Année 2019-2020

Portail Mathématiques, Informatique
Licence deuxième année (L2 Info, L2 maths, L2 MIASHS)

Algorithmique et structures de données
TD 1 – les tableaux

Exercice 1. **Suivi d'un trailer**
Nous souhaitons développer une application pour pouvoir suivre le parcours d'une personne sur un trail. Pour simplifier, nous supposerons que les points de passage sont désignés par des numéros de 0 à $n-1$, nous avons donc n points de passage au total. Le point 0 correspond à la ligne de départ et le point $n-1$ à la ligne d'arrivée.

On dispose de trois tableaux Distance, Altitude, Temps. Les tableaux Distance et Altitude donnent des informations générales connues avant le départ de la compétition.

Le tableau Distance donne la distance parcouru entre deux points de passage. Distance[i] donne ainsi la distance entre les points de passage $i-1$ et i. Par convention, nous mettrons 0 pour Distance[0].

Le tableau Altitude donne l'altitude à un point de passage. Altitude[i] est l'altitude au point de passage i.

Le troisième tableau Temps est différent pour chaque trailer. Il indique l'heure de passage à chaque point de passage.

On supposera que tous les tableaux sont des tableaux d'entiers.

Question 1. Écrivez une procédure *distancePartielle* qui prend en argument le tableau Distance, deux points de passage i et j, $i < j$, et retourne la distance pour aller du point i au point j.

Question 2. Écrivez une procédure *tempsPartiel* qui prend en argument le tableau Temps, deux points de passage i et j, $i < j$, et retourne le temps mis pour aller du point de passage i au point de passage j.

Question 3. Utilisez ces deux procédures précédentes pour écrire une procédure *vitessePartielle* qui prend en argument les tableaux Distance et Temps deux points de passage i et j, $i < j$, et retourne la vitesse de la personne entre les points de passage i et j.

On suppose qu'entre deux points de passage nous avons soit uniquement une montée (Altitude[i] - Altitude[$i-1$] > 0), soit uniquement une descente (Altitude[i] - Altitude[$i-1$] < 0). On appelle dénivelé positif la somme des montées.

Question 4. Écrivez une procédure *deniveléPositif* qui prend en argument le tableau Altitude et l'entier n et retourne le dénivelé positif.

On appelle plus grande montée du trail la plus grande différence Altitude[j] - Altitude[i] pour des entiers $0 \leq i < j \leq n-1$ tels qu'il n'y a que des montées entre i et j, c'est-à-dire Altitude[i] <Altitude[$i+1$] < ... <Altitude[j].

Question 5. Écrivez une procédure qui prend en argument le tableau Altitude et l'entier n et retourne la plus grande montée.

Exercice 2. On considère la procédure **mystere** suivante.

Procédure 1 *mystere*(n : entier) : entier

```
si n=0 alors
    retourner 2
sinon
    retourner mystere(n-1)*mystere(n-1)
```

Question 1. Quelle fonction mathématique est calculée par **mystere** ?

Question 2. On définit le coût de la procédure **mystere** comme étant le nombre de multiplications, on le note $C(n)$. Montrez par récurrence que l'on a $C(n) = 2^n - 1$.

Question 3. Proposez une procédure ayant un meilleur coût, donnez ce coût.

Exercice 3. **Impact de l'organisation des données sur le coût d'une recherche**

Question 1. Écrivez une procédure *rechercheElement(tab : tableau d'entiers, taille : entier, x : entier) : entier* qui renvoie la position du premier élément x dans tab si x est présent et -1 sinon.

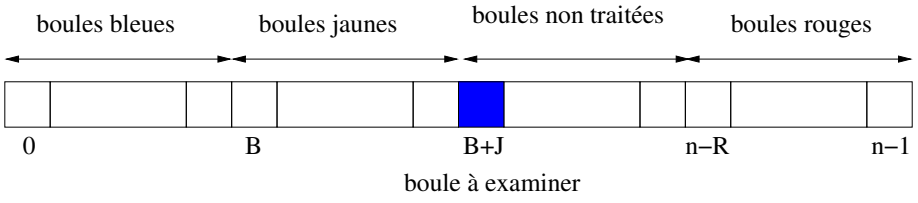
Question 2. On définit pour coût de *rechercheElement* le nombre d'éléments de tab examinés par *rechercheElement*. Calculez le coût maximum de *rechercheElement* en fonction de *taille*, le nombre d'éléments de *tab*.

Question 3. On suppose maintenant que le tableau est trié. Proposez un algorithme dichotomique pour rechercher un élément x. Donnez le coût maximum de ce nouvel algorithme dans le cas où $n = 2^k - 1$. Exécutez l'algorithme sur un exemple, par exemple $n = 15$.

Exercice 4. Drapeau hollandais

On dispose d'un tableau contenant des boules de trois couleurs : bleues, jaunes et rouges. On souhaite définir un algorithme qui trie ce tableau afin d'avoir d'abord toutes les boules bleues, puis toutes les boules jaunes et finalement toutes les boules rouges. Le coût de l'algorithme est le nombre de fois que l'algorithme regarde la couleur d'une boule, c'est-à-dire le nombre de fois que l'algorithme accède à un élément du tableau.

Notre objectif est de ne regarder qu'une seule fois la couleur de chaque boule. La méthode consiste à conserver à chaque étape la configuration suivante : on met au début toutes les boules bleues déjà rencontrées, ensuite toutes les boules jaunes déjà rencontrées, puis toutes les boules non traitées et, en dernier, toutes les boules rouges déjà rencontrées. La figure ci-dessous illustre cet arrangement, où n est le nombre total de boules, B (resp. J et R) est le nombre de boules bleues (resp. jaunes et rouges) déjà rencontrées.



Question 1. Déterminez selon la couleur de la boule examinée les instructions à effectuer.

Question 2. Écrivez l'algorithme de tri.

Question 3. Quel est le coût de cet algorithme ? Peut-on faire mieux ?