.

1. Soit $a=\{u,v\}$ une arête de G. Calculer la probabilité que a soit *coupée*, c'est-à-dire qu'une de ses extrémités appartienne à X et l'autre à Y.
2. Quelle est l'espérance de la taille de la coupe renvoyée par l'algorithme probabiliste ? *Utiliser la linéarité de l'espérance.*
3. En déduire que l'espérance de la taille de la coupe renvoyée est $\geq \frac{1}{2}\textsc{opt}$ où $\textsc{opt}$ est la taille d'une coupe maximale.

Exercice 3. MaxSat

On considère des *formules sous forme normale conjonctive* (formule CNF), comme par exemple $(x_1\vee\neg x_2\vee x_3)\wedge(\neg x_1\vee x_3)\wedge(\neg x_3)\wedge(x_2\vee x_3)$: c'est une *conjonction de clauses*, chaque clause est une *disjonction de littéraux* et chaque littéral est soit une variable booléenne soit sa négation. Le problème Sat consiste, étant donné une formule CNF, à décider s'il existe une affectation qui satisfait la formule.
Dans cet exercice, on s'intéresse à une variante du problème : MaxSat. Étant donné la formule CNF, il s'agit de trouver l'affectation qui satisfait *le plus possible de clauses*.

1. Justifier que si on sait résoudre de manière exacte MaxSat, alors on peut résoudre Sat.
2. Vérifier que dans la formule de l'exemple, au plus 3 clauses sur 4 peuvent être satisfaites simultanément.
3. Quel algorithme (déjà vu) peut-on modifier pour résoudre MaxSat ? Quelle est sa complexité ?
4. On propose l'algorithme suivant : on renvoie une affectation des variables choisie uniformément, c'est-à-dire qu'on choisit, pour chaque variable, la valeur VRAI avec probabilité $\frac{1}{2}$ ou FAUX avec probabilité $\frac{1}{2}$. *On suppose que chaque variable apparaît au plus une fois dans chaque clause.*

   i. Quelle est la complexité de cet algorithme ?
   ii. Soit C une clause de taille k. Montrer que la probabilité que C soit satisfaite est $1-1/2^k$.
   iii. En déduire que l'espérance du nombre de clauses satisfaites est $\geq m/2$ où m est le nombre total de clauses.
   iv. Que peut-on dire sur le facteur d'approximation ?