Lycée Marcelin BERTHELOT
BCPST 1C – Mme MOREL

Tri par insertion

1 Principe

Le **tri par insertion** consiste à insérer successivement chaque élément à sa place parmi les éléments de la liste **qui le précèdent**: on insère tour à tour les éléments de la liste à leur place parmi les premiers éléments.
On part donc d'une liste de nombres quelconque `L`, et en lui appliquant le tri par insertion, on obtient une liste de nombres rangés dans l'ordre croissant.

Notons `n` la longueur de la liste `L`.

- Par défaut, on suppose `L[0]` à sa place.
- **Première étape : on met l'élément `L[1]` à sa place**
  Pour cela, on compare $L[0]$ et $L[1]$:
  - Si $L[0] > L[1]$, on décale $L[0]$ vers la droite.
  - Sinon, on les laisse à leur place.

  Au terme de la première étape, les deux premiers éléments `L[0]` et `L[1]` de la liste `L` sont triés (rangés dans l'ordre croissant).

- **Deuxième étape : on met l'élément `L[2]` à sa place**
  Pour cela, on le compare aux deux premiers éléments `L[0]` et `L[1]` de la façon suivante :
  - On commence par le comparer à `L[1]`:
    - Si $L[2] > L[1]$, alors on laisse $L[2]$ en troisième position, donc on n'y touche pas !
    - Si $L[1] > L[2]$ alors on décale $L[1]$ vers la droite.
  - On le compare ensuite avec `L[0]`:
    - S'il est plus grand que `L[0]` alors il est à sa place en deuxième position et on le place et on n'y touche plus !
    - Sinon, on on décale `L[0]` vers la droite et on le place ensuite à sa bonne place : la première position.

  Au terme de la deuxième étape, les trois premiers éléments `L[0]`, `L[1]` et `L[2]` de la liste `L` sont triés (rangés dans l'ordre croissant).

- **Troisième étape : on met l'élément à sa place**
  Pour cela, on le compare aux éléments de la façon suivante :
  - On commence par le comparer à :
    -
    -
  - On le compare ensuite avec :
    -
    -
  - On le compare ensuite avec :
    -
    -

  Au terme de la troisième étape, les premiers éléments de la liste `L` sont triés (rangés dans l'ordre croissant).

- On itère n fois jusqu'à avoir trié la totalité de la liste.
  À la fin de l'étape numéro $\boxed{k}$, les $\boxed{k+1}$ premiers éléments de la liste sont dans l'ordre croissant.
  Lors de l'étape $\boxed{k}$, pour insérer l'élément $L[k]$ à la bonne place, il suffit de travailler avec les $\boxed{k}$ premiers éléments de la liste (`L[0]`, ..., `L[k-1]` qui eux sont déjà placés par ordre croissant): tant que `L[k]` est mal placé (c'est-à-dire plus petit que l'élément précédent), on décale les éléments précédents vers la droite. Sinon, on s'arrête, et on place `L[k]`.

2 Un exemple

On considère la liste `L=[3,5,2,1,8,9,7,3]`

- **Première Étape**
  - Quel élément va-t-on mettre à sa place ?
  - À l'issue de cette étape , quels sont les éléments triés ? Détailler les décalages nécessaires :

- **Deuxième Étape**
  - Quel élément va-t-on mettre à sa place ?
  - À l'issue de cette étape , quels sont les éléments triés ? Détailler les décalages nécessaires :

- **Troisème Étape** *À vous !*

3 Code `Python`

Algorithme 1 La boucle `for k in range(1,n)` correspond à la $\boxed{k}$-ème étape dans la description ci-dessus, et donc au placement de l'élément `L[k]`.
Donc la k-ème itération de cette boucle place l'élément `L[k]` à la bonne place, de sorte qu'à la fin de cette étape, les éléments

`L[0]`, ... , `L[k]` sont triés.
Pour trouver le bon emplacement `j` pour `x`, on commence par supposer que `x` est déjà à la bonne place. On fait ensuite décroître `j` (`j=j-1`) tant que que l'élément à gauche `L[j-1]` est plus grand que `x`. En effectuant cette recherche, on décale les éléments rencontrés qui doivent se placer in fine à droite de `x` , et une fois le bon emplacement `j` trouvé, on peut donc y placer `x`.

```
def tri_inser(L):
    n = len(L)
    for k in range(1,n):
        a =.................                  # élément à placer au tour de boucle k
        j =..................                   # a est à la bonne place donc son indice est ...
        while ( j>=1 and ................. ):    # tant que a est mal placé ... (avez-vous compris j>=1 ?)
             .................                   # les éléments se décalent vers la droite
            j = j-1
        L[j] = .....................             # on insère l'élément a à sa bonne place
    return L
```

Remarque 1 Chaque passage dans la boucle `while` réalise une seule comparaison entre deux éléments de la liste. Cette boucle while comporte au maximum k étapes, il y a donc au maximum k telles comparaisons. Comme k prend les valeurs de 1 à $n-1$ on réalise en tout

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

comparaisons. La complexité de l'algorithme du tri par insertion est donc de l'ordre de n^2 où n est la longueur de la liste à trier.