Université Paris Cité

L2 informatique - Année 2023–2024

TD d'Éléments d'Algorithmique n° 3

Dans ce TD, "trié" signifie "trié par ordre croissant".

*** Les exercices marqués d'une étoile sont à faire à la maison.**

Exercice 1. *Algorithme du cours.*

Appliquez l'algorithme du tri par insertion sur les tableaux suivants :

8	4	10	-5	1
1	2	3	4	5
5	4	3	2	1

Comptez dans chaque cas le nombre de comparaisons effectuées, ainsi que le nombre d'affectations à des cases du tableau.

Exercice 2. *Tri insertion selon la parité.*

On veut trier un tableau d'entiers de telle manière que les entiers pairs apparaissent dans la première partie du résultat, triés, suivis des entiers impairs, triés. Par exemple :

2	1	6	8	5	-3

devient :

2	6	8	-3	1	5

Adaptez l'algorithme de tri par insertion à ce cas.

Exercice 3. *Trier des cartes.*

Une carte contient deux informations, une couleur qui est un entier compris entre 1 et 4 (1 pour pique, 2 pour cœur, 3 pour carreau et 4 pour trèfle) et un rang qui est un entier entre 1 et 13. Le dix de pique sera représenté par [1,10], l'as de carreau par [3,1]. Un ensemble de cartes sera donc représenté par un tableau de tableaux d'entiers.

Lorsqu'on trie un jeu de cartes, on met les cartes ayant la même couleur ensemble — pique avant cœur, cœur avant carreau, etc.

En vous inspirant des exercices précédents, donnez un algorithme qui trie un tableau de cartes.

Exercice 4. *Tri stable.*

En algorithmique, on trie habituellement des entiers. En programmation, on trie généralement des structures plus compliquées. Par exemple, en Java on pourrait avoir une structure `Etudiant` :

```
public class Etudiant {
    public String nom;
    public int note;
}
```

et utiliser l'ordre défini par : `e1` $\leq$ `e2` lorsque `e1.note` $\leq$ `e2.note`.

1. Montrez que l'ensemble des étudiants muni de la relation définie ci-dessus n'est pas un ordre. (Indication : exhibez deux éléments distincts `e1` et `e2` qui sont *équivalents* pour l'ordre, c'est-à-dire tels que `e1` $\leq$ `e2` et `e2` $\leq$ `e1`.) Un tel espace s'appelle un *préordre*.

2. On dit qu'un tri est *stable* lorsque'il préserve l'ordre relatif des éléments équivalents : si `e1` était placé avant `e2` dans le tableau d'origine, et `e1` et `e2` sont équivalents, alors `e1` est encore avant `e2` dans le tableau trié. Le tri par insertion est-il stable ? Le tri par selection est-il stable ?
3. Quand cette propriété est-elle importante ?

Exercice 5. *Lecture d'algorithme.*

T est un tableau d'entiers de longueur n.

```
1  Fonc (T) :
2  i=0;
3  while i < n-1 {
4    if T[i] > T[i+1] {
5      exchange T[i] and T[i+1];
6      i=0;
7    } else {
8      i++;
9    }
10 }
```

1. Exécuter cette fonction sur | 3 | 1 | 2 | 0 |
2. Que fait `Fonc(T)` ?
3. Combien de comparaisons sont effectuées ?

Exercice 6. *Fusion de tableaux triés *.*

On suppose que notre tableau est constitué de deux tableaux collés l'un à l'autre dont chacun est lui-même trié. Par exemple :

T : | 2 | 9 | 4 | 6 | 7 | 10 |

1. Donnez un algorithme qui prend en entrée un tableau qui a cette propriété et retourne un nouveau tableau trié, qui contient les mêmes éléments. (Remarquez que la frontière entre les deux tableaux n'est pas donnée.)
2. Combien de comparaisons sont effectuées dans le pire des cas ?