TD 13 – Révisions partie 2

Exercice 1. *Charpentier*

1. *(difficile)* Etant données n baguettes rigides de longueur entières $a_1, a_2, \ldots, a_n$, pouvant être reliées dans cet ordre bout-à-bout par des charnières, et étant donné un entier k, assembler toutes les baguettes de manière qu'en repliant la chaîne obtenue la longueur totale ne dépasse pas k (voir la Figure 1).
   Indication : réduire à partir de 2-Partition.

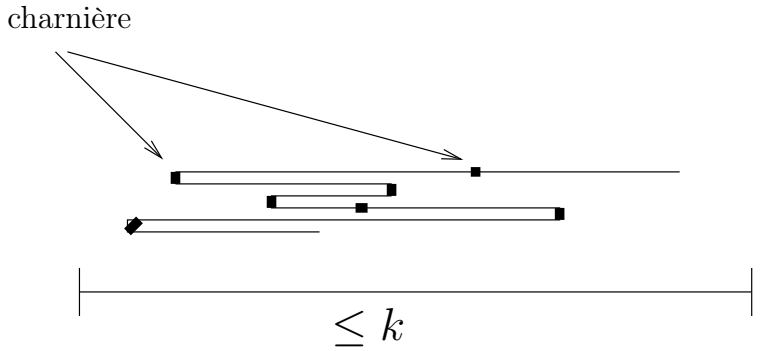

FIGURE 1 – Pliage des baguettes

Exercice 2. *On s'approche...*

Nous considérons ici le problème de la couverture par sommets : étant donné un graphe non-orienté $G = (V, E)$ et une fonction de poids sur les sommets $w : V \to \mathbb{Q}+$, trouver $S \subseteq V$ de poids minimum qui couvre les arêtes (c'est à dire $\forall\, e \in E, \exists\, s \in S : s \in e$). Nous allons étudier ici une 2-approximation. Pour ce faire nous nous intéressons à un type de fonction de poids particulier : les fonctions de poids par degré. w est une fonction de poids par degré si $\exists c > 0, \forall v \in V : w(v) = c \cdot deg(v)$.

1. Soit $\{G = (V, E), w\}$ une instance du problème telle que w est une fonction de poids par degrés. Montrer que $w(V) \leq 2 \cdot OPT$ où $w(V) = \sum_{v \in V} w(v)$.

On sait donc traiter les instances avec une fonction de poids par degrés. La méthode du mille-feuille consiste alors à décomposer une instance quelconque en une famille d'instances avec fonction de poids par degrés.

2. À chaque étape de la décomposition, on cherche la plus grande fonction de poids par degrés p inférieure à w. Expliquer comment calculer p.

L'algorithme du mille-feuille est le suivant :

Données:
$G = (V, E)$;
une fonction de poids quelconque w;
début
1 $t \leftarrow 0$;
2 $G_0 \leftarrow G$;
3 $w_0 \leftarrow w$;
4 **tant que** $G_t = (V_t, E_t)$ *contient une arête* **faire**
5 $D_t \leftarrow \{u \in V_t : deg_t(u) = 0\}$;
6 $p_t \leftarrow$ plus grande fonction de poids par degrés inférieure à w_t dans G_t;
7 $S_t \leftarrow \{u \in V_t : p_t(u) = w_t(u)\}$;
8 $G_{t+1} \leftarrow G_t \setminus (D_t \cup S_t)$;
9 $w_{t+1} \leftarrow w_t - p_t$;
10 $t \leftarrow t + 1$;
11 **retourner** $C = \bigcup_{k=0}^{t-1} S_k$

3. Montrer que l'algorithme termine en temps polynomial.
4. Montrer que l'ensemble C de sommets retournés par l'algorithme est bien une couverture.
5. Pour tout $v \in C$, exprimer $w(v)$ en fonction des poids par degrés p_k. Qu'en est-il pour $v \notin C$? Remarque : on pourra poser $p_k(u) = 0$ pour tout sommet $u \notin G_k$.

À partir de maintenant, on notera C^* une couverture par sommets optimale pour l'instance de départ $\{G = (V, E), w\}$.

6. Pour toute étape i de l'algorithme, comparer $p_i(C \cap G_i)$ et $p_i(C^* \cap G_i)$.
Indication : on remarquera que $C \cap G_i$ et $C^* \cap G_i$ sont deux couvertures par sommets de G_i.
7. Montrer que l'algorithme est une 2-approximation du problème de la couverture par sommet minimum avec des poids arbitraires. Trouver un exemple où l'algorithme renvoie effectivement une solution de poids $2.OPT$.

Exercice 3. *SetCover*

Le problème de décision Set-Cover est défini comme suit :
Entrée : Un ensemble X contenant n éléments, ainsi que m sous-ensembles $S_1, \ldots, S_m$ de X, et un entier $k \leq n$.
Sortie : Peut-on trouver au plus k ensembles parmi les k S_i de façon à couvrir tous les éléments de X ? Plus formellement, existe-t-il $I \subseteq \{1, \ldots, m\}$ tel que $|I| \leq k$ et $\cup_{i \in I} S_i = X$?
On admettra que Set-Cover est un problème $\mathcal{NP}$-complet [1](vous pouvez le chercher, ce n'est pas si compliqué !). Le problème d'optimisation associé est de trouver la plus petite couverture.
Pour cela, considérons l'algorithme glouton suivant :
– Prendre un ensemble S_j qui couvre le plus grand nombre d'élément de X.
– Remplacer X par $X \setminus S_j$ et tous les S_i par $S_i \setminus S_j$.
– Recommencer tant que X n'est pas vide.

1. Est-ce que l'algorithme glouton est optimal ?
2. Prouver que si une solution optimale contient k sous-ensembles, alors l'algorithme glouton prend au plus $\mathcal{O}(k \ln(n))$ sous-ensembles.
3. Est-ce que l'algorithme glouton est une λ-approximation de Set-Cover ? Si oui, quelle est la valeur de λ ?

1. M.R. Garey et D.S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979