1.1 Tableau non trié

Lorsqu'on recherche une valeur dans un tableau quelconque non trié on n'a pas d'autre choix que de parcourir séquentiellement ce tableau. La complexité de recherche est alors linéaire (proportionelle au nombre n d'éléments du tableau soit $O(n)$).

```
  def recherche(tableau, valeur)
    for element in tableau:
        if element == valeur:
            return True
    return False
```

On utilise la même méthode de parcours séquentiel pour déterminer un minimum ou un maximum.

```
def recherche_min(tableau):
    minimum = tableau[0]
    for i in range(1, len(tableau)):
        if tableau[i] < minimum:
            minimum = tableau[i]
    return minimum
```

1.2 Tableau trié

Dans le cas d'un tableau trié, on recherche un élément de manière plus fine. En effet puisque le tableau est trié, on a une indication en comparant la valeur cherché à un élement du tableau.
Un exemple de recherche dichotomique dans la vie courante est la recherche d'un mot quelconque dans un dictionnaire. On ouvre le dictionnaire au milieu on lit le mot en haut à gauche.

Si le mot est celui cherché on l'a trouvé, sinon s'il est avant on recommence le procédé avec la première partie du dictionnaire s'il est après on recommence le procédé avec la seconde partie du dictionnaire.
Considérons l'exemple suivant `[1, 4, 11, 13, 18, 25, 32, 40, 51]` et on cherche si la valeur 10 est présente. Pour cela si elle est présente on renvoit l'indice correspondant sinon -1.

Ce tableau contient 9 élements. Il commence à l'indice 0, se termine à l'indice 8 et l'inidice du milieu est 4

- Initialisation

```
def recherche(tableau, valeur):
    debut = 0
    fin = 8
```

- Boucle

```
    while debut < fin :
        milieu = (debut + fin) // 2
        if tableau[milieu] == valeur:
            return milieu
        elif valeur < tableau[milieu]:
            fin = milieu - 1
        else:
            debut = milieu +1
    return -1
```

Dérouler cet algorithme pour l'exemple.