# Elements d'Algorithmique
# CMTD5: Recherche dichotomique

Daniela Petrişan
Université de Paris, IRIF

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Université de Paris

# Trouver une valeur dans un tableau

**Entrée :** un tableau `T` et un élément $x$

**Sortie :** i tel que `T[i]` = x ou `NonTrouvé`

```
fonction RECHERCHESÉQUENTIELLE (T, x)
    n ← longueur de T
    pour i ← 0 à n − 1 faire
        si T[i]=x alors
            retourne i
    retourne NonTrouvé
```

# Trouver une valeur dans un tableau

**Entrée :** un tableau `T` et un élément $x$

**Sortie :** i tel que `T`[i] = x ou `NonTrouvé`

```
fonction RECHERCHESÉQUENTIELLE (T, x)
    n ← longueur de T
    pour i ← 0 à n − 1 faire
        si T[i]=x alors
            retourne i
    retourne NonTrouvé
```

Complexité dans le pire des cas : O(n)

# Trouver une valeur dans un tableau

**Entrée :** un tableau `T` et un élément $x$

**Sortie :** i tel que `T[i]` = x ou `NonTrouvé`

```
fonction RECHERCHESÉQUENTIELLE (T, x)
    n ← longueur de T
    pour i ← 0 à n − 1 faire
        si T[i]=x alors
            retourne i
    retourne NonTrouvé
```

Complexité dans le pire des cas : O(n)

Nous pouvons faire mieux si le tableau est trié !

# Recherche dichotomique: Principe

**Exemple.** Rechercher l'élément 27 dans le tableau suivant :

| l | | | | | mid | | | | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | | | | | ↓ | | | | | ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

**Exemple.** Rechercher l'élément 27 dans le tableau suivant :

| l | | | | | mid | | | | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | | | | | ↓ | | | | | ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$27 > 12$

**Exemple.** Rechercher l'élément 27 dans le tableau suivant :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| l ↓ | | | | | mid ↓ | | | | | r ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$27 > 12$

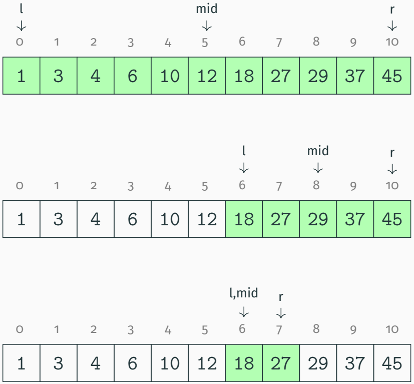
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l ↓ | | mid ↓ | | r ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$

$27 < 29$

**Exemple.** Rechercher l'élément 27 dans le tableau suivant :




$$\mathtt{l} = 0, \mathtt{r} = 10$$
$$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$$
$$27 > 12$$

$$\mathtt{l} = 6, \mathtt{r} = 10$$
$$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$$
$$27 < 29$$

$$\mathtt{l} = 6, \mathtt{r} = 7$$
$$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 6$$
$$27 > 18$$

**Exemple.** Rechercher l'élément 27 dans le tableau suivant :

| l | | | | | mid | | | | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$27 > 12$

| | | | | | | l | | mid | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$

$27 < 29$

| | | | | | | l,mid | r | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 7$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 6$

$27 > 18$

| | | | | | | | l,mid,r | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 7, \mathtt{r} = 7$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 7$

$27 = 27$, `donc renvoie 7`

**Exemple.** Rechercher l'élément 25 dans le tableau suivant :

| l |  |  |  |  | mid |  |  |  |  | r |
|---|---|---|---|---|---|---|---|---|---|---|
| ↓ |  |  |  |  | ↓ |  |  |  |  | ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

**Exemple.** Rechercher l'élément 25 dans le tableau suivant :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| l ↓ | | | | | mid ↓ | | | | | r ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$25 > 12$

**Exemple.** Rechercher l'élément 25 dans le tableau suivant :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| l ↓ | | | | | mid ↓ | | | | | r ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$$\mathtt{l} = 0, \mathtt{r} = 10$$
$$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$$
$$25 > 12$$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l ↓ | | mid ↓ | | r ↓ |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$$\mathtt{l} = 6, \mathtt{r} = 10$$
$$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$$
$$25 < 29$$

**Exemple.** Rechercher l'élément 25 dans le tableau suivant :

| l | | | | | mid | | | | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$25 > 12$

| | | | | | | l | | mid | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$

$25 < 29$

| | | | | | | l,mid | r | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 7$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 6$

$25 > 18$

**Exemple.** Rechercher l'élément 25 dans le tableau suivant :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| l | | | | | mid | | | | | r |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 0, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 5$

$25 > 12$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l | | mid | | r |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 10$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 8$

$25 < 29$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l,mid | r | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 6, \mathtt{r} = 7$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 6$

$25 > 18$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | l,mid,r | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

$\mathtt{l} = 7, \mathtt{r} = 7$

$\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2 = 7$

$25 < 27$, $\mathtt{donc\ \ r} := 6 < \mathtt{l}$, $\mathtt{renvoie\ \ NonTrouvé}$

# Recherche dichotomique: Principe

- La recherche dichotomique est un algorithme permettant de trouver une valeur dans un tableau trié. Cet algorithme s’appuie sur l’approche «diviser pour régner».

# Recherche dichotomique: Principe

- La recherche dichotomique est un algorithme permettant de trouver une valeur dans un tableau trié. Cet algorithme s'appuie sur l'approche «diviser pour régner».
- **Idée clé :** Si le tableau est trié, on peut comparer la valeur recherchée avec le milieu du tableau et on peut ignorer une moitié du tableau pour le reste de l'exécution.

# Recherche dichotomique: Principe

- La recherche dichotomique est un algorithme permettant de trouver une valeur dans un tableau trié. Cet algorithme s'appuie sur l'approche «diviser pour régner».
- **Idée clé :** Si le tableau est trié, on peut comparer la valeur recherchée avec le milieu du tableau et on peut ignorer une moitié du tableau pour le reste de l'exécution.
- L'algorithme de recherche dichotomique répète cette procédure, en divisant à chaque étape par deux la taille du tableau restant.

# Recherche dichotomique: Principe

- La recherche dichotomique est un algorithme permettant de trouver une valeur dans un tableau trié. Cet algorithme s'appuie sur l'approche «diviser pour régner».
- **Idée clé :** Si le tableau est trié, on peut comparer la valeur recherchée avec le milieu du tableau et on peut ignorer une moitié du tableau pour le reste de l'exécution.
- L'algorithme de recherche dichotomique répète cette procédure, en divisant à chaque étape par deux la taille du tableau restant.
- Soit la valeur est trouvée $\Rightarrow$ retourner l'index;
  soit un tableau vide est obtenu $\Rightarrow$ retourner NonTrouvé.

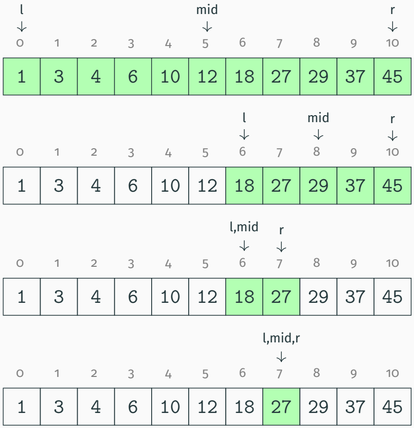
# Recherche dichotomique: Pseudo-code recursif

**Entrée :** un tableau trié $\mathtt{T}$ et un élément $\mathtt{x}$

**Sortie :** $\mathtt{i}$ tel que $\mathtt{T[i]} = \mathtt{x}$ ou $\mathtt{NonTrouvé}$

```
fonction RechDicho(T, x)
    n ← longueur de T
    retourne RechDichoRec(T, 0, n − 1, x)
```

**Exemple.** Rechercher l'élément 27



**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

**fonction** RECHDICHO($\mathtt{T}, \mathtt{x}$)

  $\mathtt{n} \leftarrow$ longueur de $\mathtt{T}$

  **retourne** RECHDICHOREC($\mathtt{T}, 0, \mathtt{n} - 1, \mathtt{x}$)

**fonction** RECHDICHOREC($\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x}$)

**Exemple.** Rechercher l'élément 27

| l ↓ | | | | | mid ↓ | | | | | r ↓ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | l ↓ | | mid ↓ | | r ↓ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | l,mid ↓ | r ↓ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

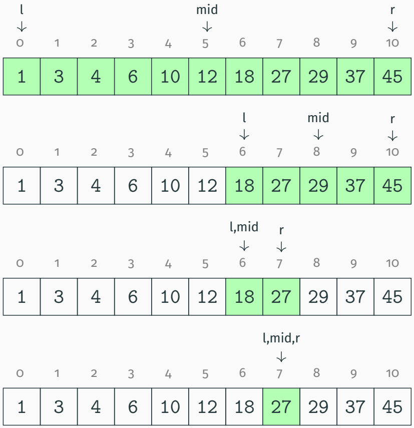
| | | | | | | | l,mid,r ↓ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

**fonction** RECHDICHO($\mathtt{T}, \mathtt{x}$)
    $\mathtt{n} \leftarrow$ longueur de $\mathtt{T}$
    **retourne** RECHDICHOREC($\mathtt{T}, 0, \mathtt{n} - 1, \mathtt{x}$)
**fonction** RECHDICHOREC($\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x}$)
    **si** $\mathtt{r} < \mathtt{l}$ **alors retourne** `NonTrouvé`

**Exemple.** Rechercher l'élément 27

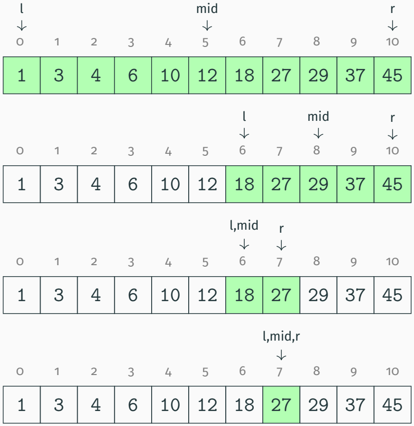


**Entrée :** un tableau trié $\mathtt{T}$ et un élément $\mathtt{x}$

**Sortie :** $\mathtt{i}$ tel que $\mathtt{T[i]} = \mathtt{x}$ ou $\mathtt{NonTrouvé}$

**fonction** $\textsc{RechDicho}(\mathtt{T}, \mathtt{x})$
    $\mathtt{n} \leftarrow$ longueur de $\mathtt{T}$
    **retourne** $\textsc{RechDichoRec}(\mathtt{T}, 0, \mathtt{n} - 1, \mathtt{x})$
**fonction** $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$
    **si** $\mathtt{r} < \mathtt{l}$ **alors retourne** $\mathtt{NonTrouvé}$
    $\mathtt{mid} \leftarrow (\mathtt{l} + \mathtt{r})/2$

**Exemple.** Rechercher l'élément 27



**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

```
fonction RECHDICHO(T, x)
    n ← longueur de T
    retourne RECHDICHOREC(T, 0, n − 1, x)
fonction RECHDICHOREC(T, l, r, x)
    si r < l alors retourne NonTrouvé
    mid ← (l + r)/2
    si T[mid]=x alors
```

**Exemple.** Rechercher l'élément 27

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| l | | | | | mid | | | | | r |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l | | mid | | r |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | l,mid | r | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

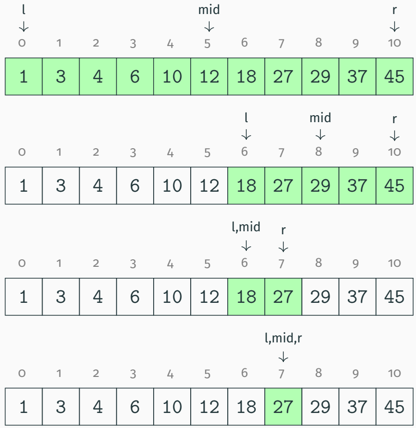
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | l,mid,r | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

**fonction** RECHDICHO($\mathtt{T}, \mathtt{x}$)
  $\mathtt{n} \leftarrow$ longueur de T
  **retourne** RECHDICHOREC($\mathtt{T}, 0, \mathtt{n} - 1, \mathtt{x}$)
**fonction** RECHDICHOREC($\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x}$)
  **si** $\mathtt{r} < \mathtt{l}$ **alors retourne** `NonTrouvé`
  $\mathtt{mid} \leftarrow (\mathtt{l} + \mathtt{r})/2$
  **si** `T[mid]=x` **alors retourne** `mid`

**Exemple.** Rechercher l'élément 27



**Entrée :** un tableau trié $\mathtt{T}$ et un élément $\mathtt{x}$

**Sortie :** $\mathtt{i}$ tel que $\mathtt{T[i]} = \mathtt{x}$ ou $\mathtt{NonTrouvé}$

**fonction** $\textsc{RechDicho}(\mathtt{T}, \mathtt{x})$

  $\mathtt{n} \leftarrow$ longueur de $\mathtt{T}$

  **retourne** $\textsc{RechDichoRec}(\mathtt{T}, 0, \mathtt{n} - 1, \mathtt{x})$

**fonction** $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$

  **si** $\mathtt{r} < \mathtt{l}$ **alors retourne** $\mathtt{NonTrouvé}$

  $\mathtt{mid} \leftarrow (\mathtt{l} + \mathtt{r})/2$

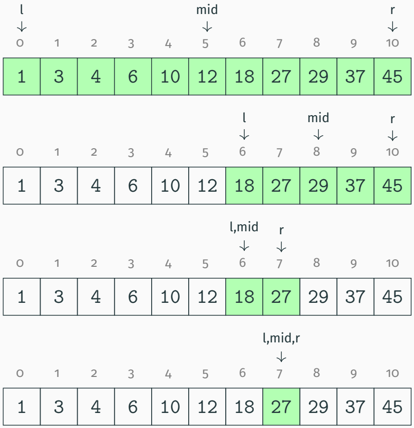
  **si** $\mathtt{T[mid]=x}$ **alors retourne** $\mathtt{mid}$

  **sinon**

    **si** $\mathtt{T[mid]<x}$ **alors**

      **retourne** $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{mid} + 1, \mathtt{r}, \mathtt{x})$

**Exemple.** Rechercher l'élément 27

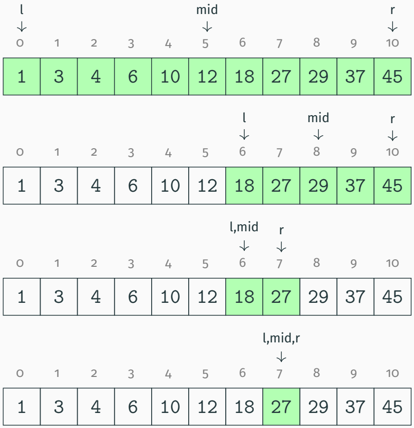


**Entrée :** un tableau trié T et un élément x

**Sortie :** i tel que T[i] = x ou NonTrouvé

```
fonction RECHDICHO(T, x)
    n ← longueur de T
    retourne RECHDICHOREC(T, 0, n − 1, x)
fonction RECHDICHOREC(T, l, r, x)
    si r < l alors retourne NonTrouvé
    mid ← (l + r)/2
    si T[mid]=x alors retourne mid
    sinon
        si T[mid]<x alors
            retourne RECHDICHOREC(T, mid + 1, r, x)
        sinon
            retourne RECHDICHOREC(T, l, mid − 1, x)
```

**Exemple.** Rechercher l'élément 27



# Recherche dichotomique:
# Pseudo-code itératif

**Entrée :** un tableau trié $\mathtt{T}$ et un élément $\mathtt{x}$

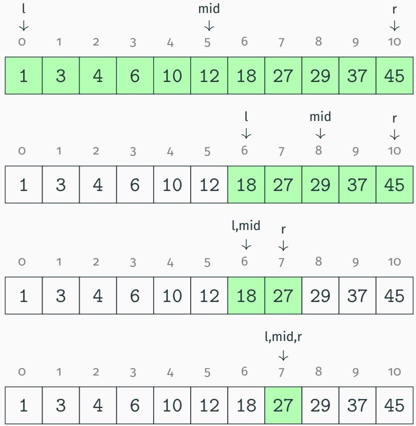
**Sortie :** $\mathtt{i}$ tel que $\mathtt{T[i]} = \mathtt{x}$ ou $\mathtt{NonTrouvé}$

**fonction** RECHDICHOITÉRATIVE($\mathtt{T}, \mathtt{x}$)

$\quad \mathtt{n} \leftarrow$ longueur de $\mathtt{T}$, $\mathtt{l} \leftarrow 0$, $\mathtt{r} \leftarrow \mathtt{n} - 1$

$\quad$ **tant que** $\mathtt{l} \leq \mathtt{r}$ **faire**

**Exemple.** Rechercher l'élément 27



**Entrée :** un tableau trié $\mathtt{T}$ et un élément $\mathtt{x}$

**Sortie :** $\mathtt{i}$ tel que $\mathtt{T[i]} = \mathtt{x}$ ou $\mathtt{NonTrouvé}$

**fonction** RECHDICHOITÉRATIVE($\mathtt{T}, \mathtt{x}$)

    $\mathtt{n} \leftarrow$ longueur de $\mathtt{T}$, $\mathtt{l} \leftarrow 0$, $\mathtt{r} \leftarrow \mathtt{n} - 1$

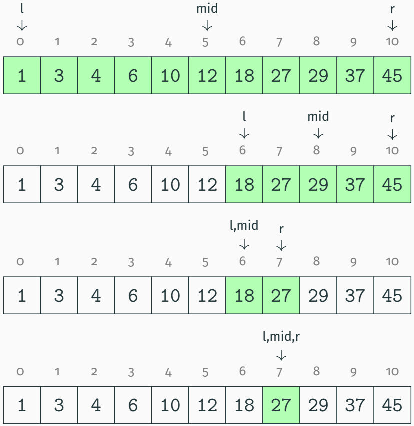
    **tant que** $\mathtt{l} \leq \mathtt{r}$ **faire**

        $\mathtt{mid} \leftarrow (\mathtt{l} + \mathtt{r})/2$

        **si** $\mathtt{T[mid]} = \mathtt{x}$ **alors**

            **retourne** $\mathtt{mid}$

**Exemple.** Rechercher l'élément 27

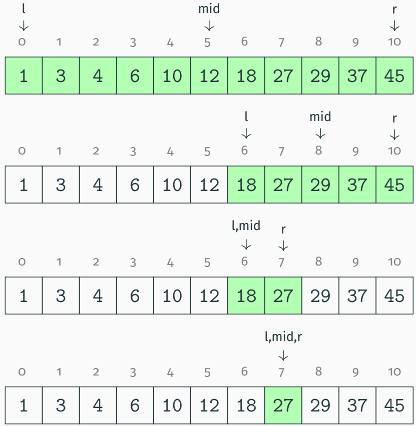


**Entrée :** un tableau trié $\texttt{T}$ et un élément $\texttt{x}$

**Sortie :** $\texttt{i}$ tel que $\texttt{T[i]} = \texttt{x}$ ou $\texttt{NonTrouvé}$

**fonction** RECHDICHOITÉRATIVE($\texttt{T}, \texttt{x}$)
    $\texttt{n} \leftarrow$ longueur de $\texttt{T}$, $\texttt{l} \leftarrow 0$, $\texttt{r} \leftarrow \texttt{n} - 1$
    **tant que** $\texttt{l} \leq \texttt{r}$ **faire**
        $\texttt{mid} \leftarrow (\texttt{l} + \texttt{r})/2$
        **si** $\texttt{T[mid]=x}$ **alors**
            **retourne** $\texttt{mid}$
        **sinon**
            **si** $\texttt{T[mid]<x}$ **alors**
                $\texttt{l} \leftarrow$

**Exemple.** Rechercher l'élément 27

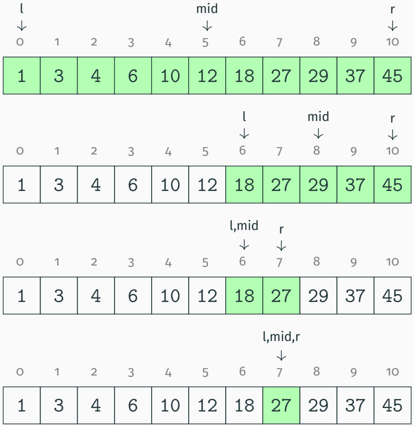


**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

```
fonction RECHDICHOITÉRATIVE(T, x)
    n ← longueur de T, l ← 0, r ← n − 1
    tant que l ≤ r faire
        mid ← (l + r)/2
        si T[mid]=x alors
            retourne mid
        sinon
            si T[mid]<x alors
                l ← mid + 1
```

**Exemple.** Rechercher l'élément 27



**Entrée :** un tableau trié `T` et un élément `x`

**Sortie :** `i` tel que `T[i]` = `x` ou `NonTrouvé`

```
fonction RECHDICHOITÉRATIVE(T, x)
    n ← longueur de T, l ← 0, r ← n − 1
    tant que l ≤ r faire
        mid ← (l + r)/2
        si T[mid]=x alors
            retourne mid
        sinon
            si T[mid]<x alors
                l ← mid + 1
            sinon
                r ←
```

**Exemple.** Rechercher l'élément 27

| l | | | | | mid | | | | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | l | | mid | | r |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

| | | | | | | l,mid | r | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

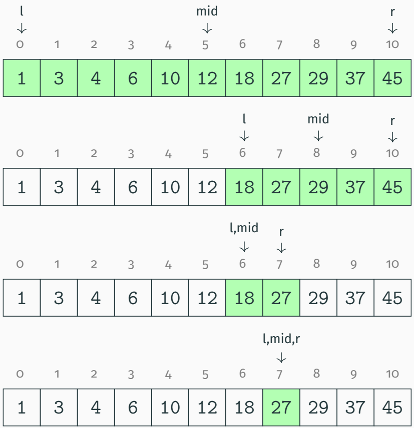
| | | | | | | | l,mid,r | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3 | 4 | 6 | 10 | 12 | 18 | 27 | 29 | 37 | 45 |

**Entrée :** un tableau trié T et un élément x

**Sortie :** i tel que T[i] = x ou NonTrouvé

```
fonction RECHDICHOITÉRATIVE(T, x)
    n ← longueur de T, l ← 0, r ← n − 1
    tant que l ≤ r faire
        mid← (l + r)/2
        si T[mid]=x alors
            retourne mid
        sinon
            si T[mid]<x alors
                l ← mid + 1
            sinon
                r ← mid − 1
```

**Exemple.** Rechercher l'élément 27

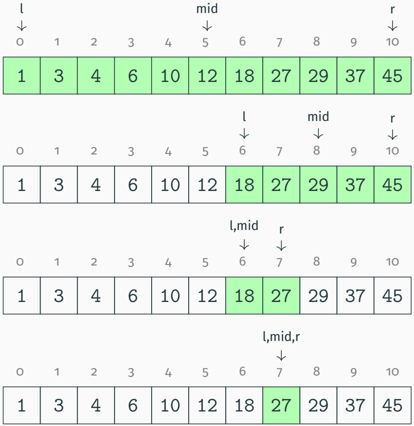


**Entrée :** un tableau trié T et un élément x

**Sortie :** i tel que T[i] = x ou NonTrouvé

```
fonction RECHDICHOITÉRATIVE(T, x)
    n ← longueur de T, l ← 0, r ← n − 1
    tant que l ≤ r faire
        mid← (l + r)/2
        si T[mid]=x alors
            retourne mid
        sinon
            si T[mid]<x alors
                l ← mid + 1
            sinon
                r ← mid − 1
    retourne NonTrouvé
```

**Exemple.** Rechercher l'élément 27



# Recherche dichotomique: Correction

RECHDICHOREC($\texttt{T}, \texttt{l}, \texttt{r}, \texttt{x}$) retourne `i` tel que $\texttt{l} \le \texttt{i} \le \texttt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

RECHDICHOREC($\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x}$) retourne `i` tel que $\mathtt{l} \leq \mathtt{i} \leq \mathtt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

$\textsc{RechDichoRec}(\texttt{T}, \texttt{l}, \texttt{r}, \texttt{x})$ retourne `i` tel que $\texttt{l} \leq \texttt{i} \leq \texttt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

Cas de base: `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

RECHDICHOREC($T, l, r, x$) retourne `i` tel que $l \leq i \leq r$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

Cas de base: `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

Hérédité : Nous supposons que la propriété est vraie lorsque $r - l + 1 < n$, où $n \geq 1$, et nous la prouvons pour $r - l + 1 = n$.

$\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$ retourne `i` tel que $\mathtt{l} \leq \mathtt{i} \leq \mathtt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

<u>Cas de base:</u> `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

<u>Hérédité :</u> Nous supposons que la propriété est vraie lorsque $\mathtt{r} - \mathtt{l} + 1 < \mathtt{n}$, où $\mathtt{n} \geq 1$, et nous la prouvons pour $\mathtt{r} - \mathtt{l} + 1 = \mathtt{n}$. Soit $\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2$; il y a trois cas :

$\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$ retourne `i` tel que $\mathtt{l} \leq \mathtt{i} \leq \mathtt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

Cas de base: `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

Hérédité : Nous supposons que la propriété est vraie lorsque $\mathtt{r} - \mathtt{l} + 1 < \mathtt{n}$, où $\mathtt{n} \geq 1$, et nous la prouvons pour $\mathtt{r} - \mathtt{l} + 1 = \mathtt{n}$. Soit $\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2$; il y a trois cas :

- `T[mid]=x`. Dans ce cas $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$ retourne `mid`, ce qui est correct puisque `x` se trouve dans le tableau à la position `mid`.

$\textsc{RechDichoRec}(\texttt{T}, \texttt{l}, \texttt{r}, \texttt{x})$ retourne `i` tel que $\texttt{l} \leq \texttt{i} \leq \texttt{r}$ et `T[i] = x` ou `NonTrouvé` si `x` n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

Cas de base: `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

Hérédité : Nous supposons que la propriété est vraie lorsque $\texttt{r} - \texttt{l} + 1 < \texttt{n}$, où $\texttt{n} \geq 1$, et nous la prouvons pour $\texttt{r} - \texttt{l} + 1 = \texttt{n}$. Soit $\texttt{mid} = (\texttt{l} + \texttt{r})/2$; il y a trois cas :

- `T[mid]=x`. Dans ce cas $\textsc{RechDichoRec}(\texttt{T}, \texttt{l}, \texttt{r}, \texttt{x})$ retourne `mid`, ce qui est correct puisque `x` se trouve dans le tableau à la position `mid`.
- `T[mid]<x`. Dans ce cas l'algorithme renvoie $\textsc{RechDichoRec}(\texttt{T}, \texttt{mid} + 1, \texttt{r}, \texttt{x})$. Vu que le tableau $T$ est trié, s'il existe $\texttt{l} \leq \texttt{i} \leq \texttt{r}$ tel que $\texttt{T}[\texttt{i}] = \texttt{x}$, alors $\texttt{mid} + 1 \leq \texttt{i} \leq \texttt{r}$; on a aussi $\texttt{r} - (\texttt{mid} + 1) + 1 < \texttt{n}$. En utilisant l'hypothèse de récurrence, le résultat renvoyé par $\textsc{RechDichoRec}(\texttt{T}, \texttt{mid} + 1, \texttt{r}, \texttt{x})$ est correct.

$\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$ retourne $\mathtt{i}$ tel que $\mathtt{l} \leq \mathtt{i} \leq \mathtt{r}$ et $\mathtt{T[i]} = \mathtt{x}$ ou `NonTrouvé` si $\mathtt{x}$ n'est pas dans le tableau dans l'intervalle `T[l,...,r]`.

**Preuve.** Récurrence sur la taille de l'intervalle `r-l+1`.

Cas de base: `r-l+1=0`. Dans ce cas `r<l`, c'est-à-dire la section du tableau est vide, donc `x` n'est pas là. L'algorithme retourne correctement `NonTrouvé` dans ce cas.

Hérédité : Nous supposons que la propriété est vraie lorsque $\mathtt{r} - \mathtt{l} + 1 < \mathtt{n}$, où $\mathtt{n} \geq 1$, et nous la prouvons pour $\mathtt{r} - \mathtt{l} + 1 = \mathtt{n}$. Soit $\mathtt{mid} = (\mathtt{l} + \mathtt{r})/2$; il y a trois cas :

- `T[mid]=x`. Dans ce cas $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{l}, \mathtt{r}, \mathtt{x})$ retourne `mid`, ce qui est correct puisque `x` se trouve dans le tableau à la position `mid`.
- `T[mid]<x`. Dans ce cas l'algorithme renvoie $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{mid} + 1, \mathtt{r}, \mathtt{x})$. Vu que le tableau $T$ est trié, s'il existe $\mathtt{l} \leq \mathtt{i} \leq \mathtt{r}$ tel que $\mathtt{T}[\mathtt{i}] = \mathtt{x}$, alors $\mathtt{mid} + 1 \leq \mathtt{i} \leq \mathtt{r}$; on a aussi $\mathtt{r} - (\mathtt{mid} + 1) + 1 < \mathtt{n}$. En utilisant l'hypothèse de récurrence, le résultat renvoyé par $\textsc{RechDichoRec}(\mathtt{T}, \mathtt{mid} + 1, \mathtt{r}, \mathtt{x})$ est correct.
- `T[mid]>x`. Similaire.

# Recherche dichotomique: Terminaison et Complexité

→ L’algorithme s’exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l’algorithme n’est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s’incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

→ L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

→ L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

- → L'algorithme s'exécute tant que $l \le r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ?

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par $2$ :

→ L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

→ Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
Après 1 exécution,

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
  Après 1 exécution,
  - Soit $r - l + 1 \leftarrow \frac{2r}{2} - \left(\left\lfloor \frac{l+r}{2} \right\rfloor + 1\right) + 1 = \left\lfloor \frac{l+r+1}{2} \right\rfloor = \left\lfloor \frac{t}{2} \right\rfloor$.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par $2$ :
  Après 1 exécution,
  - Soit $r - l + 1 \leftarrow \frac{2r}{2} - \left( \left\lfloor \frac{l+r}{2} \right\rfloor + 1 \right) + 1 = \left\lfloor \frac{l+r+1}{2} \right\rfloor = \left\lfloor \frac{t}{2} \right\rfloor$.
  - Soit $r - l + 1 = \left( \left\lfloor \frac{l+r}{2} \right\rfloor - 1 \right) - \frac{2l}{2} + 1 \leq \left\lfloor \frac{l+r+1}{2} \right\rfloor = \left\lfloor \frac{t}{2} \right\rfloor$.

→ L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

→ Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par $2$ :
  Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
  Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \dots r]$) ?
  Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par $2$ :
  Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
  Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.
  Après 3 exécutions, $r - l + 1 \leq \lfloor t/2^3 \rfloor$.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \dots r]$) ? Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
  Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
  Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.
  Après 3 exécutions, $r - l + 1 \leq \lfloor t/2^3 \rfloor$.
  …Après $i$ exécutions, $r - l + 1 \leq \lfloor t/2^i \rfloor$.

→ L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).

→ Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ?
Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.
Après 3 exécutions, $r - l + 1 \leq \lfloor t/2^3 \rfloor$.
…Après $i$ exécutions, $r - l + 1 \leq \lfloor t/2^i \rfloor$.
Alors, pour que $r - l + 1 < 1$ il faut que $t/2^i < 1 \Rightarrow i > \log t$ exécutions.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ?
  Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
  Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
  Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.
  Après 3 exécutions, $r - l + 1 \leq \lfloor t/2^3 \rfloor$.
  …Après $i$ exécutions, $r - l + 1 \leq \lfloor t/2^i \rfloor$.
  Alors, pour que $r - l + 1 < 1$ il faut que $t/2^i < 1 \Rightarrow i > \log t$ exécutions.
  Après au plus $\log t + 1$ exécutions de la boucle, l'algorithme sera fini.

- → L'algorithme s'exécute tant que $l \leq r$. Or, après chaque exécution de la boucle où l'algorithme n'est pas encore fini, soit $l \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor + 1$ ($l$ s'incremènte au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ ($r$ se diminue au moins par un).
- → Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \ldots r]$) ?
  Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle `t = r-l+1` se divise par 2 :
  Après 1 exécution, $r - l + 1 \leq \lfloor t/2 \rfloor$.
  Après 2 exécutions, $r - l + 1 \leq \lfloor t/2^2 \rfloor$.
  Après 3 exécutions, $r - l + 1 \leq \lfloor t/2^3 \rfloor$.
  ...Après $i$ exécutions, $r - l + 1 \leq \lfloor t/2^i \rfloor$.
  Alors, pour que $r - l + 1 < 1$ il faut que $t/2^i < 1 \Rightarrow i > \log t$ exécutions.
  Après au plus $\log t + 1$ exécutions de la boucle, l'algorithme sera fini. Au pire de cas, on fait une affectation et deux comparaisons dans la boucle, alors au total $3(\log t + 1)$ comparaisons et affectations.