A. Un autre algorithme de tri

On considère l’algorithme de tri suivant :

**def tri\_myst(liste): n = len(liste)**

**for i in range(n-1):**

**for j in range(n-i-1):**

**if liste[j] > liste[j+1]:**

**liste[j], liste[j+1] = liste[j+1], liste[j]**

1. Expliquer les deux dernières lignes du script Python.

si le nombre est plus grand que celui d'après alors ils échangent de place

1. On considère la liste 𝐿 = [5,2,7,3,1]. Faire fonctionner cet algorithme sur cette liste.Déterminer le nombre d’opérations élémentaires effectuées (comparaison et affectation)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L | i | j | Tests | Affectations |
| [5,2,7,3,1] | 0 | 0 | 1 | 1 |
| [2,5,7,3,1] | 0 | 1 | 1 | 0 |
| [2,5,7,3,1] | 0 | 2 | 1 | 1 |
| [2,5,3,7,1] | 0 | 3 | 1 | 1 |
| [2,5,3,1,7] | 1 | 0 | 1 | 0 |
| [2,5,3,1,7] | 1 | 1 | 1 | 1 |
| [2,3,5,1,7] | 1 | 2 | 1 | 1 |
| [2,3,1,5,7] | 2 | 0 | 1 | 0 |
| [2,3,1,5,7] | 2 | 1 | 1 | 1 |
| [2,1,3,5,7] | 3 | 0 | 1 | 1 |

1. Expliquer « comment » fonctionne cet algorithme.

flm

1. Correction de l’algorithme :
   1. Justifier que l’algorithme se termine.

Il y a que des boucles for et le nb d’itération est égal au nombre d’element dans la liste

* 1. Montrer que l’algorithme est correct(On pourra remarquer qu’à chaque itération de i, l’élément le plus grand se trouve à droite !).

à chaque itération de i, l’élément le plus grand se trouve à droite

1. Complexité :Pour une liste de 1 valeur, on compte 0 opérations élémentaires. Pour une liste de 2 valeurs classées en ordre décroissant, on compte 3 opérations (1 test et 2 affectations). Déterminer le nombre d’opérations élémentaires (comparaison et affectation) pour :
   1. une liste de 3 valeurs classées dans l’ordre décroissant.

6

* 1. une liste de 4 valeurs classées dans l’ordre décroissant.
  2. une liste de 5 valeurs classées dans l’ordre décroissant.
  3. une liste de n valeurs classées dans l’ordre décroissant.