Elements d'Algorithmique CMTD5: Recherche dichotomique

Daniela Petrişan Université de Paris, IRIF





Trouver une valeur dans un tableau

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

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

1: fonction RECHERCHESÉQUENTIELLE (T, x)

2: n ← longueur de T

3: pour i ← O à n − 1 faire

4: si T[i]=x alors

5: retourne i

6: 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

1: fonction RECHERCHESÉQUENTIELLE (T, x)

2: n ← longueur de T

3: pour i ← O à n − 1 faire

4: si T[i]=x alors

5: retourne i

6: 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

1: fonction RECHERCHESÉQUENTIELLE (T, x)

2: n ← longueur de T

3: pour i ← O à n − 1 faire

4: si T[i]=x alors

5: retourne i

6: retourne NonTrouvé
```

Complexité dans le pire des cas : O(n)

Nous pouvons faire mieux si le tableau est trié!



l ↓	mid ↓											
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

$$1 = 0, r = 10$$

 $mid = (1 + r)/2 = 5$
 $27 > 12$

$$1 = 0, r = 10$$

 $mid = (1+r)/2 = 5$
 $27 > 12$
 $1 = 6, r = 10$
 $mid = (1+r)/2 = 8$

27 < 29

$$1 = 0, r = 10$$

 $mid = (1 + r)/2 = 5$
 $27 > 12$

mid =
$$(1+r)/2 = 8$$

27 < 29
 $1 = 6, r = 7$

1 = 6, r = 10

$$mid = (1+r)/2 = 6$$

27 > 18

$$1 = 7, r = 7$$

mid = $(1 + r)/2 = 7$
 $27 = 27$, donc renvoie 7

l ↓	mid ↓												
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

$$1 = 0, r = 10$$

 $mid = (1 + r)/2 = 5$
 $25 > 12$

$$1 = 0, r = 10$$

 $mid = (1 + r)/2 = 5$
 $25 > 12$
 $1 = 6, r = 10$
 $mid = (1 + r)/2 = 8$

25 < 29

$$1 = 0, r = 10$$

 $mid = (1 + r)/2 = 5$
 $25 > 12$

mid = (1 + r)/2 = 8

$$25 < 29$$

 $1 = 6, r = 7$
 $mid = (1 + r)/2 = 6$

1 = 6, r = 10

25 > 18

$$l = 7, r = 7$$

mid = $(l + r)/2 = 7$

mid = (1+r)/2 = 725 < 27, donc r := 6 < 1, renvoie NonTrouvé

• 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».

- 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.

- 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.

- 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 ⇒ retourner l'index;
 soit un tableau vide est obtenu ⇒ retourner NonTrouvé.

Pseudo-code recursif

Recherche dichotomique:

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

1: **fonction** RECHDICHO(T, x)

1: IOIICTIOII RECHDICHO(1, x

2: $n \leftarrow \text{longueur de } T$

3: **retourne** RECHDICHOREC(T, 0, n - 1, x)

 	P.c.						/			
l ↓ 0	1	2	3	4	mid ↓ 5	6	7	8	9	r ↓ 10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	3	4	5	l → 6	7	mid ↓ 8	9	r → 10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	3	4	5	l,mid ↓ 6	r ↓ 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
	_		_							l

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

1: **fonction** RECHDICHO(T, x)

2: $n \leftarrow longueur de T$

3: **retourne** RECHDICHOREC(T, 0, n - 1, x)

4: **fonction** RECHDICHOREC(T, 1, r, x)

Exemple. Rechercher l'élément 27

	p.c.						/			
l ↓	1	2	3	,	mid ↓ 5	6	7	8	0	r ↓ 10
0	1	- 2	3	4	5	0		0	9	10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	3	4	5	l → 6	7	mid ↓ 8	9	r ↓ 10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	3	4	5	l,mid ↓ 6	r ↓ 7	8	9	10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	2	,	-	6	l,mid,r ↓		0	10

10

12 18 27

Fntrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

1: **fonction** RECHDICHO(T, x)

 $n \leftarrow longueur de T$

retourne RECHDICHOREC(T, 0, n - 1, x)

fonction RECHDICHOREC(T, 1, r, x)

si r < 1 alors retourne NonTrouvé

Evennle Recharcher l'élément 27

_		pic.	ILCC	IICIC	iici	ı cıc		11 2/			
	l ↓ 0	1	2	3	4	mid ↓ 5	6	7	8	9	r ↓ 10
	1	3	4	6	10	12	18	27	29	37	45
	0	1	2	3	4	5	l → 6	7	mid ↓ 8	9	r ↓ 10
	1	3	4	6	10	12	18	27	29	37	45
	0	1	2	3	4	5	l,mid ↓ 6	r ↓ 7	8	9	10
	1	3	4	6	10	12	18	27	29	37	45

l.mid.r

12 18 27

10

Fntrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

- $n \leftarrow longueur de T$
- retourne RECHDICHOREC(T, 0, n 1, x)
- fonction RECHDICHOREC(T, 1, r, x)
- si r < 1 alors retourne NonTrouvé
- $mid \leftarrow (1+r)/2$ 6:

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

0	1	2	3
1	3	4	6
0	1	2	3
1	3	4	6



							l,mid,r ↓	
0	1	2	3	4	5	6	7	8

-		_				-	,	-	-	10
1	3	4	6	10	12	18	27	29	37	45

Fntrée: un tableau trié T et un élément x

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

2:
$$n \leftarrow longueur de T$$

3: **retourne** RECHDICHOREC(T, 0,
$$n - 1$$
, x)

6:
$$mid \leftarrow (1+r)/2$$

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

0	1	2	3	4
1	3	4	6	10
0	1	2	3	4
4	2	1	6	10

1	0	4	0	10	12	10
0	1	2	3	4	5	l,mi ↓
						_
1	3	4	6	10	12	18

1	3	4	6	10	12	18	27	29	37	45
							l,mid,r			
							+			

0										
1	3	4	6	10	12	18	27	29	37	45

Fntrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

1: **IONCLION** RECHDICHO(1,
$$x$$
)
2: $x \leftarrow x$ longueur de T

3: **retourne** RECHDICHOREC
$$(T, 0, n - 1, x)$$

6:
$$mid \leftarrow (1+r)/2$$

l ↓ 0	1	2	3	4	mid ↓ 5	6	7	8	9	r ↓ 10
1	3	4	6	10	12	18	27	29	37	45
0	1	2	3	4	5	ι ↓ 6	7	mid ↓ 8	9	r ↓ 10
0	3	2	3	10	5 12	18	7	\downarrow	9	+

0	1	2	3	4	5	l → 6	7
1	3	4	6	10	12	18	27
0	1	2	3	4	5	l,mid ↓ 6	r ↓ 7

0	1	2	3	4	5	ь	7	8	
1	3	4	6	10	12	18	27	29	
							l,mid,r ↓		
0	1	2	2	/.	E	6	7		

0										
1	3	4	6	10	12	18	27	29	37	45

Fntrée: un tableau trié T et un élément x

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

- 1: fonction RECHDICHO(T, x)
- $n \leftarrow longueur de T$
- retourne RECHDICHOREC(T. 0, n 1, x)
- fonction RECHDICHOREC(T, 1, r, x)
- si r < 1 alors retourne NonTrouvé
- $mid \leftarrow (1+r)/2$ 6:
- si T[mid] = x alors retourne mid 7:
- sinon 8.
- si T[mid] < x alors 9:
- retourne RECHDICHOREC(T, mid + 1, r, x) 10:

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											
							ι		mid ↓		r L







10 12 18 27

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

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

- 1: **fonction** RECHDICHO(T, x)
- 2: $n \leftarrow \text{longueur de T}$
- 3: **retourne** RECHDICHOREC(T, 0, n 1, x)
- 4: **fonction** RECHDICHOREC(T, 1, r, x)
- 5: **si** r < 1 **alors retourne** NonTrouvé
- 6: $mid \leftarrow (1+r)/2$
- 7: **si** T [mid] =x **alors retourne** mid
- 8: **sinon**
- 9: si T[mid] < x alors
- 10: **retourne** RECHDICHOREC(T, mid + 1, r, x)
- 11: sinon
- 12: **retourne** RECHDICHOREC(T, 1, mid 1, x)

+	1	2	3	4	mid ↓ 5	6	7	8	9	↓ 10
1	3	4	6	10	12	18	27	29	37	45







Recherche dichotomique: Pseudo-code itératif

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

- 1: fonction RECHDICHOITÉRATIVE(T, x)
- 2: $n \leftarrow \text{longueur de T, 1} \leftarrow 0, r \leftarrow n-1$
- 3: tant que $1 \le r$ faire

	ACIII	pie.	Nec	nerc	.iiei	i eie	inci	11 2/			
	l ↓ 0	1	2	3	4	mid ↓ 5	6	7	8	9	r ↓ 10
	1	3	4	6	10	12	18	27	29	37	45
	0	1	2	3	4	5	l → 6	7	mid ↓ 8	9	r → 10
	1	3	4	6	10	12	18	27	29	37	45
	0	1	2	3	4	5	l,mid ↓ 6	r ↓ 7	8	9	10
	1	3	4	6	10	12	18	27	29	37	45
,	0	1	2	3	4	5	6	l,mid,r ↓ 7	8	9	10
	1	3	Δ	6	10	12	1.8	27	29	37	45

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

- 1: **fonction** RECHDICHOITÉRATIVE(T, x)
- 2: $n \leftarrow \text{longueur de T, 1} \leftarrow 0, r \leftarrow n-1$
- 3: tant que $1 \le r$ faire
- 4: $mid \leftarrow (1+r)/2$
- 5: si T [mid] =x alors
- 6: **retourne** mid

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

						l,r
0	1	2	3	4	5	
1	3	4	6	10	12	1

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		
4	0	4	6	40	40	40	07	20	07	4.5		

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

- 1: **fonction** RECHDICHOITÉRATIVE(T, x)
- $n \leftarrow longueur de T. 1 \leftarrow 0, r \leftarrow n-1$
- tant que $1 \le r$ faire 3:
- $mid \leftarrow (1+r)/2$
- si T[mid] = x alors 5:
- retourne mid 6:
- sinon
- si T[mid] < x alors
- 1 ← 9:

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

1	2	1	6	10	10	
0	1	2	3	4	5	

0	1	2	3	4	5	6
1	3	4	6	10	12	18

						.,,.			
						+			
1	2	3	4	5	6	7	8	9	10

Entrée: un tableau trié T et un élément x **Sortie:** i tel que T[i] = x ou NonTrouvé

2:
$$n \leftarrow \text{longueur de T, } 1 \leftarrow 0, r \leftarrow n-1$$

3: tant que
$$1 \le r$$
 faire

4:
$$mid \leftarrow (1+r)/2$$

7: sinon

8: si T[mid] < x alors

9:
$$1 \leftarrow mid + 1$$



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

10

l.mid.r

45

18 27

```
Entrée: un tableau trié T et un élément x Sortie: i tel que T[i] = x ou NonTrouvé
```

2:
$$n \leftarrow \text{longueur de T, 1} \leftarrow 0, r \leftarrow n-1$$

3: tant que
$$1 \le r$$
 faire

4:
$$mid \leftarrow (1+r)/2$$

9:
$$1 \leftarrow \text{mid} + 1$$

Exemple. Rechercher l'élément 27

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

10

18

```
Entrée: un tableau trié T et un élément x Sortie: i tel que T[i] = x ou NonTrouvé
```

2:
$$n \leftarrow \text{longueur de T, } 1 \leftarrow 0, r \leftarrow n-1$$

3: tant que
$$1 \le r$$
 faire

4:
$$mid \leftarrow (1+r)/2$$

9:
$$1 \leftarrow \text{mid} + 1$$

11:
$$r \leftarrow mid - 1$$

Exemple. Rechercher l'élément 27

3

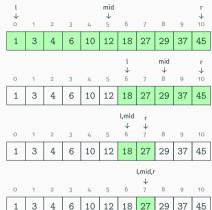
	l ↓ 0	1	2	3	4	mid ↓ 5	6	7	8	9	r ↓ 10	
	1	3	4	6	10	12	18	27	29	37	45	
	0	1	2	3	4	5	l → 6	7	mid ↓ 8	9	r → 10	
	1	3	4	6	10	12	18	27	29	37	45	
,	0	1	2	3	4	5	l,mid ↓ 6	r ↓ 7	8	9	10	
	1	3	4	6	10	12	18	27	29	37	45	
	l,mid,r ↓											
	0	1	2	3	4	5	6	7	8	9	10	

10

12 18 27

```
Entrée: un tableau trié T et un élément x
Sortie: i tel que T[i] = x ou NonTrouvé
 1: fonction RECHDICHOITÉRATIVE(T, x)
        n \leftarrow longueur de T. 1 \leftarrow 0, r \leftarrow n-1
        tant que 1 \le r faire
 3:
            mid \leftarrow (1+r)/2
            si T[mid] = x alors
                 retourne mid
 6:
            sinon
                si T[mid] < x alors
                     1 \leftarrow mid + 1
 9:
                 sinon
10:
                     r \leftarrow mid - 1
 11:
        retourne NonTrouvé
```

12:



Recherche dichotomique: Correction

RECHDICHOREC(T, 1, r, x) retourne i tel que $1 \le i \le r$ et T[i] = x ou NonTrouvé si x n'est

pas dans le tableau dans l'intervalle T[1,...,r].

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

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

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

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

<u>Cas de base:</u> r-1+1=0. Dans ce cas r<1, 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 r - 1 + 1 < n, où $n \ge 1$, et nous la prouvons pour r - 1 + 1 = n.

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

<u>Cas de base:</u> r-1+1=0. Dans ce cas r<1, 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 r-1+1 < n, où $n \ge 1$, et nous la prouvons pour r-1+1=n. Soit mid=(1+r)/2; il y a trois cas :

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

<u>Cas de base:</u> r-1+1=0. Dans ce cas r<1, 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 r - 1 + 1 < n, où $n \ge 1$, et nous la prouvons pour r - 1 + 1 = n. Soit mid = (1 + r)/2; il y a trois cas :

• T[mid]=x. Dans ce cas RECHDICHOREC(T,1,r,x) retourne mid, ce qui est correct puisque x se trouve dans le tableau à la position mid.

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

<u>Cas de base</u>: r-1+1=0. Dans ce cas r<1, 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 r - l + 1 < n, où $n \ge 1$, et nous la prouvons pour r - l + 1 = n. Soit mid = (1 + r)/2; il y a trois cas :

- T[mid]=x. Dans ce cas RECHDICHOREC(T,1,r,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 RECHDICHOREC(T, mid + 1, r, x). Vu que le tableau T est trié, s'il existe 1 ≤ i ≤ r tel que T[i] = x, alors mid + 1 ≤ i ≤ r; on a aussi r (mid + 1) + 1 < n. En utilisant l'hypothèse de récurrence, le résultat renvoyé par RECHDICHOREC(T, mid + 1, r, x) est correct.

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

<u>Cas de base:</u> r-1+1=0. Dans ce cas r<1, 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 r - 1 + 1 < n, où $n \ge 1$, et nous la prouvons pour r - 1 + 1 = n. Soit mid = (1 + r)/2; il y a trois cas :

- T[mid]=x. Dans ce cas RECHDICHOREC(T,1,r,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 RECHDICHOREC(T, mid + 1, r, x). Vu que le tableau T est trié, s'il existe 1 ≤ i ≤ r tel que T[i] = x, alors mid + 1 ≤ i ≤ r; on a aussi r (mid + 1) + 1 < n. En utilisant l'hypothèse de récurrence, le résultat renvoyé par RECHDICHOREC(T, mid + 1, r, x) est correct.
 - T[mid] >x. Similaire.

Recherche dichotomique: Terminaison et Complexité

 \rightarrow 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 $l \leftarrow \left| \frac{l+r}{2} \right| + 1$ (l s'incremènte au moins par un), soit $r \leftarrow \left| \frac{l+r}{2} \right| - 1$ (r se diminue au moins par un).

un), soit $r \leftarrow \left| \frac{l+r}{2} \right| - 1$ (r se diminue au moins par un).

l'algorithme n'est pas encore fini, soit
$$l \leftarrow \left\lfloor \frac{r}{2} \right\rfloor + 1$$
 (l s'incremente au moins par un), soit $r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$ (r se diminue au moins par un).

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow Combien de fois la boucle s'exécute-t-elle au pire de cas (ex. quand $x \notin T[l \dots r]$)?

un), soit $r \leftarrow \left| \frac{l+r}{2} \right| - 1$ (r se diminue au moins par un).

un), soit $r \leftarrow \left| \frac{l+r}{2} \right| - 1$ (r se diminue au moins par un). \rightarrow 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-1+1 se divise par 2 :

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow 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-1+1 se divise par 2 :

Après 1 exécution,

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow 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

Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle
$$t = r-1+1$$
 se divise par 2 :

• Soit
$$r - l + 1 \leftarrow \frac{2r}{2} - \left(\left| \frac{l+r}{2} \right| + 1 \right) + 1 = \left| \frac{l+r+1}{2} \right| = \left| \frac{t}{2} \right|$$
.

→ 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 $r \leftarrow \left| \frac{l+r}{2} \right| - 1$ (r se diminue au moins par un).

$$\rightarrow$$
 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-1+1 se divise par 2 :

• Soit $r-l+1=\left(\left|\frac{l+r}{2}\right|-1\right)-\frac{2l}{2}+1\leq \left|\frac{l+r+1}{2}\right|=\left|\frac{t}{2}\right|.$

Fintervalle
$$t = r-1+1$$
 se divise par 2:
Après 1 exécution,
• Soit $r-l+1 \leftarrow \frac{2r}{2} - \left(\left| \frac{l+r}{2} \right| + 1 \right) + 1 = \left| \frac{l+r+1}{2} \right| = \left| \frac{t}{2} \right|$.

Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle
$$t = r-1+1$$
 se divise par 2 :

Après 1 exécution,

 $2r \left(\left| l+r \right| \right) = \left| l+r+1 \right| + t \right|$

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow 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

Après 1 exécution, $r - l + 1 \le |t/2|$.

l'intervalle t = r-1+1 se divise par 2 :

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow 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

Après chaque exécution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle
$$t = r-1+1$$
 se divise par 2 :

l'intervalle t = r-1+1 se divise par 2 : Après 1 exécution, $r - l + 1 \le |t/2|$.

Après 2 exécutions, $r - l + 1 \le |t/2^2|$.

un), soit
$$r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$$
 (r se diminue au moins par un).

 \rightarrow 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

Après 1 exécution
$$r = l + 1 < |t/2|$$

Après 2 exécutions, $r - l + 1 \le |t/2^2|$. Après 3 exécutions, $r - l + 1 \le |t/2^3|$.

l'intervalle
$$t = r-1+1$$
 se divise par 2 :
Après 1 exécution, $r - l + 1 \le \lfloor t/2 \rfloor$.

un), soit
$$r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor - 1$$
 (r se diminue au moins par un).

 \rightarrow 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

Apres chaque execution (et si l'algorithme toujours n'a pas fini), la taille de l'intervalle
$$t = r-1+1$$
 se divise par 2 :

Après 2 exécutions,
$$r - l + 1 \le \lfloor t/2 \rfloor$$
.
Après 3 exécutions, $r - l + 1 \le \lfloor t/2^2 \rfloor$.
Après i exécutions, $r - l + 1 \le \lfloor t/2^3 \rfloor$.

l'intervalle
$$t = r-1+1$$
 se divise par 2 :

Après 1 exécution, $r - l + 1 \le \lfloor t/2 \rfloor$.

un), soit
$$r \leftarrow \left| \frac{l+r}{2} \right| - 1$$
 (r se diminue au moins par un).

 \rightarrow 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-1+1 se divise par 2 :

Après 1 exécution,
$$r - l + 1 \le \lfloor t/2 \rfloor$$
.
Après 2 exécutions, $r - l + 1 \le \lfloor t/2^2 \rfloor$.

Après 3 exécutions, $r - l + 1 \le |t/2^3|$ Après i exécutions, $r - l + 1 \le |t/2^i|$.

Alors, pour que
$$r - l + 1 < 1$$
 il faut que $t/2^i < 1 \Rightarrow i > \log t$ exécutions.

un), soit
$$r \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor$$
 – 1 (r se diminue au moins par un).

 \rightarrow 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, $r - l + 1 \le \lfloor t/2 \rfloor$. Après 2 exécutions, $r - l + 1 \le \lfloor t/2^2 \rfloor$.

Après 3 exécutions, $r - l + 1 \le \lfloor t/2^3 \rfloor$Après *i* exécutions, $r - l + 1 \le \lfloor t/2^i \rfloor$.

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 $l \le r$. 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).

 \rightarrow 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 2 exécutions, $r - l + 1 \le \lfloor t/2^2 \rfloor$. Après 3 exécutions, $r - l + 1 \le \lfloor t/2^3 \rfloor$.

Après 1 exécution, $r - l + 1 \le |t/2|$.

... Après *i* exécutions, $r - l + 1 \le \lfloor t/2^i \rfloor$.

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.