

Algorithmes de recherche, dichotomie

Ambre Le Berre

MPSI

2025/2026

Plan du cours

1. Maximum, minimum
2. Recherche d'un élément dans une liste
3. D'autres applications de la dichotomie

Maximum, minimum

Recherche simple de maximum

C'est très courant de chercher le maximum / minimum d'une liste.

Recherche simple de maximum

C'est **très courant** de chercher le maximum / minimum d'une liste.

```
def minimum(L):  
    """renvoie le minimum de la liste L et son indice"""  
    val_min = L[0]  
    i_min = 0  
    #on commence à 1 car on a déjà vu l'élément 0  
    for i in range(1, len(L)):  
        if L[i] < val_min:  
            val_min = L[i]  
            i_min = i  
  
    return (val_min, i_min)
```

Autres recherches

```
def minimum(L):  
    """renvoie le minimum de la liste L et son indice"""  
    val_min = L[0]  
    i_min = 0  
    #on commence à 1 car on a déjà vu l'élément 0  
    for i in range(1, len(L)):  
        if L[i] < val_min:  
            val_min = L[i]  
            i_min = i  
  
    return (val_min, i_min)
```

Modifiez le code précédent pour trouver le deuxième maximum.

Recherche d'un élément dans une liste

Recherche “naïve”

```
def recherche(L, x):  
    """renvoie True si x est dans L, False sinon."""  
    for y in L:  
        if x == y:  
            return True  
    return False
```


Recherche dans une liste triée : la dichotomie

On a vu comment on peut trier une liste. Comment utiliser ce tri pour chercher un élément plus vite ?

Recherche dans une liste triée : la dichotomie

On a vu comment on peut trier une liste. Comment utiliser ce tri pour chercher un élément plus vite ?

Idée : on regarde au milieu de la liste.

- Si milieu $< x$, alors x est dans la deuxième moitié
- Sinon, dans la première moitié

Recherche dans une liste triée : la dichotomie

Recherche de “2” dans la liste [-1, 0, 1, 2, 3, 4, 5, 10, 123, 124]

-1	0	1	2	3	4	5	10	123	124
-1	0	1	2	3	4	5	10	123	124
-1	0	1	2	3	4	5	10	123	124
-1	0	1	2	3	4	5	10	123	124

Recherche dans une liste triée : la dichotomie

```
def recherche_dicho(L, x):  
    """Si L est une liste triée dans l'ordre croissant, renvoie  
    l'indice de x dans L, ou bien None si x n'est pas dans L."""  
    i = 0 #borne inf inclue  
    j = len(L) #borne sup exclue  
  
    while (j - i) > 1:  
        milieu = (i + j) // 2  
        if L[milieu] > x:  
            i = milieu + 1  
        elif L[milieu] < x:  
            j = milieu  
        else: #cas ou L[milieu] = x  
            return milieu  
    return None
```

Combien de tour de boucle au maximum pour une liste de taille 8 ? De taille 16 ? Et pour 2^n en général ?

Complexité

Combien de tour de boucle au maximum pour une liste de taille 8 ? De taille 16 ? Et pour 2^n en général ?

En fait, la complexité est **logarithmique**, soit en $O(\log(n))$.

D'autres applications de la dichotomie

Recherche du 0 d'une fonction

Supposons qu'on ait une fonction f quelconque et qu'on cherche où elle s'annule sur un intervalle $[a, b]$, tel que $f(a) < 0$ et $f(b) > 0$.

On peut appliquer le même principe !

Recherche du 0 d'une fonction

```
def recherche_zero(f, a, b, eps):  
    """Si  $f(a) < 0$  et  $f(b) > 0$ , renvoie un zero de  $f$  sur  $[a, b]$  à eps  
    près."""  
    while b - a > eps:  
        milieu = a + b / 2  
        if f(milieu) < 0:  
            a = milieu  
        else:  
            b = milieu  
    return (a + b) / 2
```