Université Larbi Ben Mhidi

Département MI

Année : 2024-2025

Niveau : 1ère année Master AD

Module : Méthodes algorithmiques

Série de TD 1 : Rappel sur la complexité

Exercice 01 : Complexité temporelle

Étant donné un tableau d'entiers arr de taille n, vous devez trouver la somme de tous les éléments du tableau.

Comparez les deux approches données pour calculer cette somme :

Approche 1	***Approche 2:***

def somme_parcours_unique(arr):
somme = 0
for elem in arr:
somme += elem
return somme
	def somme_parcours_double(arr):
	somme = 0
	for i in range(len(arr)):
	for j in range(len(arr)):
	if i == j:
	somme += arr[i]
	return somme

Exercice 02 : Complexité spatiale

Posant un tableau d'entiers `arr`, vous devez trouver le tableau inversé :

Approche 1 :	***Approche 2 :***

def inversion_in_place(arr):
gauche = 0
droite = len(arr) - 1
while gauche < droite:
arr[gauche], arr[droite] = arr[droite], arr[gauche]
gauche += 1
droite -= 1
return arr
	def inversion_avec_temp(arr):
	temp = [0] * len(arr)
	for i in range(len(arr)):
	temp[len(arr) - 1 - i] = arr[i]
	return temp

Exercice 3 : Analyse de l'espace utilisé dans un tri rapide (QuickSort)

Voici une implémentation classique de QuickSort :

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
        gauche = [x for x in arr if x < pivot]
    milieu = [x for x in arr if x == pivot]
    droite = [x for x in arr if x > pivot]
    return quicksort(gauche) + milieu + quicksort(droite)
```

Analysez la complexité en espace de cet algorithme dans le meilleur et le pire des cas.