

Recherche - exercices de révision

1. On vous donne ci-dessous l'algorithme de recherche binaire vu au cours 9, où on a rajouté une instruction `print()` dans la boucle `while`.

```
def recherche_binaire(L, x):  
    bas = 0  
    haut = len(L)-1  
  
    while haut >= bas:  
  
        milieu = (haut + bas)//2  
        print(f"{bas = }, {haut = }, {milieu = }, L[{milieu}] = {L[milieu]} ",\  
              end = "")  
  
        if L[milieu] == x:  
            return milieu  
  
        if L[milieu] > x:  
            haut = milieu - 1  
            print(">", x)  
  
        if L[milieu] < x:  
            bas = milieu + 1  
            print("<", x)
```

- (a) Qu'affichent les instructions suivantes?

```
L = [10, 12, 14, 16, 18, 20, 22]  
x = 17  
print(f"\nrecherche_binaire(L,{x}) retourne {recherche_binaire(L, x)}")
```

- (b) Même question avec `x = 20`.
(c) Même question avec `x = 23`.

Solution : Exécutez le code. Testez d'autres valeurs de `L` et de `x`.

2. Voici maintenant l'algorithme de recherche binaire récursif vu à la série 9, où on a rajouté une instruction `print()`.

```
def recherche_binaire_rec(L, bas, haut, x):

    milieu = (bas+haut)//2
    print(f"Je suis dans recherche_binaire_rec(L, {bas}, {haut}, {x}),", "\n",
          end = " ")
    print(f"L[{milieu}] = {L[milieu]}")

    if haut < bas:
        print(f"{haut} < {bas}, je retourne!")
        return None
    elif L[milieu] == x:
        return milieu
    elif L[milieu] > x:
        return recherche_binaire_rec(L, bas, milieu-1, x)
    else:
        return recherche_binaire_rec(L, milieu+1, haut, x)
```

- (a) Qu'affichent les instructions suivantes ?

```
L = [10, 12, 14, 16, 18, 20, 22]
x = 17
print(f"\nrecherche_binaire_rec(L,0,6,{x}) retourne {recherche_binaire_rec(L,0,6,x)}")
```

- (b) Même question avec $x = 20$.
 (c) Même question avec $x = 23$.

Solution : Exécutez le code. Testez d'autres valeurs de L et de x .

3. Implémentez une version récursive (sans boucle) de l'algorithme de recherche linéaire vu au cours 9. Votre algorithme `recherche_rec` doit :

- prendre en entrée une liste de nombres L , un élément x à rechercher dans la liste, et un indice `debut` qui correspond au début de la sous-liste de L dans laquelle on recherche x
- retourner un indice $i \geq \text{debut}$ tel que $L[i] = x$ si un tel indice existe, `None` sinon.

Etant donné une liste L et un élément x , le premier appel à votre algorithme sera `recherche_rec(L, x, 0)`.

Solution : L'algorithme suivant recherche récursivement un élément dans une liste :

```
def recherche_rec(L, x, debut):
    if debut == len(L):
        return
    if L[debut] == x:
        return debut
    return recherche_rec(L, x, debut + 1)
```

Par exemple, l'appel `print(recherche_rec([10, 12, 14, 16, 18], 16, 0))` retourne 3 et `print(recherche_rec([10, 12, 14, 16, 18], 17, 0))` retourne `None`.