Soutien en algorithmique et programmation

Séance 5 : réorganisations de tableaux

Introduction

On va travailler sur un algorithme de réorganisation de tableau bien connu : le pivot. On pourra utiliser la fonction principale ci-dessous pour tester les fonctions à écrire, en commentant ce qui n'est pas encore implanté.

```
def main():
    """
      Fonction principale
    """
    tab = [randint(0, 10) for _ in range(TAILLE)]
    idx_pivot = randint(0, TAILLE - 1)
    sav_tab, sav_piv = tab[:], idx_pivot
    print("Tableau initial :", tab)
    print("Indice du pivot : ", idx_pivot, ", valeur du pivot : ",
          tab[idx_pivot], sep="")
    print("Pivot en place :")
    idx_pivot = pivot(tab, idx_pivot)
    print("  indice du pivot :", idx_pivot)
    print("  tableau réorganisé :", tab[0:idx_pivot + 1], tab[idx_pivot + 1:])
    print("Drapeau en place :")
    tab, idx_pivot = sav_tab[:], sav_piv
    deb_piv, fin_piv = drapeau(tab, idx_pivot)
    print("  indices du pivot : de", deb_piv, "à", fin_piv)
    print("  tableau réorganisé :", tab[0:deb_piv], tab[deb_piv:fin_piv + 1],
          tab[fin_piv + 1:])
```

Algorithme du pivot

Le principe de cet algorithme est simple :

- on choisit un élément du tableau (souvent au hasard) et on décide qu'il devient la valeur pivot ;
- on réorganise tous les éléments du tableau de façon à ce qu'à la fin tous les éléments inférieurs ou égaux au pivot se situent dans la partie gauche du tableau, et tous les éléments strictement supérieurs au pivot se situent dans la partie droite. Les deux parties seront finalement délimitées par le pivot lui-même.

On doit donc écrire une fonction `pivot(tab, idx_pivot)` qui prend en paramètre le tableau ainsi que l'**indice** de la valeur choisie comme pivot, et qui a la fin de son exécution, renverra le nouvel indice du pivot (car il aura surement changé de place dans le tableau).

On devra travailler **en place** c'est à dire en procédant uniquement à des échanges d'éléments, sans allouer de nouveaux tableaux ou cellules.

Pour ce type d'exercice, il est intéressant de faire des dessins détaillant :

- l'état du tableau avant le début de la boucle parcourant le tableau ;
- l'état du tableau à la fin de la boucle ;
- l'état du tableau à la $i^{ème}$ itération, et ce qu'il faut faire pour passer à la $(i+1)^{ème}$ itération tout en préservant l'invariant de la boucle.

Algorithme du drapeau

On va maintenant écrire une deuxième version, que l'on appelera le drapeau, qui ressemble beaucoup à l'algorithme du pivot. La différence est qu'à la fin de la boucle, le tableau doit être divisé en 3 zones :

- la partie de gauche contenant les éléments strictement inférieurs au pivot ;
- la partie du milieu contenant les éléments égaux au pivot ;
- la partie de droite contenant les éléments strictement supérieurs au pivot.

La fonction à écrire a pour prototype `def drapeau(tab, idx_pivot)` et renvoie deux indices :

- `deb_piv` est l'indice de la première case contenant une valeur égale au pivot ;
- `fin_piv` est l'indice de la dernière case contenant une valeur égale au pivot.

Là-encore, on travaillera **en place** sans allouer de tableau intermédiaire.