Chapitre 1.6 : Recherche dans une liste

I Recherche du maximum/minimum d'une liste

On dispose d'une liste `L` de nombres et on souhaite trouver la valeur du maximum de cette liste.
Pour cela : on définit une variable auxiliaire `maxi` et on parcourt la liste `L` à l'aide d'une boucle `for` d'indice `i` de sorte qu'à chaque itération dans la boucle, `maxi` contienne le maximum de la liste `L[:i+1]`. A la fin de la boucle, la variable `maxi` contiendra donc le maximum de la liste `L`. Ce qui donne :

```
def maximum(L:list):
    '''

    '''
```

Bien sûr, on peut adapter cet algorithme pour renvoyer le minimum, une position du maximum, la liste de toutes les positions possibles, etc... (voir exercices).

II Recherche d'un élément dans une liste

On dispose d'une liste `L` et d'un objet `e` et on souhaite déterminer si `e` est un élément de `L` ou non.

II.1 Recherche linéaire

La première façon de procéder fonctionne à tous les coups : on parcourt la liste `L` à l'aide d'une boucle et dès qu'on trouve un élément de `L` égal à `e`, on peut renvoyer `True`. Enfin, si on arrive au bout de la boucle sans jamais avoir renvoyé `True`, l'élément `e` n'est pas dans la liste. Ce qui donne :

```
def element(L:list, e) -> bool:
    '''

    '''
```

Bien entendu, on peut adapter l'algorithme précédent pour renvoyer un indice d'apparition de l'élément `e`, etc... (voir exercices)

II.2 Recherche dichotomique

Cet algorithme ne fonctionne que lorsque `L` est une liste triée par ordre croissant. C'est donc un inconvénient par rapport à la recherche linéaire. Par contre, la recherche dichotomique est beaucoup plus rapide (en terme de complexité).

On suppose donc que la liste L est triée. On utilise le principe de **dichotomie** (qui vient du grec pour division en deux parties).
L'idée est simple :

- pour déterminer si un élément e est dans une liste triée L, on compare e à l'élément m qui est au milieu de L.
- si e est égal à m, on a trouvé e !
- si e est inférieur strictement à m, on sait alors que e ne peut se trouver qu'à gauche de m.

- si e est supérieur strictement à m, on sait alors que e ne peut se trouver qu'à droite de m.

À l'aide d'une seule comparaison, on élimine la moitié des éléments du tableau ! Il ne reste plus qu'à recommencer l'opération avec la liste de gauche ou de droite suivant le cas.
Par exemple, on cherche si e = 9 se trouve dans L = [1, 3, 5, 6, 9, 12, 14] :

Indices dans L	0	1	2	3	4	5	6
On cherche e dans L[0 :7]	1	3	5	6	9	12	14
On compare e avec L[3]	1	3	5	6	9	12	14
On cherche e dans L[4 :7]					9	12	14
On compare e avec L[5]					9	12	14
On cherche e dans L[4 :5]					9		
On compare e avec L[4]					9		

III Exercices

Exercice 1

1. Écrire une fonction `maxmin(L)` qui renvoie le maximum et le minimum de la liste `L`.
2. Écire une fonction `posmax(L)` qui renvoie la liste des positions du maximum de la liste `L`. Par exemple, `posmax([1, 3, 0, 2, 3, -1])` doit renvoyer `[1, 4]`.

■

Exercice 2 Écrire une fonction `maxdico(d)` qui prend un dictionnaire et qui renvoie la plus grande valeur du dictionnaire. ■

Exercice 3 Écrire une fonction `element(L, e)` qui renvoie la liste des indices d'apparition de `e` dans `L`. ■