Université Paris Diderot - L2 informatique Année 2015–2016

TD d'Éléments d'Algorithmique n° 2

Dans tout le TD comme on dira que des élements d'un tableau sont triés, il faudra comprendre "triés par ordre croissant".

Exercice 1. *Tri Bourrin.*

```
1   pour tout indice i entre 0 et n-1 faire
2        pour tout indice j entre i+1 et n-1 faire
3             si T[i]>T[j] alors echange T[i] et T[j]
4             fin si
5        fin faire
6   fin faire
```

1. Cet algorithme vous semble-t-il valide pour trier un tableau de taille `n` ? A quel algorithme du cours pourriez vous le comparer ?
2. Que dire de son nombre de comparaisons ? son nombre d'affectations ?

Exercice 2. *Tri par Insertion dichotomique.*

On se propose ici d'écrire une variante du tri par insertion vu en cours en améliorant la phase d'insertion. On rappelle le principe du tri par insertion : à chaque étape de la boucle, on suppose que les k premiers éléments sont déjà triés et on cherche à trouver où insérer le $(k+1)$-ième élément, on l'insère alors en décalant vers la droite les éléments plus grands.

L'insertion dichotomique fonctionne ainsi : on commence par regarder si T[k] doit être inséré entre l'indice 0 et k/2 ou entre l'indice k/2 et k-1. Une seule comparaison suffit pour faire ça (laquelle ?). Ensuite on recommence en coupant de nouveau en deux le sous tableaux obtenu, et ainsi de suite, jusqu'à qu'on ait exactement identifié la bonne position pour T(k).

1. Effectuer à la main l'insertion dichotomique de `T[8]` dans le sous-tableau `T[0]..T[7]`.

T =	2	6	12	13	16	18	28	30	3

2. Ecrire une algorithme qui fait l'insertion dichotomique du k-ième élément de façon itérative. La fonction prendra en entrée T et k et renverra l'entier correspondant à la bonne position de T(k).
3. Ecrire une algorithme qui fait la même chose de façon récursive.
4. Combien de comparaisons va t'on effectuer pour cette insertion ? Combien d'affectations ?

Exercice 3. *Tri Caillou et Tri Shaker.*

Vous avez vu le tri à bulles. Voici une variante, qui trie un tablesu `T` de taille `n`.

```
TRI_CAILLOU(T)

1   debut<- 0
2   k <- n-1
3   tant que(debut<n-1) faire
4      tant que k>debut faire
5         si T[k] > T[k-1] alors echanger T[k] et T[k-1]
6         fin si
7         k <- k-1
8      fin faire
9      debut <- debut+1
10     k <- n-1
11  fin faire
```

1. Comment le comparer au tri à bulles ?
2. Il peut arriver que le tableau soit déjà trié avant même que `debut` soit égal à `n-1`. Donner un petit exemple de tableau où cela se produit.
3. Montrer comment en ajoutant une variable de test on peut corriger ce problème et faire en sorte que la procédure s'arrête dès que le tableau est trié.
4. On se propose d'étudier une variante de tri à bulles et tri caillou appelée tri shaker. L'idée est de balayer alternativement de la gauche vers la droite (comme dans bulles) puis de la droite vers la gauche (comme dans caillou). Ecrire l'algorithme correspondant.
5. En quoi cet algorithme vous semble-t-il meilleur ?

Exercice 4. *Tri de Permutation.*

On considère les procédures suivantes :

```
Retourne(T,g,d)
1   Tant que (g<d) faire
2           c <- T[g]
3           T[g] <- T[d]
4           T[d] <- c
5           g++
6           d--
7   fin faire
**
PlaceZero(T)
1   Tant que T[0]!=0 faire
2        Retourne(T,0,T[0])
3   fin faire
```

On applique la procedure `PlaceZero` à un tableau de taille n contenant tous les nombres de 0 à $n-1$. Montrer que la procédure termine. Donner une borne sur sa complexité.

Exercice 5. *Tri Bizarre.*

On considère l'algorithme suivant sur un Tableau T et deux indices i et j dans ce tableau.

```
Tri_Bizarre (T,i,j)
1       si T[i] > T[j] alors échanger T[i] et T[j]
2       si i <= j-2 alors faire
3           k <- (j-i+1)/3
4           Tri_Bizarre (T,i, j-k)
5           Tri_Bizarre (T,i+k, j)
6           Tri_Bizarre (T,i, j-k)
```

Montrer que `Tri_Bizarre(T,0,n-1)` trie correctement T de longueur n.