Sylvain, Raphaël
(111 124 564)

Conception et analyse d'algorithmes
IFT-3001

Travail 2

Travail présenté à
Yanick Ouellet

Département d'informatique et de génie logiciel
Univesité Laval
Hiver 2019

Sylvain, Raphaël
111 124 564

Question 2

Description

Soit un menu R où pour un item x, il y a un nombre a_x d'ailes et b_x de pintes de bière pour un coût c_x d'associé.

Définition du tableau

Le tableau M contient le prix minimum.

Définition des dimensions du tableau

La première dimension va de 0 jusqu'au nombre d'item dans le menu.
La deuxième dimension va de 0 jusqu'au nombre de d'ailes commandées.
La troisième dimension va de 0 jusqu'au nombre de bières commandées.

Définition d'une cellule

La cellule $M[i, j, k]$ contient le prix minimum pour une commande de j ailes et k bières. Elle contient l'infini si cette combinaisons de j ailes et k bières est impossible.

Conditions initiales

La cellule $M[0, 0, 0] = 0$
La cellule $M[0, j, k] = \infty$ $(\forall j, k \in \mathbb{N} | j + k > 0)$

Récurrence

$$M\left[i, j, k\right] = \begin{cases} M\left[i-1, j, k\right] & \text{si } k - b_i < 0 \\ M\left[i-1, j, k\right] & \text{si } j - a_i < 0 \\ \min\left(M\left[i, j-a_i, k-b_i\right] + c_i, M\left[i-1, j, k\right]\right) & \text{sinon} \end{cases}$$

Analyse de la fonction commande

Le temps d'exécution de l'algorithme dépend du nombre d'item n dans le menu, le nombre d'ailes a à commandées et le nombre de pintes de bières b commandées.

Cette méthode est composée de deux appels à des fonctions. Nous analyserons donc chacune des fonctions et nous pourrons donner notre réponse selon le maximum des deux.

Analyse de la fonction genererTableau

Le temps d'exécution de l'algorithme dépend du nombre d'item n dans le menu, le nombre d'ailes a à commandées et le nombre de pintes de bières b commandées.

Nous devons séparer l'analyse en plusieurs blocs, puisqu'il y a des appels de fonction.

Bloc A

Le bloc A est tout ce qui se trouve à dessus des boucles `for`.

Nous avons deux appels de fonction ici. Un appel à `vector::size` et au constructeur de `Tableau::Tableau`

Ils sont tout les deux exécuté une seule fois.

L'appel à `vector::size` ce fait en tant constant $\Theta(1)$.

L'appel à `Tableau::Tableau` ce fait en temps linéaire sur le nombre de cas du tableau. Il n'y a pas de pire cas. Dans notre cas, il se fait donc en tout temps à une complexité de

$$\begin{aligned}
& \Theta\left((n+1)*(a+1)*(b+1)\right) \\
= & \qquad \langle \text{ Étendre polynôme } \rangle \\
& \Theta\left(nab+na+nb+ab+n+a+b+1\right) \\
= & \qquad \langle \text{ Règle du maximum } \rangle \\
& \Theta\left(nab\right)
\end{aligned}$$

Donc, le bloc A a une complexité de $\Theta\left(\max\left(1+nab\right)\right)=\Theta\left(nab\right)$

Bloc B

Le bloc B est constitué de la boucle `for` et de ces sous-boucles.

L'opération de base est la comparaison `i == 0`, car c'est l'opération exécuter le plus souvent et que tout les appels de fonction se font en temps constant, incluant `Tableau::at`.

Il n'y a pas de pire cas.

Le nombre de fois que cette opération peut être exécuter nous est données par la sommation suivante :

$$
\begin{array}{ll}
 & C^B(n,a,b) \\
= & \qquad \langle \text{ Définition de la sommation selon l'algorithme } \rangle \\
 & \sum_{i=0}^{n} \sum_{j=0}^{a} \sum_{k=0}^{b} 1 \\
= & \qquad \langle \text{ Règle de sommation } \rangle \\
 & \sum_{i=0}^{n} \sum_{j=0}^{a} ((b-0+1)\cdot 1) \\
= & \qquad \langle \text{ Simplification } \rangle \\
 & \sum_{i=0}^{n} \sum_{j=0}^{a} (b+1) \\
= & \qquad \langle \text{ Règle de sommation } \rangle \\
 & \sum_{i=0}^{n} ((a-0+1)\cdot(b+1)) \\
= & \qquad \langle \text{ Simplification } \rangle \\
 & \sum_{i=0}^{n} ((a+1)\cdot(b+1)) \\
= & \qquad \langle \text{ Règle de sommation } \rangle \\
 & (n-0+1)\cdot((a+1)\cdot(b+1)) \\
= & \qquad \langle \text{ Simplification } \rangle \\
 & (n+1)\cdot(a+1)\cdot(b+1) \\
= & \qquad \langle \text{ Simplification } \rangle \\
 & nab+na+nb+ab+n+a+b+1 \\
\in & \qquad \langle \text{ Notation aymptotique } \rangle \\
 & \Theta(nab+na+nb+ab+n+a+b+1) \\
= & \qquad \langle \text{ Règle du maximum } \rangle \\
 & \Theta(nab)
\end{array}
$$

Conclusion

Puisque le bloc A a une complexité de $\Theta(nab)$ et le bloc B une complexité de $\Theta(nab)$, la fonction genererTableau a une complexité de $\Theta(nab)$, selon la règle du maximum.

Analyse de la fonction solutionnerTableau