Aix*Marseille université

Domaine Sciences et Technologies
Portail 1 René Descartes

Introduction à l'informatique : TD 4

Code UE : SPO1U57

Année 2023–2024

Tableaux et matrices

Exercice 1 (recherche séquentielle) En cours, on a pu rechercher un élément dans un tableau, à l'aide, par exemple, de l'algorithme de recherche séquentielle :

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

1. Modifier l'algorithme de recherche séquentielle pour qu'il renvoie `True` ou `False`, selon que l'élément a été trouvé dans le tableau ou non : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ devra renvoyer `True`, alors que la recherche de 9 dans ce même tableau devra renvoyer `False`.
2. Modifier ensuite l'algorithme pour qu'il renvoie l'indice de la *dernière occurrence* de l'élément recherché : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ renverra désormais 4.
3. Rappeler la complexité dans le pire des cas de l'algorithme de recherche séquentielle donné plus haut, en terme de l'ordre de grandeur du nombre d'opérations élémentaires en fonction de la longueur n du tableau.
4. Améliorer l'algorithme de recherche séquentielle dans le cas où le tableau donné en entrée est supposé trié, pour qu'il s'arrête dans son parcours du tableau dès lors qu'il a trouvé l'élément à chercher, ou bien qu'il est sûr que l'élément à chercher ne se trouve pas dans le tableau.
5. Quelle est l'ordre de grandeur de complexité dans le pire des cas de votre nouvel algorithme ?

Exercice 2 (Statistiques) Étant donné un tableau de nombres réels en entrée :

1. Écrire un algorithme qui calcule la moyenne des valeurs du tableau et fournir une analyse de complexité, y compris en notation « grand O ».
2. Écrire un algorithme qui calcule la variance des valeurs du tableau et fournir une analyse de complexité, y compris en notation « grand O ». Pour rappel, la variance des valeurs $x_1, \ldots, x_n$ est calculée comme $\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$, où μ est la moyenne des valeurs.

Exercice 3 (Rotations d'un tableau) Écrire un algorithme qui prend en entrée un tableau `t` et un entier `k` et renvoie la `k`-ième rotation de `t` vers la droite. Par exemple :

```
>>> rotation([1,2,3,4,5], 2)
[3, 4, 5, 1, 2]
>>> rotation([1,2,3,4,5], 0)
[1, 2, 3, 4, 5]
>>> rotation([1,2,3,4,5], -3)
[3, 4, 5, 1, 2]
```

Fournir également une analyse de complexité, y compris en notation « grand O ».

Exercice 4 (Parcours de matrices) Écrire trois algorithmes qui prennent en entrée une matrice carrée `M` de taille n (c'est-à-dire un tableau de n tableaux chacun de longueur n) et qui affichent (avec la fonction `print`) les éléments de la matrice :

1. par lignes (`parcours_lignes`) ;
2. par colonnes (`parcours_colonnes`) ;
3. par diagonales descendantes à droite (`parcours_diagonales`). Ici une diagonale est considérée toujours de longueur n, et si elle sort du côté droit de la matrice on continue avec l'élément plus à gauche de la ligne suivante.

Par exemple, étant donné

```
M = [[ 1,  2,  3,  4],
     [ 5,  6,  7,  8],
     [ 9, 10, 11, 12],
     [13, 14, 15, 16]]
```

les trois algorithmes afficheront :

```
>>> parcours_lignes(M)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
>>> parcours_colonnes(M)
1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
>>> parcours_diagonales(M)
1 6 11 16 2 7 12 13 3 8 9 14 4 5 10 15
```

Fournir également une analyse de complexité, y compris en notation « grand O ».