TD6 : AP.

1 Recherches séquentielles renvoyant un indice

Nous avons vu en cours des algorithmes de recherche renvoyant une valeur booléenne. Vous allez dans cet exercice réaliser des fonctions donnant l'indice de l'élément trouvé. Le type des fonctions à réaliser est donc

`list[A], int, int, A, Callable[[A, A], int]` → `int`

où `A` est un type et `Callable[[A, A], int]` désigne le type de la fonction de comparaison (sa signature est `A, A` → `int`)

1. Avant de vous lancer dans ces implantations, il faut préciser la spécification de ces fonctions de recherche :
   1. Si l'élément recherché est présent plusieurs fois dans la liste, quel indice renvoyer ?
   2. Idem si cet élément n'est pas dans la liste ?
2. Réalisez une implantation de la recherche séquentielle dans une liste non triée, puis dans une liste triée, qui donne le plus petit indice d'un élément présent dans la liste.
3. Idem pour le plus grand indice.

1. Si l'élément est présent plusieurs fois, on renvoie sa première occurrence.
Si l'élément n'est pas présent, on renvoie la taille de la liste.

2.
```
def recherche_seq(l, a, b, x, comp):
    i = a
    while i < b and comp(l[i], x) != 0:
        i += 1
    return i
```

3.
```
def recherche_seq(l, a, b, x, comp):
    i = b
    while i > a and comp(l[i], x) != 0:
        i -= 1
    return i
```

2 Point trop n'en faut!

Voici une implantation en Python erronée de la recherche laborieuse vue en cours.

```
def recherche_laborieuse_erronee(li: list[int], a: int,b: int ,x: int) -> bool:
    trouve=False
    for i in range(a,b):
        if li[i]==x:
            trouve=True
        else:
            trouve=False
    return trouve
```

1. Pour chacun des deux appels suivants à cette fonction de recherche, présentez sous forme d'un tableau les valeurs successives que prennent les variables `i` et `trouve` , ainsi que la valeur renvoyée par la fonction.
   1. `recherche_laborieuse_erronee([3, 1, 2, 4, 1], 0, 5, 1)`
   2. `recherche_laborieuse_erronee([3, 1, 2, 4, 5], 0, 5, 1)`
2. Quelle est l'erreur de programmation manifestement commise ? Proposez une réparation de ce code.
3. En fonction de n la longueur de la liste, combien de comparaisons sont effectuées lors d'une recherche laborieuse erronée ?
4. Proposez un nouveau corps de fonction, qui calcule exactement la même chose, donc produit des résultats erronés, mais bien plus efficacement que `recherche_laborieuse_erronee`

1

appel 1.

tab	i	trouve
0	0	False
1	1	True
2	2	False
3	3	False
4	4	True

-> renvoi : True.

appel 2

tab	i	trouve
0	0	False
1	1	True
2	2	False
3	3	False
4	4	False

-> renvoi : False.

2. Il faut enlever else : trouve = False.

3. O(n).

3 Pour comprendre la recherche dichotomique

Soit t la liste d'entiers :

0	2	4	6	8	10	12	14	16	18	20	22

1. Donnez la suite des tranches de liste parcourue lors de la recherche dichotomique de $x = 11$ dans la liste t. Combien de comparaisons d'éléments de la liste sont-elles effectuées ?
2. Même question avec $x = 12$.

Même question avec $x = 10$.

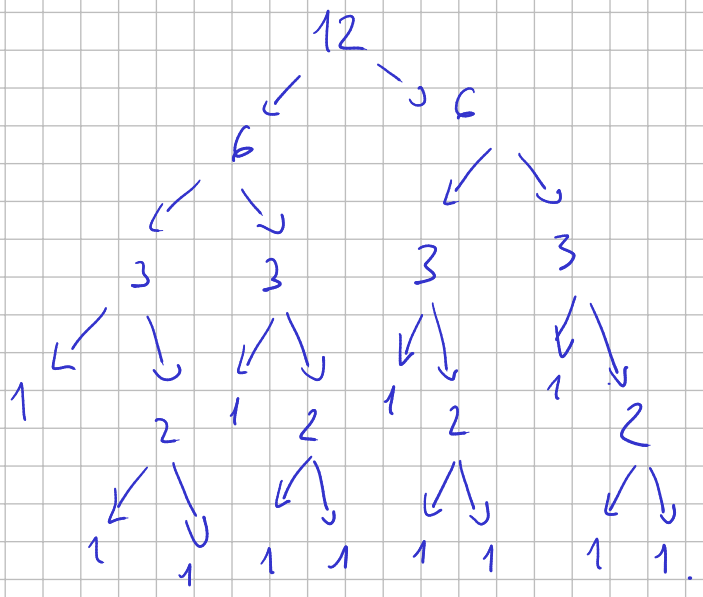
3. Dessinez l'arbre des recherches possibles pour une recherche dichotomique dans une liste de longueur 12.

1. 0, 2, 4, 6, 8, 10 -> 6, 8, 10 -> 10. = 4 comparaisons

2. 12 -> 1 comp.

3. 0, 2, 4, 6, 8, 10 -> 6, 8, 10 -> 10 -> 4 comp.

4.



4 Une petite amélioration

L'algorithme de recherche dichotomique tel qu'il a été présenté en cours est décit par une boucle tantque dont la seule condition est $d < f$, d et f désignant les indices de début et de fin de la tranche courante de la recherche. Cependant il se peut que l'élément recherché soit trouvé avant que cette condition ne soit plus réalisée.

Proposez une nouvelle version de l'algorithme de recherche dichotomique qui tient compte de cette remarque.

```
liste, x, d, f, comp

m ← (d+f)/2
pendant que d < f et comp(x, l[m]) ≠ 0
    m ← (d+f)/2
    cmp ← comp(x, l[m])
    si cmp = 1 alors
        f ← m - 1
    si cmp = -1 alors
        d ← m + 1
Retourner comp(x, l[m]) = 0.
```

5 Recherche dichotomique renvoyant un indice

1. Réalisez une implantation de la recherche dichotomique qui renvoie un indice plutôt qu'une valeur booléenne.
2. Dans le cas où l'élément recherché est présent plusieurs fois dans la liste, quel est l'indice qui est renvoyé par votre fonction ?

1.
```
def Recherche_dicho(x, l, a, b, comp):
    m = (a+b)//2
    while a<b and comp(x, l[m]) != 0:
        m = (a+b)//2
        cmp = comp(x l[m])
        if cmp == 1:
            b = m - 1
        else:
            a = m+1
    Return len(liste[a:b]) if comp(x, l[m]) != 0 else m.
```

2. L'indice renvoyé sera celui de l'élément au milieu

6 Rechercher l'ensemble des indices

Étant donnés une structure séquentielle ℓ deux indices a et b, et un élément x, réalisez une recherche de l'ensemble des indices i tels que $\ell[i] = x$ et $a \leq i < b$. Votre fonction renvoie un ensemble d'entiers. Réalisez la pour les différents cas de figures

1. ℓ non nécessairement triée ;
2. ℓ triée, recherche séquentielle ;
3. ℓ triée, recherche dichotomique.

1.
```
def Recherche(x, l, a, b, comp):
    Res = set()
    i = a
    while i < b :
        if comp(x, l[i]) == 0: Res.add(i) ; i+=1.
    Return i.
```

2. def Recherche(x, l, a, b, comp):
    Res = set()
    i = a
    while i < b and x <= l[i]:
        if comp(x, l[i]) == 0: Res.add(i)
        i += 1
    return Res

a = 0, b = 7.

3. def Recherche(x, l, a, b, comp):
    if a == b:
        return [] if comp(x, l[a]) != 0 else [a]
    else:
        m = (a+b)//2
        Res = [] if comp(x, l[i]) != 0 else [m]
        return(Recherche(x, l, a, m-1, comp) + Res +
            Recherche(x, l, m+1, b, comp).