

Tris par Insertion

1 Introduction

Le tri par insertion est un algorithme de tri classique. La plupart des personnes l'utilisent naturellement pour trier des cartes à jouer.

Cet algorithme est simple, mais considéré comme inefficace seul, sauf si les données sont déjà presque triées. Il est utilisé en pratique en combinaison avec d'autres méthodes comme le tri rapide. Il s'exécute en temps quadratique (voir Fig. 1) par rapport au 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 .

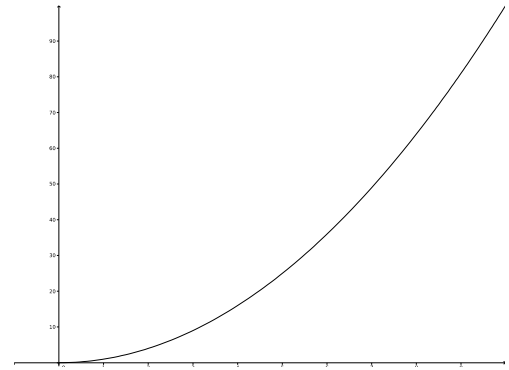


Fig. 1: Fonction quadratique
 $f(x) = x^2$

2 Le concept

On parcourt les éléments à trier du début à la fin. À chaque étape, on considère une case d'indice i du tableau, tous les éléments précédents l'indice i sont déjà triés. L'objectif de l'étape est d'insérer la valeur de la case i à sa place parmi les éléments précédents. Il faut trouver où l'élément doit être inséré en le comparant aux autres puis décaler les éléments pour pouvoir l'insérer. En pratique, on effectue les deux étapes en une passe, on dit que l'on fait *remonter* l'élément petit à petit jusqu'à rencontrer un élément plus petit (voir Fig. 2). Afin de trier tout le tableau, le premier indice i sera la seconde case du tableau, 1.

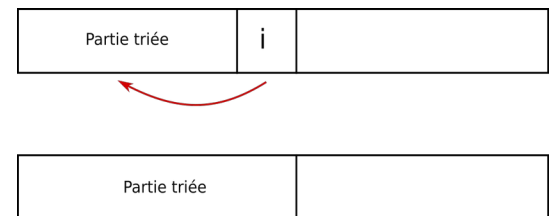


Fig. 2: schéma de la remontée d'une valeur

3 Exemple

Tableau à trier : $tab = [6, 5, 3, 1, 8, 7, 2, 4]$

$i = 1$:	$[6, 5, 3, 1, 8, 7, 2, 4] \rightarrow [5, 6, 3, 1, 8, 7, 2, 4]$	$tab[i] = 5$	$5 < 6$	$\Rightarrow 5$ remonte à l'indice 0
$i = 2$:	$[5, 6, 3, 1, 8, 7, 2, 4] \rightarrow [3, 5, 6, 1, 8, 7, 2, 4]$	$tab[i] = 3$	$3 < 5 < 6$	$\Rightarrow 3$ remonte à l'indice 0
$i = 3$:	$[3, 5, 6, 1, 8, 7, 2, 4] \rightarrow [1, 3, 5, 6, 8, 7, 2, 4]$	$tab[i] = 1$	$1 < 3 < \dots < 6$	$\Rightarrow 1$ remonte à l'indice 0
$i = 4$:	$[1, 3, 5, 6, 8, 7, 2, 4] \rightarrow [1, 3, 5, 6, 8, 7, 2, 4]$	$tab[i] = 8$	$1 < \dots < 6 < 8$	$\Rightarrow 8$ reste à l'indice i ($=4$)
$i = 5$:	$[1, 3, 5, 6, 8, 7, 2, 4] \rightarrow [1, 3, 5, 6, 7, 8, 2, 4]$	$tab[i] = 7$	$\dots < 6 < 7 < 8$	$\Rightarrow 7$ remonte à l'indice 4
$i = 6$:	$[1, 3, 5, 6, 7, 8, 2, 4] \rightarrow [1, 2, 3, 5, 6, 7, 8, 4]$	$tab[i] = 2$	$1 < 2 < 3 < \dots$	$\Rightarrow 2$ remonte à l'indice 1
$i = 7$:	$[1, 2, 3, 5, 6, 7, 8, 4] \rightarrow [1, 2, 3, 4, 5, 6, 7, 8]$	$tab[i] = 4$	$\dots < 3 < 4 < 5 < \dots$	$\Rightarrow 4$ remonte à l'indice 3

4 Algorithme

L'algorithme du tris par insertion peut-être décomposé en deux parties : Une fonction qui effectue la remontée, c'est à dire qui *insère* au bon emplacement une valeur dans un tableau déjà trié. Le bon emplacement étant choisi de façon à ce que le tableau reste trié. Et une fonction qui permet de répéter cette opération sur tout le tableau, ce sera la fonction principale de *tri par insertion*.

```
Variables :  
tab : tableau d'entiers  
tab_len : entier  
i_trie : entier  
Algorithme :        Tri par insertion  
i_trie = 0;  
tab_len = taille(tab);  
Pour i_trie=1; i_trie<tab_len; i_trie++:  
    insere(tab,i_trie);  
Fin Pour;  
Fin Algorithme;
```

```
Variables :  
tab : tableau d'entiers  
i_ins, tmp : entiers  
Algorithme :        fonction insere(tab, i_trie)  
i_ins= i_trie;  
tmp = tab[i_trie];  
Tant que i_ins > 0 et tmp < tab[i_ins - 1]:  
    tab[i_ins] = tab[i_ins - 1];  
    i_ins = i_ins - 1;  
Fin Tant que;  
tab[i_ins] = tmp;  
Fin Algorithme;
```

5 Exercice(s)

1. Coder en python l'algorithme de tri par insertion. Il sera codé sous la forme de deux fonctions python : `tri_insert(tab)` et `insere(tab,i_trie)` pour, respectivement, l'algorithme de tri et la fonction de remonter.
2. Tester l'algorithme pour des tableaux contenant 10, 50, 100, 500, 1000 et 5000 valeurs entières aléatoires.
3. Relever le temps d'exécution 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.
5. Tracer un graphique qui représente les temps d'exécutions des algorithmes de tri par insertion et par selection par rapport au nombre d'éléments. Que pouvez-vous conclure ?