ISM MPSI/PCSI, Informatique
Année 2025/2026

TP Informatique 26

Exercice 1

Le *tri par sélection* consiste à itérer le procédé suivant : si les $k-1$ premiers éléments sont triés, on place en position k le minimum de la liste extraite à partir du k-ième.

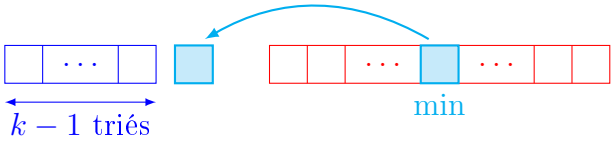

FIGURE 1 – Tri par sélection

Ainsi, pour une liste `T`, on détermine une position de son minimum et on l'échange avec l'élément en première position, puis on détermine une position du minimum de la liste extraite `T[1:]` et on l'échange avec l'élément en deuxième position, etc. ...

1. Écrire une fonction `mini(T,deb,fin)` d'arguments `T` une liste non vide de nombres, `deb` et `fin` des entiers et qui renvoie un indice dans la liste `T` du minimum de la liste extraite `T[deb:fin]`. On n'utilisera pas de techniques de slicing.
2. Écrire une fonction `tri_select(T)` d'argument `T` une liste non vide de nombres qui modifie directement cette liste en la triant selon l'algorithme du tri par sélection.
3. L'implémentation de ce tri par sélection est-elle en place ? Déterminer sa complexité temporelle.

Exercice 2

Le *tri par insertion* consiste à itérer le procédé suivant : si les k premiers éléments sont triés, on vient insérer le $k+1$-ème à sa place avec les k premiers.

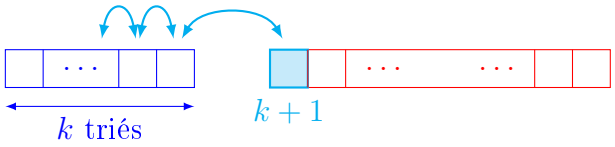

FIGURE 2 – Tri par insertion

Ainsi, pour une liste `T`, on examine les deux premiers éléments et on les échange éventuellement pour qu'ils soient ordonnés, puis on considère le troisième élément et on vient, par échanges successives, l'insérer vis-à-vis des deux premiers déjà triés, etc. ...

1. Écrire une fonction `tri_ins1(T)` d'argument `T` une liste non vide de nombres et qui modifie directement cette liste en la triant selon l'algorithme du tri par insertion.
2. L'implémentation de ce tri par sélection est-elle en place ? Déterminer sa complexité temporelle.
3. Écrire une version améliorée `tri_ins2` du tri par insertion qui évite les échanges en privilégiant des « remontées » de certaines valeurs dans la liste au cours du processus.

Exercice 3

On reprend le tri par insertion. Lors de la phase d'insertion, on peut effectuer une recherche dichotomique dans la zone triée pour déterminer l'indice où insérer l'élément courant.

1. Écrire une fonction `rech_dicho(T,deb,fin,elt)` d'arguments `T` une liste non vide de nombres, `deb` et `fin` des entiers et `elt` un nombre et qui détermine, par une recherche dichotomique sur la zone triée `T[deb:fin]` la position où insérer `elt` pour conserver le caractère ordonné.
2. Écrire une fonction `tri_ins3(T)` d'argument `T` une liste non vide de nombres et qui modifie directement cette liste en la triant selon l'algorithme du tri par insertion en utilisant la recherche dichotomique précédemment codée.
3. Cette version est-elle préférable par rapport à la précédente ?