Tris par Selection

1 Introduction

Le tri par sélection (ou tri par extraction) est un algorithme de tri par comparaison. Cet algorithme est simple, mais considéré comme inefficace car il s'exécute en temps quadratique (voir Fig. 1) en le nombre d'éléments à trier (n) : $O(n^2)$. C'est-à-dire que le temps de calcul est lié au carré du nombre d'élément à trier n.

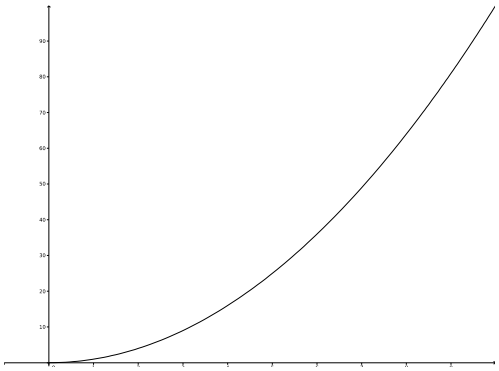

Fig. 1: Fonction quadratique f(x) = x²

2 Le concept

Dans le tris par selection, on inverse successivement les plus petits élements avec les cases du début du tableau (voir Fig. 2).

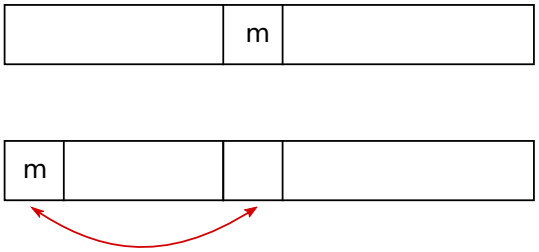

Fig. 2: schéma de l'inversion de deux valeurs

On peut décomposer cela en étapes :

1. On recherche le plus petit élément du tableau et le place à la première position.
2. On recherche le deuxième élément le plus petit et le place en deuxième position.
3. On continue de la même façon jusqu'à ce que le tableau soit entièrement trié.

3 Exemple

Tableau à trier : [10, 9, 5, 7, 3]

#1	10 |	9	5	7	3	Tableau à trier
#2	3 |	9	5	7	10	3 est l'élément le plus petit. On l'échange avec 10.
#3	3	5 |	9	7	10	On échange 5 et 9. Partie triée : [3, 5]
#4	3	5	7 |	9	10	On échange 7 et 9. Partie triée : [3, 5, 7]
#5	3	5	7	9 |	10	On échange rien. Partie triée : [3, 5, 7, 9]
#6	3	5	7	9	10 |	On échange rien. Partie triée : [3, 5, 7, 9, 10]. ⇒ Le tableau est trié.

4 Algorithme

L'algorithme du tris par sélection est composé de deux parties : La *recherche de minimum dans la partie non triée du tableau* et le *tris* en lui même.

```
Variables :
tab : tableau d'entiers
tab_len : entier
i_min, i_trie, tmp : entiers
Algorithme :      Trie par sélection
i_trie = 0;
i_min = 0;
tmp = 0;
tab_len = taille(tab);
Pour i_trie =0; i_trie < tab_len; i_trie++:
      i_min = minimum(tab, i_trie);
      Si i_min != i_trie Alors:
            tmp = tab[i_min];
            tab[i_min]  = tab[i_trie];
            tab[i_trie] = tmp;
      Fin Si;
Fin Pour;
Fin Algorithme;
```

```
Variables :
tab : tableau d'entiers
tab_len : entier
i_min, val_min, k : entiers
Algorithme :      fonction minimum(tab, i_trie)
k = 0;
i_min = i_trie;
val_min = tab[i_trie];
tab_len = taille(tab);
Pour k = i_trie+1; k < tab_len; k++:
      Si tab[k] < val_min Alors:
            val_min = tab[k];
            i_min = k;
      Fin Si;
Fin Pour;
Renvoie i_min;
Fin Algorithme;
```

5 Exercice(s)

1. Coder en python l'algorythme de tri par sélection. Il sera codé sous la forme de deux fonctions python : tri_selec(tab) et min_tab(tab,i_trie) pour, respectivement, l'algorithme de tri et la fonction de recherche de minimum.
2. Tester l'algorythme pour un tableau contenant 10, 50, 100, 500, 1000 et 5000 valeurs entières aléatoires.
3. Relever les temps d'exécutions des exécutions précédentes. Utiliser la fonction *time()* du module *time.*
4. Tracer un graphique qui représente les temps d'exécutions par rapport au nombre d'éléments.