Université Abdelmalek Essaâdi

Faculté des Sciences et Techniques de Tanger

Compte Rendu de Travaux Pratiques

TP 8 : Algorithmes de Tri

Module : Algorithmique et Programmation

Encadrante : Pr. Ouafae EL BOUHADI

Filière : Cycle Ingénieur

Année universitaire : 2024 – 2025

Réalisé par :

Hamza Boulahrouf

1 Introduction

Les algorithmes de tri sont des outils fondamentaux en informatique permettant dorganiser efficacement les données. Ce travail pratique vise à implémenter, analyser et optimiser différents algorithmes de tri classiques.

2 Objectifs du TP

— Comprendre le fonctionnement des algorithmes de tri.
— Optimiser le tri à bulles.
— Appliquer des tris partiels.
— Trier des structures complexes.
— Vérifier létat de tri dune liste.

3 Exercice 1 : Tri à bulles optimisé

La figure suivante illustre limplémentation du tri à bulles optimisé.

```python
def Tri_a_bulles(liste):
    n = len(liste)
    N=0
    # On fait n-1 passages
    for i in range(n):

        # Parcours jusqu'à n-i-1 pour éviter de revisiter les max déjà placés
        for j in range(0, n - i - 1):
            # Si la valeur est plus grande que la suivante, on échange
            if liste[j] > liste[j + 1]:
                liste[j], liste[j + 1] = liste[j + 1], liste[j]
                N+=1
        if N == 0:
            return print("la liste est deja ordonee")

    return liste
```

FIGURE 1 – Tri à bulles optimisé

4 Exercice 2 : Tri partiel avec le tri à bulles

4.1 Recherche du deuxième plus grand élément

```python
#Utiliser Tri à bulles  pour trier partiellement la liste afin de trouver le 2° plus grand élément. Afficher ce 2° plus grand élément.
def deuxieme_plus_grand(liste):
    n = len(liste)
    for i in range(2):
        for j in range(0, n - i - 1):
            if liste[j] > liste[j + 1]:
                liste[j], liste[j + 1] = liste[j + 1], liste[j]
    return liste[-2]

```

FIGURE 2 – Recherche du deuxième plus grand élément

4.2 Tri des cinq premiers éléments

```python
#Trier uniquement les 5 premiers éléments avec Tri à bulles
def Tri_a_bulles5(liste):
    n = len(liste)

    # On fait n-1 passages
    for i in range(5):

        # Parcours jusqu'à n-i-1 pour éviter de revisiter les max déjà placés
        for j in range(0, n - i - 1):

            # Si la valeur est plus grande que la suivante, on échange
            if liste[j] > liste[j + 1]:
                liste[j], liste[j + 1] = liste[j + 1], liste[j]

    return  liste

```

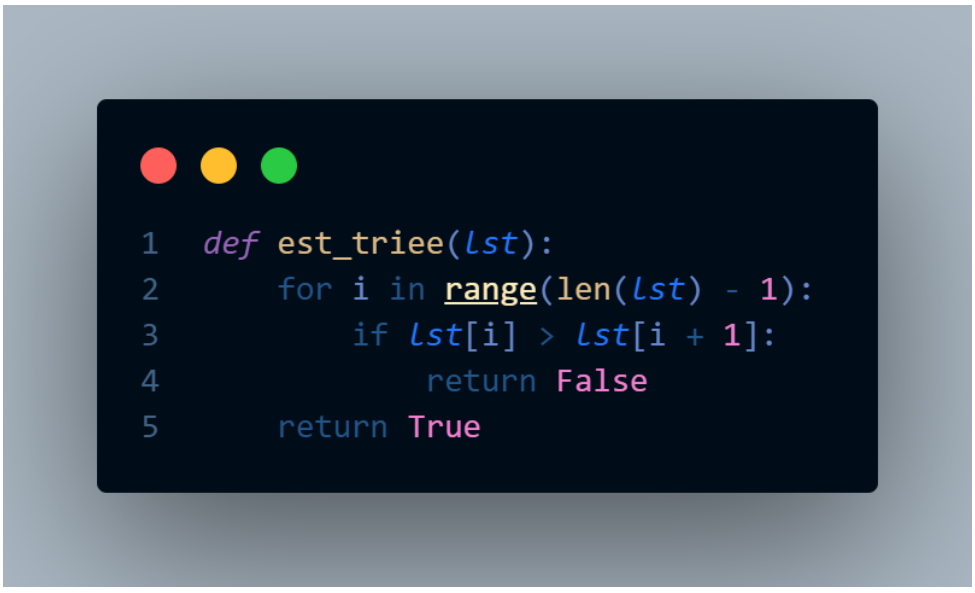
FIGURE 3 – Tri des cinq premiers éléments

5 Exercice 3 : Tri par sélection

```python
etudiants = [
{"nom": "Ali", "note": 78},
{"nom": "Sara", "note": 92},
{"nom": "Youssef", "note": 85},
{"nom": "Leila", "note": 92}
]
def tri_selection_notes(etudiants):
    n = len(etudiants)
    for i in range(n - 1):
        max_index = i
        for j in range(i + 1, n):
            if etudiants[j]["note"] > etudiants[max_index]["note"]:
                max_index = j
        etudiants[i], etudiants[max_index] = etudiants[max_index], etudiants[i]
    return etudiants
```

FIGURE 4 – Tri par sélection des étudiants

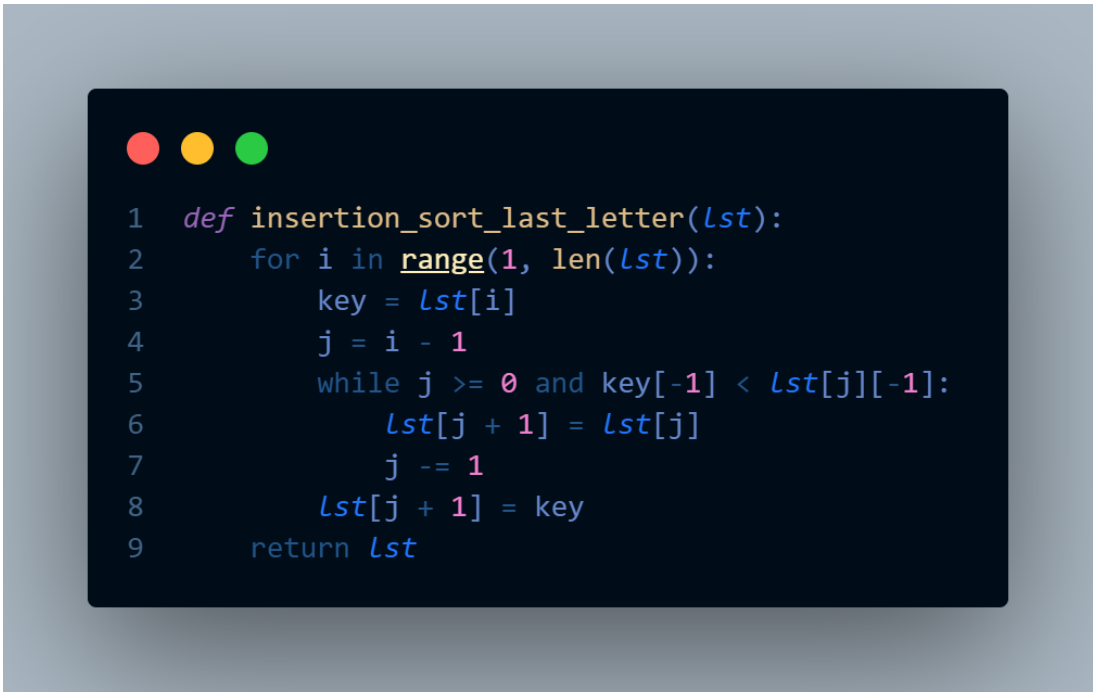
6 Exercice 4 : Vérification du tri



```python
def est_triee(lst):
    for i in range(len(lst) - 1):
        if lst[i] > lst[i + 1]:
            return False
    return True
```

FIGURE 5 – Fonction de vérification du tri

7 Exercice 5 : Tri par insertion



```python
def insertion_sort_last_letter(lst):
    for i in range(1, len(lst)):
        key = lst[i]
        j = i - 1
        while j >= 0 and key[-1] < lst[j][-1]:
            lst[j + 1] = lst[j]
            j -= 1
        lst[j + 1] = key
    return lst
```

FIGURE 6 – Tri par insertion selon la dernière lettre

8 Conclusion

Ce travail pratique a permis de consolider la compréhension des algorithmes de tri et de leurs optimisations. Les différentes méthodes étudiées montrent limportance du choix de lalgorithme selon le contexte et les besoins.