Corrigé du TD Bin Packing

1] 3 boîtes : $\{2,8\}$, $\{5,4,1\}$ et $\{7,3\}$

2] Entrée : C, $X = x_0, \dots, x_{n-1}$ et k

Sortie : Oui ssi il est possible de répartir les elts de X dans moins de k boîtes de capacités C chacune.

3] SubSetSum $\leq_p$ Partition

$\varphi : A, s \to A \cup \{2s, \sum_{x \in A} x\} = C$

↳ Si (A, s) est une instance positive de SubSetSum alors

$\exists B \subset A$ t.q $\sum_{x \in B} x = s$

Ainsi $\sum_{x \in A \setminus B} x = \sum_{x \in A} x - s$

on pose $I = B \cup \{\sum_{x \in A} x\}$ et ainsi $C \setminus I = A \setminus B \cup \{2s\}$

$\sum_{x \in I} x = s + \sum_{x \in A} x = \sum_{x \in C \setminus I} x$

On a donc bien vérifié que C est une instance positive de partition.

↳ Réciproquement, soit $I \subset C$ t.q $\sum_{x \in I} x = \sum_{x \in C \setminus I} x$, autrement dit, supposons que C est une instance positive de Partition. Puisque les entiers sont naturels : $2s + \sum_{x \in A} x \geqslant \sum_{x \in A} x$ et $2s$ et $\sum_{x \in A} x$ ne peuvent
($s > 0$)

pas être tous les deux ds I ou ds son complémentaire. Supposons spg que $2s \in I$ et on pose $I' = I \setminus \{2s\}$

$$\sum_{x \in I'} x = \sum_{x \in I} x - 2s = \sum_{x \in A \setminus I} x + \sum_{x \in A} x - 2s$$

En ajoutant $\sum_{x \in A \setminus I} x$ de part et d'autre, on obtient

$$\cancel{\sum_{x \in A}} x = 2 \sum_{x \in A \setminus I} x + \cancel{\sum_{x \in A}} x - 2s$$

$$\Rightarrow \sum_{x \in A \setminus I} x = s$$

et finalement $B = A \setminus I$ convient pour garantir que (A, s) est une instance positive de SubSetSum.

Ainsi la réduction annoncée est garantie et en montrant que Partition $\in NP$, on obtient que c'est un pb NP-complet.

[4] On peut montrer facilement que BPD est NP en utilisant une descript° de l'ensemble des boîtes comme certificat.
On montre ensuite que Partition $\leqslant_p$ BPD

Soit $(x_0, \dots, x_{n-1})$ une instance de partition, on considère $(x_0, \dots, x_{n-1}) = X$, $k = 2$ et $C = \left\lfloor \frac{\sum_0^{n-1} x_i}{2} \right\rfloor$

On vérifie qu'il existe $I \subset [\![0, n-1]\!]$ t.q

$$\sum_{i \in I} x_i = \sum_{i \notin I} x_i$$ ssi on peut répartir les $\{x_i\}$ ds moins de 2 boîtes de capacité C.

5 Si on avait $v_i + v_{i+1} \leq C$ alors le contenu de B_{i+1} aurait pu (dû) être inséré ds B_i et on n'aurait pas crée la boîte B_{i+1}.

6 $$\sum_{i=0}^{m-1} v_i = V \quad \text{et} \quad m^* C \geq V$$

or $$\sum_{i=0}^{m-2} (v_i + v_{i+1}) > C(m-1) \quad \text{d'après 5}$$

$$2\left(\sum_{i=0}^{m-1} v_i\right) - v_0 - v_{m-1}$$

donc: $$2V \geq 2\underbrace{\left(\sum_{i=0}^{m-1} v_i\right)}_{=V} - v_0 - v_{m-1} > C(m-1)$$

et $$2m^* C > C(m-1)$$ (entiers)

donc $$\underline{2m^* \geq m}$$

7 Tous les objets mis avant x ainsi que x lui $\hat{m}$ sont mis ~~tous seuls dans leur boîte~~ dans des boîtes distinctes dans la solution optimale et tout seul dans leur boîte dans la solution obtenue par l'algo. Ainsi $m^* \geq$ cardinal de cette famille d'elts $= j+1$

Donc: $m^* \geq \frac{2m}{3} \Rightarrow m \leq \frac{3}{2} m^*$.

[8] Chaque elt qui est inséré ds les boîtes $B_j \to B_{m-1}$ a un volume $> v$ car sinon aurait été inséré ds la boîte en $q°$. De plus, tous ces elts ont un poids $\leq x \leq \frac{C}{2}$ (*) et donc chacune des boîtes $B_j \to B_{m-1}$ est en capacité de contenir au $\ominus$ deux elts et va donc effectivement en contenir au $\ominus$ deux sauf éventuellement B_{m-1} qui peut n'en avoir qu'un seul.

(*) x est le 1^e elt inséré ds une des $B_j \to B_{m-1}$ car de poids max ds B_j.

On obtient : $\sum_{l=j}^{m-1} v_l = v_{m-1} + \sum_{l=j}^{m-2} v_l > v + 2v(m-1-j)$

[9] On utilise de nouveau : $C m^* \geq \sum_{i=0}^{m-1} v_i$

$$> \sum_{i=0}^{j-1} v_i + v + 2v(m-1-j)$$

De plus, $\forall i \in [\![0, j-1]\!] \quad v_i \geq C - v$

$$\begin{aligned} C m^* &> j(C-v) + v + 2v(m-1-j) \\ &> jC + v(1 - j + 2m - 2 - 2j) \\ &= jC + v(\underbrace{2m - 3j - 1}_{\geq 0}) \end{aligned}$$

Sans le -1, on pourrait conclure $j < m^*$ et donc $\approx m \leq \frac{3}{2} m^*$ mais il est là donc il faut regarder précisément ce qui se passe avec les ~~~~ parties entières.

cas 1 : $m \equiv 0 [3]$ donc $j = \frac{2m}{3} \in \mathbb{N}$

On obtient $C m^* > \frac{2m}{3} C - v$

$$\Rightarrow m^* > \frac{2m}{3} - \frac{v}{C} > \frac{2m}{3} - 1$$

or $0 \leq \frac{v}{C} < 1$ car $v < C$

$$\Rightarrow m^* \geq \frac{2m}{3} \Rightarrow \boxed{m \leq \frac{3}{2} m^*}$$

cas 2 : $m \not\equiv 0 [3]$ donc $j < \frac{2m}{3} \Rightarrow 3j \leq 2m - 1$

et donc $C m^* > jC \Rightarrow m^* > j$

$$\Rightarrow m^* \geq j+1 \geq \frac{2m}{3}$$

$$\Rightarrow \boxed{m \leq \frac{3}{2} m^*.}$$

[10] Supposons qu'il existe un algo qui fournit une $\frac{3}{2} - \varepsilon$ approx pour Bin Packing en tps polynomial alors utilisons le pour résoudre partition.

Soit $x_0, \dots, x_{n-1}$ une instance de partition.

On pose $(2x_0, \dots, 2x_{n-1})$ et $C = \sum_{i=0}^{n-1} x_i$

↳ Si $x_0, \dots, x_{n-1}$ est une instance ≥ 0 de partition alors il existe une solution à Bin Packing avec deux boîtes et l'algo approché va renvoyer une valeur entre 2 et $\left(\frac{3}{2} - \varepsilon\right) \times 2 < 3$ et va donc renvoyer 2.

↳ Si $x_0, \dots, x_{n-1}$ n'est pas une instance positive de Partition alors une soluté optimale de Bin Packing vaudra au ⊖3 et donc l'algo approché renverra au ⊖3. En effet une soluté à 2 boîtes garantit que $x_0, \dots, x_{n-1}$ est positive pour partition.

Ainsi en utilisant l'algo approché pour Bin Packing, on peut décider le pb Partition suivant qu'il trouve 2 boîtes ou strictement plus en tps polynomial.
Comme Partit° est NP-complet, le fait qu'il soit dsP donnerait que $\underline{P = NP}$.

Question II.5. Programmer des fonctions `output` et `vide_liste` de prototypes respectifs `void output(void)` et `void vide_liste(void)` ; `output` affiche le mot codé par la liste de caractères sur la sortie standard (suivi d'un retour à la ligne), tandis que `vide_liste` vide la liste (retire tous ses éléments) et libère la mémoire dynamique associée. Quelle est la complexité en temps de ces deux fonctions ?