Construire un algorithme : approche incrémentale

1. Supposer que l'algorithme fonctionne et qu'une partie des données ont été traitées avec succès.

Imaginer l'état des données ou de la structure de données.

Construire un algorithme : approche incrémentale

1. Supposer que l'algorithme fonctionne et qu'une partie des données ont été traitées avec succès.
2. Définir les actions à mener pour traiter la donnée suivante (incrémenter l'action).

Les données traitées ou la structure de données doivent être dans le même état que précédemment mais avec une donnée traitée en plus

Construire un algorithme : approche incrémentale

1. Supposer que l'algorithme fonctionne et qu'une partie des données ont été traitées avec succès.
2. Définir les actions à mener pour traiter la donnée suivante (incrémenter l'action).
3. Définir les conditions de début et de fin de l'algorithme

Réfléchir à comment commencer
et quand s'arrêter

Construire un algorithme : approche incrémentale

1. Supposer que l'algorithme fonctionne et qu'une partie des données ont été traitées avec succès.
2. Définir les actions à mener pour traiter la donnée suivante (incrémenter l'action).
3. Définir les conditions de début et de fin de l'algorithme

Exemple ?

Recherche dans un tableau

```
def recherche(t, n, x):
    i=0
    trouve = False
    while(trouve == False and i<n):
      if (t[i]== x) :
        trouve = True
      else :
        i=i+1
    if (trouve == True):
        return(i)
    return(-1)
```

Recherche dans un tableau

```
def recherche(t, n, x):
    i=0
    trouve = False
    while( not trouve and i<n):
      if (t[i]== x) :
        trouve = True
      else :
        i=i+1
    if (trouve):
        return(i)
    return(-1)
```

Complexité en temps ?
Complexité en mémoire ?
Retourne le 1er si doublon :

Recherche dans un tableau

```
def recherche(t, n, x):
    i=0
    trouve = False
    while( not trouve and i<n):
      if (t[i]== x) :
        trouve = True
      else :
        i=i+1
    if (trouve):
        return(i)
    return(-1)
```

Complexité en temps ? $\Omega(1)$ $O(n)$
Complexité en mémoire ? $\Theta(1)$
Retourne le 1er si doublon : oui

Recherche dans un tableau

```
def rechercheDichotomique(t, n, x):
    g = 0
    d = n-1
    trouve = False
    while(not trouve and g<=d):
        m=(g+d)//2
        if (t[m]==x):
            trouve = True
        else :
            if (t[m]<x):
                g = m+1
            else :
                d = m-1
    if (trouve):
        return(m)
    return(-1)
```

k fois

que vaut k ?

Complexité en temps ?

Complexité en mémoire ?

Retourne le 1er si doublon :

Recherche dans un tableau

```
def rechercheDichotomique(t, n, x):
    g = 0
    d = n-1
    trouve = False
    while(not trouve and g<=d):
        m=(g+d)//2
        if (t[m]==x):
            trouve = True
        else :
            if (t[m]<x):
                g = m+1
            else :
                d = m-1
    if (trouve):
        return(m)
    return(-1)
```

k fois

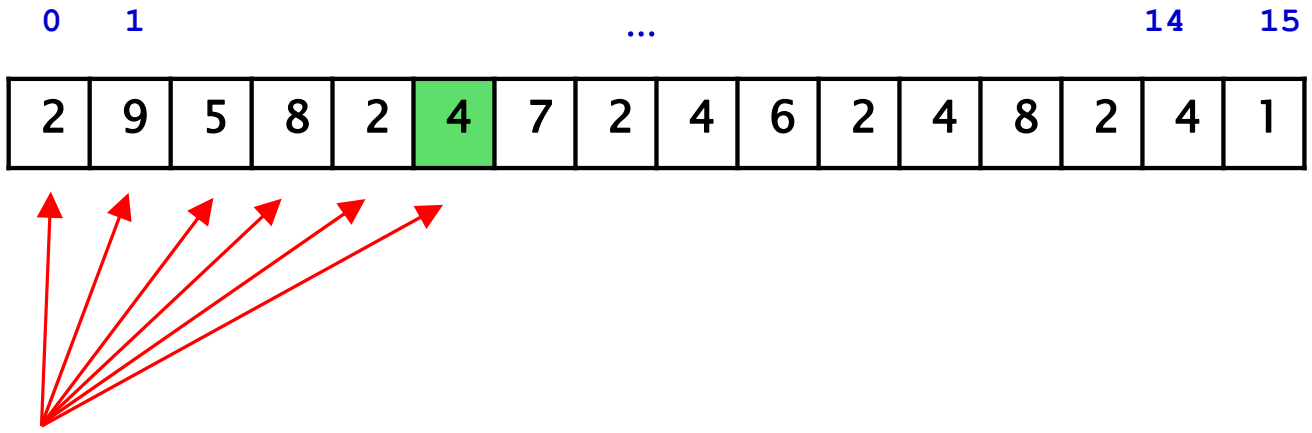
que vaut k ?

Complexité en temps ? $\Omega(1)$ $O(\log_2(n))$

Complexité en mémoire ? $\Theta(1)$

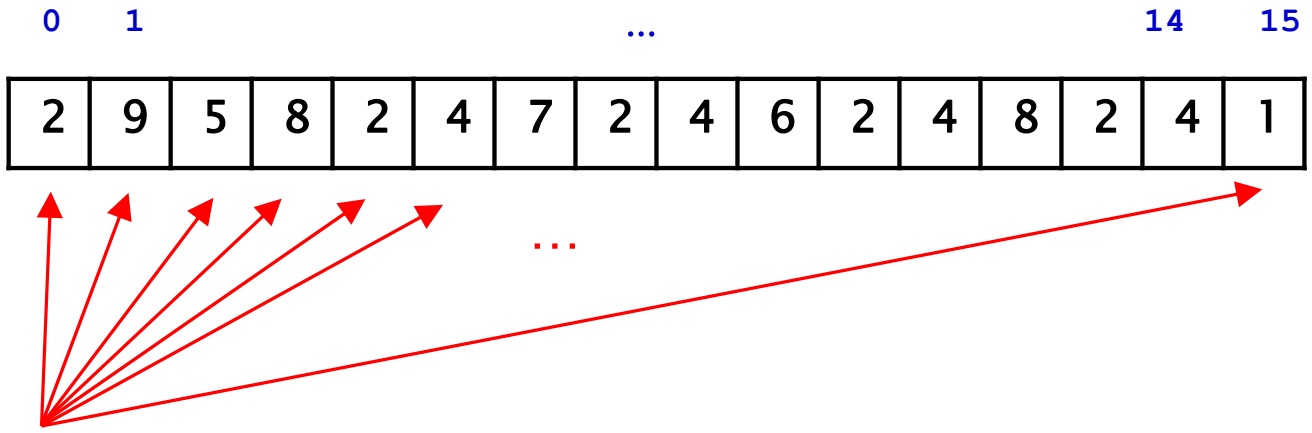
Retourne le 1er si doublon : non

Recherche dans un tableau : exemples



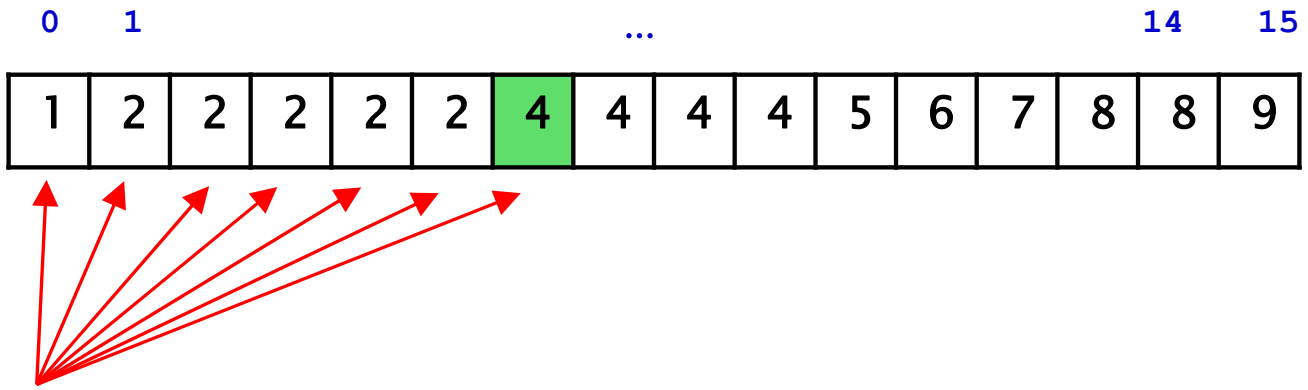
Rechercher 4 dans le tableau : `return 5`

Recherche dans un tableau : exemples



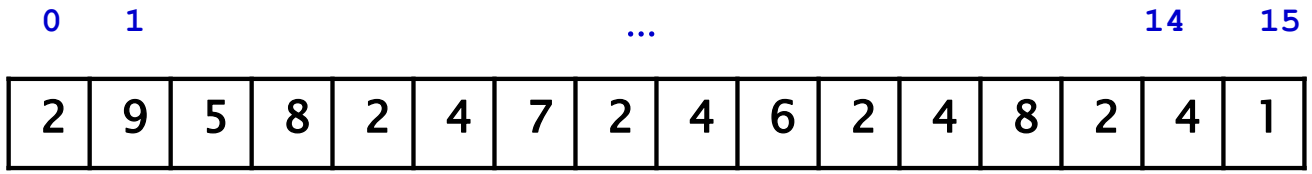
Rechercher 3 dans le tableau : `return -1`

Recherche dans un tableau : exemples



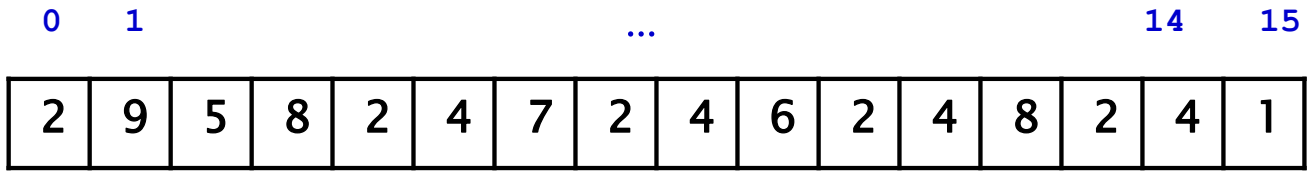
Rechercher 4 dans le tableau : `return 6`

Recherche dans un tableau : exemples



Recherche dichotomique de 4 dans le tableau : ???

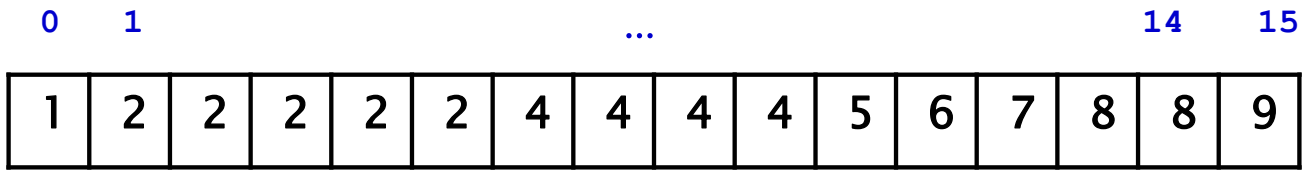
Recherche dans un tableau : exemples



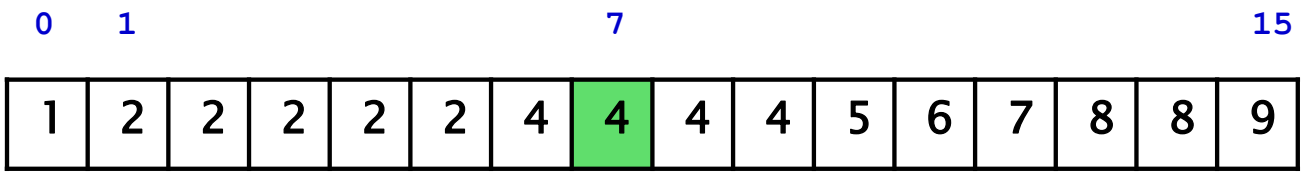
Recherche dichotomique de 4 dans le tableau :

Impossible le tableau n'est pas trié !!!

Recherche dans un tableau : exemples



Recherche dichotomique de 4 dans le tableau :

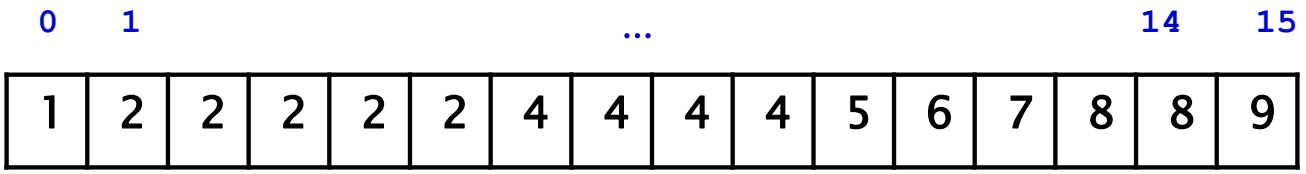


Dès le début de l'algorithme avec g==15 et d==0 on a :

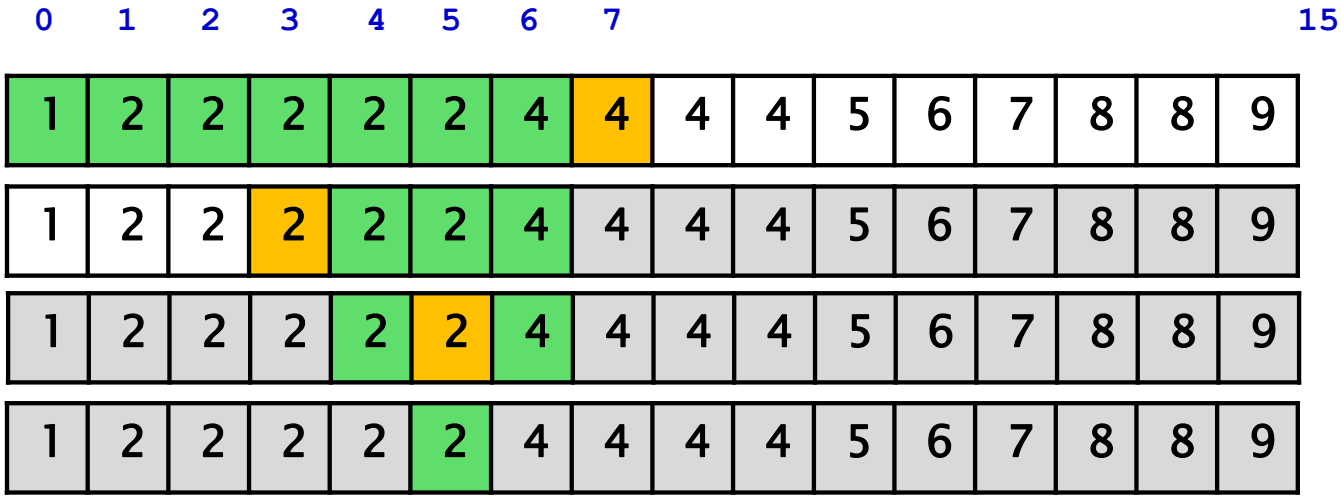
```
m=(g+d)//2
if (t[m]==x):
    trouve = True
```

La boucle s'arrête, 7 sera retourné, ce n'est pas l'indice du 1er « 4 » présent dans le tableau.

Recherche dans un tableau : exemples



Recherche dichotomique de 3 dans le tableau :



`d==f, t[d]!=3` à l'itération suivante on aura **`f<g`**
donc 3 n'est pas dans le tableau