Chapitre 3 : Algorithmes de recherche

On a souvent besoin de rechercher la position d'un élément ou un élément donné dans un tableau. Il existe deux types d'algorithmes de recherche, suivant que le tableau soit trié ou non.

1 Recherche séquentielle

La méthode de recherche la plus simple est la recherche séquentielle qui s'effectue en temps linéaire : étudier les éléments les uns après les autres. Elle ne nécessite pas d'avoir une structure de données triée. La fonction suivante recherche de manière séquentielle si `tab` contient la valeur `element`.

```
def recherche_sequentielle(element, tab):
    n =len(tab)
    i = 0
    while i <  n :
        if tab[i] == element :
            return i
        i=i+1
    return -1
```

Complexité : La complexité de l'algorithme ci-dessus est linéaire dans le pire des cas.

2 La dichotomie

Si le tableau dans lequel on recherche une valeur est trié, on peut utiliser cette propriété pour effectuer une recherche plus rapide : la recherche dichotomique.

Idée générale de la dichotomie

Soit `tab` est tableau trié et `i` un indice valide du tableau :

- si `element` $\leq$ `tab[i]` alors tous les éléments "à droite" de `tab[i]` (d'indice `j` avec `j > i`) sont supérieurs ou égaux à `element`,
- si `element > tab[i]` alors tous les éléments "à gauche" de `tab[i]` (d'indice `j` avec `j < i`) sont inférieurs ou égaux à `element`.

Quand on regarde la valeur situé au milieu du tableau, si celle-ci est plus petite que l'élément à trouver, on sait qu'il faut regarder dans la partie droite du tableau (du milieu du tableau à la fin). Autrement, il faut regarder dans la partie gauche du tableau (du début du tableau jusqu'au milieu).

À chaque itération, on considère seulement "une moitié de tableau".

```
def recherche_dichotomique(element, tab ):
    debut = 0
    fin = len(tab)-1
    milieu = (debut+fin)//2
    while debut <= fin :
        if tab[milieu] == element :  #On a trouvé l'élément
            return milieu
        elif tab[milieu] > element :
            fin = milieu-1           #On considère la partie gauche du tableau
        else :
            debut = milieu+1         #On considère la partie droite du tableau
        milieu = (debut+fin)//2
    return -1
```

Remarques :

- Complexité logarithmique dans le pire des cas : $log_2(n)$.
- La recherche dichotomique est plus rapide que la recherche séquentielle en général (complexité en moyenne et dans le pire des cas) mais dans certains cas, la recherche séquentielle peut être plus rapide :
  - si les tableaux sont de petite taille,
  - si la valeur à chercher se trouve dans le début du tableau.