DLST L1 — Travaux Pratiques
Université Grenoble Alpes — INF202 2018/2019

Feuille 3 : expérimentation du tri par insertion sur des séquences aléatoires

Note On mettra dans le fichier tri-1.py donné (partiellement) tout ce qui sera programmé (et testé). L'objectif du TP est double :

- étudier une première méthode de tri, le tri par insertion.
- apprendre à construire des séquences aléatoires de données dans un format adapté aux tests sur les tris.

Ennoncé

Le tri par insertion est un algorithme de tri classique et intuitif. Il est par exemple utilisé naturellement par les joueurs de cartes pour trier leurs mains.

On considère une séquence de nombres de taille T à trier par ordre croissant. Le trie par insertion consiste à parcourir la séquence du début ($i = 1$) à la fin ($i = T - 1$) et à chaque itération i :

- on considère que les éléments de 0 à $i - 1$ de la séquence sont déjà triés,
- on déplace le i-ème élément de manière à ce que la séquence de 0 à i soit triée. Il faut pour cela trouver où l'élément doit être inséré en le comparant, après chaque déplacement, à l'élément qui le précéde, puis le faisant "redescendre" si sa valeur est inférieur. Ce processus est réitéré tant que la valeur de l'élement à trier est inférieur au voisin qui le précède dans la séquence.

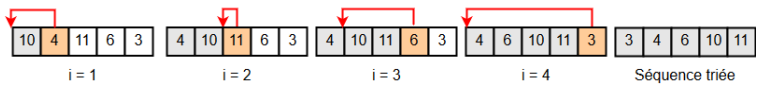

FIGURE 1 – Exemple de tri par insertion. Les cases gris clair représentent les éléments déjà triés et la case orange l'élément à placer à la bonne position.

1. Compléter la fonction *gen_seq_aleat(taille,valeur_min,valeur_max)* qui permet de générer une séquence de longueur "*taille*" d'entiers dont la valeur est aléatoirement comprise entre "*valeur_min*" et "*valeur_min*". Pour chaque élément généré de la séquence, on associera un indice initial qui correspond à l'indice de l'élément au moment de la création, et un indice final qui correspond à l'indice de l'élément après le tri.
2. Ecriver l'algorithme de tri par insertion en pseudo-code.
3. Compléter la fonction de tri par insertion *tri_insertion(seq)* qui permet de trier la séquence passée en argument.
4. Tester la fonction *tri_insertion* sur différentes séquences aléatoire.
5. Ajouter un compteur du nombre de comparaisons effectuées, et vérifier que le coût de ce tri est en moyenne quadratique. Pour cela, écrire un programme itérant sur un nombre donné de séquences de même longueur.

6. Pour quel type de séquence le coût est-il minimal ? Faire un programme générant un telle séquence et vérifier que le coût est linéaire, en procédant comme pour la question précédente.
7. Même question pour le coût maximal.