Examen août 2016 - Question 4

Cédric Evrard

January 5, 2020

1 Analyse de l'énoncé

Il faut comprendre que lors d'une découpe, on traverse toutes de toutes les "sous-planches" qui ont déjà été découpées dans l'autre direction.

Ce nombre de planches (ou de découpe + 1) doit être multiplié par le coût de la découpe.

Il faut réaliser une découpe de chaque prix donné, dans n'importe quel ordre, pour avoir le coût total minimum.

2 Spécification du problème

Nous aurons deux tableaux, un pour les découpes verticales et un pour les découpes horizontales, contenant le nombre de découpes qui ont été faites par rapport au prix au même indice dans le tableau des prix.

Deux variables seront créées avec le nombre de découpes effectuées dans chaque sens (initialisées à 0) et une variable avec la somme totale des découpes.

Voici la spécification de la pré- et post- condition de l'algorithme[1].

Listing 1: Spécification JML de la question 4

```
1 /*@
2   @ normal_behavior;
3   @ assignable \nothing;
4   @ requires verticalPrices != null && horizontalPrices != null
5   @ requires (\forall int i; i < verticalPrices.length;
      verticalPrices[i] > 0);
6   @ requires (\forall int i; i < horizontalPrices.length;
      horizontalPrices[i] > 0);
7   @ ensures \result == (\sum int i; i < verticalCuts.length;
      verticalPrices[i] * verticalCuts[i]) + (\sum int i; i <
      horizontalCuts.length; horizontalPrices[i] *
      horizontalCuts[i]);
8   @*/
```

3 Choix et étapes de la méthode

Pour réaliser le calcul du coût le plus faible, nous allons utiliser un algorithme glouton. La contrainte sur nos découpes sera de prendre celle dont le coût est le plus élevé (que ce soit verticalement ou horizontalement.)

[1] Il manque un *ensures* pour indiquer que la solution trouvée est bien la plus petite mais je ne vois pas comment l'écrire, donc si vous avez une idée, je suis preneur :-)

En cas de découpe ayant le même coût, on choisit en premier celle dont le nombre de découpes dans le sens opposé est le plus faible.

Les étapes de la résolution peuvent donc être résumées comme ceci

1. Trie en ordre descendants des éléments des tableaux de prix.
2. Récupération de l'élément ayant le coût le plus élevé dans les tableaux
3. Multiplication du nombre de découpes par le coût
4. Ajout de la somme au total

4 Construction de l'algorithme

Listing 2: Algorithme de résolution du problème

```
1  public class Solver {
2      private static int currentVerticalCuts = 1;
3      private static int currentHorizontalCuts = 1;
4
5      private static int currentVerticalPriceIndex = 0;
6      private static int currentHorizontalPriceIndex = 0;
7
8      private static int totalPrice = 0;
9
10     public static int GetBestPrice(Integer[] verticalPrices,
           Integer[] horizontalPrices) {
11         currentVerticalCuts = 1;
12         currentHorizontalCuts = 1;
13         currentVerticalPriceIndex = 0;
14         currentHorizontalPriceIndex = 0;
15
16         totalPrice = 0;
17
18         Arrays.sort(verticalPrices, Collections.reverseOrder());
19         Arrays.sort(horizontalPrices, Collections.reverseOrder());
20
21         for (int i = 0; i < verticalPrices.length +
               horizontalPrices.length; i++) {
22             if (currentHorizontalPriceIndex ==
                   horizontalPrices.length ||
                   currentVerticalPriceIndex ==
                   verticalPrices.length) {
23                 if (currentVerticalPriceIndex <
                       verticalPrices.length) {
24                     HandleCut(verticalPrices, horizontalPrices,
                           Direction.Vertical);
25                 }
26                 if (currentHorizontalPriceIndex <
                       horizontalPrices.length) {
27                     HandleCut(verticalPrices, horizontalPrices,
                           Direction.Horizontal);
```

```
28                  }
29              } else {
30                  if (verticalPrices[currentVerticalPriceIndex] >
                       horizontalPrices[currentHorizontalPriceIndex])
                       {
31                      HandleCut(verticalPrices, horizontalPrices,
                           Direction.Vertical);
32                  } else if
                       (horizontalPrices[currentHorizontalPriceIndex]
                       > verticalPrices[currentVerticalPriceIndex]) {
33                      HandleCut(verticalPrices, horizontalPrices,
                           Direction.Horizontal);
34                  } else {
35                      if (currentVerticalCuts <=
                           currentHorizontalCuts) {
36                          HandleCut(verticalPrices,
                               horizontalPrices, Direction.Vertical);
37                      } else {
38                          HandleCut(verticalPrices,
                               horizontalPrices,
                               Direction.Horizontal);
39                      }
40                  }
41              }
42          }
43
44          return  totalPrice;
45      }
46
47      private static void HandleCut(Integer[] verticalPrices,
           Integer[] horizontalPrices, Direction direction) {
48          switch (direction) {
49              case Vertical:
50                  totalPrice +=
                       verticalPrices[currentVerticalPriceIndex] *
                       currentHorizontalCuts;
51                  currentVerticalCuts++;
52                  currentVerticalPriceIndex++;
53                  break;
54              case Horizontal:
55                  totalPrice +=
                       horizontalPrices[currentHorizontalPriceIndex]
                       * currentVerticalCuts;
56                  currentHorizontalCuts++;
57                  currentHorizontalPriceIndex++;
58                  break;
59          }
60      }
61  }
```

5 Ordre de grandeur de l'algorithme

Les deux tris s'effectuent en un temps $\mathcal{O}(nlogn)$ (Ligne 18- Listing 2) et $\mathcal{O}(mlogm)$ (Ligne 19- Listing 2).

Ensuite, nous avons une boucle principale (Ligne 23 - Listing 2). Cette boucle ne contient que des opérations qui sont effectuées en $\mathcal{O}(1)$.

La boucle prend un temps $\mathcal{O}(m+n)$, qui est l'addition de la longueur des deux tableaux des prix.

Nous avons donc un algorithme qui possède une complexité temporelle de $\mathcal{O}(nlogn)$ ou n est la valeur du plus long tableau.

Pour l'espace en mémoire, nous avons les deux tableaux fourni qui possèdent la liste des prix ainsi que 5 variables entières. Nous pouvons donc dire que nous avons une complexité en place mémoire de $\mathcal{O}(m+n)$ où m et n ont la même signification que pour la complexité temporelle.