

















Series d'exercices corrigés en Python



 Structures de controle

 Structures de données

-  TD N°1
-  TD N°2
-  TD N°3
-  TD N°4
-  TD N°5
-  TD N°6 - Tableaux
-  TD N°7 - Tableaux
-  TD N°8 - Tableaux
-  TD N°9 - Tableaux
-  Problème de séquencement des tâches
-  Problème de la sélection d'activités
-  Problème d'installation des étagères
-  Sous-ensemble de produits minimum d'un tableau
-  La recherche dichotomique

 Les fonctions

 Gestion de fichiers

 Programmation OO

 Sujets avancés

LA RECHERCHE DICHOTOMIQUE

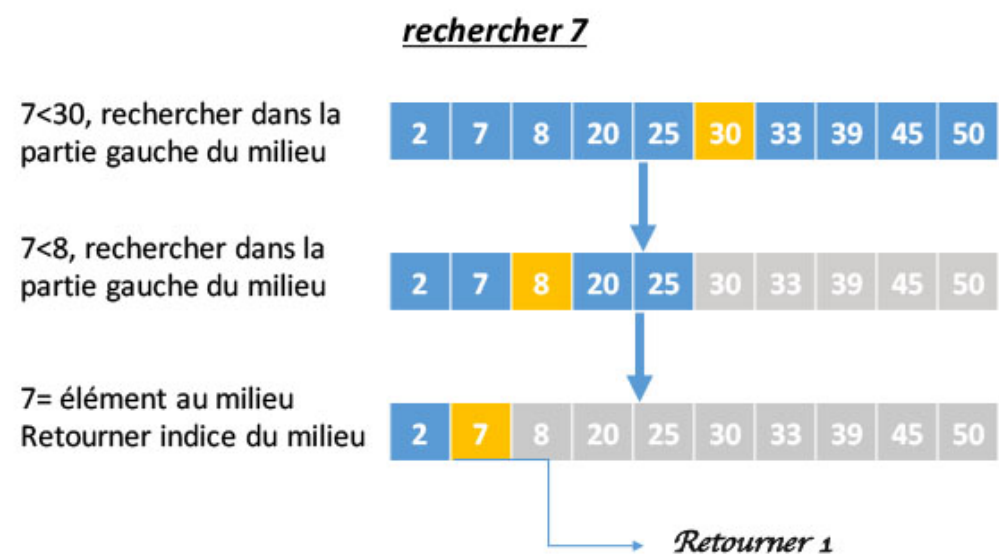
 01 Mai 2019 /  M. ESSADDOUKI /  Diviser pour régner /  4646 Visites

Problème ! Etant donné un tableau trié `tab[]` de `n` éléments, écrivez une fonction pour rechercher un élément donné `x` dans `tab[]`.

Une approche simple consiste à effectuer une recherche linéaire. La complexité temporelle de l'algorithme ci-dessus est $O(n)$. Une autre approche pour effectuer la même tâche consiste à utiliser la recherche binaire.

La recherche dichotomique consiste à rechercher dans un tableau trié en divisant de manière récursive l'intervalle de recherche en deux.

Commencez par un intervalle couvrant tout le tableau. Si la valeur de la clé de recherche est inférieure à l'élément situé au milieu de l'intervalle, limitez l'intervalle à la moitié inférieure. Sinon, le réduire à la moitié supérieure. Vérifiez à plusieurs reprises jusqu'à ce que la valeur soit trouvée ou que l'intervalle soit vide.



L'idée de la recherche dichotomique est de profiter du tableau trié et de réduire la complexité à $O(\text{Log } n)$.

Implémentation récursive de la recherche dichotomique

Algorithmme

Python

Nous ignorons initialement la moitié des éléments juste après une seule comparaison.

1. Comparez x avec l'élément du milieu.
2. Si x correspond à l'élément du milieu, nous retournons l'indice du milieu.
3. sinon Si x est supérieur à l'élément milieu, alors x ne peut se situer que dans la moitié droite du sous-tableau après l'élément milieu.
4. Sinon recherchez dans la moitié gauche.

Implémentation itérative de la recherche dichotomique

```
1 def RechercheBinaire2(tab, x):
2     debut, fin = 0, len(tab)
3     while debut <= fin:
4
5         milieu = (debut + fin)//2
6         if tab[milieu] == x:
7             return milieu
8
9         elif tab[milieu] < x:
10            debut = milieu + 1
11
12        else:
13            fin = milieu - 1
14
15        # l'element n'existe pas
16        return -1
17
18
19 tab = [2, 7, 8, 20, 25, 30, 33, 39, 45, 50]
20 x = 8
21
22 result = RechercheBinaire2(tab, x)
23
24 if result != -1:
25     print("L'élément est présent à l'indice % d" % result)
26 else:
27     print("L'élément n'est pas présent dans le tableau")

```

1 | L'élément est présent à l'indice 2

Tableaux

Partager ce cours avec tes amis :