T.D. 2

Exercice 1: Recherche dans un tableau

Soit T un tableau de n cellules contenant des valeurs numériques (dont certaines éventuellement identiques) et soit v une valeur numérique.

1. Ecrire un programme python qui renvoie le plus petit indice i tel que $T[i] = v$ ou 0 s'il n'en existe pas (le tableau est parcouru par indice croissant).
2. Donner la complexité de ce programme en fonction de n.

Exercice 2: Recherche du plus grand indice

Soit T un tableau de n cellules contenant des valeurs numériques (dont certaines éventuellement identiques) et soit v une valeur numérique.

1. Ecrire un programme python qui renvoie le plus grand indice i tel que $T[i] = v$ ou -1 s'il n'en existe pas (le tableau est parcouru par indice croissant).
2. Donner la complexité de ce programme en fonction de n.
3. Quel est l'inconvénient de ce programme par rapport au précédent ?
4. La boucle while permet de réaliser des boucles dont l'indice est décroissant. Utiliser cette commande pour proposer un algorithme plus efficace "symétrique" de l'algorithme de la question 1.

Exercice 3: Recherche dans un tableau trié

On suppose maintenant que le tableau T est trié.

1. Ecrire un programme python qui renvoie un indice i (quelconque) tel que $T[i] = v$ ou -1 s'il n'en existe pas. Le principe du programme est le suivant :
   (a) Il calcule $milieu = (n + 1)/2$ et il compare v avec $T[milieu]$. S'il y a égalité, il renvoie $milieu$.
   (b) Sinon si $v < T[milieu]$ et $milieu > 0$, il s'appelle récursivement avec pour entrée la partie de T allant de 1 à $milieu - 1$ et renvoie le résultat de cet appel.
   (c) Sinon si $v > T[milieu]$ et $milieu < n - 1$, il s'appelle récursivement avec pour entrée la partie de T allant de $milieu + 1$ à $n - 1$ et renvoie le résultat de cet appel.
   (d) Sinon il renvoie -1 (i.e., pas trouvé).
2. Modifier le programme pour qu'il renvoie le plus petit indice comme dans la question 1.

Exercice 4: Complexité de la recherche dans un tableau trié

On étudie la complexité du dernier algorithme.

1. Démontrer que lorsque l'algorithme s'appelle récursivement, la taille du nouveau tableau est inférieure ou égale à $n/2$.
2. Soit $Time(n)$, le pire temps d'exécution de cet algorithme pour un tableau de taille inférieure ou égale à n. En déduire que pour une constante c bien choisie :

   $Time(1) \leq c$

   $\forall n \geq 2, Time(n) \leq c + Time(n/2)$
3. Démontrer par récurrence que $Time(n) \leq c(log_2(n)+1)$ et donc que la complexité de l'algorithme est en $O(log_2(n))$.

Exercice 5: Algorithme itératif de recherche

Algorithme 1 : Un algorithme itératif de recherche par partition.

$Cherche(T, n, v)$: entier

Entrées : T un tableau trié, n sa dimension, v une valeur

Sorties : l'indice d'une cellule de tableau contenant v ou -1 s'il n'en existe pas

Données : bas, $haut$: entiers

$bas \leftarrow 1$

$haut \leftarrow n$

tant que $bas \leq haut$ faire

    $milieu \leftarrow (bas + haut)/2$
    si $v = T[milieu]$ alors
        retourner $milieu$
    sinon si $v < T[milieu]$ alors
        $haut \leftarrow milieu - 1$
    sinon
        $bas \leftarrow milieu + 1$
    finsi

fin tant que

retourner -1

1. Expliquer comment l'algorithme 1 procède pour obtenir un fonctionnement similaire au premier programme de la question 3. En quoi est-il plus efficace?

2. Programmer en python l'algorithme 1.

3. Modifier le programme pour qu'il renvoie le plus petit indice comme dans la question 1.

4. Modifier le programme de la question précédente pour qu'il renvoie le nombre d'indices i tels que $T[i] = v$.