fiche 6

Tableaux et tris

Le but du TD est de déterminer le coût temporel d'algorithmes simples sur les tableau. Pour montrer le coût par des exemples, on pourra créer un tableau aléatoire de taille n en utilisant la fonction suivante :

```
from random import randint

def tableau_alea (n) :
    tab =[ randint(1,1000) for i in range(n)]
    return tab
mon_tab = tableau_alea (1000000)
```

De plus, pour mesurer les différents coûts, on pourra utiliser le module `time` :

```
def temps_fonction (fonction,parametre) :
    import time

    # Debut du decompte du temps
    start_time = time.time()

    # affichage du resultat de la fonction

    if len(parametre) == 2 :
        print(fonction(parametre[0],parametre[1]))
    else :
        print(fonction(parametre))
    # affichage du temps d execution
    print("Temps d execution : %s secondes ---" % (time.time() -
   start_time))
```

(Attention à ne pas prendre en compte le temps de construction d'un tableau aléatoire...)

1 Parcours dans un tableau

Exercice 1 :

1. Écrire la fonction `occurence` qui prend en argument un tableau et un élément, et renvoie le nombre de fois où l'élément apparait dans le tableau.
2. Déterminer le nombre de boucle nécessaire pour effectuer le comptage.
3. Tester la fonction avec des tableaux de taille proportionnelle.

```
taille = 100000
t_1 = tableau_alea(taille)
t_2 = tableau_alea(taille*10)
t_3 = tableau_alea(taille*20)
temps_fonction (occurrence,(t_1, 200))
temps_fonction (occurrence,(t_2, 200))
temps_fonction (occurrence,(t_3, 200))
```

Exercice 2 :

1. Écrire la fonction `stat` qui prend en argument un tableau et renvoie un dictionnaire dans lequel on récupèrera :
   - Le nombre de terme.
   - La somme des termes.
   - La moyenne.
   - Le minimum.
   - Le maximum.

   Exemple :
   ```
   {'nombre': 2000000, 'somme': 1000810163, 'moyenne': 500.4050815,
       'minimum': 1, 'maximum': 1000}
   ```
2. Déterminer le nombre de boucle nécessaire pour effectuer le comptage.
3. Tester la fonction avec des tableaux de taille proportionnelle.
   ```
   taille = 100000
   t_1 = tableau_alea(taille)
   t_2 = tableau_alea(taille*10)
   t_3 = tableau_alea(taille*20)
   temps_fonction (stat,t_1)
   temps_fonction (stat,t_2)
   temps_fonction (stat,t_3)
   ```

2 Tris d'un tableau

2.1 Tris par sélection

Exercice 3 :

1. Écrire la fonction `echange` qui prend en argument un tableau et deux indices i et j, et la fonction change la position des éléments du tableau d'indice i et j. Exemple :
   ```
   >>> t = [1,2,3,4,5,6]
   >>> echange(t,2,4)
   >>> t
   [1, 2, 5, 4, 3, 6]
   ```
2. Écrire la fonction `mini_i` qui retourne l'indice du minimum d'un tableau à partir de l'indice i.
   Exemple :
   ```
   >>> mini_i([8,3,4,1,7,9,2], 2)
   3
   >>> mini_i([8,3,4,1,7,9,2], 5)
   6
   ```

3. Écrire la fonction `tri_select` qui prend un tableau, et trie dans l'ordre croissant le tableau par la méthode du tri par sélection.

4. Tester la fonction avec des tableaux de taille proportionnelle.

```
taille = 100
t_1 = tableau_alea(taille)
t_2 = tableau_alea(taille*10)
t_3 = tableau_alea(taille*100)
temps_fonction (tri_select,t_1)
temps_fonction (tri_select,t_2)
temps_fonction (tri_select,t_3)
```

2.2 Tris par insertion

Exercice 4 :

1. Écrire la fonction $\texttt{insere}_\texttt{i}$ qui prend en argument un tableau et un entier i. L'entier correspond à l'indice à partir duquel le tableau n'est plus trié. (le tableau est argument est donc trié dans le sens croissant entre les indices 0 et $i-1$.)
Le but de la fonction est de placer l'élément est position i au bon endroit dans les $i+1$ premiers élément du tableau. (c'est l'insertion...) Exemple : Dans l'exemple suivant, le tableau `t` est trié jusqu'à l'indice 4. (les 5 premiers termes...)
La fonction va donc procéder au tri des 6 premiers éléments, en insérant le terme d'indice 5 au bon endroit.

```
>>> t = [2,5,9,12,15,7,5,6,4,8,]
>>> inser_i(t,5)
>>> t
[2, 5, 7, 9, 12, 15, 5, 6, 4, 8]
```

2. Écrire la fonction `tri_insert` qui prend un tableau, et trie dans l'ordre croissant le tableau par la méthode du tri par insertion.

3. Tester la fonction avec des tableaux de taille proportionnelle.

```
taille = 100
t_1 = tableau_alea(taille)
t_2 = tableau_alea(taille*10)
t_3 = tableau_alea(taille*100)
temps_fonction (tri_insert,t_1)
temps_fonction (tri_insert,t_2)
temps_fonction (tri_insert,t_3)
```

2.3 Tri à bulle

Exercice 5 : On considère l'algorithme suivant :

```
def tri_bulle ( tab) :
    n = len(tab)
    inversion = True
    while inversion :
        inversion = False
        for i in range(n-1) :
            if tab[i]>tab[i+1] :
                echange (tab, i , i+1)
                inversion = True
    return tab
```

1. Tester la fonction avec la tableau $[5, 9, 12, 15, 20]$.
2. Déterminer le nombre d'étapes nécessaires.
3. Tester la fonction avec la tableau $[15, 13, 12, 8, 5]$.
4. Déterminer le nombre d'étapes nécessaires.