Exercices : Algorithmes de tri sur les tableaux

Exercice 1

Compléter les lignes vides correspondant aux différentes étapes de la méthode de tri par insertion.

Tableau initial	59	46	54	89	17	65	66	12	34	22
Passage 1 dans la boucle	46	59	54	89	17	65	66	12	34	22
Passage 2 dans la boucle										
Passage 3 dans la boucle										
Passage 4 dans la boucle										
Passage 5 dans la boucle										
Passage 6 dans la boucle										
Passage 7 dans la boucle										
Passage 8 dans la boucle										
Passage 9 dans la boucle										

Exercice 2

On rappelle ci-contre l'algorithme du tri par insertion.
On l'exécute sur un tableau de taille 8.

```
i = 1
while i < taille(tableau):

    valeur_a_inserer = tableau[i]

    j = i
    while j > 0 and valeur_a_inserer < tableau[j-1] :
        T[j] = T[j-1] (♦)
        j = j - 1

    T[j] = valeur_a_inserer

    i = i + 1
```

1. Dans le pire des cas, combien de fois est exécutée l'instruction (♦) ?
   On pourra s'aider du schéma d'une matrice 8 x 8 à colorier (avec i = indice de ligne et j = indice de colonne).

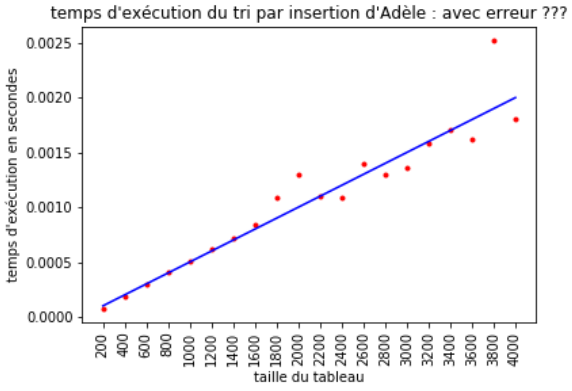
2. Pour quels tableaux de taille 8 le pire des cas se produit-il ?

3. Peut-on trouver un tableau de taille 8 tel que l'instruction (♦) n'est jamais exécutée ?

4. D'après le cours, quel est le coût dans le pire des cas pour le tri par insertion ?

   Adèle a testé un programme implémentant l'algorithme de tri par insertion.
   Pour cela elle a d'abord généré des grands tableaux de nombres entiers à donner en entrée à a son programme.

   Puis elle a chronométré les temps d'exécution de son programme en focnction de la taille du tableau donné en entrée. Elle a obtenu le nuage de points ci-contre.



   Ces temps d'exécution semblent linéaires en fonction de la taille du tableau. Proposer une explication.

Exercice 3

Compléter les lignes vides correspondant aux différentes étapes de la méthode de tri par sélection.

Tableau initial	53	66	34	89	17	65	66	12	34	22
Passage 1 dans la boucle	12	66	34	89	17	65	66	53	34	22
Passage 2 dans la boucle										
Passage 3 dans la boucle										
Passage 4 dans la boucle										
Passage 5 dans la boucle										
Passage 6 dans la boucle										
Passage 7 dans la boucle										
Passage 8 dans la boucle										
Passage 9 dans la boucle										
Passage 10 dans la boucle										

Exercice 4

On rappelle ci-contre l'algorithme du tri par sélection.

On l'exécute sur un tableau de taille 8.

```
i= 0
while i < taille(tableau):

    j_min = i
    j = j_min + 1

    while j < taille(tableau) :
        if tableau[j] < tableau[j_min] :
            j_min = j
        j = j + 1 (♦)

    tableau[i], tableau[j_min] = tableau[j_min], tableau[i]
    i= i + 1
```

1. Dans le pire des cas, combien de fois est exécutée l'instruction (♦) ?
   On pourra s'aider du schéma d'une matrice 8 x 8 à colorier (avec i = indice de ligne et j = indice de colonne).

2. Pour quels tableaux de taille 8 le pire des cas se produit-il ?

3. Peut-on trouver un tableau de taille 8 tel que l'instruction (♦) n'est jamais exécutée ?

Exercice 5

Sur sa machine, Émilie a implémenté un algorithme prenant en entrée des tableaux de taille n et ayant un coût quadratique dans le pire des cas.
On admet que le temps d'exécution est quadratique comme le coût et qu'il n'y a pas de problème de surcharge de la mémoire.(*)

1) Sur un tableau d'entiers de taille 250 son algorithme s'est exécuté en 90 ms.
   Quel sera approximativement le temps d'exécution sur un tableau d'entiers de taille 50 000 ?

2) Pour quelle taille de tableau d'entiers l'exécution prendra-t-elle environ une journée ?

3) Émilie effectue un test avec un tableau d'entiers de taille 250 000. Le temps d'exécution est de 2,324 secondes. Donner une explication possible.

(*) : *en réalité on ne passe pas aussi simplement de la complexité algorithmique au temps d'exécution sur machine.*