ALGORITHMES DE RECHERCHE

PROBLEMATIQUE

Le problème est de rechercher des informations dans une table – qui peut être une liste, un tableau, un arbre, un graphe, etc. Dans la suite, nous supposerons que :

- Les tables sont des tableaux.
- Les tableaux contiennent des nombres – on pourrait traiter de la même façon des enregistrements quelconques, par exemple formés d'un numéro de Sécurité sociale, d'un nom et d'une adresse, ou alors d'un nom et d'un numéro de téléphone.

RECHERCHE SEQUENTIELLE

Cette méthode simple consiste à parcourir le tableau à partir du premier élément et à s'arrêter dès que l'on trouve l'élément cherché – on ne cherche que la première occurrence d'un élément.

Soit T un tableau de n éléments et k l'élément que l'on recherche :

```
def rechercheSeq(T,k):
    """
    T : tableau de n éléments et un élément k
    La fonction retourne l'indice du premier élément où se trouve k.
    Si k n'est pas dans la liste la fonction retourne None.
    """
    i=0
    n=len(T)
    while i<=n and T[i]!=k:
        i+=1
    if i<=n:
        return i
    else:
        return None
```

Pour une recherche séquentielle, la complexité est linéaire, $O(n)$, puisque, dans le pire des cas, il faut parcourir tout le tableau.

RECHERCHE DICHOTOMIQUE

Cette méthode s'applique si le tableau est déjà trié et s'apparente à la méthode diviser pour régner.

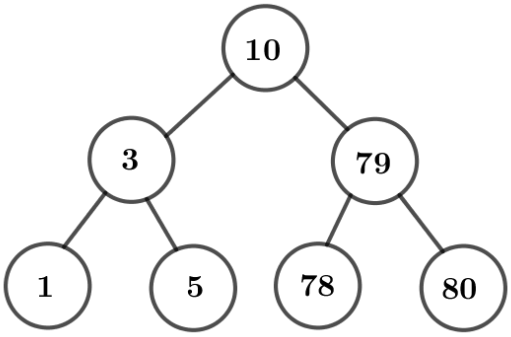
Soit T un tableau déjà trié de n éléments et k l'élément que l'on recherche. On compare k au nombre médian du tableau. Si c'est le même nombre alors on a trouvé. Sinon on recommence sur la première moitié du tableau, ou sur la seconde, selon que k est inférieur ou supérieur au nombre médian :

```
def rechercheDicho(T,k):
    """
    T : tableau ordonné de n éléments et un élément k
    La fonction retourne l'indice d'élément où se trouve k.
    Si k n'est pas dans la liste la fonction retourne None.
    """
    min=0
    max=len(T)-1
    i=(min+max)//2
    while min<=max and T[i]!=k:
        if k<T[i]:
            max=i-1
        else:
            min=i+1
        i=(min+max)//2
    if T[i]==k:
        return i
    else:
        return None
```

Pour une recherche dichotomique, la complexité est logarithmique $O(\log n)$.

On remarque que la recherche dichotomique dans un tableau revient à organiser le tableau sous la forme d'un arbre binaire. :

```
Tableau =[1,3,5,10,78,79,80]
```



Une recherche dichotomique dans le Tableau revient à suivre un chemin dans l'arbre. Un tel arbre est appelé **arbre binaire de recherche**.

PARCOURS DANS LES ARBRES

Un parcours d'arbre liste toutes les étiquettes des nœuds d'un arbre selon un ordre prédéterminé.

On distingue plusieurs méthodes.

LE PARCOURS EN LARGEUR

On liste les nœuds niveau par niveau. On commence par la racine, puis on liste les nœud qui sont à la distance 1 de la racine, puis les nœuds qui sont à la distance 2 etc.

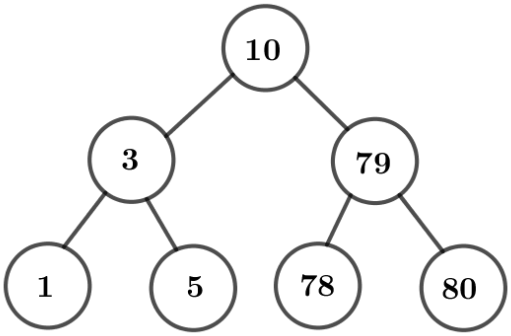
LES PARCOURS EN PROFONDEUR

Ces parcours descendent en profondeur dans l'arbre tant que c'est possible. Pour traiter un nœud n, on traite d'abord tous ses descendants et on remonte ensuite pour traiter le père de n et son autre fils. Ces parcours sont de trois type : le ***parcours infixe***, le ***parcours préfixe***, le ***parcours suffixe*** (ou *postfixe*).

- Le **parcours infixe** traite d'abord le sous-arbre gauche, puis le nœud courant puis le sous-arbre droit :
- Le **parcours préfixe** traite d'abord le nœud courant puis le sous-arbre-gauche, puis le sous-arbre droit.
- Le **parcours suffixe** (ou **postfixe**) traite d'abord le sous-arbre gauche, puis le sous-arbre droit, puis le nœud courant.

On ne peut réaliser ces différents parcours qu'en utilisant une pile, puisqu'on est obligé de commencer le parcours par la racine et d'empiler les nœuds dont on n'a pas encore rencontré. La pile peut être utilisée d'une manière explicite ou bien au moyen d'une procédure récursive.

Par exemple, pour l'arbre :



Parcours en largeur : 10, 3, 79, 1, 5, 78, 80

Parcours infixe : 1, 3, 5, 10, 78, 79, 80

Parcours préfixe : 10, 3, 1, 5, 79, 78, 80

Parcours suffixe : 1, 5, 3, 78, 80, 79, 10

Remarque : si les éléments sont rangés dans un arbre binaire de recherche, il suffit de parcourir cet arbre dans l'ordre infixe pour ranger les éléments par ordre croissant.