Algorithmes et Pensée Computationnelle

Algorithmes de recherche - AVANCE

Le but de cette séance est de se familiariser avec les algorithmes de recherche. Dans la série d'exercices, nous manipulerons des listes et collections en Java et Python. Nous reviendrons sur la notion de récursivité et découvrirons les arbres de recherche. Au terme de cette séance, l'étudiant sera en mesure d'effectuer des recherches de façon efficiente sur un ensemble de données.
Le code présenté dans les énoncés se trouve sur Moodle, dans le dossier **Ressources**.

1 Recherche séquentielle (ou recherche linéaire)

1.1 Exercices

Question 1: (*5 minutes*) Recherche séquentielle - 3 (Python)

Considérez une **liste d'entiers triés** **L** ainsi qu'un entier **e**. Écrivez un programme qui retourne l'index de l'élément **e** de la liste **L** en utilisant une recherche séquentielle. Si **e** n'est pas dans **L**, retournez -1.

Exemple

L = [1231321,3213125,3284016,4729273,5492710]
e = 3284016
Résultat attendu : 2

```
def recherche_sequentielle(L,e):
    for elem in L: #Ici, i correspond à la valeur et non l'index.
        #complètez ici

    L = [1231321,3213125,3284016,4729273,5492710]
    e = 3284016
    resultat = recherche_sequentielle(L,e)
    print(resultat)
```

Conseil

Une liste triée permet une recherche plus efficace à l'aide d'un algorithme plus simple.
Retournez l'index de la valeur dans la liste. Pensez à utiliser la fonction **index()** qui retourne l'index d'un élément au sein d'une liste en Python.
Exemple et syntaxe : **lst.index(i)** va indiqué la position de l'élément **i** dans la liste **lst**.

2 Recherche binaire

2.1 Exercices

Question 2: (*15 minutes*) Recherche binaire - plus proche élément (Python)

Soit une liste d'entiers **triés** **L** ainsi qu'un entier **e**. Écrivez un programme retournant la valeur dans **L** la plus proche de **e** en utilisant une recherche binaire (binary search).

```
def plus_proche_binaire(liste,n):
    #complètez ici


L = [1, 2, 5, 8, 12, 16, 24, 56, 58, 63]
e = 41
print(plus_proche_binaire(L,e))
# Résultat attendu : 56\\
```

Conseil

Pensez à définir des variables `min` et `max` délimitant l'intervalle de recherche et une variable booléenne **found** initialisée **false** et qui devient **true** lorsque l'algorithme a trouvé la valeur la plus proche de **e**.

Question 3: (*10 minutes*) Recherche binaire (Python)

Considérez une liste d'entiers triés **L** ainsi qu'un entier **e**. Écrivez un programme qui retourne l'index de l'élément **e** de la liste L en utilisant une recherche binaire. Si **e** n'est pas dans **L**, retournez **–1**.

Exemple

L = [1231321,3213125,3284016,4729273,5492710]

e = 3284016

Résultat attendu : 2

```
def recherche_binaire(liste,e):
    first = 0
    last = len(L)-1

    #complètez ici


L = [1231321,3213125,3284016,4729273,5492710]
e = 3284016
recherche_binaire(L,e)
```

Conseil

Inspirez-vous des conseils des exercices précédents (exercices basiques).

Question 4: (*20 minutes*) Recherche matricielle (Python)
Matrice en Python
Considérez une matrice ordonnée **m** et un élément **l**.

Pour rappel, une matrice ordonnée répond aux critères suivants :
[i][j]<=m[i+1][j] (une ligne va du plus petit au plus grand)
[i][j]<=m[i][j+1] (une colonne va du plus petit au plus grand)

Écrivez un algorithme qui retourne la position de l'élément **l** dans **m**. Si **l** n'est pas présent dans **m** alors il faut retourner **(–1, –1)**

Exemples

Exemple 1 : si m=[[1,2,3,4],[4,5,7,8],[5,6,8,10],[6,7,9,11]] et que l=7. Nous souhaitons avoir la réponse (1,2) OU (3,1) (l'une des deux, pas besoin de retourner les deux résultats).

Exemple 2 : si m=[[1,2],[3,4]] et que l=7. Nous souhaitons avoir la réponse (-1,-1) car 7 n'est pas dans la matrice m.

```
def recherche_matricielle(m,l):
    #complètez ici


m=[[1,2,3,4],[4,5,7,8],[5,6,8,10],[6,7,9,11]]
l=7
recherche_matricielle(m,l)
```

Conseil

Pour cet exercice, il est nécessaire d'utiliser des boucles **for** imbriquées, c'est-à-dire : une boucle **for** dans une autre boucle **for**. Cela permet de parcourir tous les éléments d'une liste (ou d'un tableau) à deux dimensions (dans notre cas une matrice).