Travaux Dirigés d'algorithmique n°4

Cours d'algorithmique

—Licence CFA / Troisième Année—

▸ **Exercice 1. (Recherches et accumulations dans les tableaux)**

On dispose d'un tableaux d'entiers `tab` de taille `taille`. Écrire un algorithme pour résoudre les problèmes suivants ; donner la complexité de chaque algorithme.

1. Calculer le nombre d'entiers pairs dans le tableaux ;
2. Le tableau contient-il un entier pair ?
3. Étant donné un entier `i`, trouver un l'entier le plus proche de `i` dans le tableau.
4. Le tableau est-il trié ? C'est-à-dire, l'assertion suivante est-elle vrai ?

$$\text{pour tout } \; i < \texttt{taille} - 1, \quad \text{on a} \quad \texttt{tab}[i] > \texttt{tab}[i+1] \, .$$

▸ **Exercice 2. (Manipulations de base des tableaux)**

On travaille sur les données suivantes : `tab` est tableau pour lequel on a alloué `max_taille` éléments ; Les éléments $0 \leq i < \texttt{taille} \leq \texttt{max\_taille}$ sont initialisés, les autres sont considérés comme inutilisés.

Écrire un algorithme pour résoudre les problèmes suivants ; donner la complexité.

1. insertion d'un élément à la fin ;
2. insertion d'un élément en position i ;
3. insertion d'un élément au début ;
4. suppression d'un élément à la fin ;
5. suppression d'un élément en position i ;
6. suppression d'un élément au début ;

▸ **Exercice 3. (Tri par bulles)**

On dispose d'un tableaux d'entiers `tab` de taille `taille`. On cherche à trier le tableau. Pour ceci, on peut utiliser l'algorithme suivant : on parcourt le tableau en cherchant les positions i telles que

$$0 \leq i < \texttt{taille} - 1 \qquad \text{et} \qquad \texttt{tab}[i] > \texttt{tab}[i+1] \, .$$

Quand on a trouvé un tel i, on échange $\texttt{tab}[i]$ et $\texttt{tab}[i+1]$ et ensuite on continue le parcours. On recommence tant que le tableau n'est pas trié.

1. Écrire l'algorithme du tri par bulle.
2. En fait, on se rend compte que les éléments qui ne sont pas déplacés à la fin d'un parcourt, ne seront jamais plus déplacés ensuite. Modifier le tri en tenant compte ce cette remarque à l'aide d'un index qui note le dernier élément à parcourir.
3. On peut encore améliorer le tri par bulle en parcourant alternativement dans un sens et dans l'autre. Écrire l'algorithme correspondant. On utilisera deux indices, pour le premier et le dernier éléments à parcourir.
4. Pour chacune des trois versions, dire quelle est le cas le pire et la complexité dans ce cas.

▸ **Exercice 4. (Implantation d'une pile par tableau)**

Une pile est une structure de donnée qui enregistre des informations selon le mode dernier entré premier sorti (LIFO : Last In First Out). On manipule une pile en utilisant les quatres opérations suivantes :

Création d'une nouvelle pile	PileVide() : Pile	
Teste si la pile est vide	EstVide(Pile) : Booleen	
Ajout d'un élément	Empiler(T, Pile)	modifie la pile
Suppression d'un élément	Depiler(Pile) : T	modifie la pile

Note : Depiler(p) est valide seulement si non EstVide(p).

1. En utilisant un tableau proposer une structure permettant de stocker une pile ; Dans un premier temps, on supposera que la taille est limitée.
2. Écrire précisément les invariants de la structure.
3. Écrire les quatres fonctions précédentes ;
4. Quelle est leur complexité.
5. Comment faire pour que la taille ne soit plus limité sans perdre en complexité.

▸ **Exercice 5. (Implantation d'une file par tableau)**

Une file est une structure de donnée qui enregistre des informations selon le mode premier entré premier sorti (FIFO : First In First Out). On manipule une pile en utilisant les quatres opérations suivantes :

Création d'une nouvelle file	FileVide() : File	
Teste si la pile est vide	EstVide(File) : Booleen	
Ajout d'un élément	Enfiler(T, File)	modifie la file
Suppression d'un élément	Defiler(File) : T	modifie la file

Note : Defiler(p) est valide seulement si non EstVide(p).

1. Mêmes questions que pour les piles.