Introduction à l'informatique
Algorithmes et arithmétique

Portail René Descartes

Luminy

Université d'Aix-Marseille

Plan du cours

1. Structures linéaires (fin)

2. Algorithmes sur les entiers et les réels

Plan du cours

Récursivité

- Un algorithme (ou fonction) qui s'appelle lui-même (ou elle-même) est récursif
- Exemple : calculer $n!$, pour $n \geqslant 1$

Fonction fiter(**d** n : **entier**) : **entier**
    i, f : **entier** ;
Début
    $f := 1$;
    $i := 2$;
    Tant que $(i \leqslant n)$ **faire**
        $f := f \times i$;
        $i := i + 1$;
    Fin faire
    retour f ;
Fin

Fonction frec(**d** n : **entier**) : **entier**
Début
    Si $n = 2$ **alors**
        retour 2 ;
    Sinon
        retour $n \times$ fact_rec$(n-1)$;
    Fin si
Fin

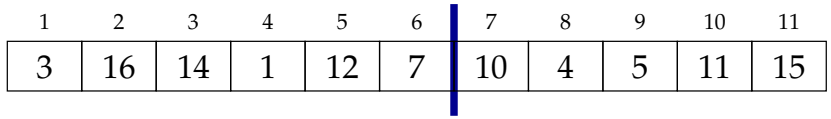
Tri fusion

- Algorithme de tri par comparaison stable
- Technique du “diviser pour régner”
- N’opérant pas en place
- Idée générale :
  1. Si le tableau n’a qu’un élément, il est déjà trié
  2. Sinon, séparer le tableau en 2 parties égales (à 1 près)
  3. trier récursivement les 2 parties avec l’algorithme
  4. Fusionner les deux tableaux triés en un seul tableau trié

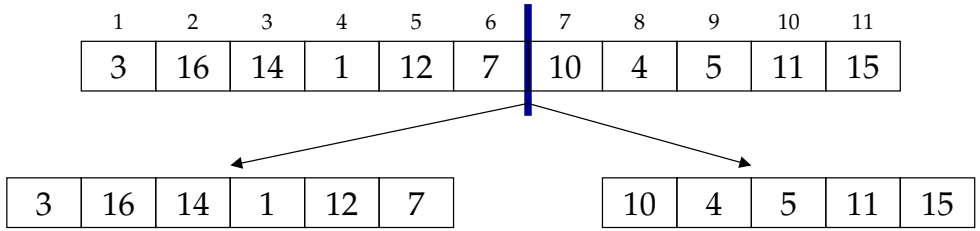
Tri fusion

1	2	3	4	5	6	7	8	9	10	11
3	16	14	1	12	7	10	4	5	11	15

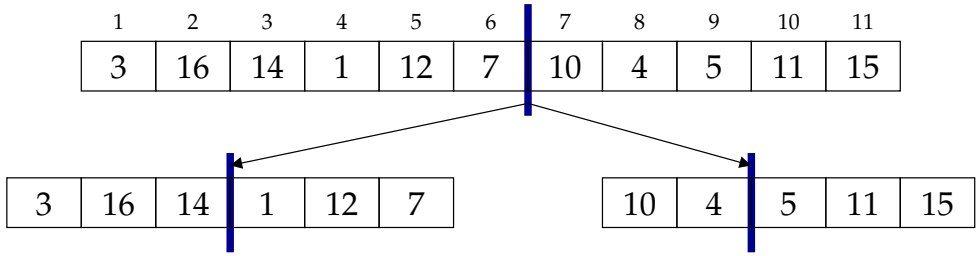
Tri fusion



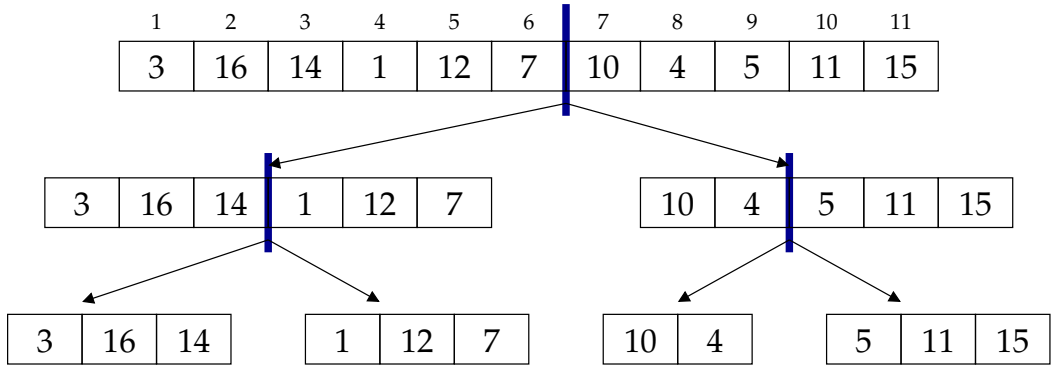
Tri fusion



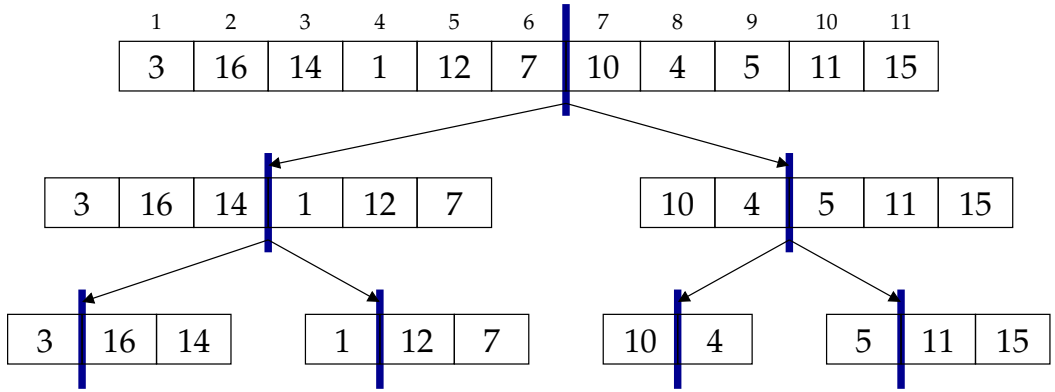
Tri fusion



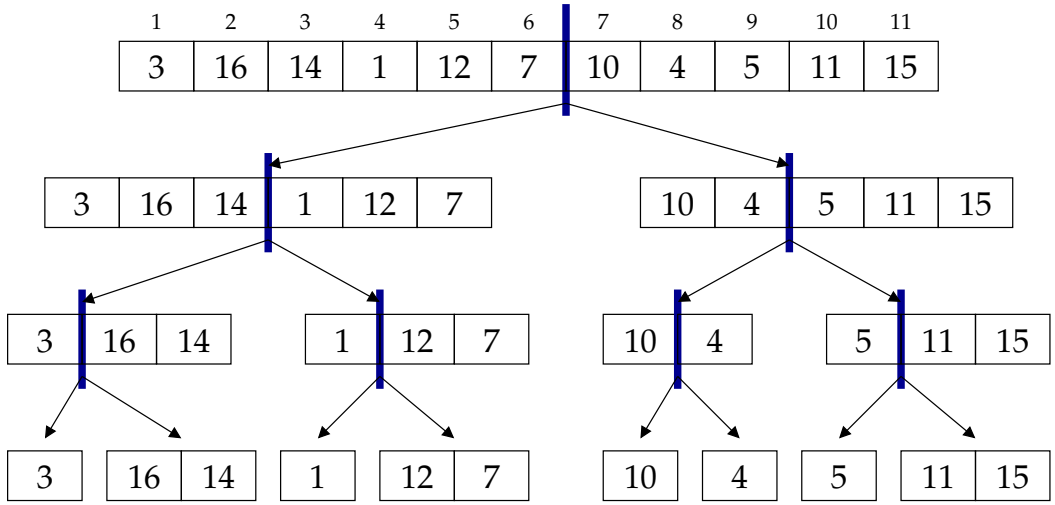
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15

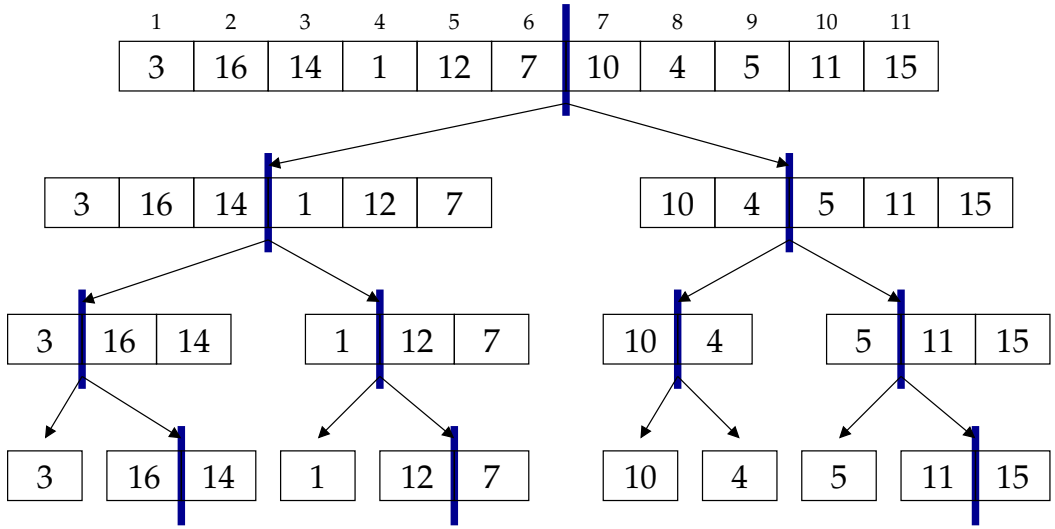
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15

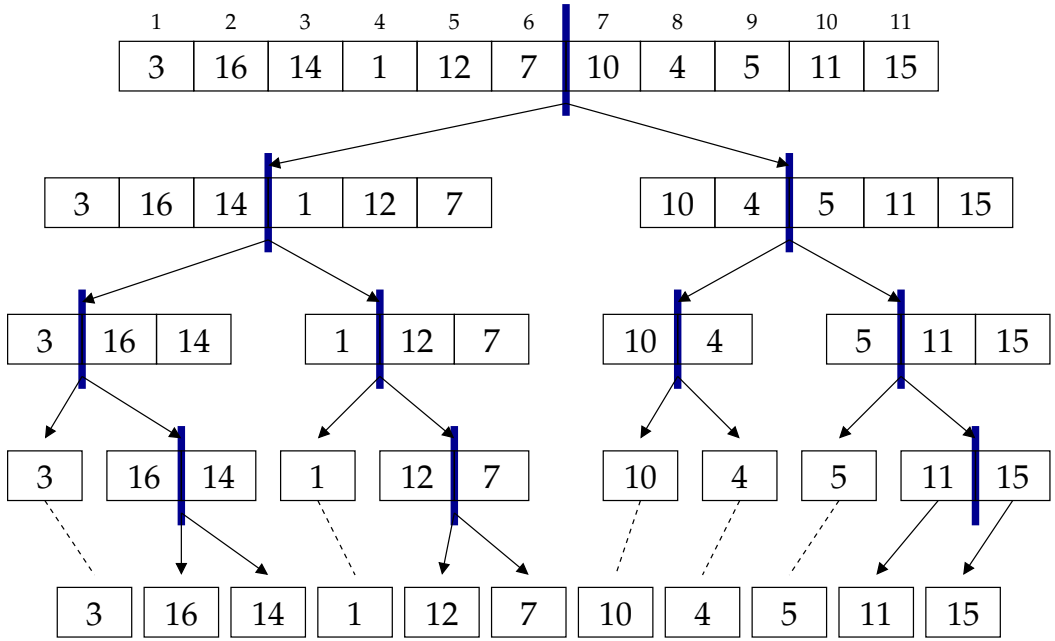
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15

Tri fusion

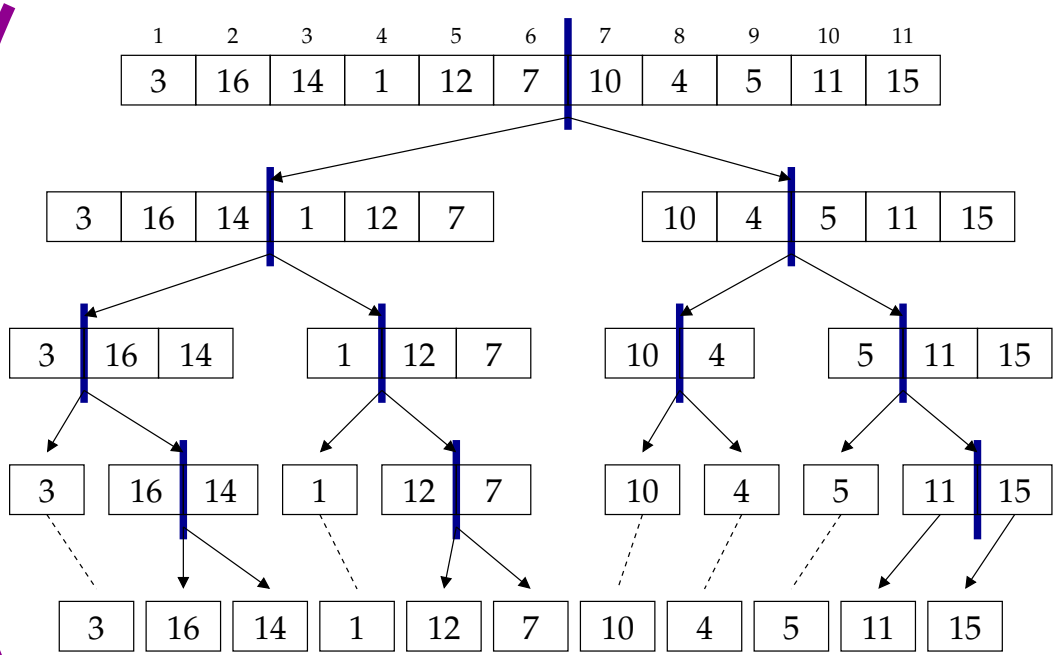

1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15

Tri fusion

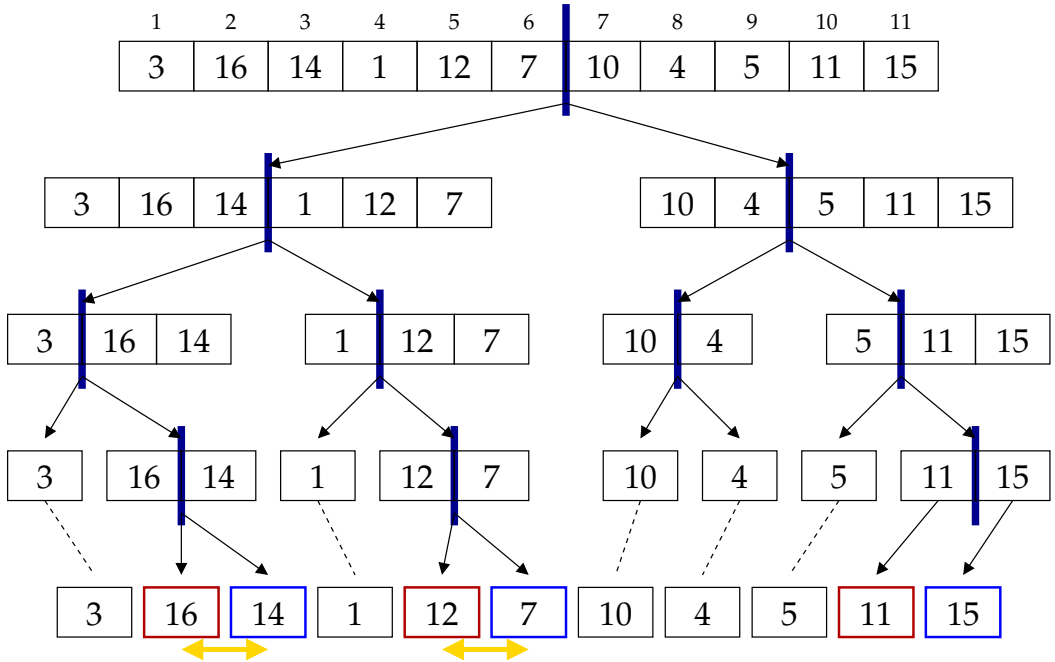

1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3 16 14 1 12 7 10 4 5 11 15

Tri fusion

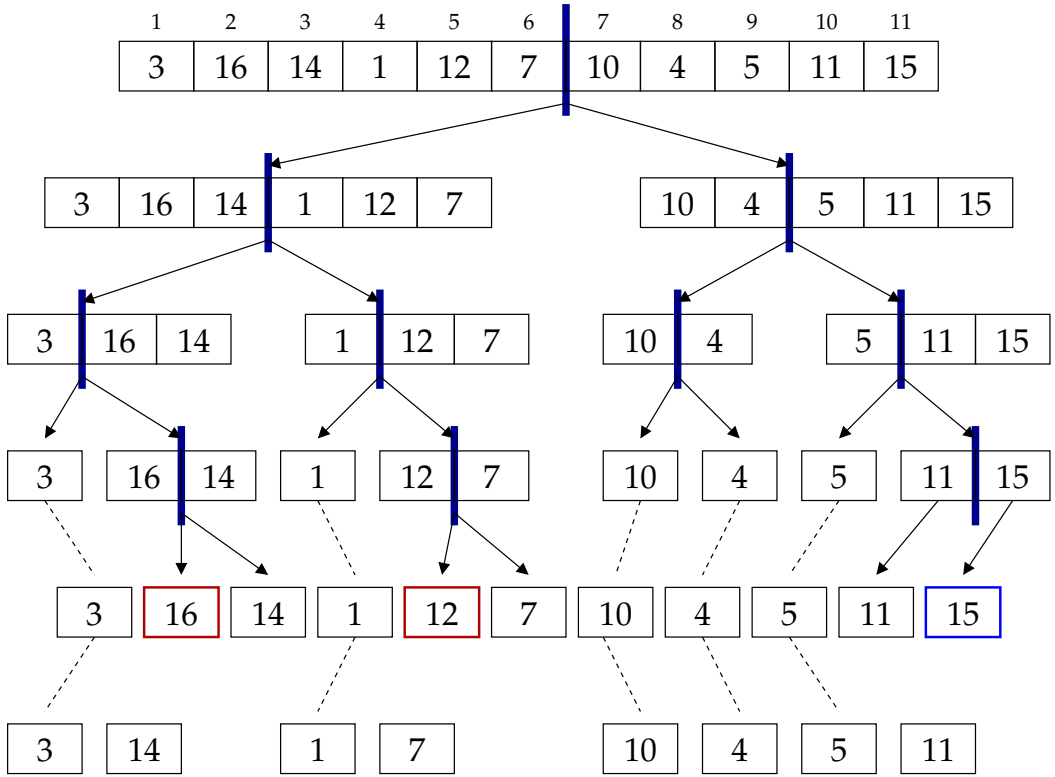
DIVISER



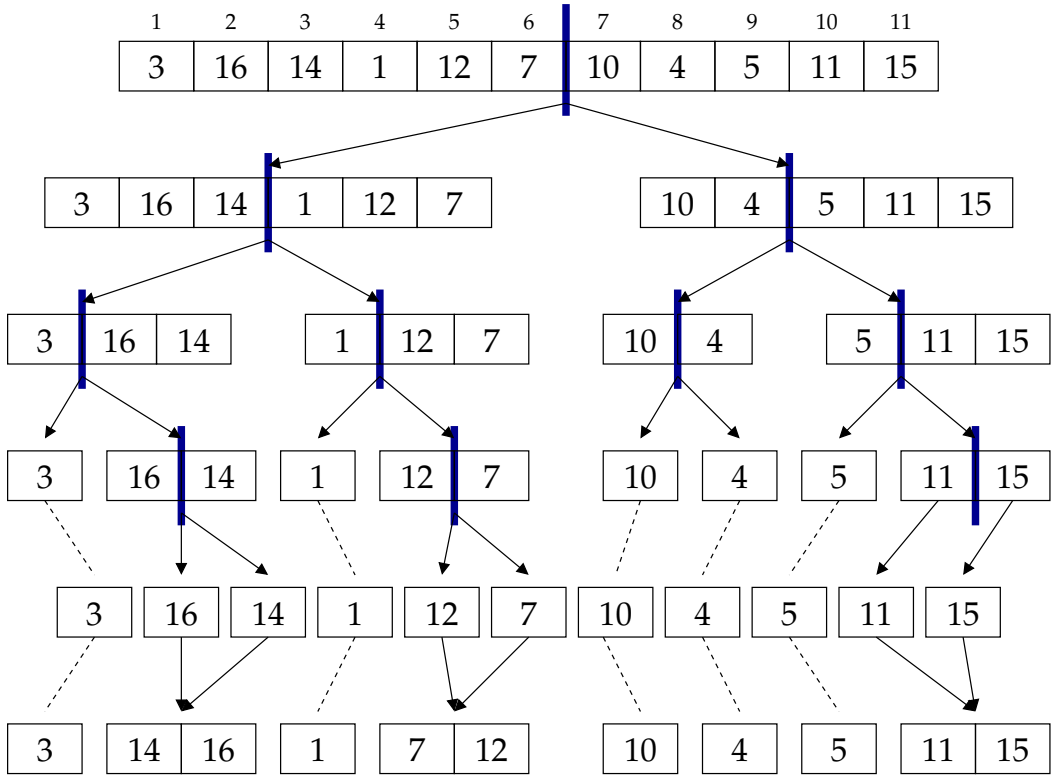
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3 16 14 1 12 7 10 4 5 11 15

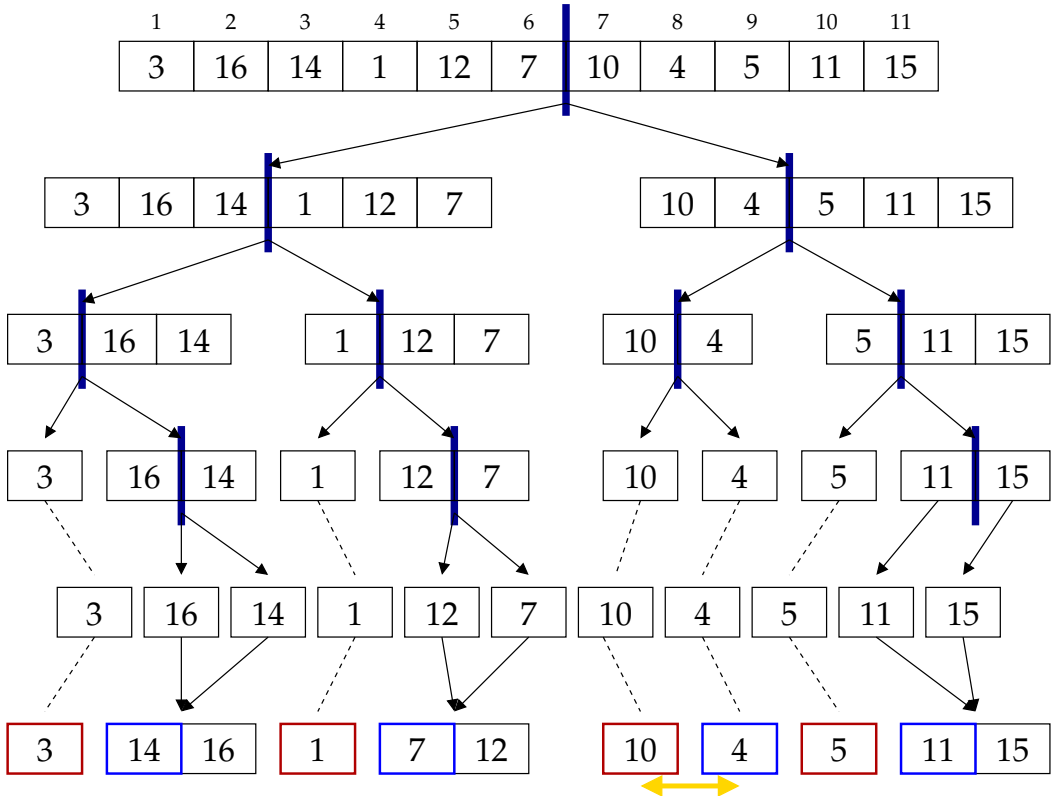
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3 16 14 1 12 7 10 4 5 11 15
3 14 1 7 10 4 5 11

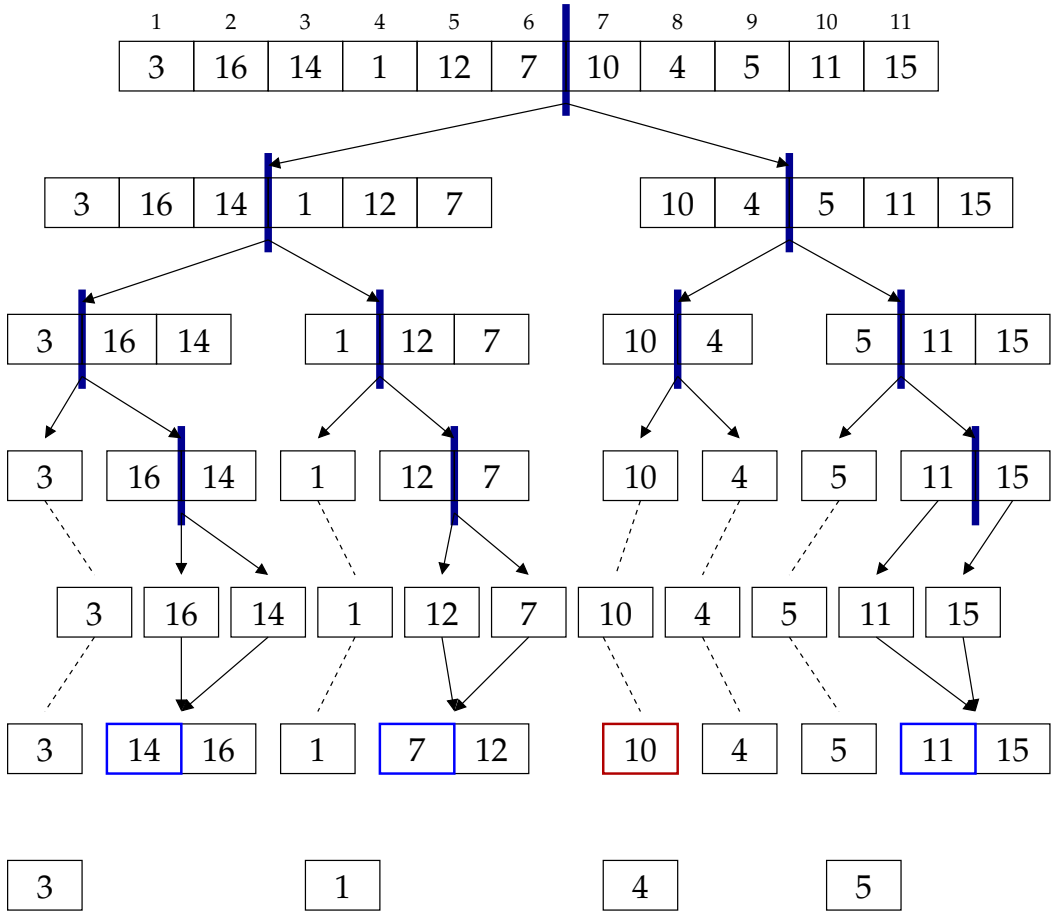
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15

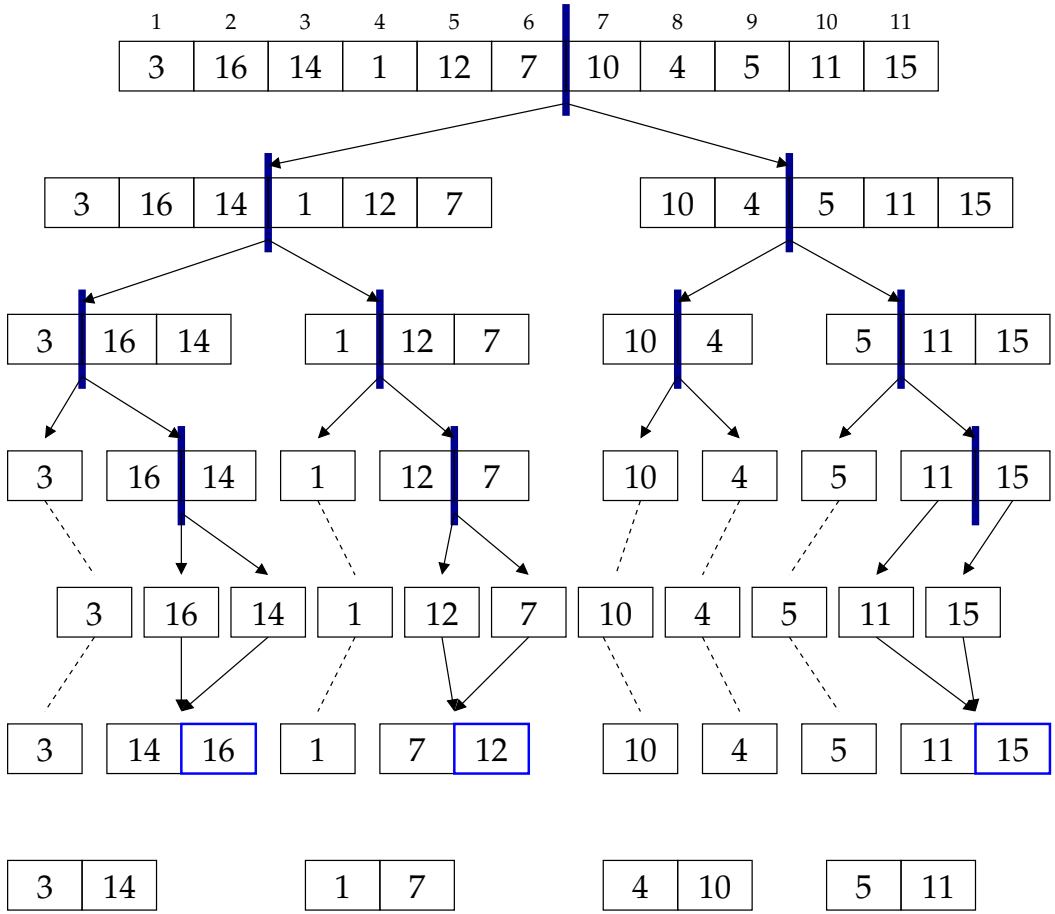
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15

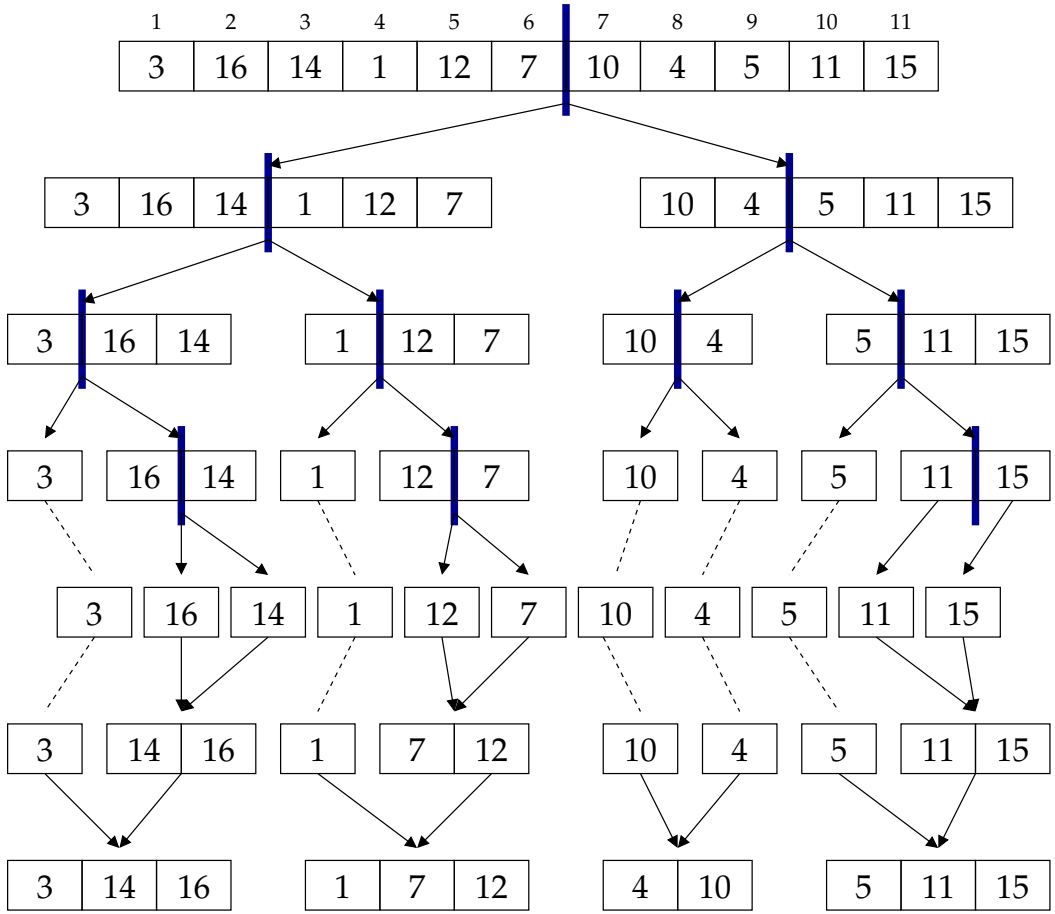
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
1
4
5

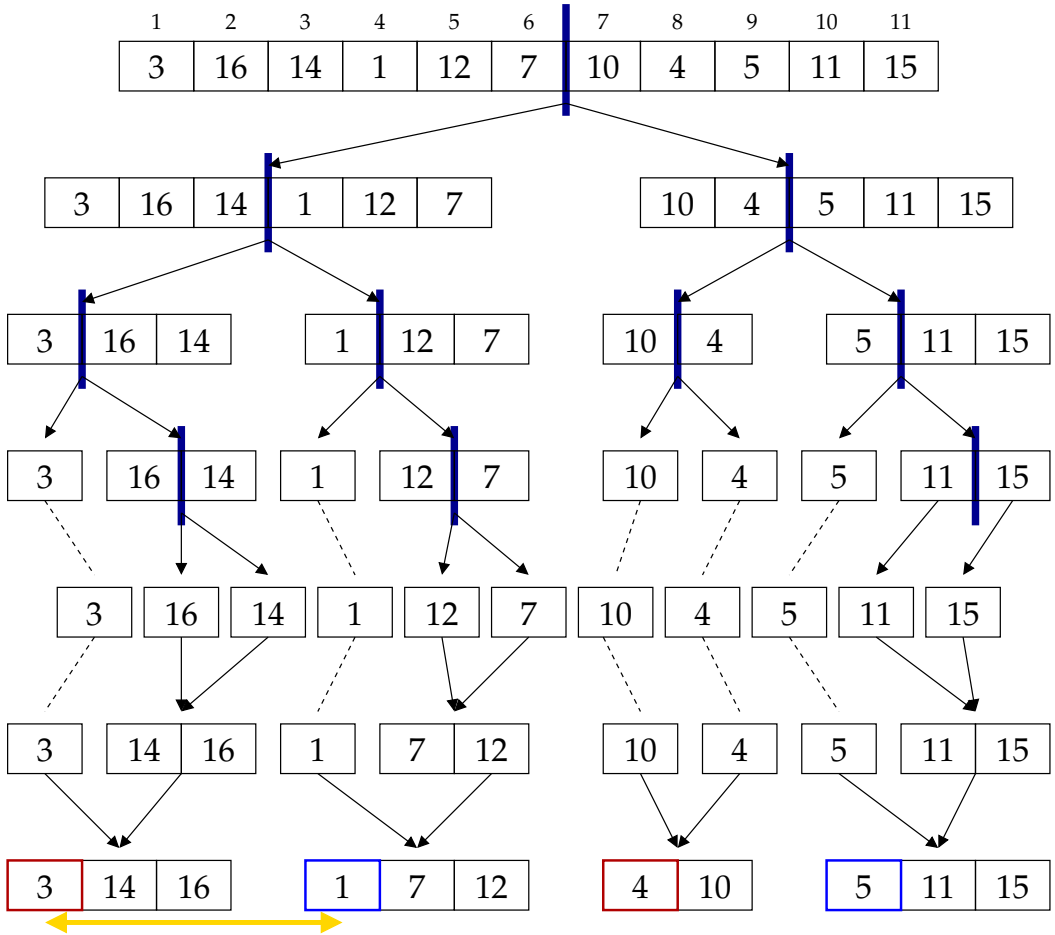
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3 14
1 7
4 10
5 11

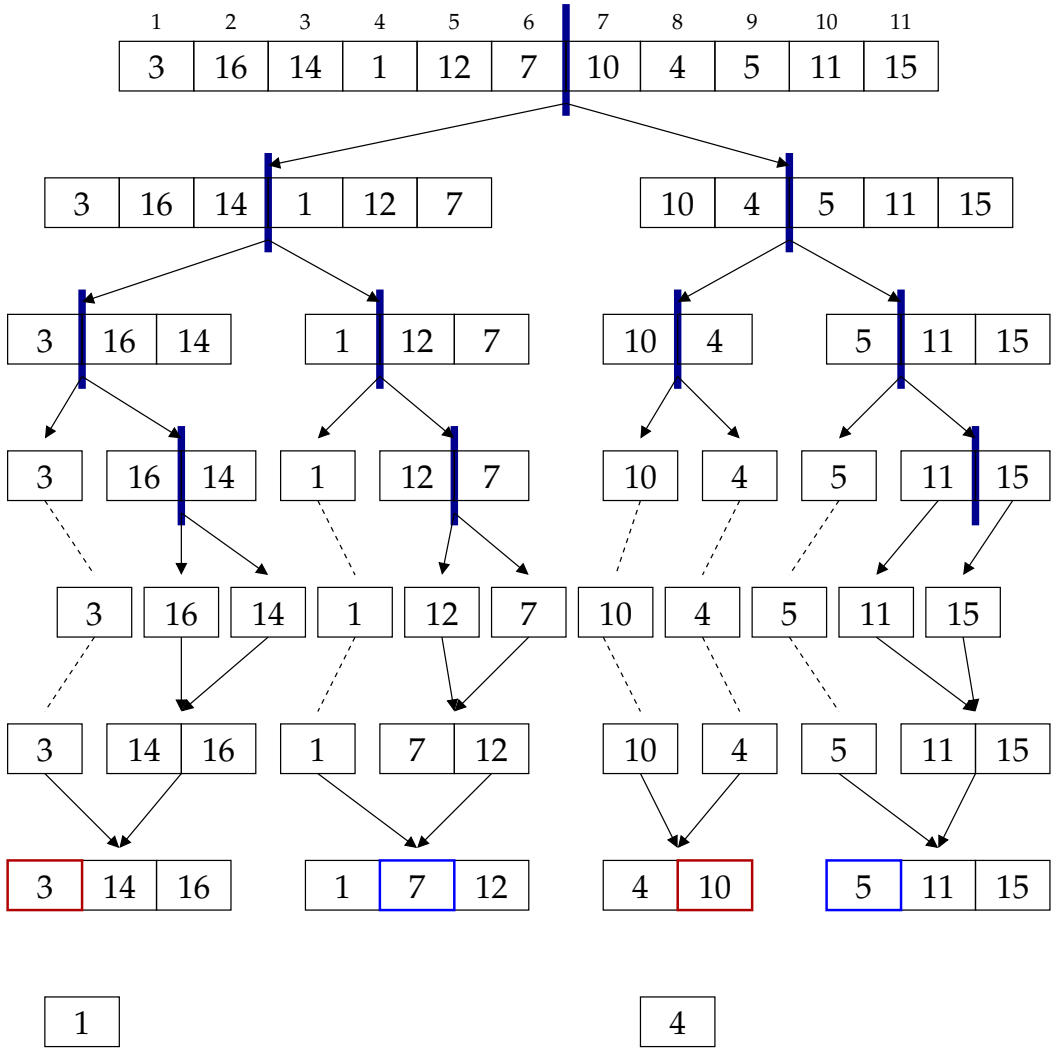
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14
16
1
7
12
4 10
5 11 15

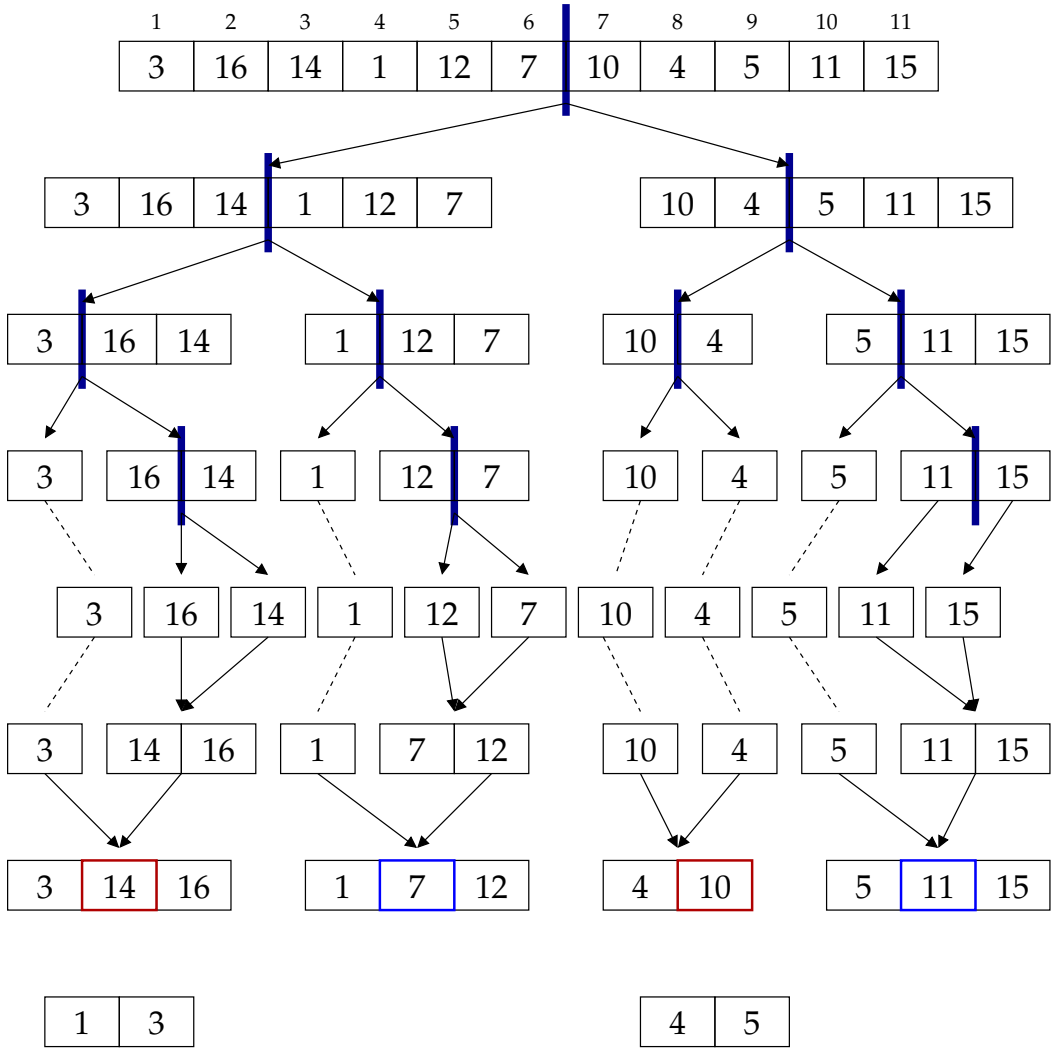
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14
16
1
7
12
4
10
5
11
15

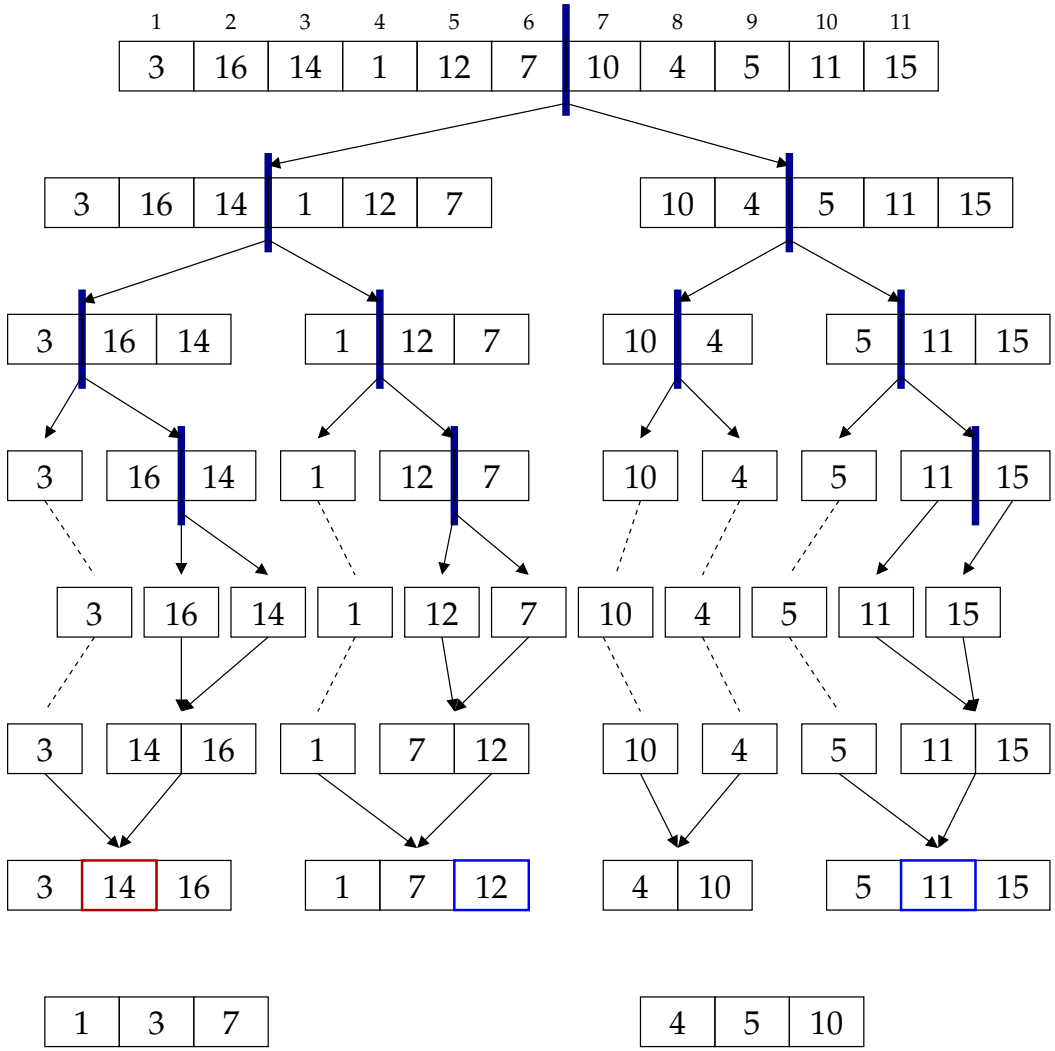
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14
16
1
7
12
4
10
5
11
15
1
4

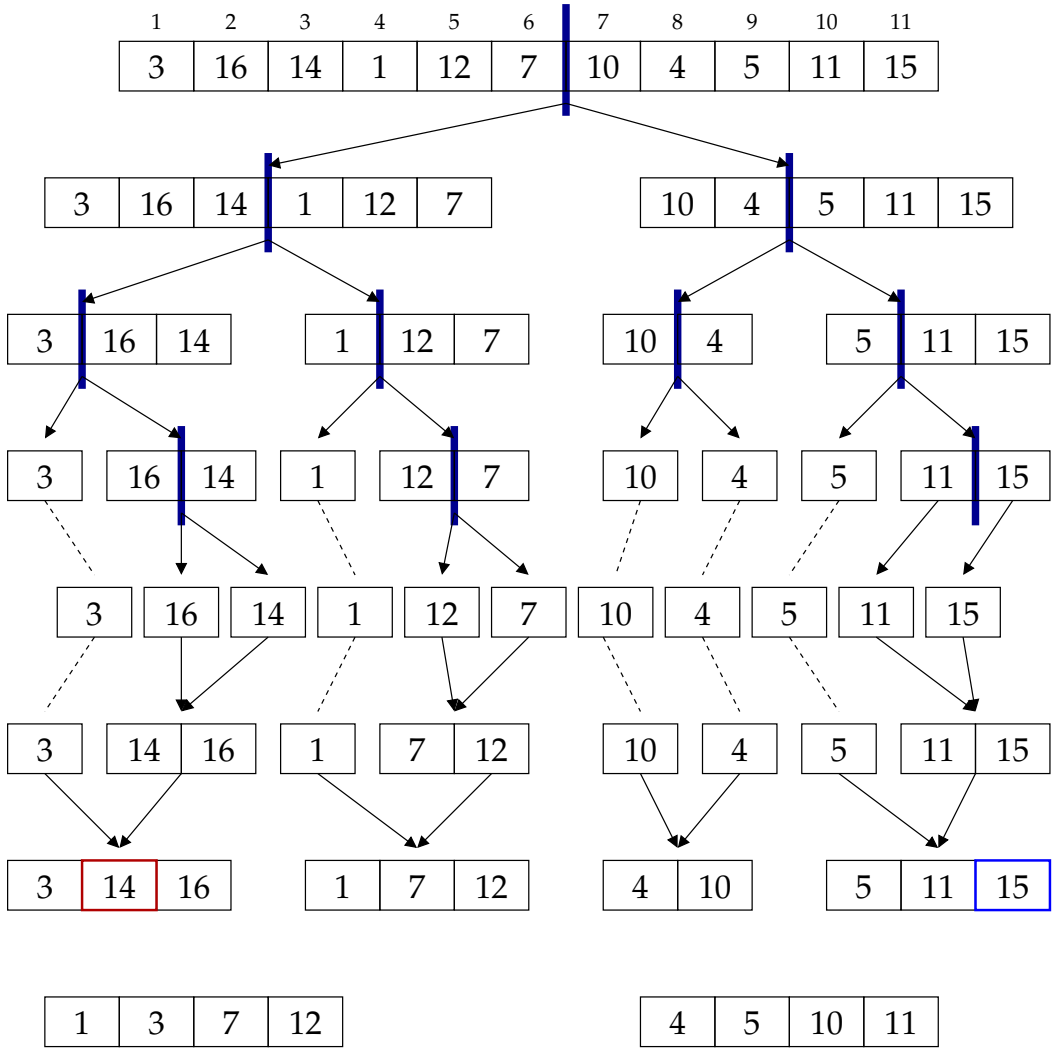
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3 14 16
1 7 12
10 4
5 11 15
3 14 16
1 7 12
4 10
5 11 15
1 3
4 5

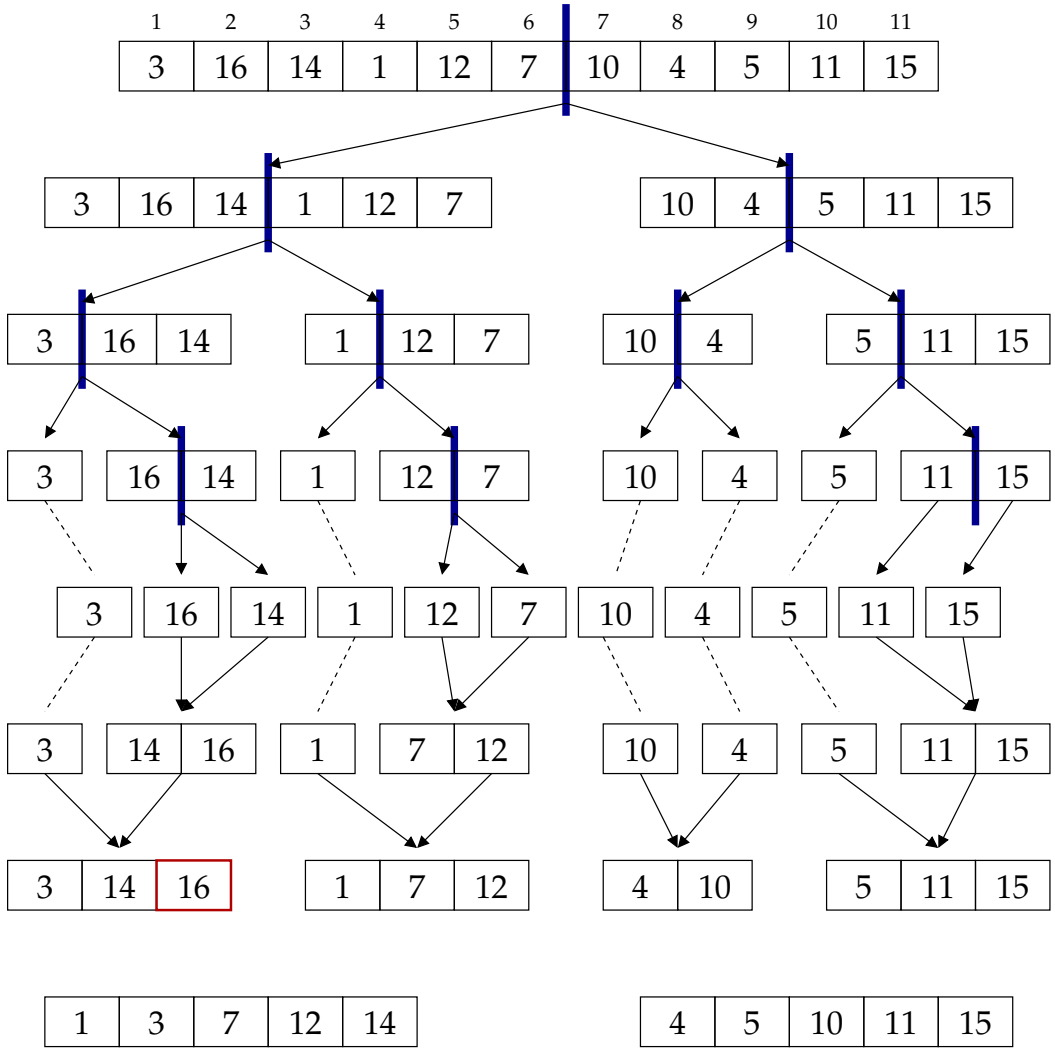
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3 14 16
1 7 12
10 4
5 11 15
3 14 16
1 7 12
4 10
5 11 15
1 3 7
4 5 10

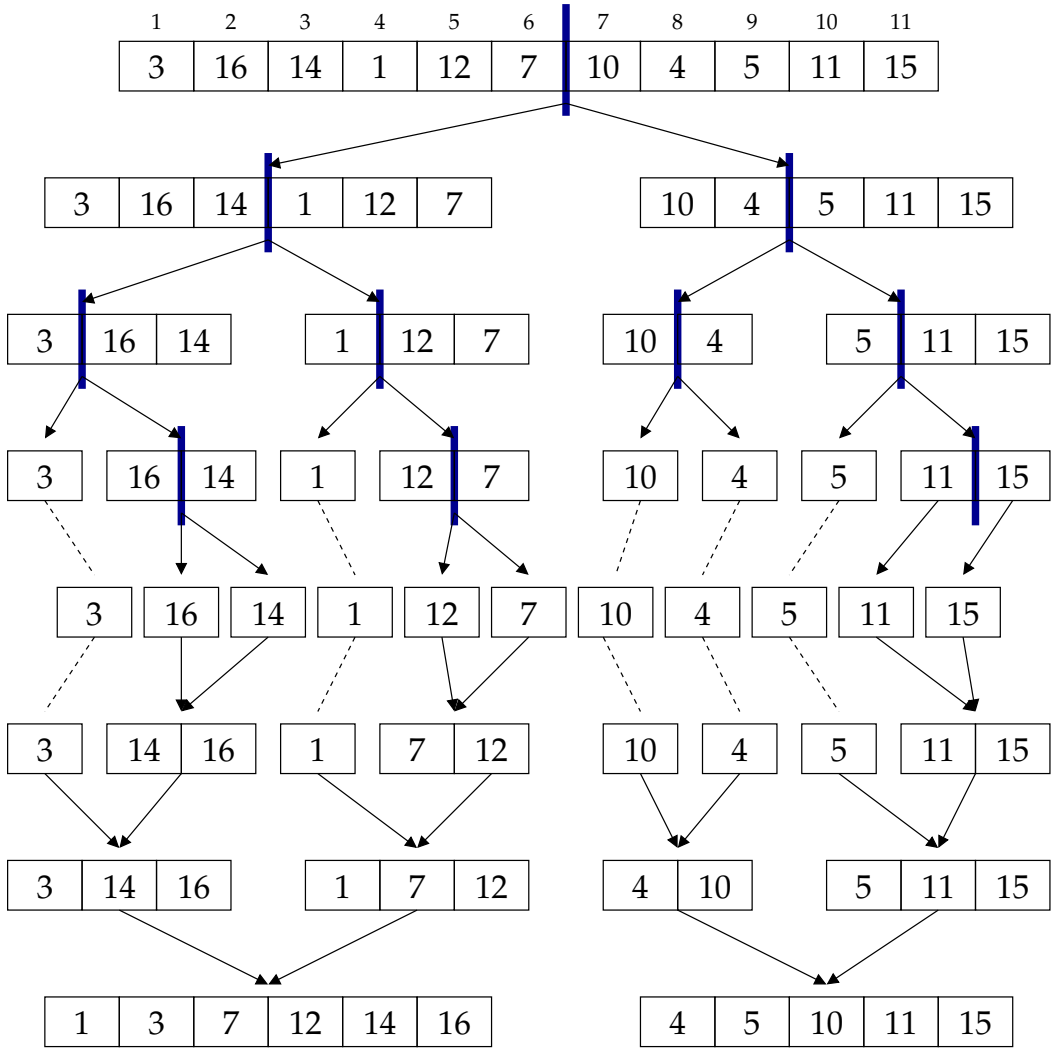
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14
16
1
7
12
4 10
5
11
15
1 3 7 12
4 5 10 11

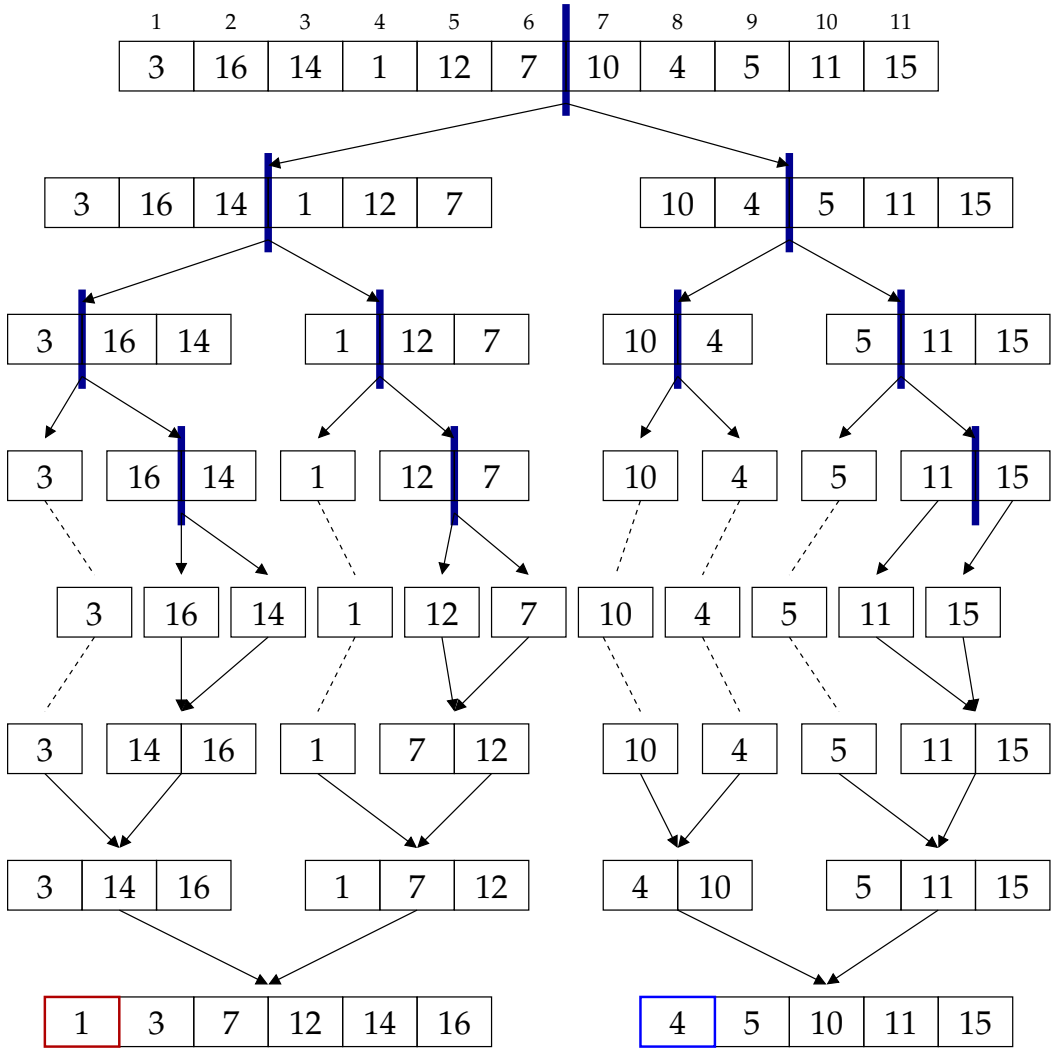
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14
16
1
7
12
4 10
5 11 15
1 3 7 12 14
4 5 10 11 15

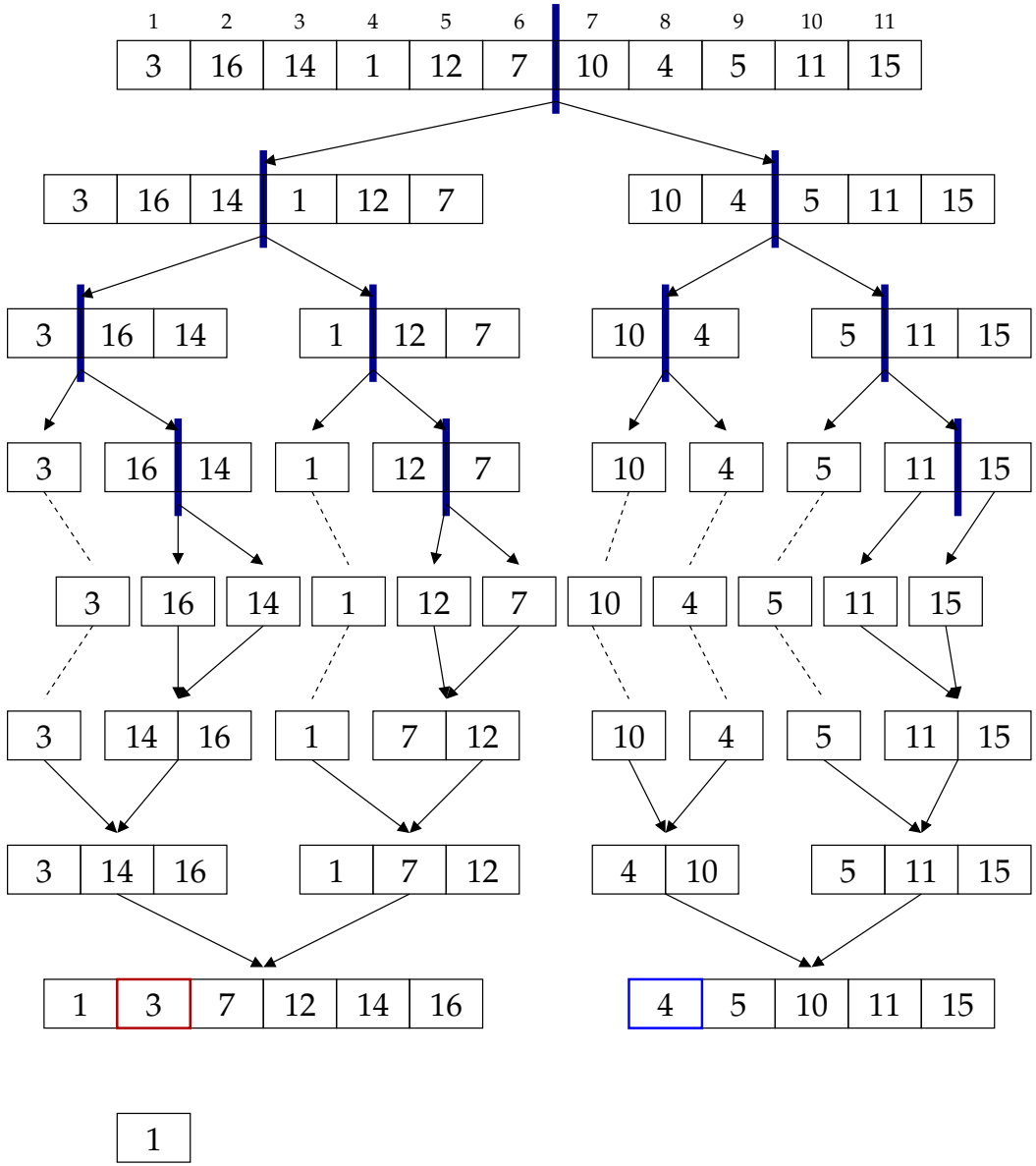
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1
7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15

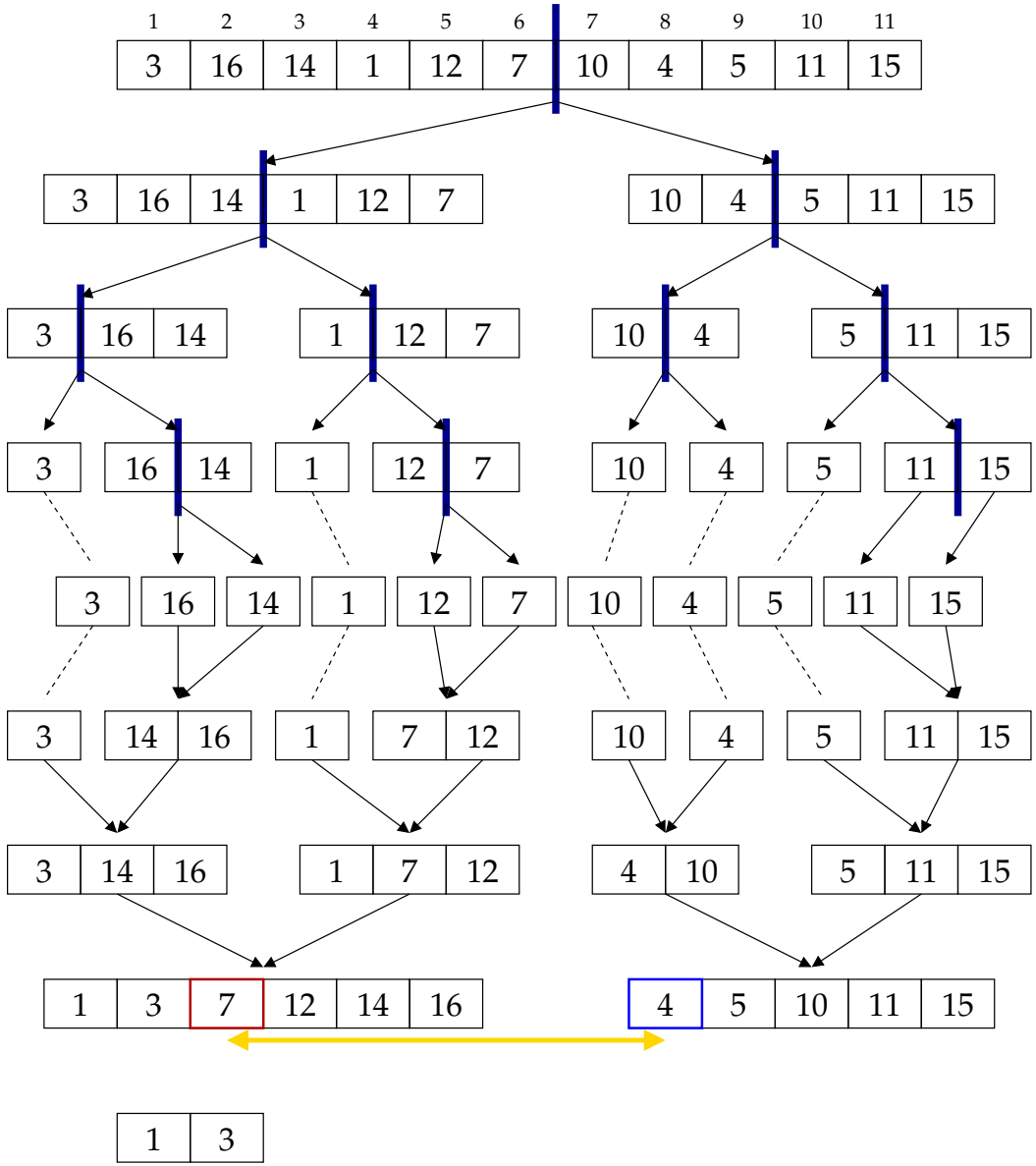
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1
7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15

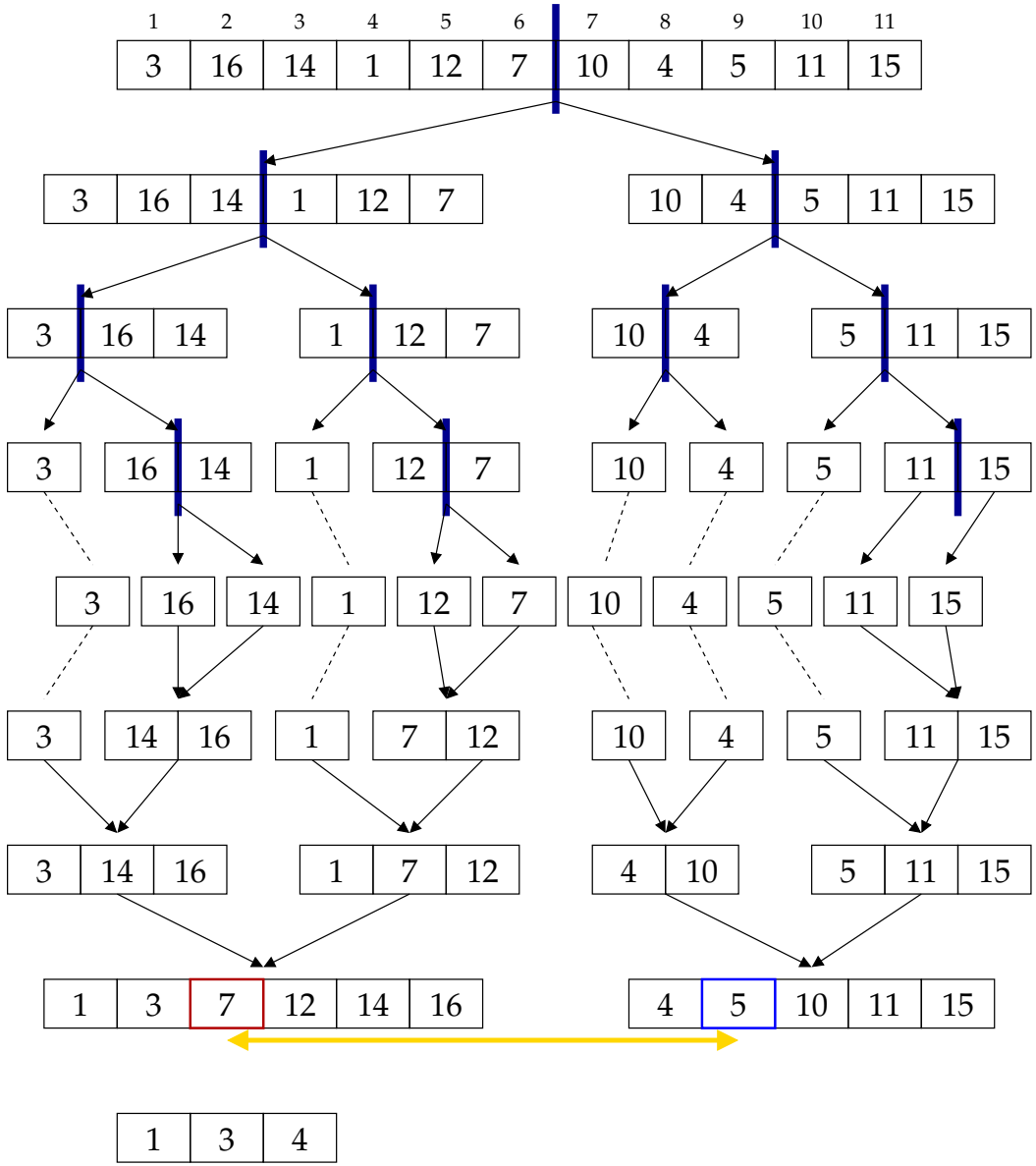
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1
7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1

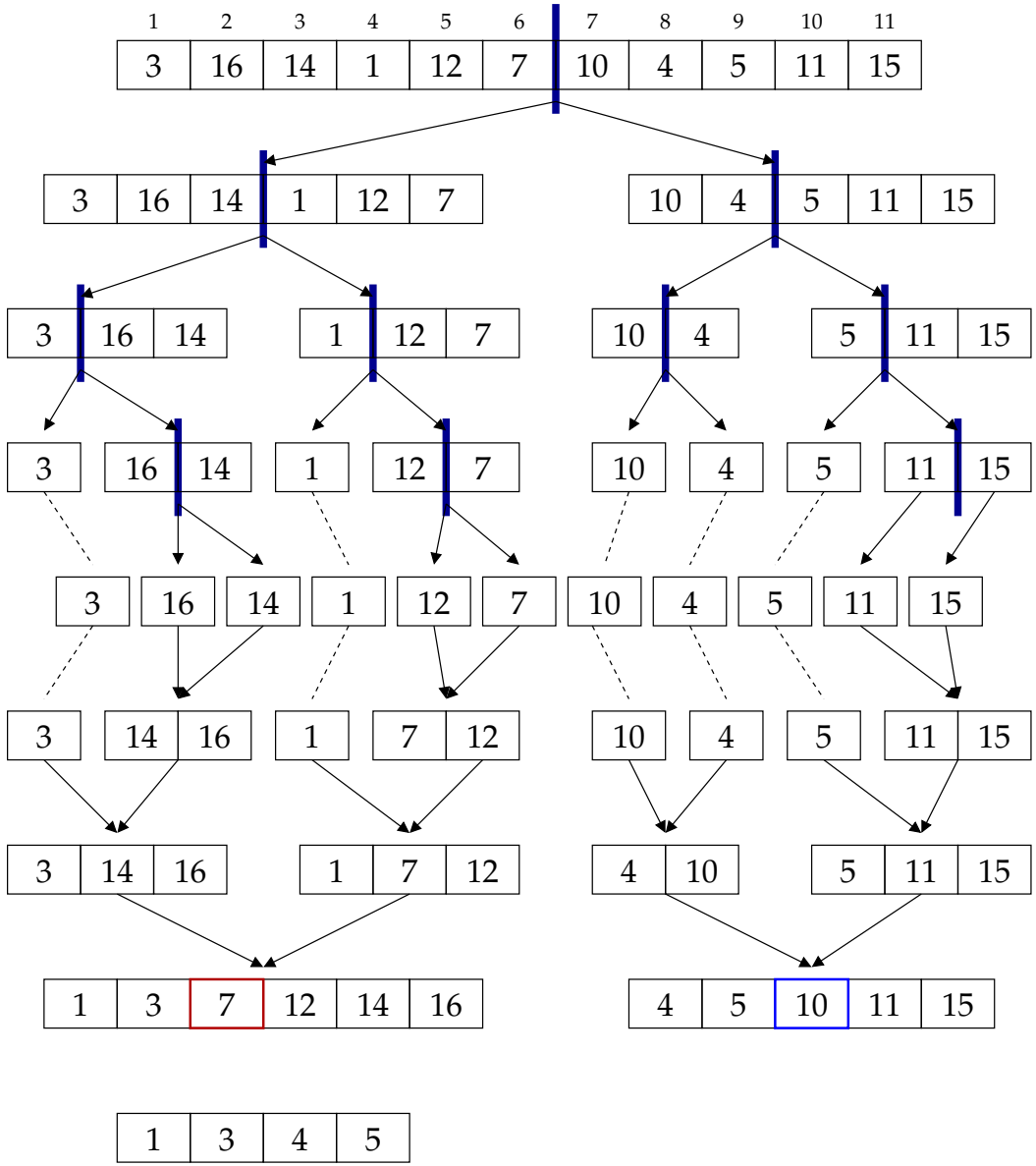
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3 14 16
1 7 12
10 4 5 11 15
3 14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3

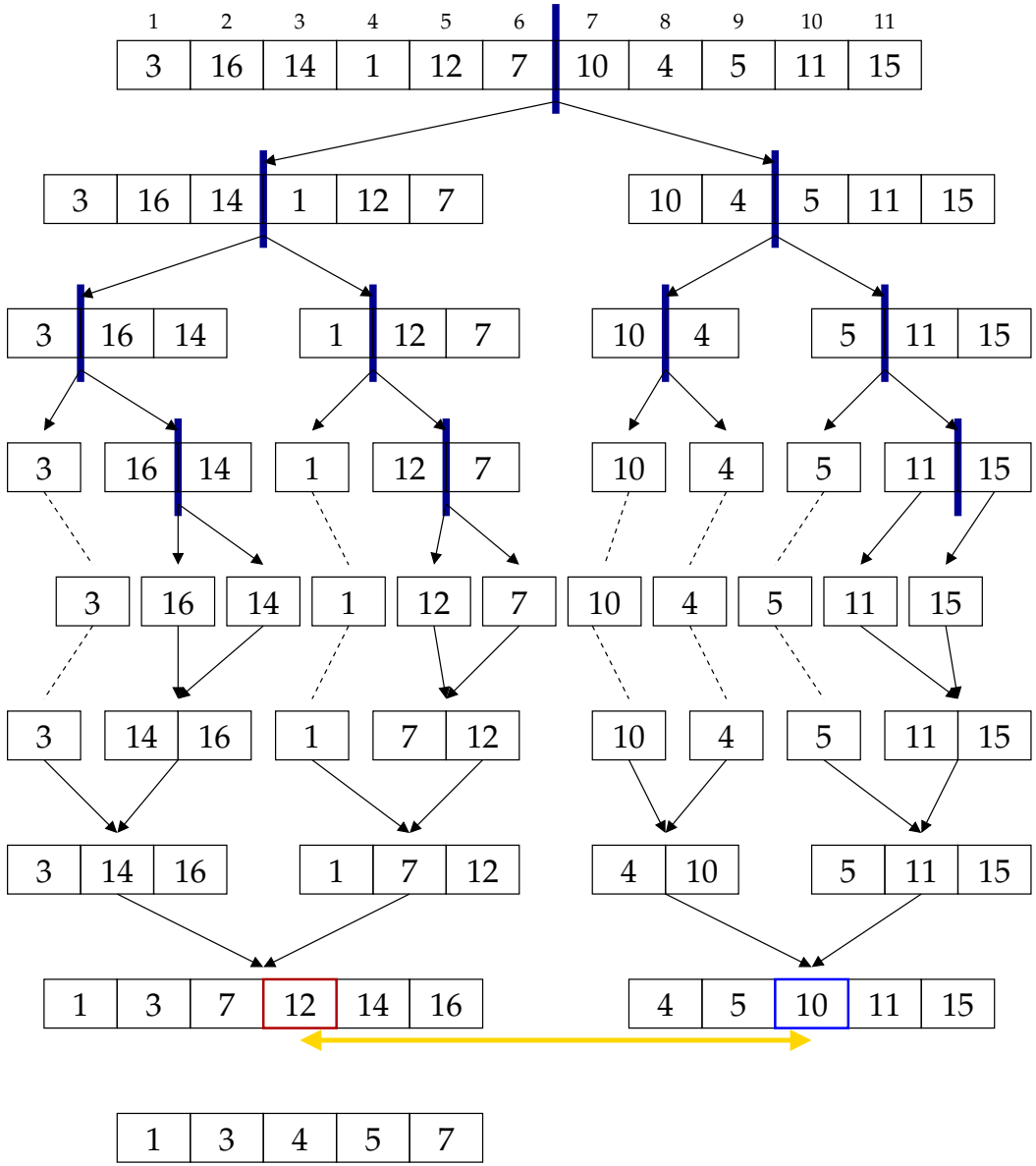
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4

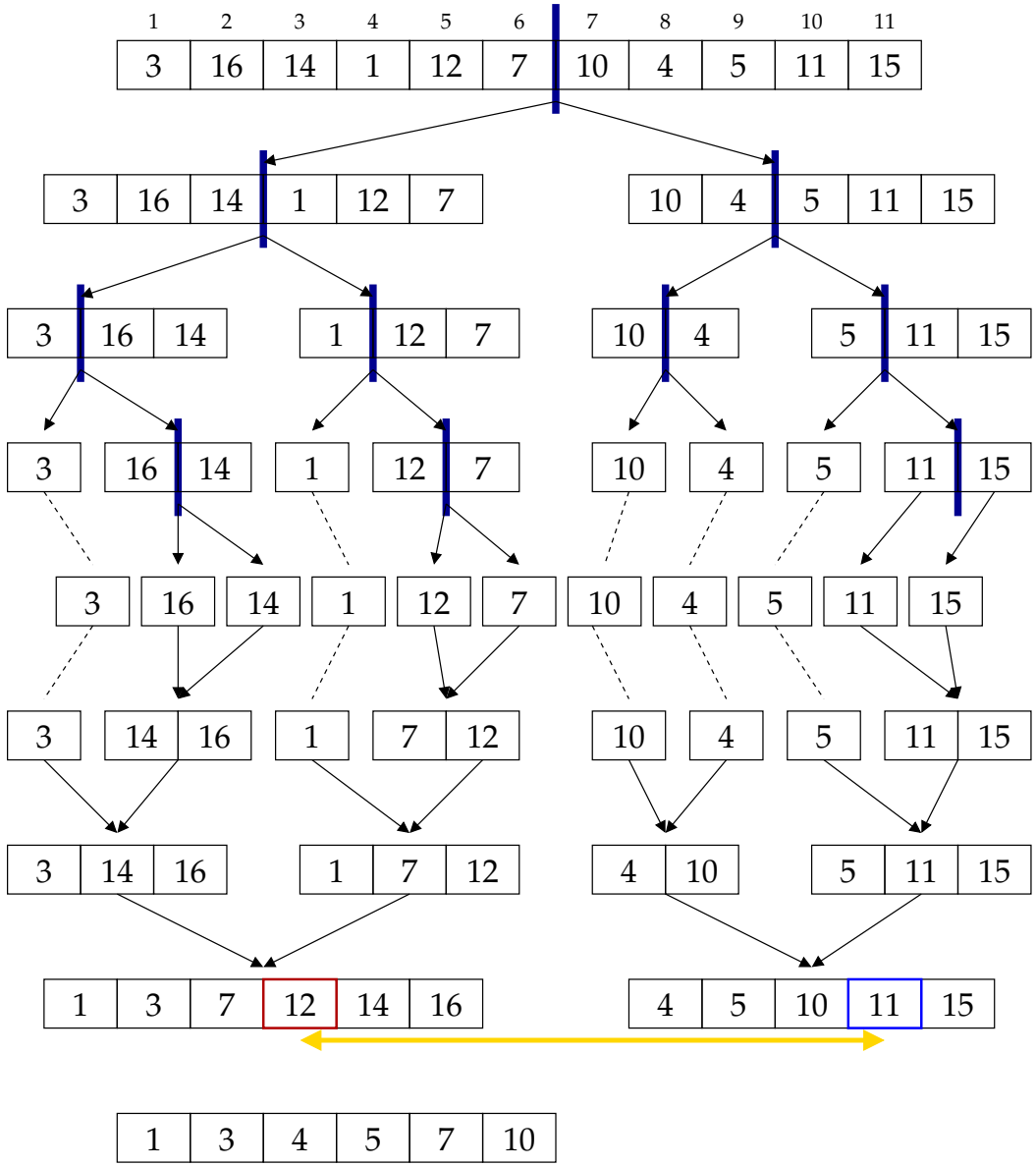
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5

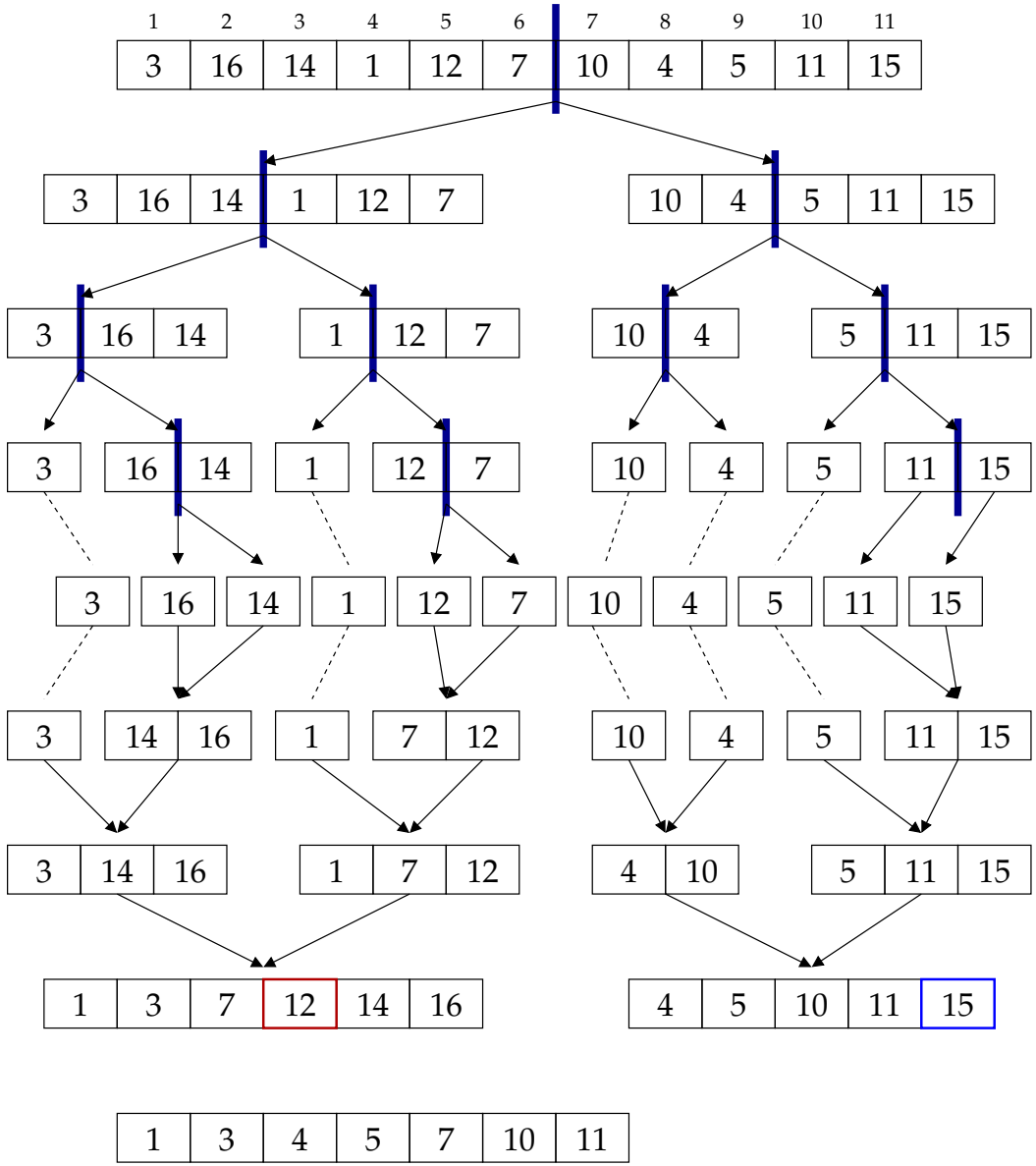
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7

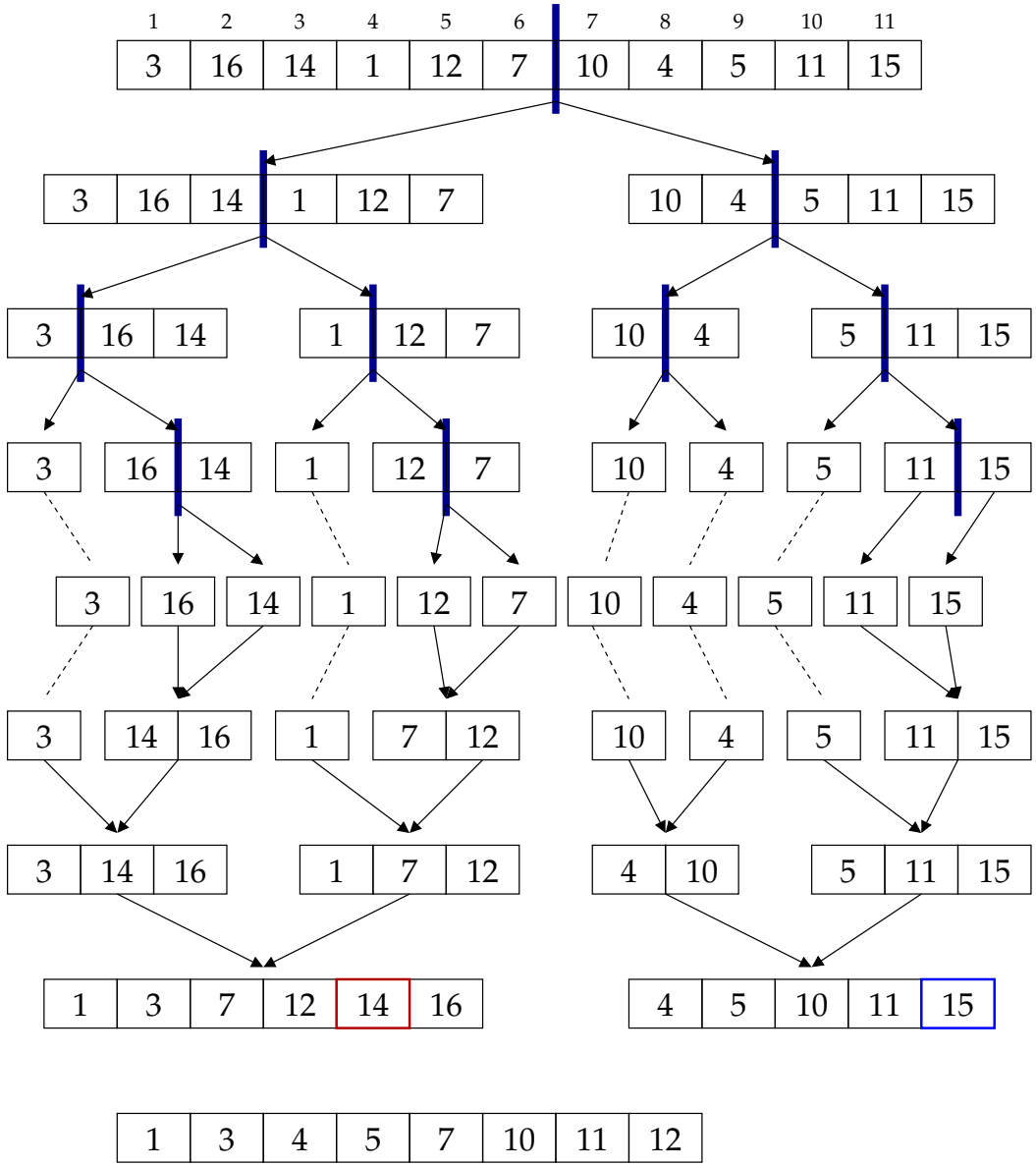
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1
7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10

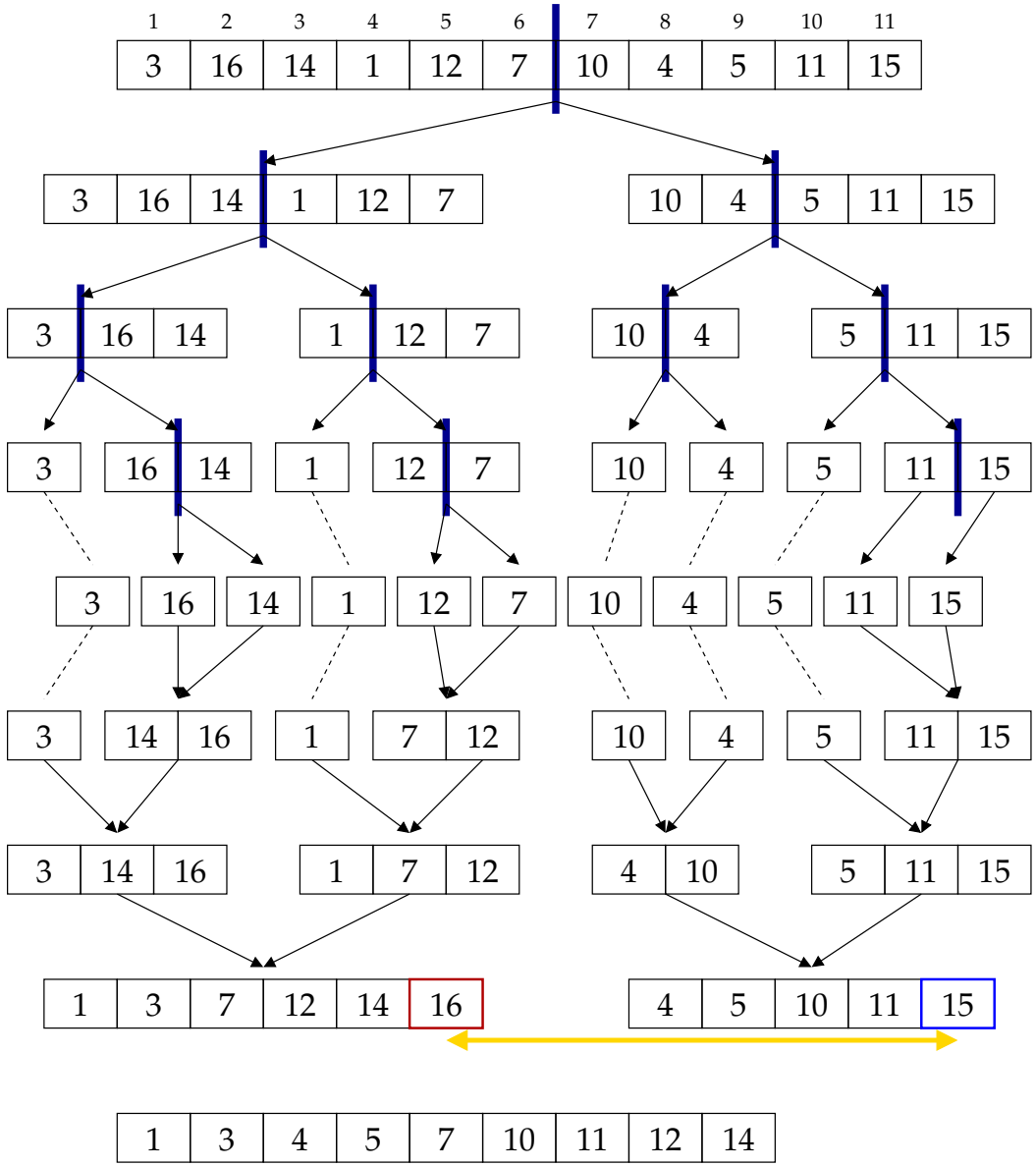
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11

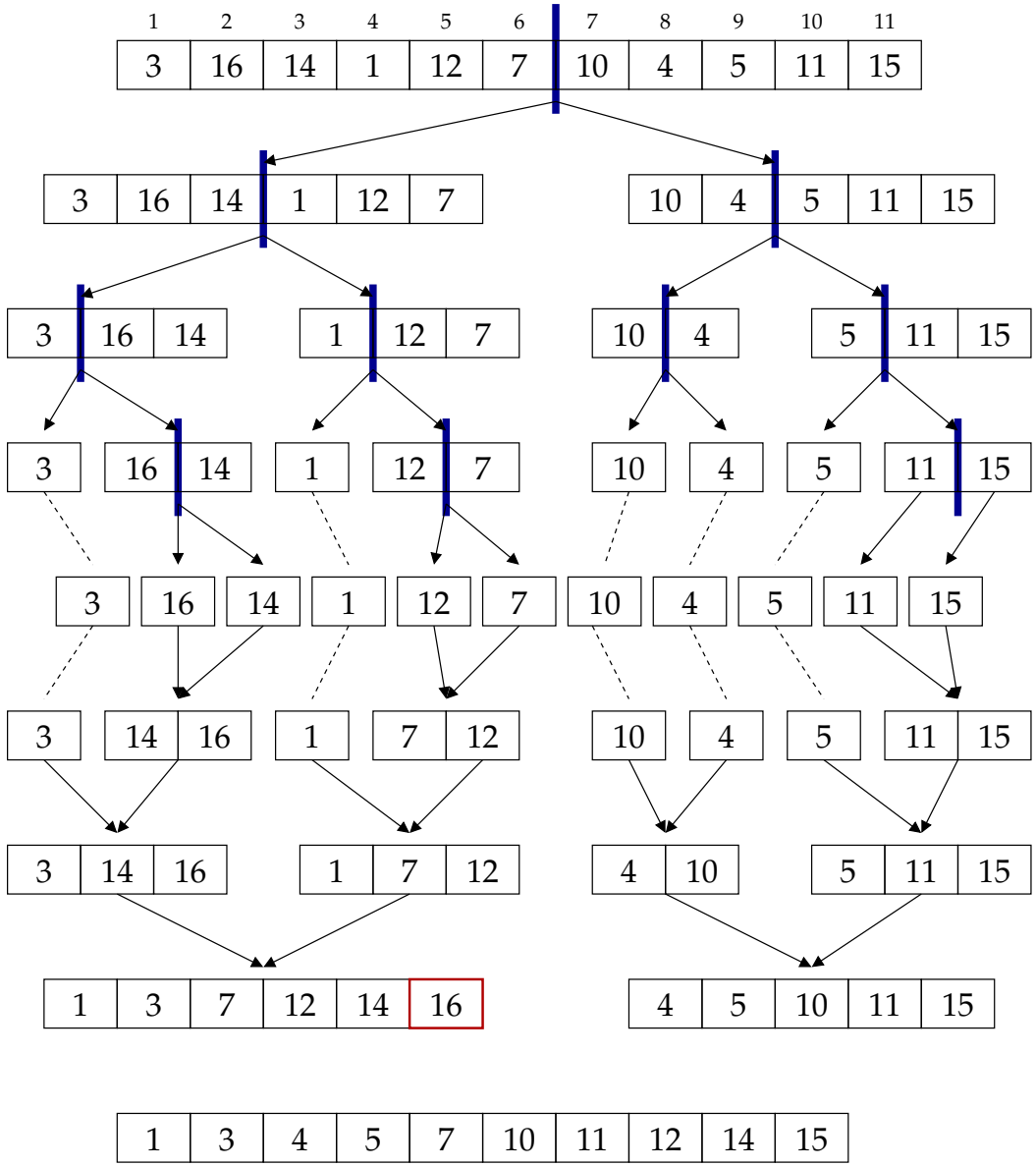
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11 12

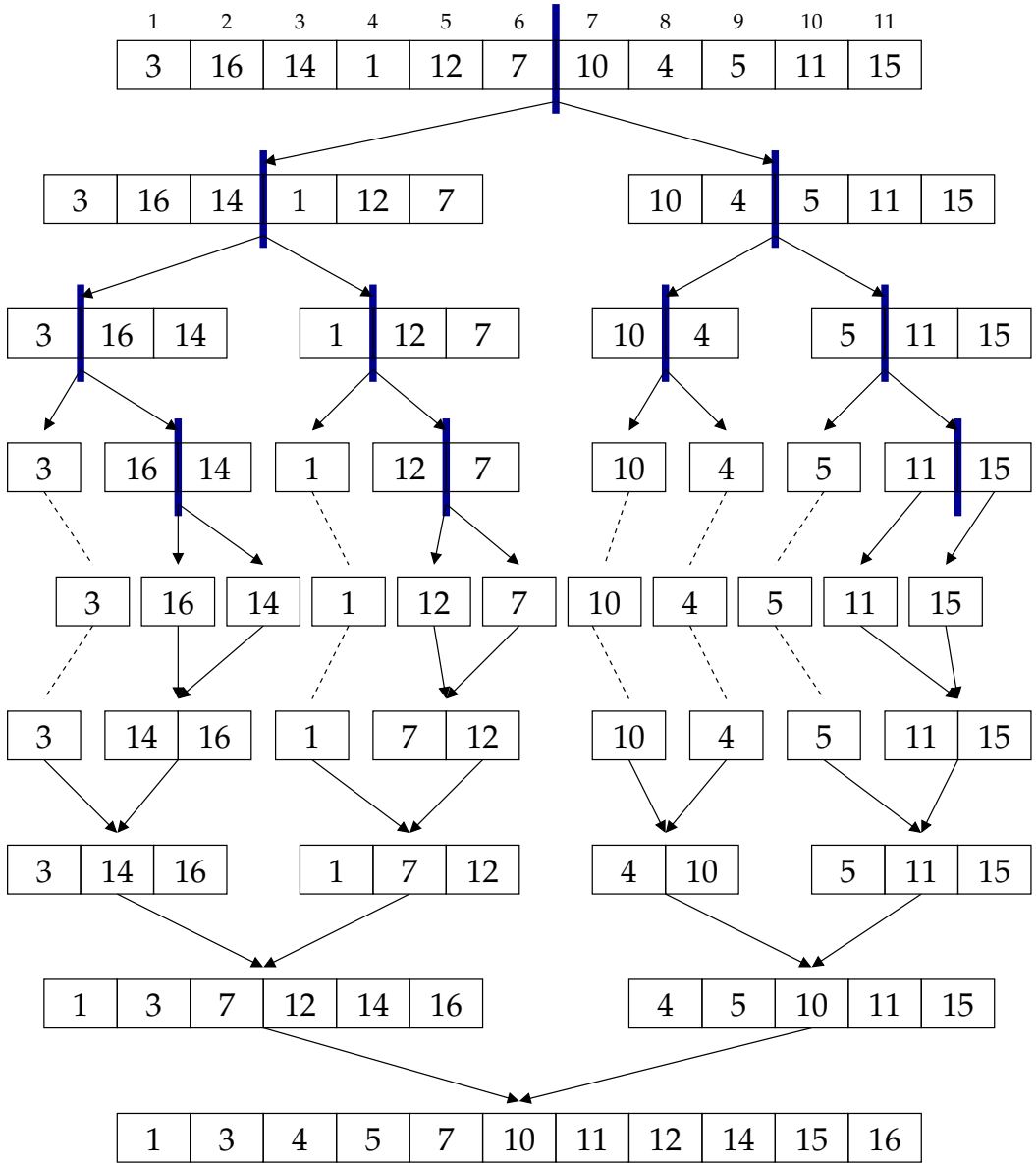
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11 12 14

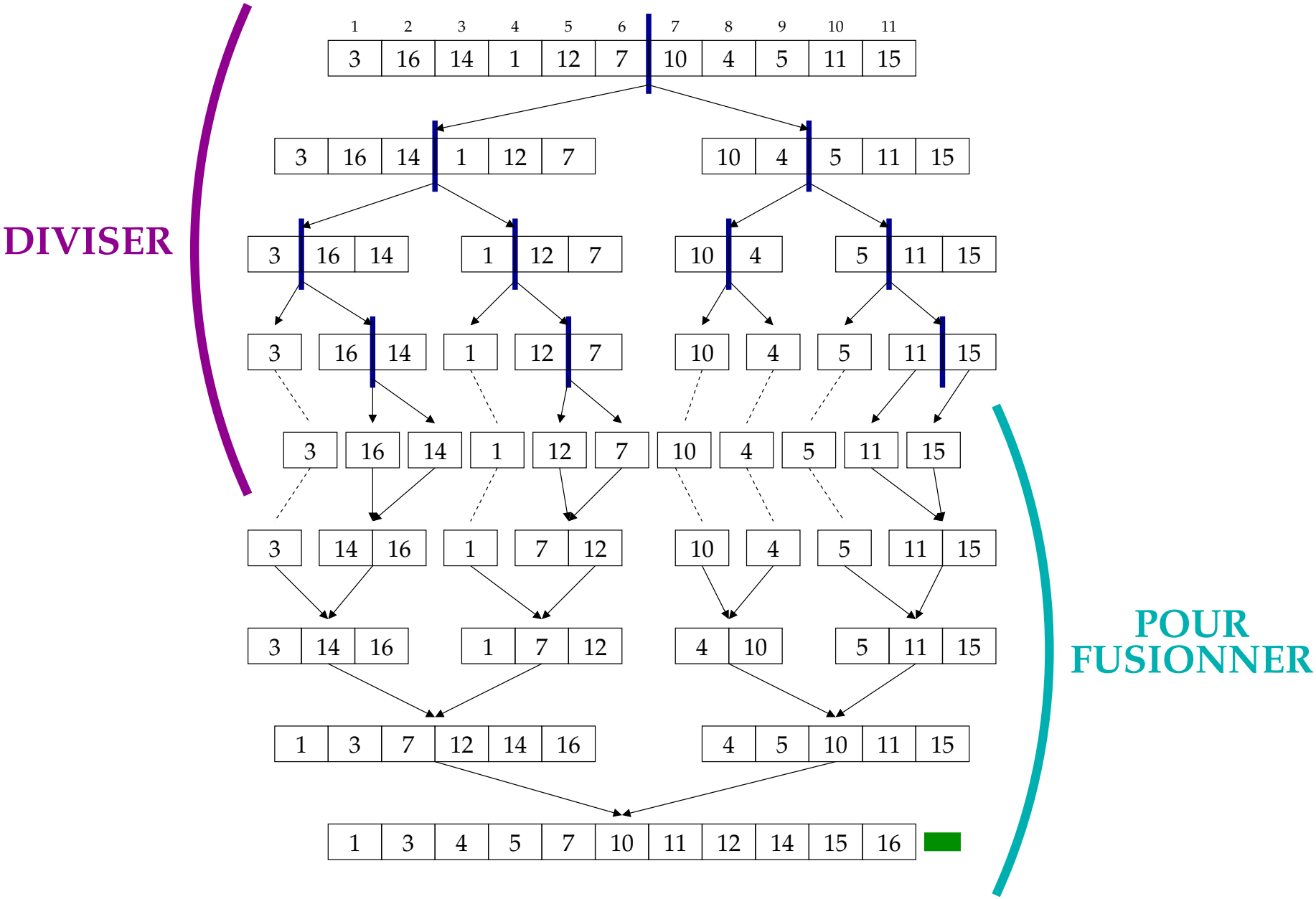
Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11 12 14 15

Tri fusion


1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1
7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11 12 14 15 16

Tri fusion


DIVISER
POUR FUSIONNER
1 2 3 4 5 6 7 8 9 10 11
3 16 14 1 12 7 10 4 5 11 15
3 16 14 1 12 7
10 4 5 11 15
3 16 14
1 12 7
10 4
5 11 15
3
16 14
1
12 7
10
4
5
11 15
3
16
14
1
12
7
10
4
5
11
15
3
14 16
1
7 12
10
4
5
11 15
3
14 16
1 7 12
4 10
5 11 15
1 3 7 12 14 16
4 5 10 11 15
1 3 4 5 7 10 11 12 14 15 16

Tri fusion

Algorithme

Fonction tri_fusion(**d** tab$[1 \dots n]$: **tab entier**) : **tab entier**
Début
    Si $(n \leqslant 1)$ **alors** **retour** tab ;
    Sinon
        retour fusion(tri_fusion(tab$[1 \dots \lfloor \frac{n+1}{2} \rfloor]$), tri_fusion(tab$[\lceil \frac{n+1}{2} \rceil \dots n]$)) ;
    Fin si
Fin

Fonction fusion(**d** $t_1[1 \dots n_1]$, **d** $t_2[1 \dots n_2]$: **tab entier**) : **tab entier**
Début
    Si $(n_1 = 0)$ **alors** **retour** t_2 ;
    Sinon si $(n_2 = 0)$ **alors** **retour** t_1 ;
    Sinon si $(t_1[1] \leqslant t_2[1])$ **alors** **retour** $t_1[1]$ :: fusion$(t_1[2 \dots n_1], t_2)$;
    Sinon **retour** $t_2[1]$ :: fusion$(t_1, t_2[2 \dots n_2])$;
    Fin si
Fin

Tri fusion

Complexité

Qu'en dites-vous ?

Tri fusion

Complexité

Qu'en dites-vous ?

$$O(n \cdot \log_2(n))$$

Tri fusion

Application

Vidéo

Plan du cours

Comment une machine fait-elle pour compter ?

$$i := i + 1$$

Comment une machine fait-elle pour compter ?

$i := i + 1$

$31 + 23$

Comment une machine fait-elle pour compter ?

$i := i + 1$

$n \equiv 4 \mod 27$

$31 + 23$

Comment une machine fait-elle pour compter ?

$i := i + 1$

$n \equiv 4 \mod 27$

$31 + 23$

$pgcd(21, 49)$

Comment une machine fait-elle pour compter ?

$i := i + 1$

$n \equiv 4 \mod 27$

$31 + 23$

$pgcd(21, 49)$

Comment une machine fait-elle pour compter ?

a et b premiers entre eux ?

$i := i + 1$

$n \equiv 4 \mod 27$

$31 + 23$

$pgcd(21, 49)$

Comment une machine fait-elle pour compter ?

a et b premiers entre eux ?

e^{19}

$i := i + 1$

$n \equiv 4 \mod 27$

$31 + 23$

$pgcd(21, 49)$

Comment une machine fait-elle pour compter ?

a et b premiers entre eux ?

$\ln(14)$

e^{19}

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

0

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

0	0

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

0	0	0

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

0	0	0	0

$$n := n + 1$$

n stocké sur un octet (tableau)

1	1	0	0	1	1	1	1

incrément (ajout de 1)

1	0	0	0	0

$$n := n + 1$$

n stocké sur un octet (tableau)	1	1	0	0	1	1	1	1
incrément (ajout de 1)	0	1	0	0	0	0		

$$n := n + 1$$

n stocké sur un octet (tableau)	1	1	0	0	1	1	1	1
incrément (ajout de 1)	1	0	1	0	0	0	0	

$$n := n + 1$$

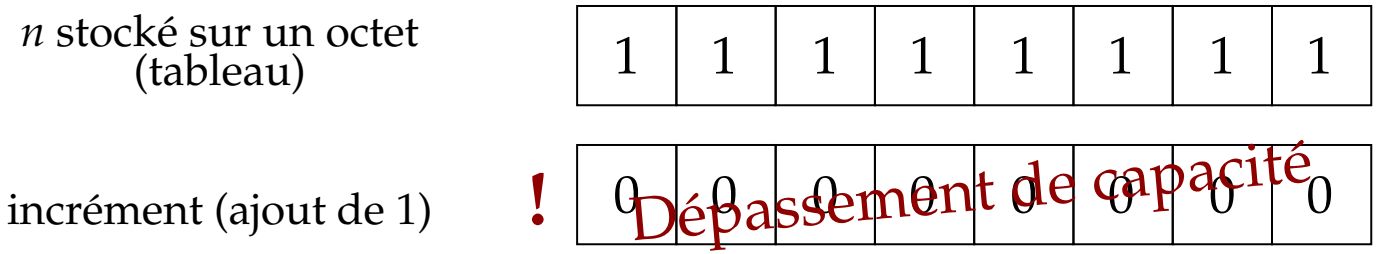
n stocké sur un octet (tableau)	1	1	0	0	1	1	1	1
incrément (ajout de 1)	1	1	0	1	0	0	0	0

$$n := n + 1$$

n stocké sur un octet (tableau)	1	1	0	0	1	1	1	1
incrément (ajout de 1)	1	1	0	1	0	0	0	0

Fonction incr(**d** $n[0..n-1]$: **tab_booléen**) : **tab_booléen**
    i : **entier** ;
Début
    $i := \text{longueur}(n) - 1$;
    Tant que $((i \geqslant 0)$ et $(n[i] = 1))$ **faire**
        $n[i] := 0$;
        $i := i - 1$;
    Fin faire
    $n[i] := 1$;
    retour n ;
Fin

$n := n + 1$

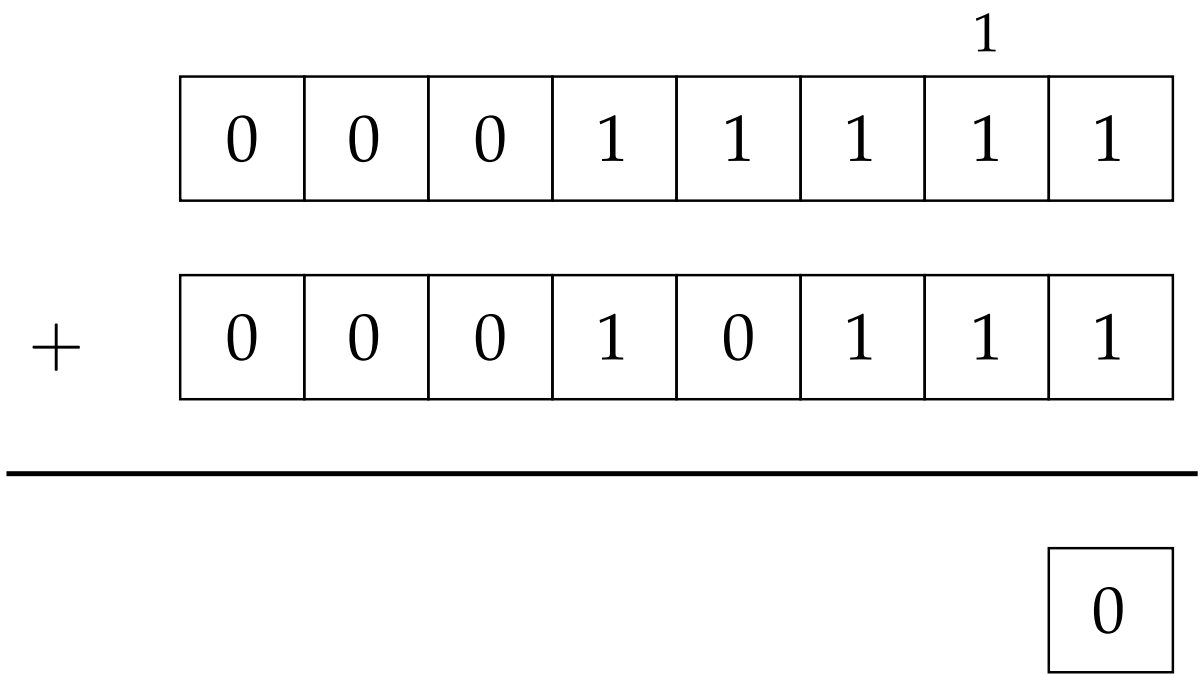



```
Fonction incr(d n[0..n-1] : tab_booléen) : tab_booléen
    i : entier ;
Début
    i := longueur(n) - 1 ;
    Tant que ((i ⩾ 0) et (n[i] = 1)) faire
        n[i] := 0 ;   i := i - 1 ;
    Fin faire
    Si (i = -1) alors
        écrire("Erreur capacité") ;   sortir ;
    Sinon
        n[i] := 1 ;
    Fin si
    retour n ;
Fin
```

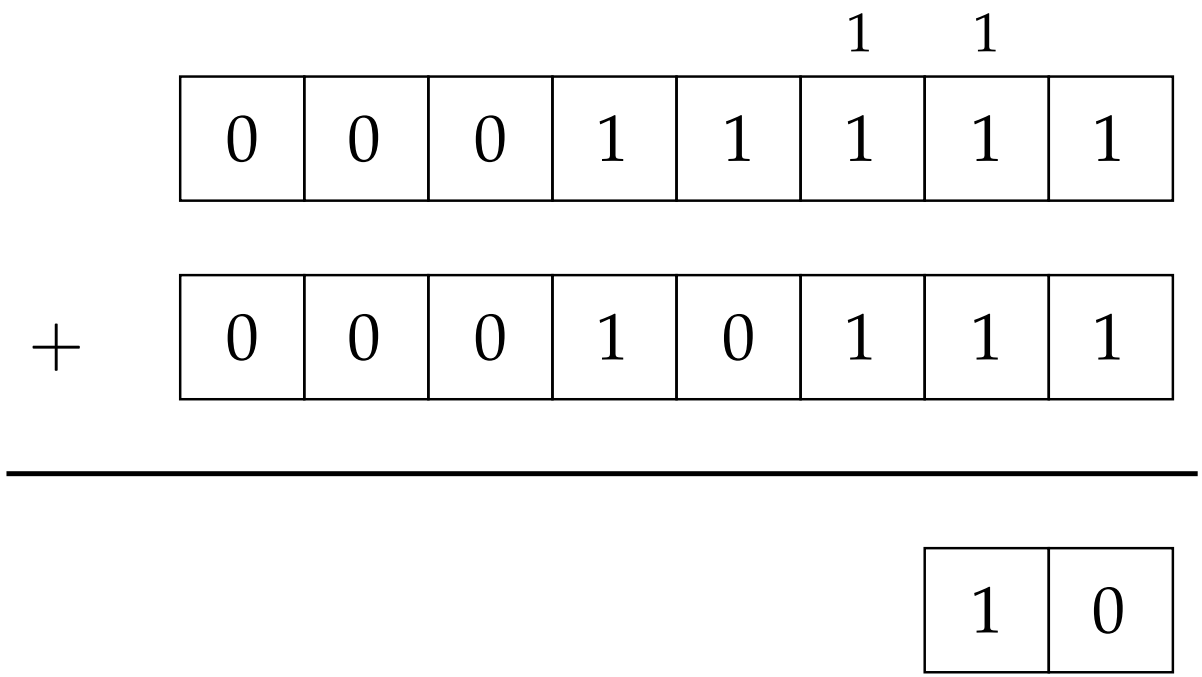
$31 + 23$

	0	0	0	1	1	1	1	1
+	0	0	0	1	0	1	1	1

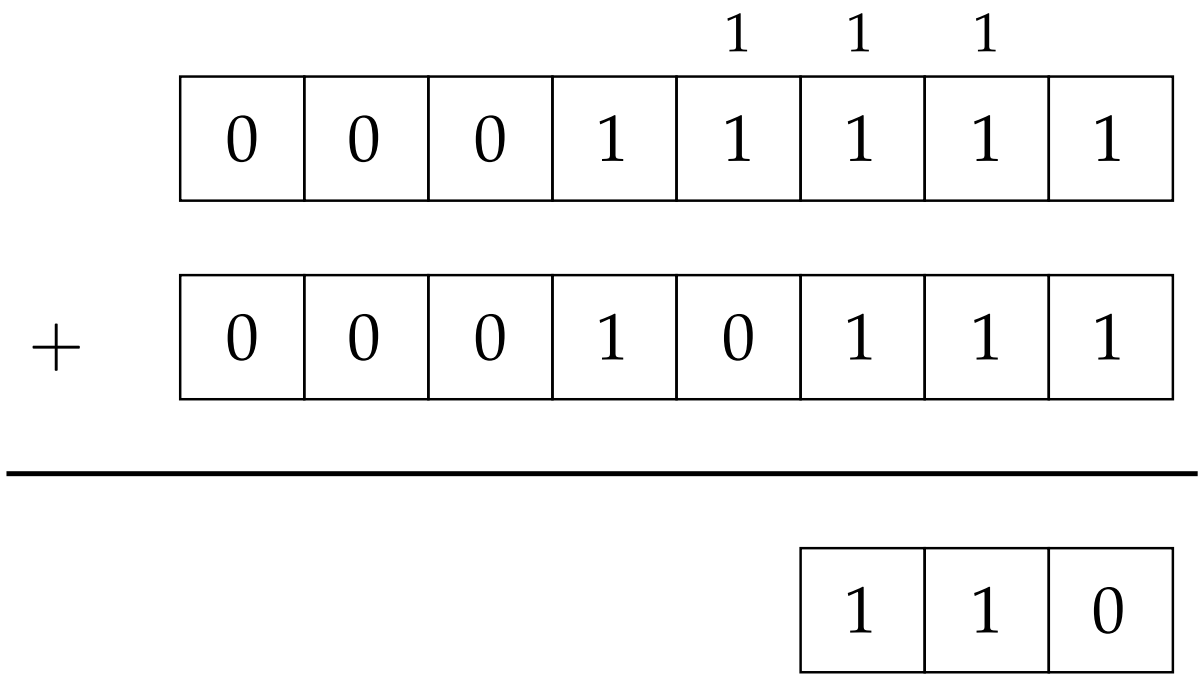
$31 + 23$



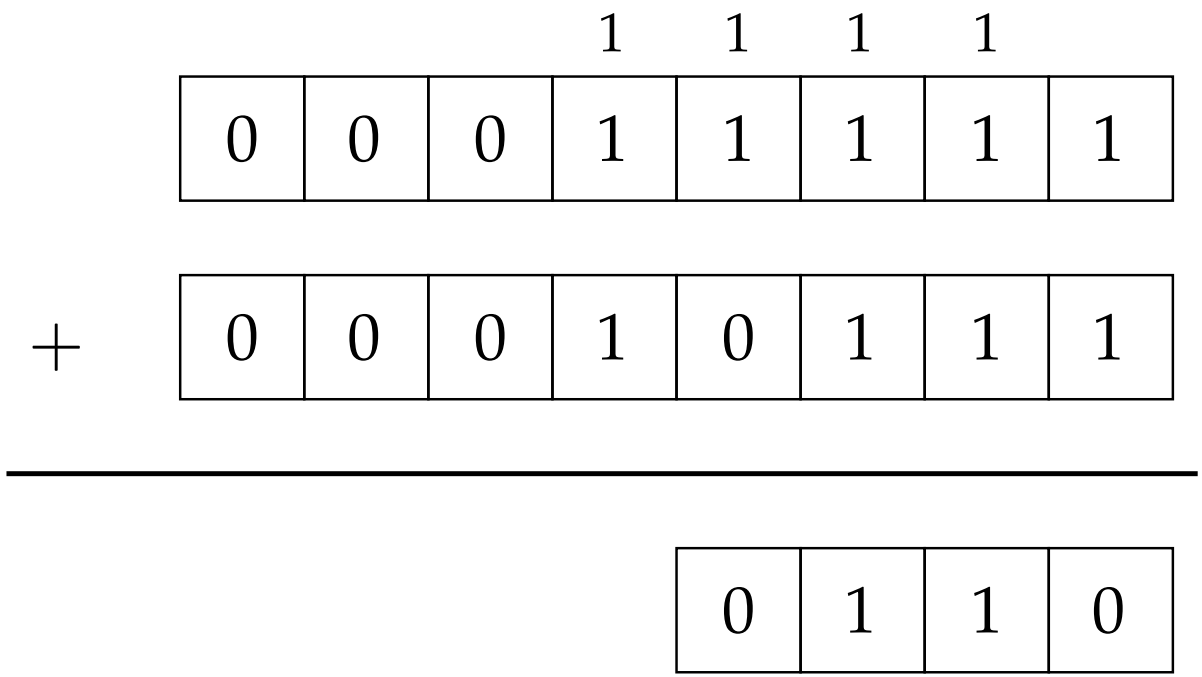
$31 + 23$



$31 + 23$



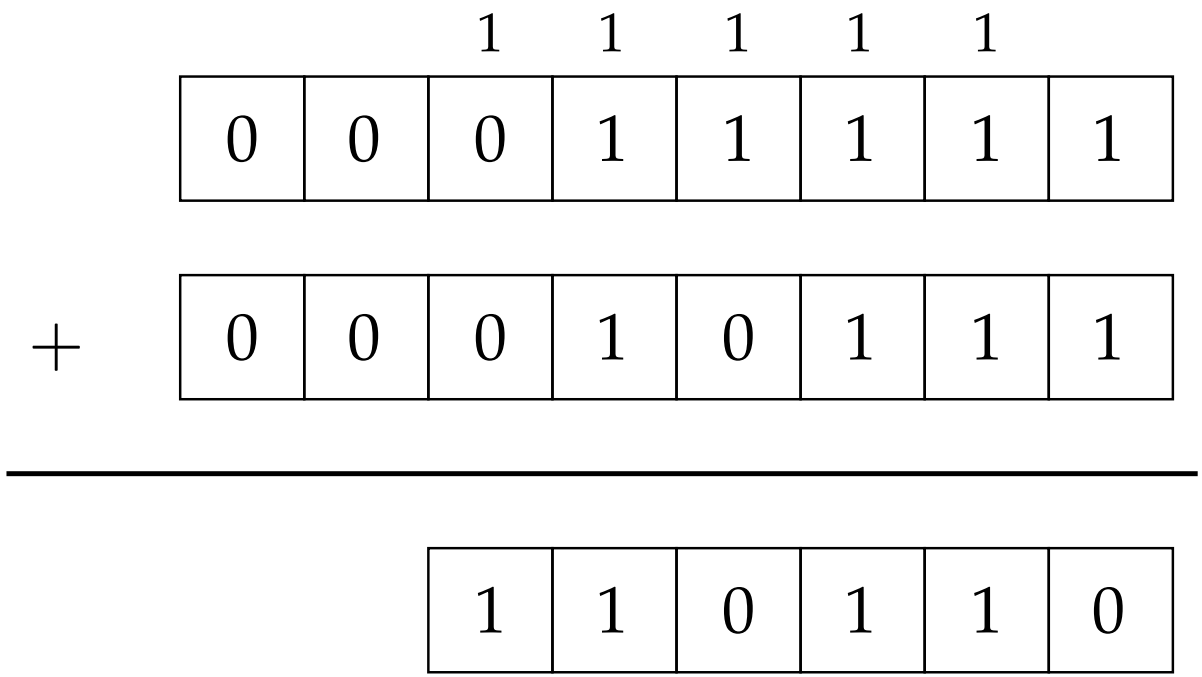
$31 + 23$



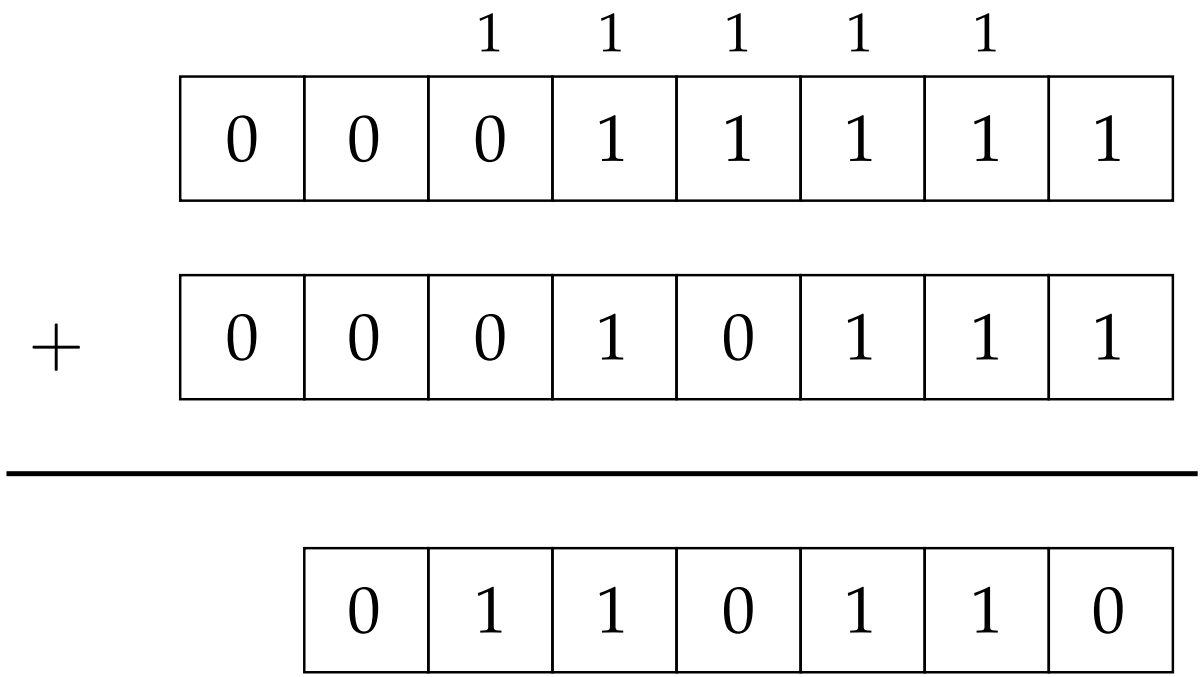
$31 + 23$

			1	1	1	1	1	
	0	0	0	1	1	1	1	1
+	0	0	0	1	0	1	1	1
			1	0	1	1	0	

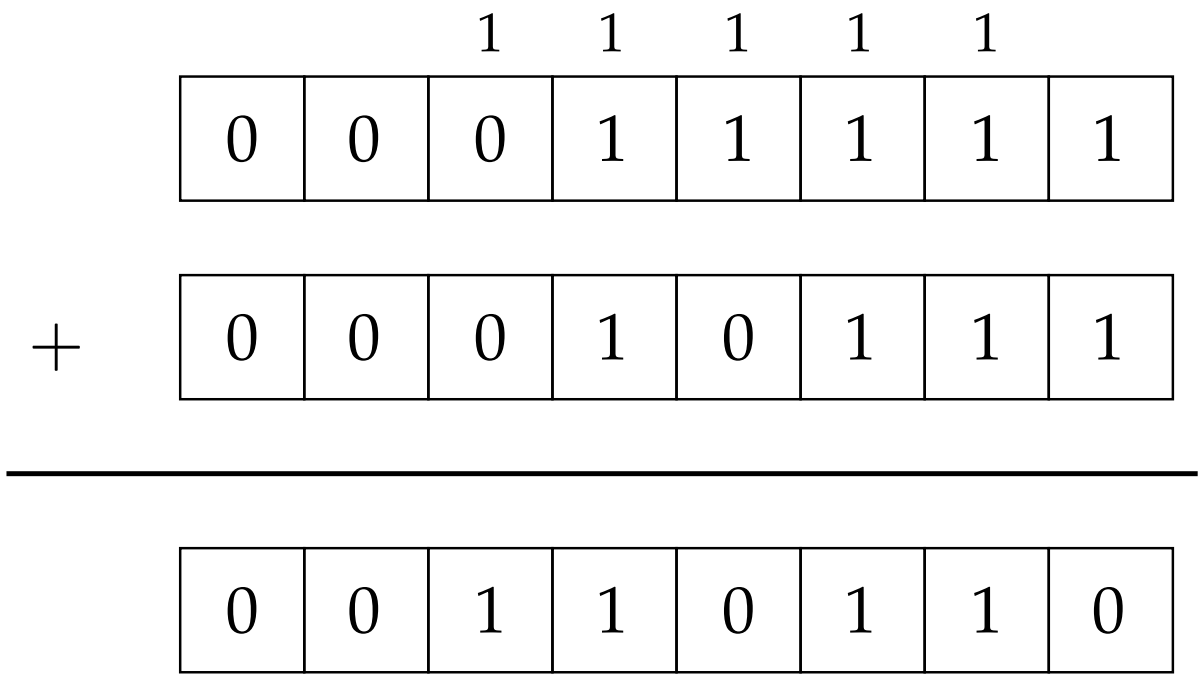
$31 + 23$



$31 + 23$



$31 + 23$

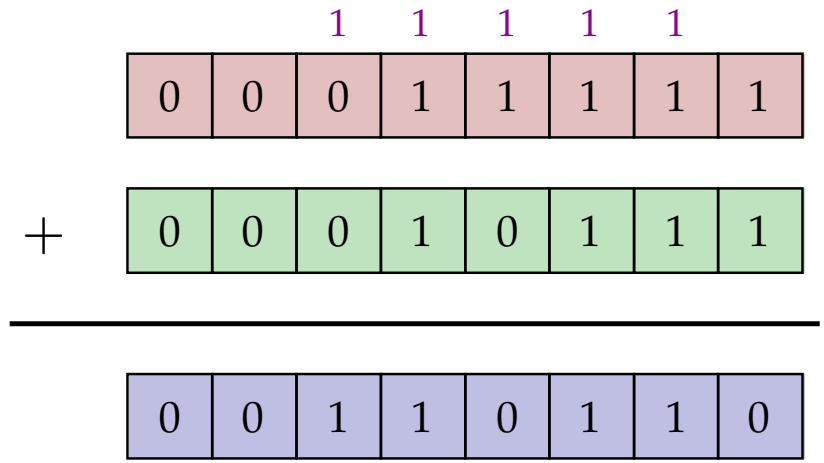


$31 + 23$

Fonction somme(**d** $m[0..n_1]$, $n[0..n-1]$: **tab_booléen**) : **tab_booléen**

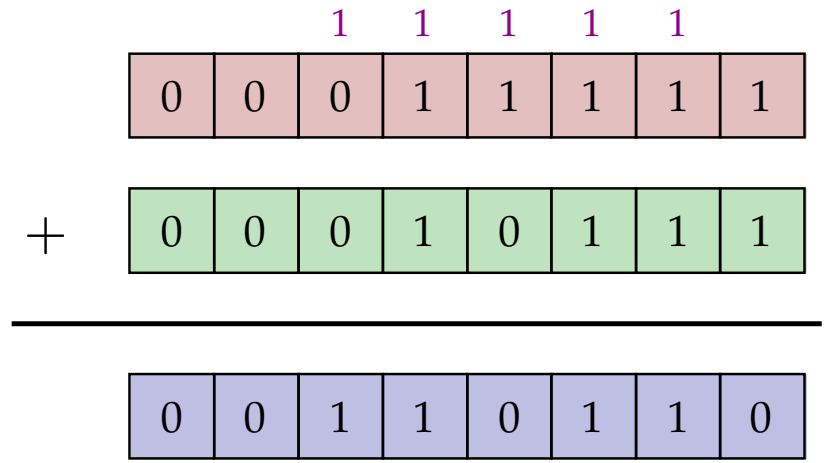
i, ℓ, r, tmp : **entier** ;

res$[0..n-1]$: **tab_booléen** ;



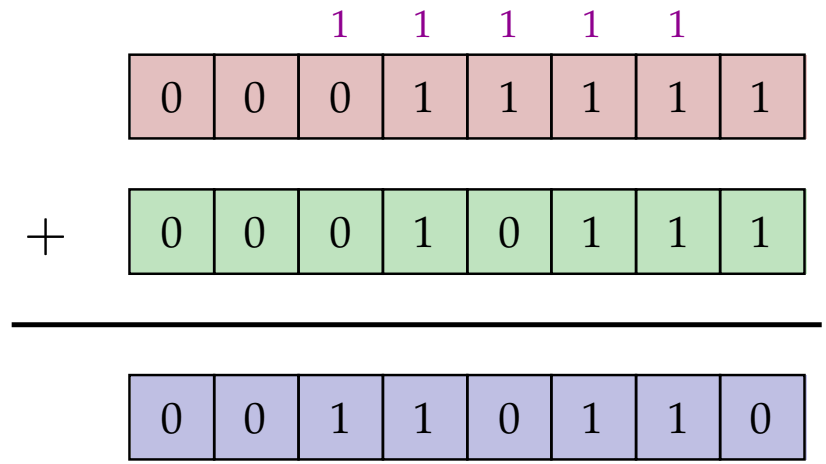
$31 + 23$

Fonction somme(**d** $m[0..n_1]$, $n[0..n-1]$: **tab_booléen**) : **tab_booléen**
    i, ℓ, r, tmp : **entier** ;
    $\text{res}[0..n-1]$: **tab_booléen** ;
Début
    $\ell := \text{longueur}(m)$;
    $r := 0$;



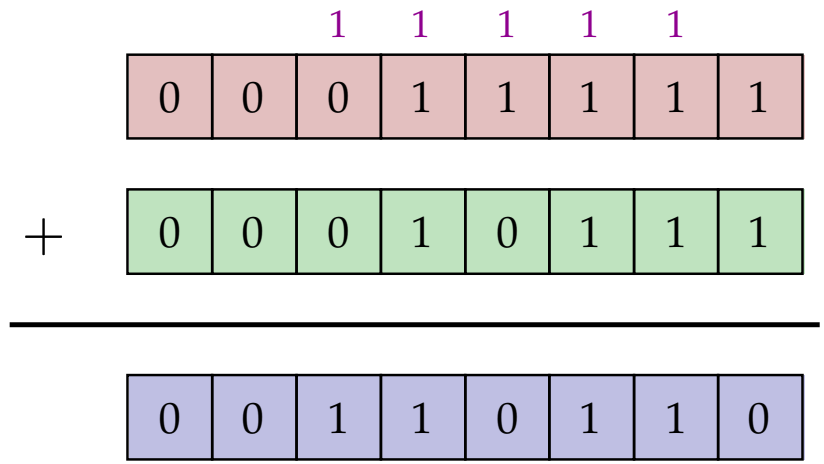
$31 + 23$

```
Fonction somme(d m[0..n_1], n[0..n − 1] : tab_booléen) : tab_booléen
    i, ℓ, r, tmp : entier ;
    res[0..n − 1] : tab_booléen ;
Début
    ℓ := longueur(m) ;
    r := 0 ;
    Pour (i de ℓ − 1 à 0) faire
        tmp = m[i] + n[i] + r ;
        res[i] := tmp  mod 2 ;
```



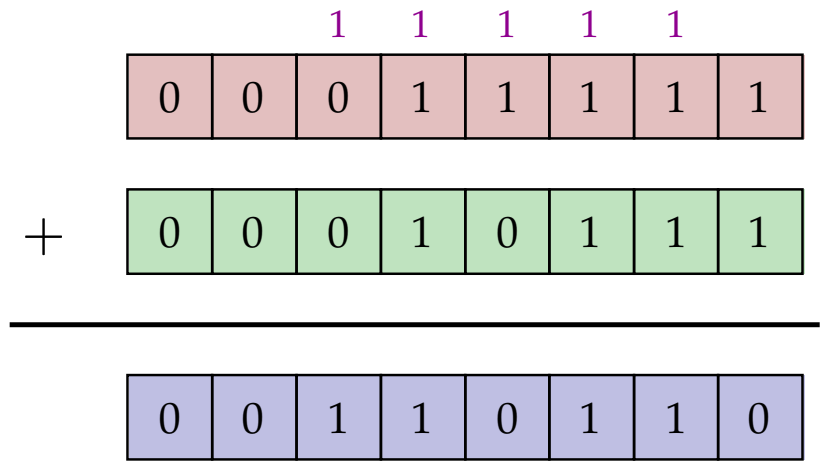
$31 + 23$

```
Fonction somme(d m[0..n_1], n[0..n − 1] : tab_booléen) : tab_booléen
    i, ℓ, r, tmp : entier ;
    res[0..n − 1] : tab_booléen ;
Début
    ℓ := longueur(m) ;
    r := 0 ;
    Pour (i de ℓ − 1 à 0) faire
        tmp = m[i] + n[i] + r ;
        res[i] := tmp  mod 2 ;
        Si (tmp ⩾ 2) alors
            r := 1 ;
        Sinon
            r := 0 ;
        Fin si
    Fin faire
```



$31 + 23$

```
Fonction somme(d m[0..n_1], n[0..n − 1] : tab_booléen) : tab_booléen
    i, ℓ, r, tmp : entier ;
    res[0..n − 1] : tab_booléen ;
Début
    ℓ := longueur(m) ;
    r := 0 ;
    Pour (i de ℓ − 1 à 0) faire
        tmp = m[i] + n[i] + r ;
        res[i] := tmp  mod 2 ;
        Si (tmp ⩾ 2) alors
            r := 1 ;
        Sinon
            r := 0 ;
        Fin si
    Fin faire
    retour res ;
Fin
```



$$n \equiv 4 \mod 27$$

Quotient et reste de la division euclidienne de n par m

$$n \equiv 4 \mod 27$$

Quotient et reste de la division euclidienne de n par m

$$n = m \times q + r, \qquad (q \in \mathbb{N}, r \in \mathbb{N}_{26})$$

$$n \equiv 4 \mod 27$$

Quotient et reste de la division euclidienne de n par m

$$n = m \times q + r, \qquad (q \in \mathbb{N}, r \in \mathbb{N}_{26})$$

Procédure division(**d** m, n : **entier** ; **r** q, r : **entier**)
Début
    $q := 0$;
    Tant que ($n \geqslant m$) **faire**
        $q := q + 1$;
        $n := n - m$;
    Fin faire
    $r := n$;
Fin

$$n \equiv 4 \mod 27$$

Quotient et reste de la division euclidienne de n par m

$$n = m \times q + r, \qquad (q \in \mathbb{N}, r \in \mathbb{N}_{26})$$

Procédure division(**d** m, n : **entier** ; **r** q, r : **entier**)
Début
    $q := 0$;
    Tant que $(n \geqslant m)$ **faire**
        $q := q + 1$;
        $n := n - m$;
    Fin faire
    $r := n$;
Fin

$\lfloor n/m \rfloor$ `n//m` en Python

$n \mod m$ `n%m` en Python

$n \equiv 4 \mod 27$

Quotient et reste de la division euclidienne de n par m

$$n = m \times q + r, \qquad (q \in \mathbb{N}, r \in \mathbb{N}_{26})$$

Procédure division(**d** m, n : **entier** ; **r** q, r : **entier**)
Début
    $q := 0$;
    Tant que $(n \geqslant m)$ **faire**
        $q := q + 1$;
        $n := n - m$;
    Fin faire
    $r := n$;
Fin

$\lfloor n/m \rfloor$ `n//m` en Python

$n \mod m$ `n%m` en Python

$$328 = 27 \times q + r$$

$383 - 27 = 355$ $193 - 27 = 166$
$355 - 27 = 328$ $166 - 27 = 139$
$228 - 27 = 301$ $139 - 27 = 112$
$301 - 27 = 274$ $112 - 27 = 85$
$274 - 27 = 247$ $85 - 27 = 58$
$247 - 27 = 220$ $58 - 27 = 31$
$220 - 27 = 193$ $31 - 27 = 4$

$$328 = 27 \times 14 + 4$$

a et b premiers entre eux ?

Définition

Soient $a, b \in \mathbb{N}^*$. a et b sont premiers entre eux si et seulement si "l'unique diviseur commun" (positif) à a et à b est 1.

Remarque

Deux nombres premiers sont premiers entre eux.

a et b premiers entre eux ?

Définition

Soient $a, b \in \mathbb{N}^*$. a et b sont premiers entre eux si et seulement si "l'unique diviseur commun" (positif) à a et à b est 1.

Remarque

Deux nombres premiers sont premiers entre eux.

- 14 et 15 sont premiers entre eux
- 14 et 49 ne sont pas premiers entre eux

a et b premiers entre eux ?

Définition

Soient $a, b \in \mathbb{N}^*$. a et b sont premiers entre eux si et seulement si "l'unique diviseur commun" (positif) à a et à b est 1.

Remarque

Deux nombres premiers sont premiers entre eux.

- 14 et 15 sont premiers entre eux
- 14 et 49 ne sont pas premiers entre eux

Définition

$\forall a, b \in \mathbb{N}^*$, a et b sont premiers entre eux $\iff \text{pgcd}(a, b) = 1$.

a et b premiers entre eux ?

Définition

Soient $a, b \in \mathbb{N}^*$. a et b sont premiers entre eux si et seulement si "l'unique diviseur commun" (positif) à a et à b est 1.

Remarque

Deux nombres premiers sont premiers entre eux.

- 14 et 15 sont premiers entre eux
- 14 et 49 ne sont pas premiers entre eux

Définition

$\forall a, b \in \mathbb{N}^*$, a et b sont premiers entre eux $\iff$ $\text{pgcd}(a, b) = 1$.

Comment calculer le PGCD de a et b ?

Algorithme d'Euclide

$\text{pgcd}(21, 49) = ?$

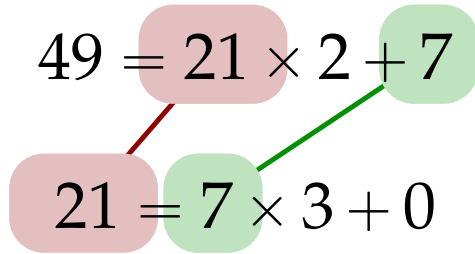
Algorithme d'Euclide

$\text{pgcd}(21, 49) = ?$

$49 = 21 \times 2 + 7$

Algorithme d'Euclide

$\text{pgcd}(21, 49) = ?$



Algorithme d'Euclide

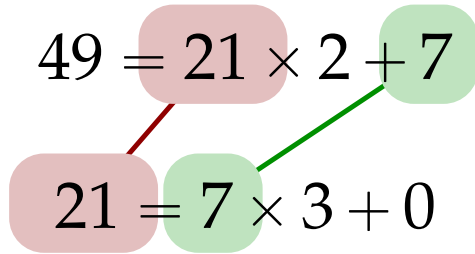
$$\text{pgcd}(21, 49) = 7$$

$$49 = 21 \times 2 + 7$$

$$21 = 7 \times 3 + 0$$

Algorithme d'Euclide

$$\text{pgcd}(21, 49) = 7$$



$$\text{pgcd}(345, 799) = ?$$

Algorithme d'Euclide

$$\text{pgcd}(21, 49) = 7$$

$$49 = 21 \times 2 + 7$$

$$21 = 7 \times 3 + 0$$

$$\text{pgcd}(345, 799) = ?$$

$$799 = 345 \times 2 + 109$$

Algorithme d'Euclide

$$\text{pgcd}(21, 49) = 7$$

$$49 = 21 \times 2 + 7$$

$$21 = 7 \times 3 + 0$$

$$\text{pgcd}(345, 799) = ?$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

Algorithme d'Euclide

$\text{pgcd}(21, 49) = 7$

$$49 = 21 \times 2 + 7$$
$$21 = 7 \times 3 + 0$$

$\text{pgcd}(345, 799) = ?$

$$799 = 345 \times 2 + 109$$
$$345 = 109 \times 3 + 18$$
$$109 = 18 \times 6 + 1$$

Algorithme d'Euclide

$\text{pgcd}(21, 49) = 7$

$$49 = 21 \times 2 + 7$$
$$21 = 7 \times 3 + 0$$

$\text{pgcd}(345, 799) = ?$

$$799 = 345 \times 2 + 109$$
$$345 = 109 \times 3 + 18$$
$$109 = 18 \times 6 + 1$$
$$18 = 1 \times 18 + 0$$

Algorithme d'Euclide

$$\text{pgcd}(21, 49) = 7$$

$$49 = 21 \times 2 + 7$$

$$21 = 7 \times 3 + 0$$

$$\text{pgcd}(345, 799) = 1$$

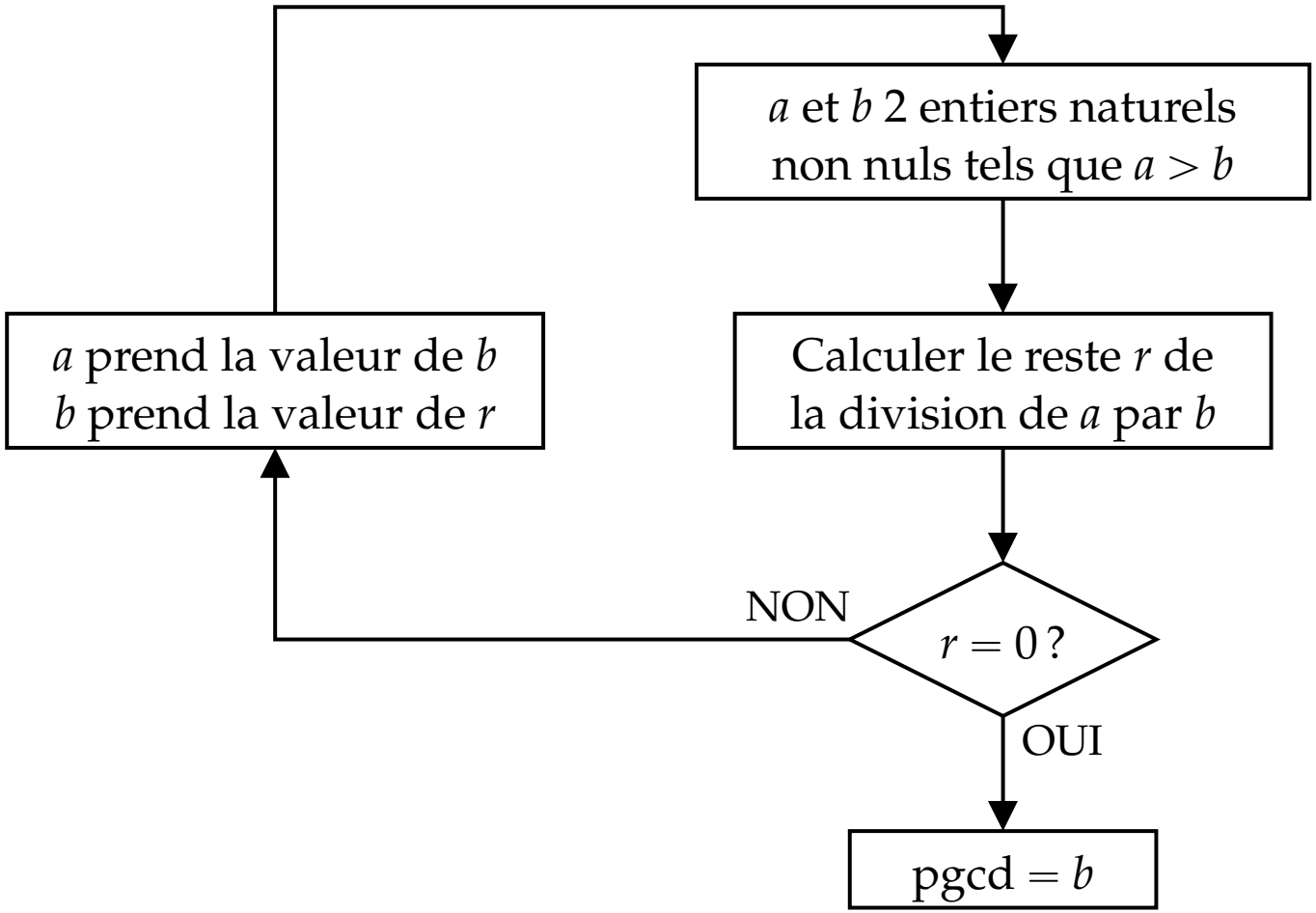
$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

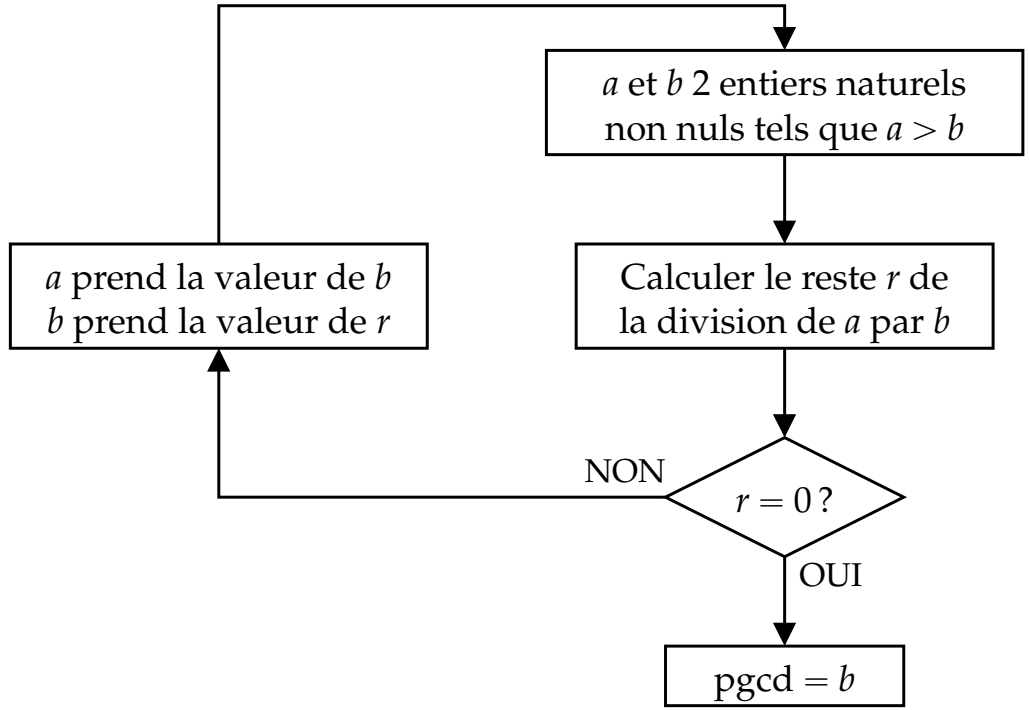
$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

Algorithme d'Euclide



Algorithme d'Euclide



Fonction pgcd (**d** a, b : **entier**) : **entier**
    # $a, b \in \mathbb{N}^*$, $a > b$
    r : **entier** ;
Début
    $r := a \mod b$;
    Tant que $(r > 0)$ **faire**
        $a := b$;
        $b := r$;
        $r := a \mod b$;
    Fin faire
    retour b ;
Fin

OK, mais est-ce utile ?

Nombres pseudo-aléatoires

Générateurs congruentiels linéaires

$$x_{n+1} = (a \cdot x_n + c) \mod m$$

(Lehmer, 1948)

$$x_n = (x_{n-j} + x_{n-k} + c) \mod m()$$

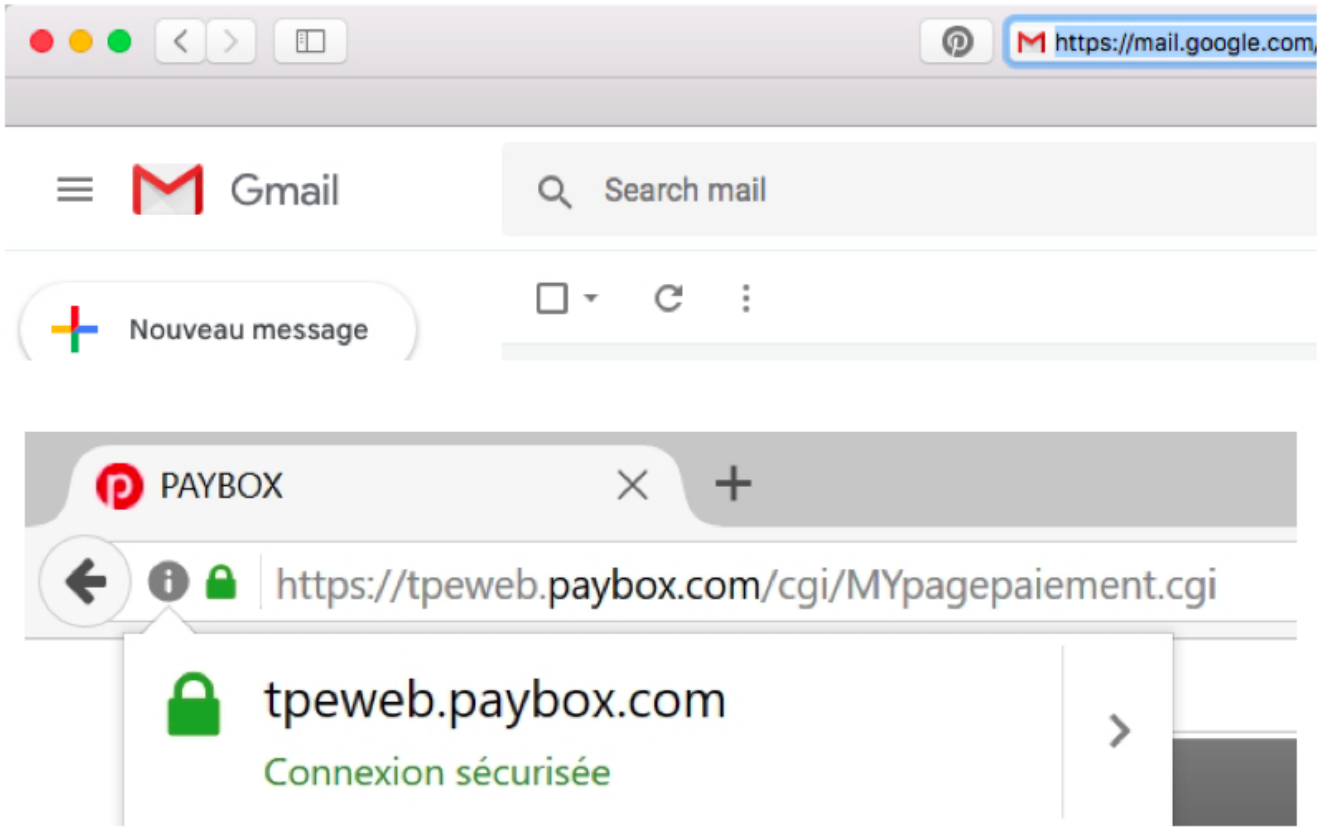
(Mitchell & Moore, 1858)

$$x_n = (x_{n-24} + x_{n-55} + c) \mod m$$

OK, mais est-ce utile ?

Communication CB ⇜⇝ TPE

OK, mais est-ce utile ?



Communication en ligne sécurisée

OK, mais est-ce utile ?

Charlie

Alice

Communication sécurisée

Alice veut envoyer un secret à Bob

Bob

OK, mais est-ce utile ?

Charlie

Communication sécurisée

Alice veut envoyer un secret à Bob

Alice

Bob

mot de passe
=
“maman1967”

chiffrement →

message chiffré
=
“X# !3y7b ?j”

déchiffrement →

message déchiffré
=
“maman1967”

Chiffrement / déchiffrement

mot de passe
=
“maman1967”

message chiffré
=
“X# !3y7b ?j”

Chiffrement / déchiffrement

mot de passe
=
“maman1967”

chiffrement →

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de César – Décalage de la la clé k dans la table ASCII

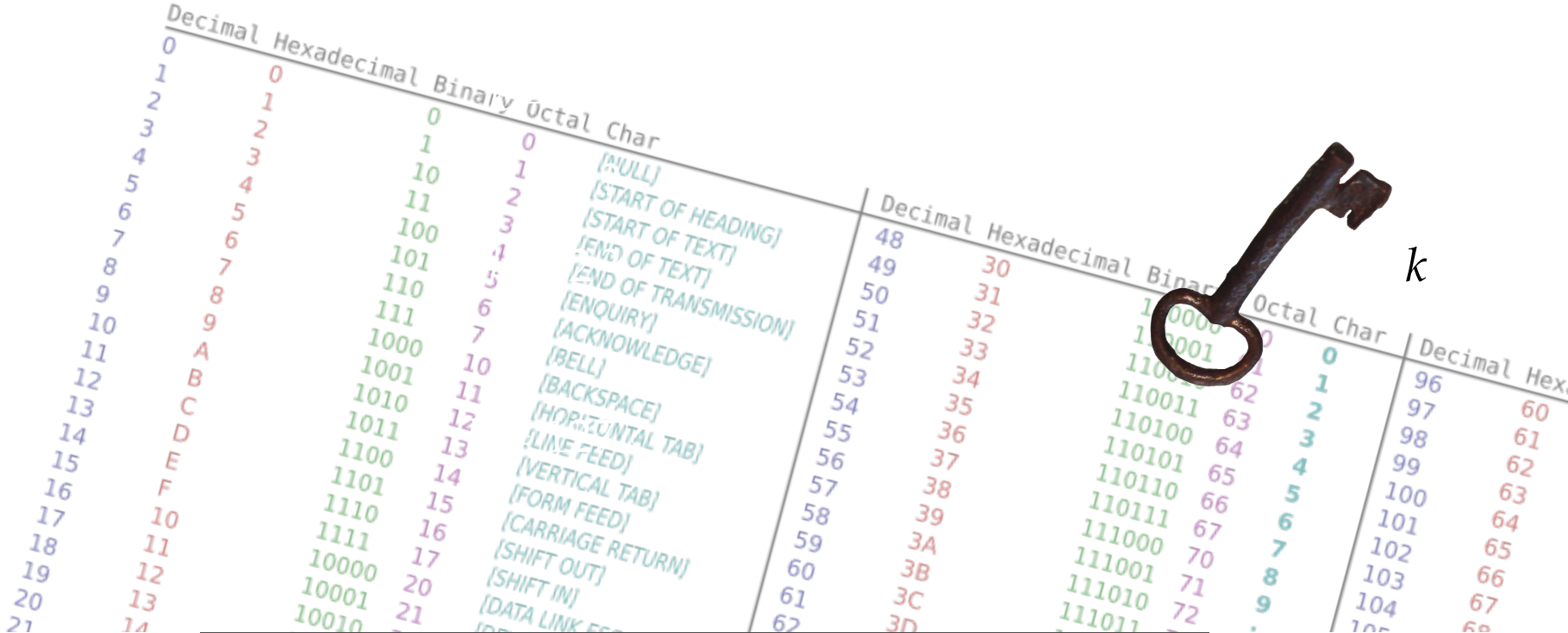
Chiffrement / déchiffrement

mot de passe
=
“maman1967”

chiffrement →

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de César – Décalage de la la clé k dans la table ASCII



Chiffrement / déchiffrement

mot de passe
=
“maman1967”

chiffrement →

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de César – Décalage de la la clé k dans la table ASCII

Charlie

$k = 0$?

$k = 1$?

$k = 2$?

...

$k = 127$?

k

Chiffrement / déchiffrement

mot de passe
=
“maman1967”

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de Vigenère – Un mot périodique dont chaque symbole encode un décalage spécifique

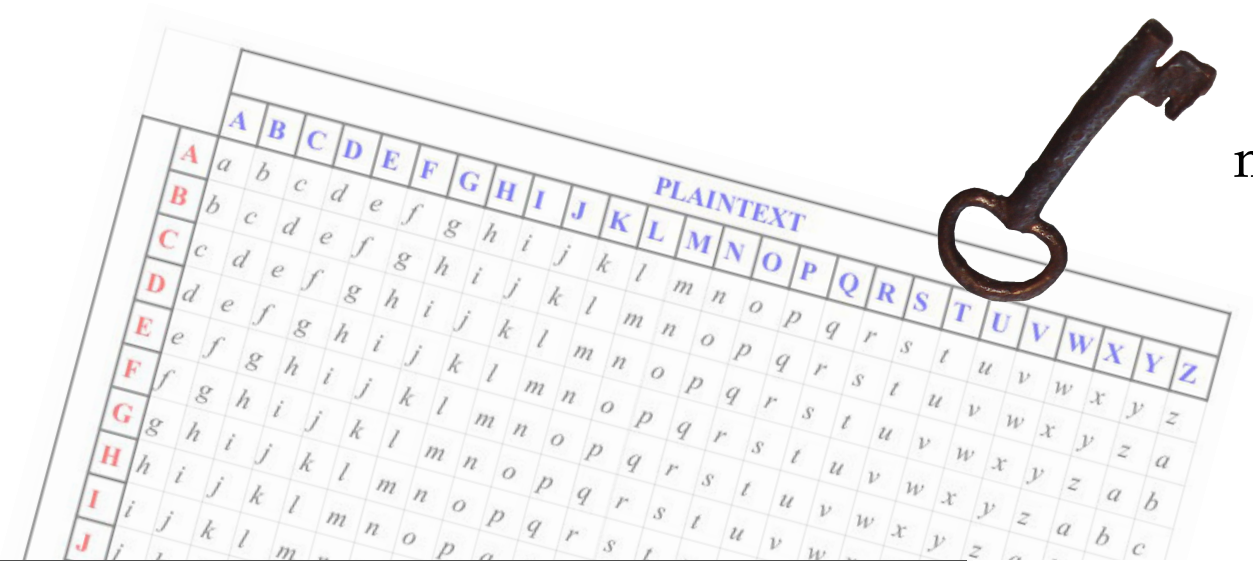
Chiffrement / déchiffrement

mot de passe
=
“maman1967”

chiffrement →

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de Vigenère – Un mot périodique dont chaque symbole encode un décalage spécifique



mot

Chiffrement / déchiffrement

mot de passe
=
“maman1967”

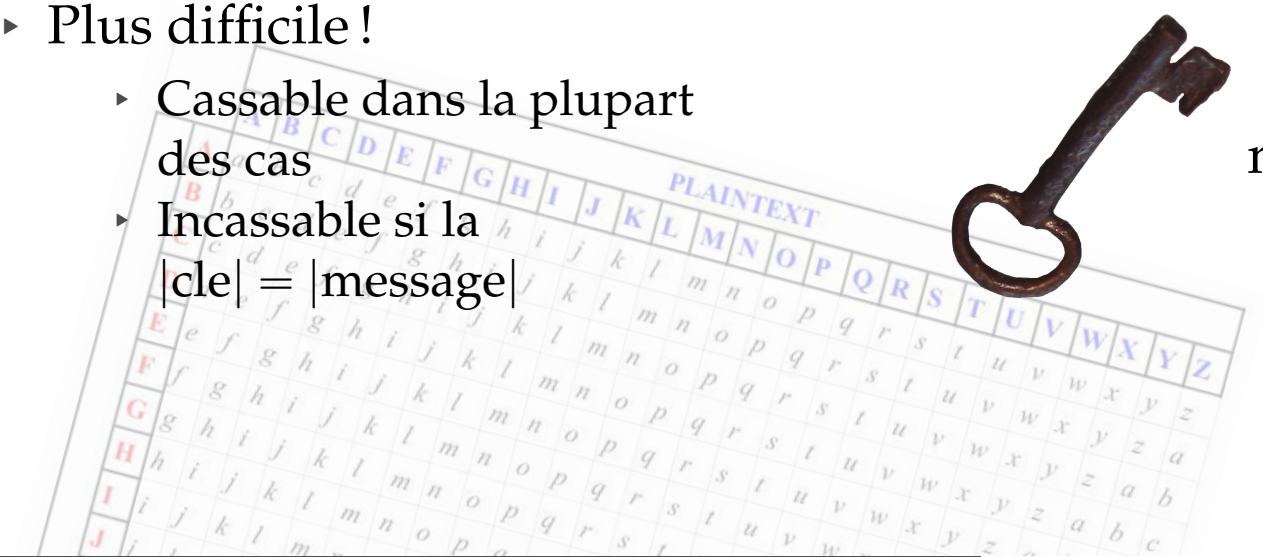
chiffrement →

message chiffré
=
“X# !3y7b ?j”

- Chiffrement de Vigenère – Un mot périodique dont chaque symbole encode un décalage spécifique

Charlie

- Plus difficile !
  - Cassable dans la plupart des cas
  - Incassable si la |cle| = |message|



mot

Chiffrement / déchiffrement

- Cryptosystème RSA – Chiffrement asymétrique, fondé sur de grands nombres premiers et le problème de la décomposition en produit de facteurs premiers

Chiffrement / déchiffrement

- Cryptosystème RSA – Chiffrement asymétrique, fondé sur de grands nombres premiers et le problème de la décomposition en produit de facteurs premiers

Alice

(e,n)

(d,n)

Bob

Chiffrement / déchiffrement

- Cryptosystème RSA – Chiffrement asymétrique, fondé sur de grands nombres premiers et le problème de la décomposition en produit de facteurs premiers

Alice

(e,n)

(d,n)

Bob

$$C \equiv M^e \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

$$M \equiv C^d \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

Choix des clés RSA

Choix des clés RSA

- Choisir $p \neq q$ deux grands nombres premiers

Choix des clés RSA

- Choisir $p \neq q$ deux grands nombres premiers
- Calculer $n = p \cdot q$

Choix des clés RSA

- Choisir $p \neq q$ deux grands nombres premiers
- Calculer $n = p \cdot q$
- Calculer $\phi(n) = (p-1)(q-1)$

Choix des clés RSA

- Choisir $p \neq q$ deux grands nombres premiers
- Calculer $n = p \cdot q$
- Calculer $\phi(n) = (p-1)(q-1)$
- Choisir un entier e premier avec $\phi(n)$ $\rightsquigarrow$ clé d'Alice

Choix des clés RSA

- Choisir $p \neq q$ deux grands nombres premiers
- Calculer $n = p \cdot q$
- Calculer $\phi(n) = (p-1)(q-1)$
- Choisir un entier e premier avec $\phi(n) \rightsquigarrow$ clé d'Alice
- Calculer l'entier d inverse de $e \mod \phi(n) \rightsquigarrow$ clé de Bob

$$d \cdot e \equiv 1 \mod \phi(n)$$

grâce la relation de Bézout

$$d \cdot e + k \cdot \phi(n) = 1$$

Correction de RSA

Alice

(e,n)

(d,n)

Bob

$$C \equiv M^e \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

$$M \equiv C^d \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

Correction de RSA

Alice

(e,n)

(d,n)

Bob

$$C \equiv M^e \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

$$M \equiv C^d \mod n, \qquad \text{avec } C, M \in \mathbb{N}$$

Théorème

$$\left(M^e\right)^d \equiv M \mod n$$

Force de RSA

- À partir de (e, n), difficile de calculer (d, n)
- Raison : à partir de n $(= p \cdot q)$, difficile de calculer $\phi(n) = (p-1)(q-1)$ si p et q sont des nombres premiers suffisamment grands

La décomposion d'un entier en produit de facteurs premiers est un problème difficile

Comment faire ?

- Déchiffrement
  - Calcul de l'entier $d \cdot e \equiv 1 \mod \phi(n)$
    - Prise en compte de la relation de Bézout $d \cdot e + k \cdot \phi(n) = 1$
    - Utilisation de l'algorithme d'Euclide étendu
- Chiffrement
  - Utilisation de l'algorithme d'exponentiation modulaire pour résoudre $C \equiv M^e \mod n$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$109 - 18 \times 6 = 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$109 - 18 \times 6 = 1$$

$$109 - (345 - 109 \times 3) \times 6 = 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$109 \times 19 - 345 \times 6 = 1$$

$$109 - 18 \times 6 = 1$$

$$109 - (345 - 109 \times 3) \times 6 = 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$109 \times 19 - 345 \times 6 = 1$$

$$109 - 18 \times 6 = 1$$

$$(799 - 345 \times 2) \times 19 - 345 \times 6 = 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109 \qquad 799 \times 19 - 345 \times 44 = 1$$

$$345 = 109 \times 3 + 18 \qquad 109 \times 19 - 345 \times 6 = 1$$

$$109 = 18 \times 6 + 1 \qquad 109 - 18 \times 6 = 1$$

$$(799 - 345 \times 2) \times 19 - 345 \times 6 = 1$$

Algorithme d'Euclide étendu

$$\text{pgcd}(345, 799) = 1$$

$$799 = 345 \times 2 + 109 \qquad 799 \times 19 - 345 \times 44 = 1$$

$$345 = 109 \times 3 + 18 \qquad 109 \times 19 - 345 \times 6 = 1$$

$$109 = 18 \times 6 + 1 \qquad 109 - 18 \times 6 = 1$$

Relation de Bézout de $(799, 345)$

$$799 \times 19 - 345 \times 44 = 1$$

e^{19} et $M^d \mod n$

- Calcul de la puissance $n$ème d'un nombre x

$$x \qquad x^2 \qquad x^3 \qquad \ldots \qquad x^n$$

e^{19} et $M^d \mod n$

- Calcul de la puissance $n$ème d'un nombre x

$$x \qquad x^2 \qquad x^3 \qquad \ldots \qquad x^n$$

```
Fonction puissance(d x, n : entier) : entier
    res : entier ;
Début
    res := x ;
    Pour i de 2 à n faire
        res := res × x ;
    Fin faire
    retour res ;
Fin
```

e^{19} et $M^d \mod n$

- Calcul de la puissance $n$ème d'un nombre x

$$x \qquad x^2 \qquad x^3 \qquad \ldots \qquad x^n$$

```
Fonction puissance(d x, n : entier) : entier
    res : entier ;
Début
    res := x ;
    Pour i de 2 à n faire
        res := res × x ;
    Fin faire
    retour res ;
Fin
```

$n-1$ multiplications

Exponentiation rapide

$$x^8 = x \times x \times x \times x \times x \times x \times x \times x$$

7 multiplications

$$x^8 = (x \times x \times x \times x) = \Big((x \times x)^2\Big)^2 = \Big(\big(x^2\big)^2\Big)^2$$

3 multiplications

Exponentiation rapide

$$x^{13} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

12 multiplications

$$\begin{aligned} x^{13} &= (x \times x \times x \times x \times x \times x)^2 \times x \\ &= \left((x \times x \times x)^2\right)^2 \times x \end{aligned}$$

5 multiplications

Exponentiation rapide

```
Fonction fastexp(d x, n : entier) : entier
    a, b, m : entier ;
Début
    a := 1 ;
    b := x ;
    m := n ;
Tant que (m > 0) faire
        Si (m ≡ 1 mod 2) alors
            a := a × b ;
        Fin si
        b := b × b ;
        m := ⌊m/2⌋ ;
    Fin faire
    retour a ;
Fin
```

	a	b	m
initialisation	1	x	13
après 1 tour	x	x^2	6
après 2 tours	x	x^4	3
après 3 tours	x^5	x^8	1
après 4 tours	x^{13}	x^{16}	0

Exponentiation rapide

```
Fonction fastexp(d x, n : entier) : entier
    a, b, m : entier ;
Début
    a := 1 ;
    b := x ;
    m := n ;
Tant que (m > 0) faire
        Si (m ≡ 1 mod 2) alors
            a := a × b ;
        Fin si
        b := b × b ;
        m := ⌊m/2⌋ ;
    Fin faire
    retour a ;
Fin
```

	a	b	m
initialisation	1	x	13
après 1 tour	x	x^2	6
après 2 tours	x	x^4	3
après 3 tours	x^5	x^8	1
après 4 tours	x^{13}	x^{16}	0

$O(\log_2(n))$ multiplications

$\ln(14)$ $\log_2(152)$

Table des logarithmes népériens entre 0,01 et 1

N	ln(N)	N	ln(N)	N	ln(N)	N	ln(N)	N	ln(N)
0,01	–4,605 17	0,21	–1,560 65	0,41	–0,891 6	0,61	–0,494 3	0,81	–0,210 72
0,02	–3,912 02	0,22	–1,514 13	0,42	–0,867 5	0,62	–0,478 04	0,82	–0,198 45
0,03	–3,506 56	0,23	–1,469 68	0,43	–0,843 97	0,63	–0,462 04	0,83	–0,186 33
0,04	–3,218 88	0,24	–1,427 12	0,44	–0,820 98	0,64	–0,446 29	0,84	–0,174 35
0,05	–2,995 73	0,25	–1,386 29	0,45	–0,798 51	0,65	–0,430 78	0,85	–0,162 52
0,06	–2,813 41	0,26	–1,347 07	0,46	–0,776 53	0,66	–0,415 52	0,86	–0,150 82
0,07	–2,659 26	0,27	–1,309 33	0,47	–0,755 02	0,67	–0,400 48	0,87	–0,139 26
0,08	–2,525 73	0,28	–1,272 97	0,48	–0,733 97	0,68	–0,385 66	0,88	–0,127 83
0,09	–2,407 95	0,29	–1,237 87	0,49	–0,713 35	0,69	–0,371 06	0,89	–0,116 53
0,1	–2,302 59	**0,3**	–1,203 97	**0,5**	–0,693 15	**0,7**	–0,356 67	**0,9**	–0,105 36
0,11	–2,207 27	0,31	–1,171 18	0,51	–0,673 34	0,71	–0,342 49	0,91	–0,094 31
0,12	–2,120 26	0,32	–1,139 43	0,52	–0,653 93	0,72	–0,328 5	0,92	–0,083 38
0,13	–2,040 22	0,33	–1,108 66	0,53	–0,634 88	0,73	–0,314 71	0,93	–0,072 57
0,14	–1,966 11	0,34	–1,078 81	0,54	–0,616 19	0,74	–0,301 11	0,94	–0,061 88
0,15	–1,897 12	0,35	–1,049 82	0,55	–0,597 84	0,75	–0,287 68	0,95	–0,051 29
0,16	–1,832 58	0,36	–1,021 65	0,56	–0,579 82	0,76	–0,274 44	0,96	–0,040 82
0,17	–1,771 96	0,37	–0,994 25	0,57	–0,562 12	0,77	–0,261 36	0,97	–0,030 46
0,18	–1,714 8	0,38	–0,967 58	0,58	–0,544 73	0,78	–0,248 46	0,98	–0,020 2
0,19	–1,660 73	0,39	–0,941 61	0,59	–0,527 63	0,79	–0,235 72	0,99	–0,010 05
0,2	–1,609 44	**0,4**	–0,916 29	**0,6**	–0,510 83	**0,8**	–0,223 14	**1**	0

Table des logarithmes népériens entre 1 et 100

N	ln(N)	N	ln(N)	N	ln(N)	N	ln(N)	N	ln(N)
1	0	21	3,044 52	41	3,713 57	61	4,110 87	81	4,394 45
2	0,693 15	22	3,091 04	42	3,737 67	62	4,127 13	82	4,406 72
3	1,098 61	23	3,135 49	43	3,761 2	63	4,143 13	83	4,418 84
4	1,386 29	24	3,178 05	44	3,784 19	64	4,158 88	84	4,430 82
5	1,609 44	25	3,218 88	45	3,806 66	65	4,174 39	85	4,442 65
6	1,791 76	26	3,258 1	46	3,828 64	66	4,189 65	86	4,454 35
7	1,945 91	27	3,295 84	47	3,850 15	67	4,204 69	87	4,465 91
8	2,079 44	28	3,332 2	48	3,871 2	68	4,219 51	88	4,477 34
9	2,197 22	29	3,367 3	49	3,891 82	69	4,234 11	89	4,488 64

$\ln(14)$ $\log_2(152)$

Admettons savoir calculer les puissances de e et de 2
avec des exposants flottants pas seulement entiers

$$x = \ln(14) \iff e^x = 14$$
$$\iff e^x - 14 = 0$$

$$x = \log_2(152) \iff 2^x = 152$$
$$\iff 2^x - 152 = 0$$

$\ln(14)$ $\log_2(152)$

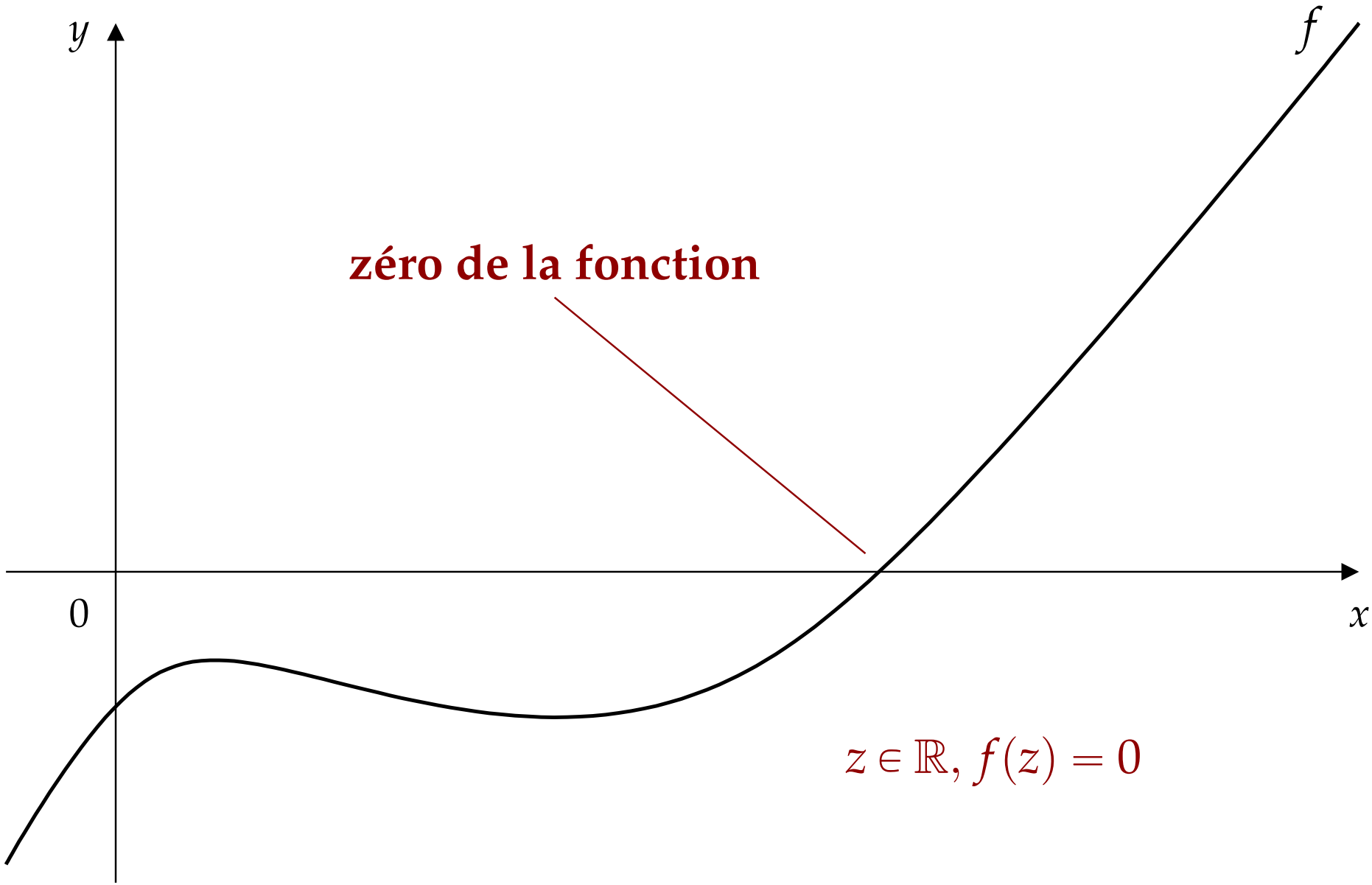
Admettons savoir calculer les puissances de e et de 2
avec des exposants flottants pas seulement entiers

$$x = \ln(14) \iff e^x = 14$$
$$\iff e^x - 14 = 0$$

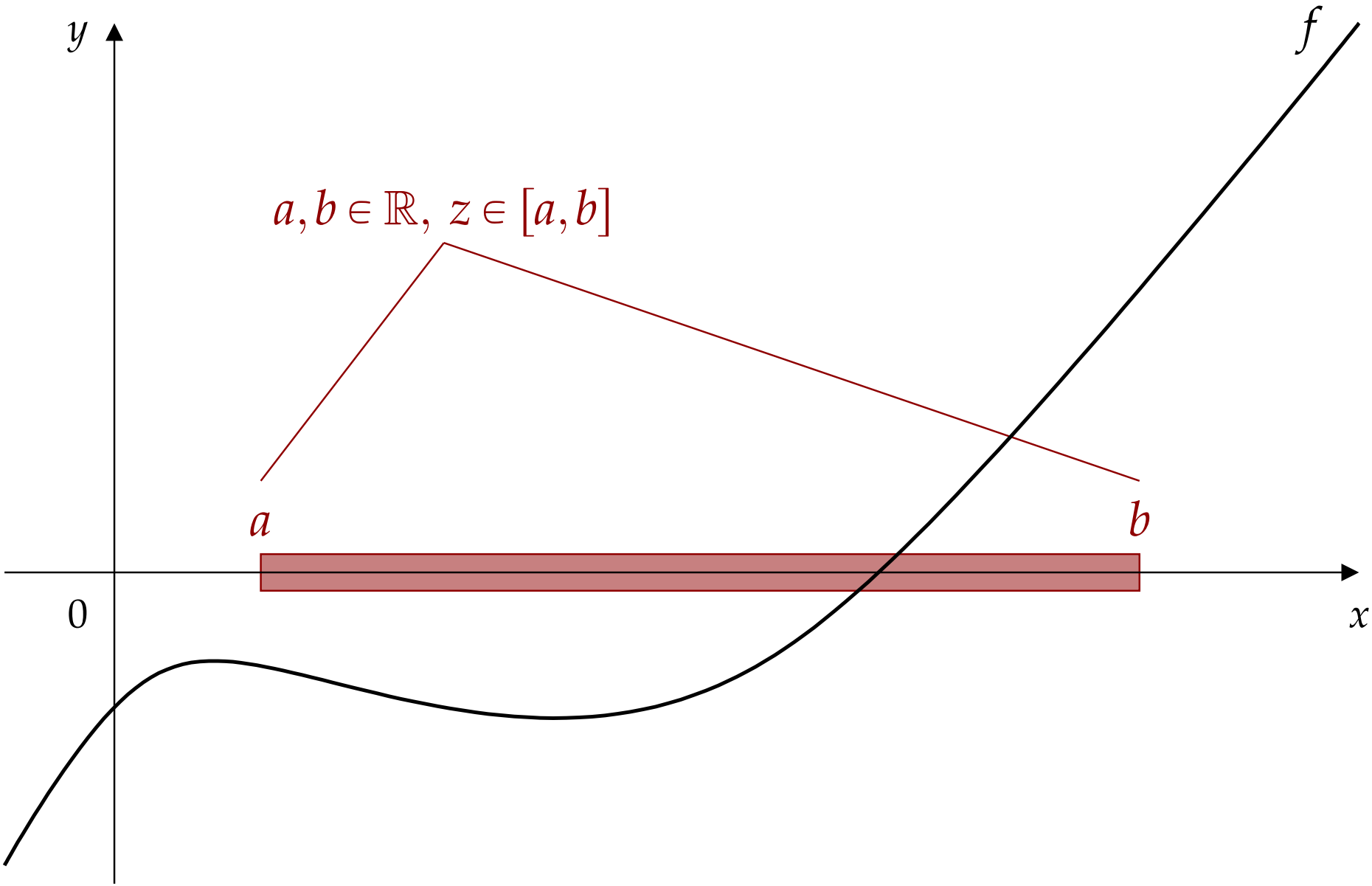
$$x = \log_2(152) \iff 2^x = 152$$
$$\iff 2^x - 152 = 0$$

Il faut trouver le zéro d'une fonction

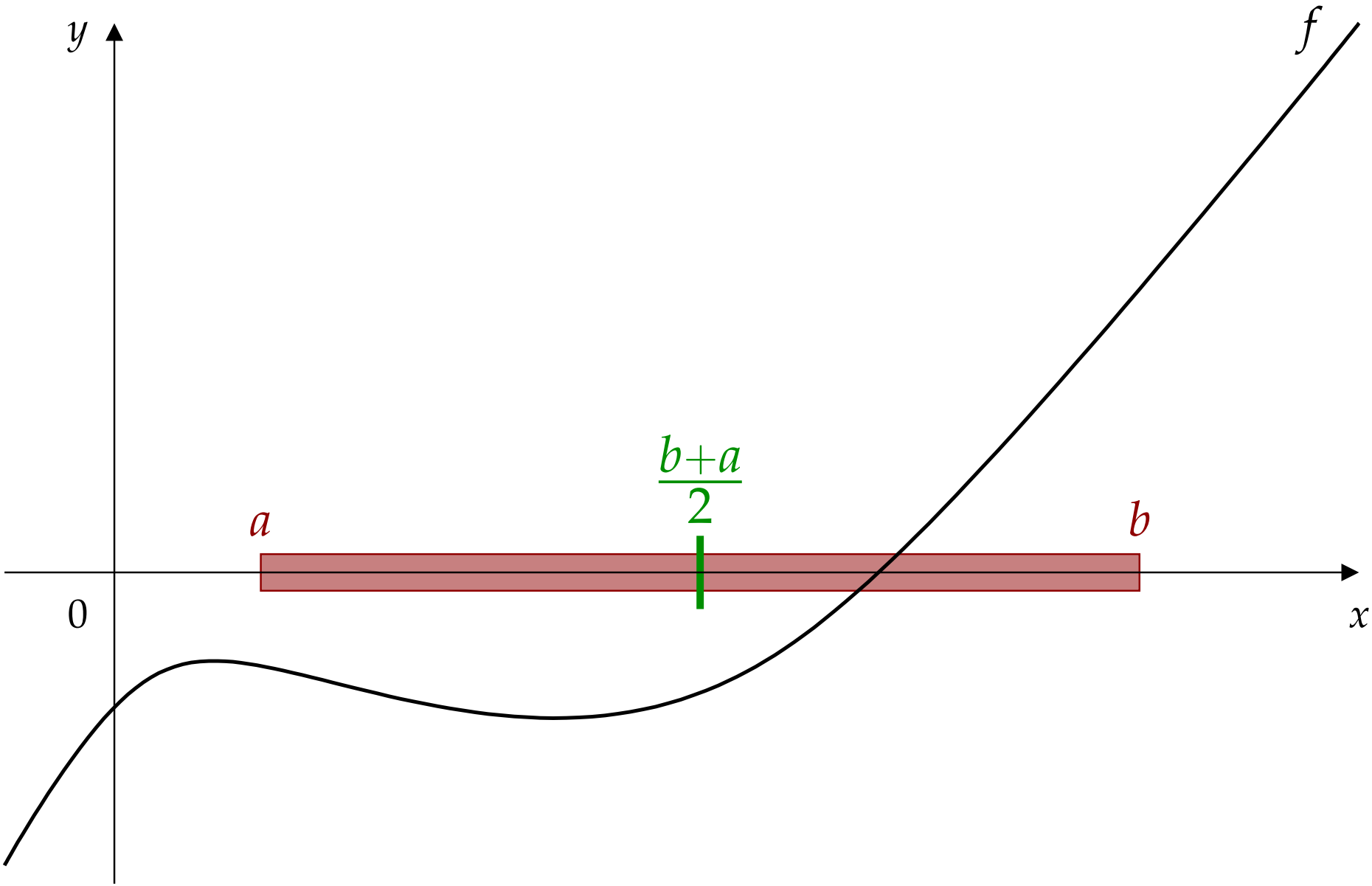
Approximation du zéro



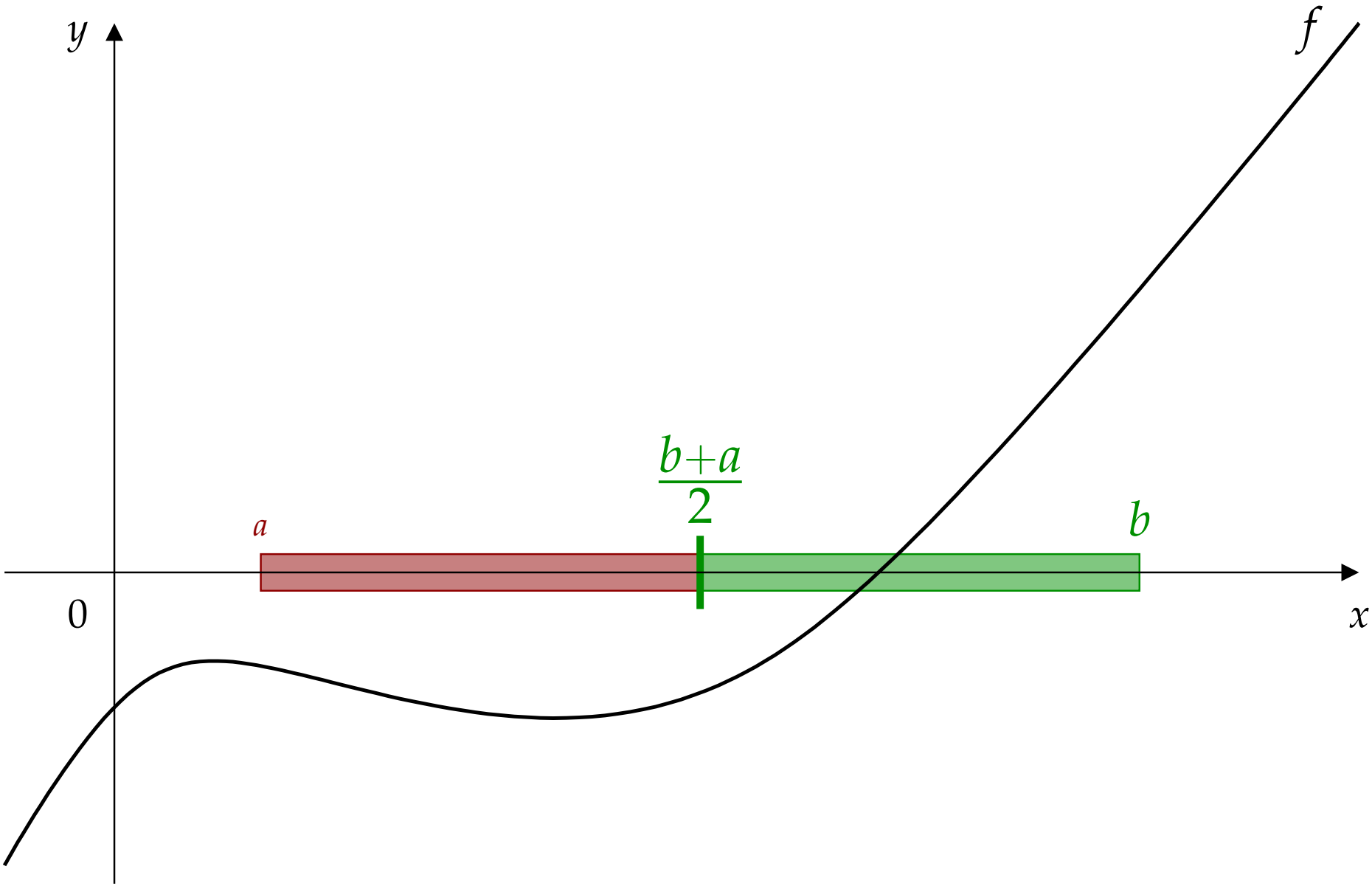
Recherche dichotomique



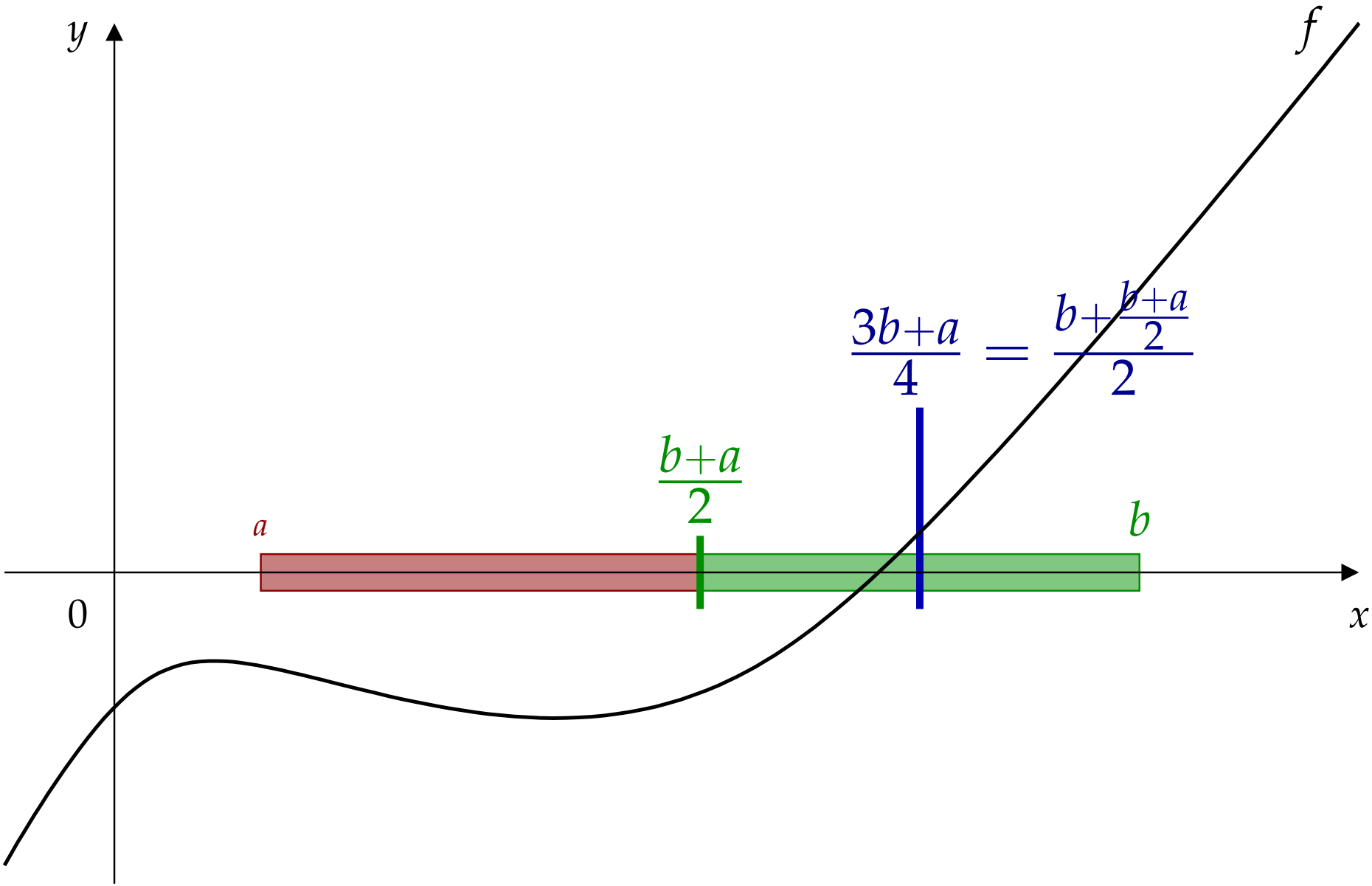
Recherche dichotomique



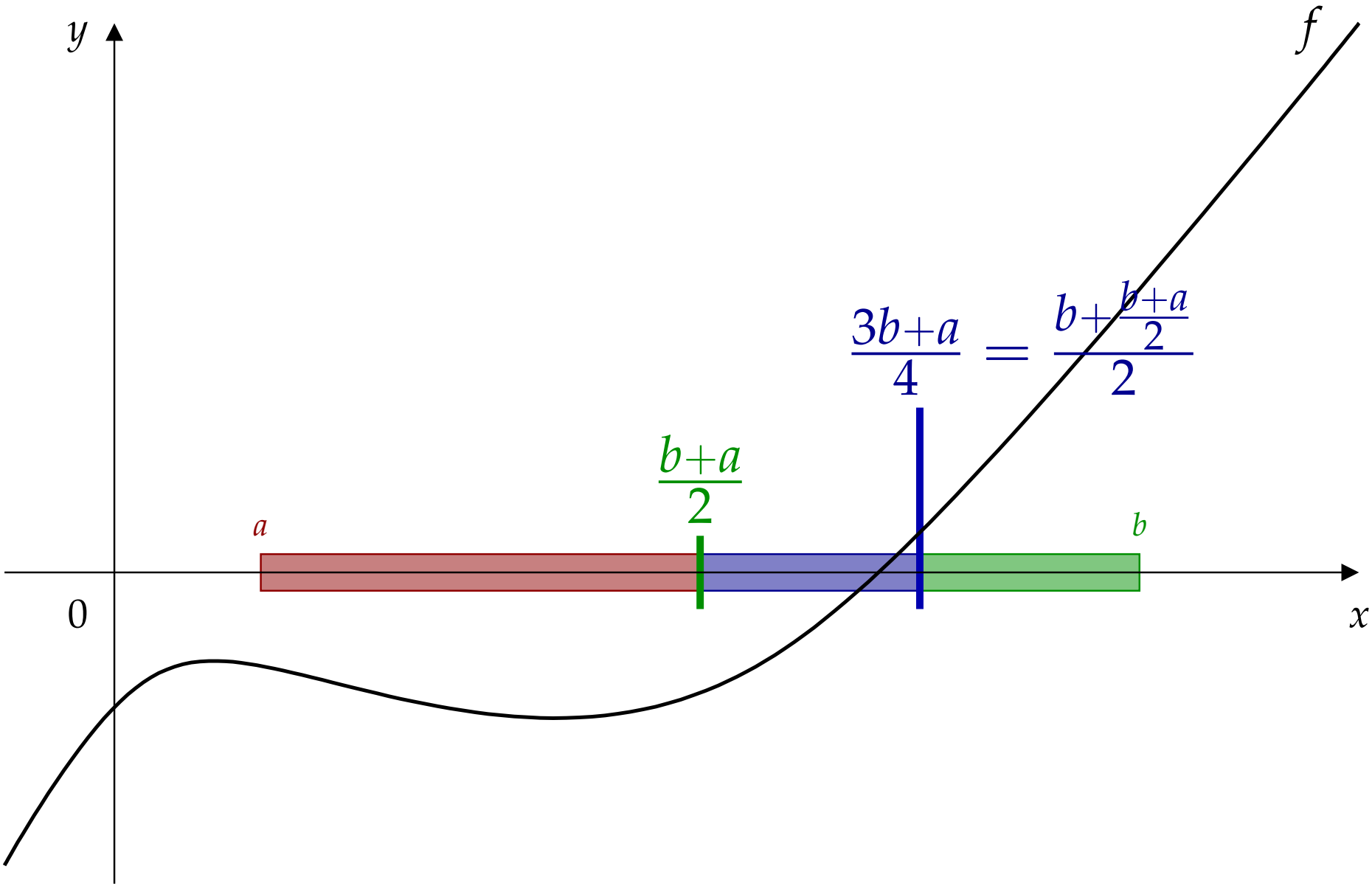
Recherche dichotomique



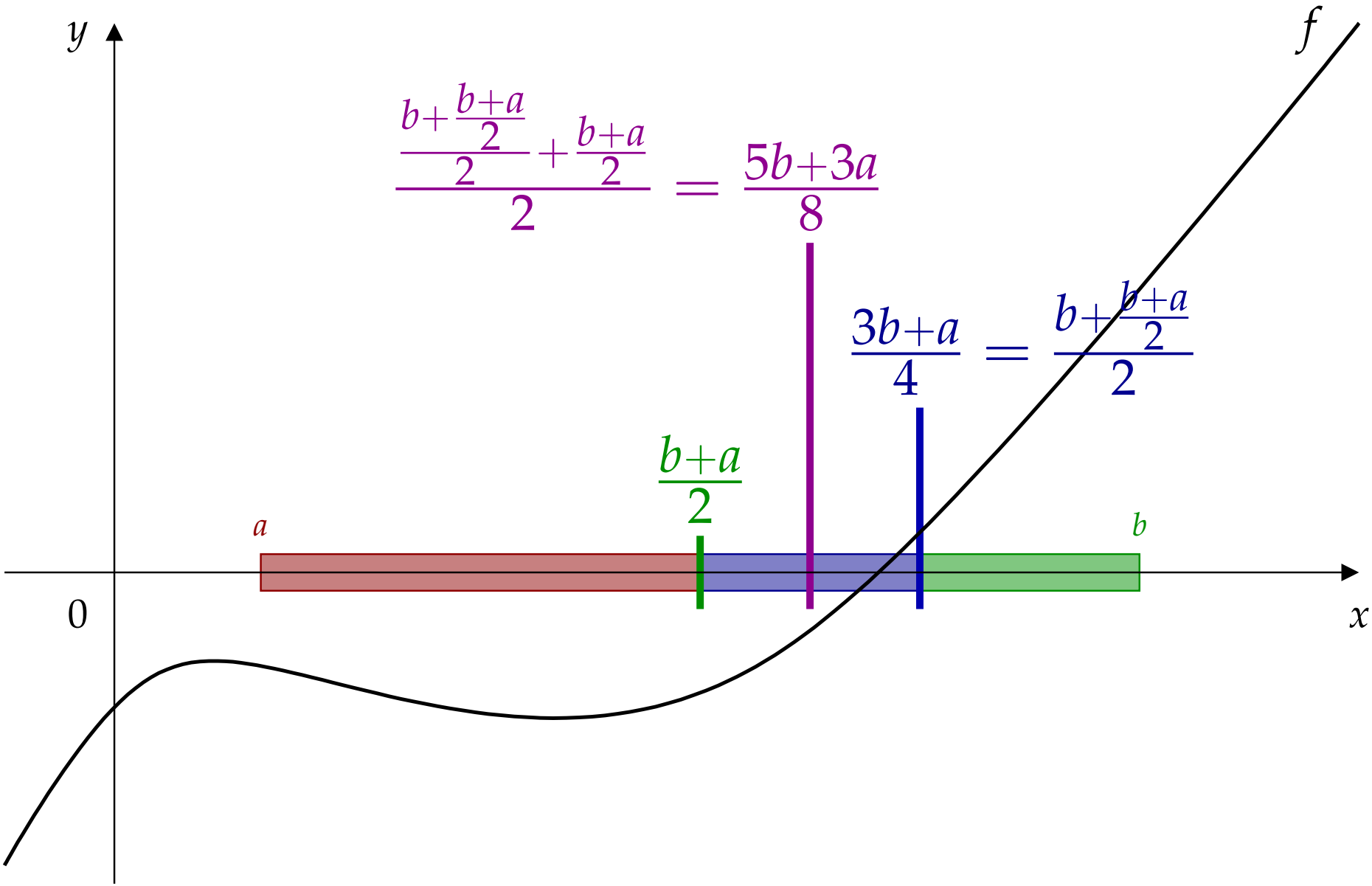
Recherche dichotomique



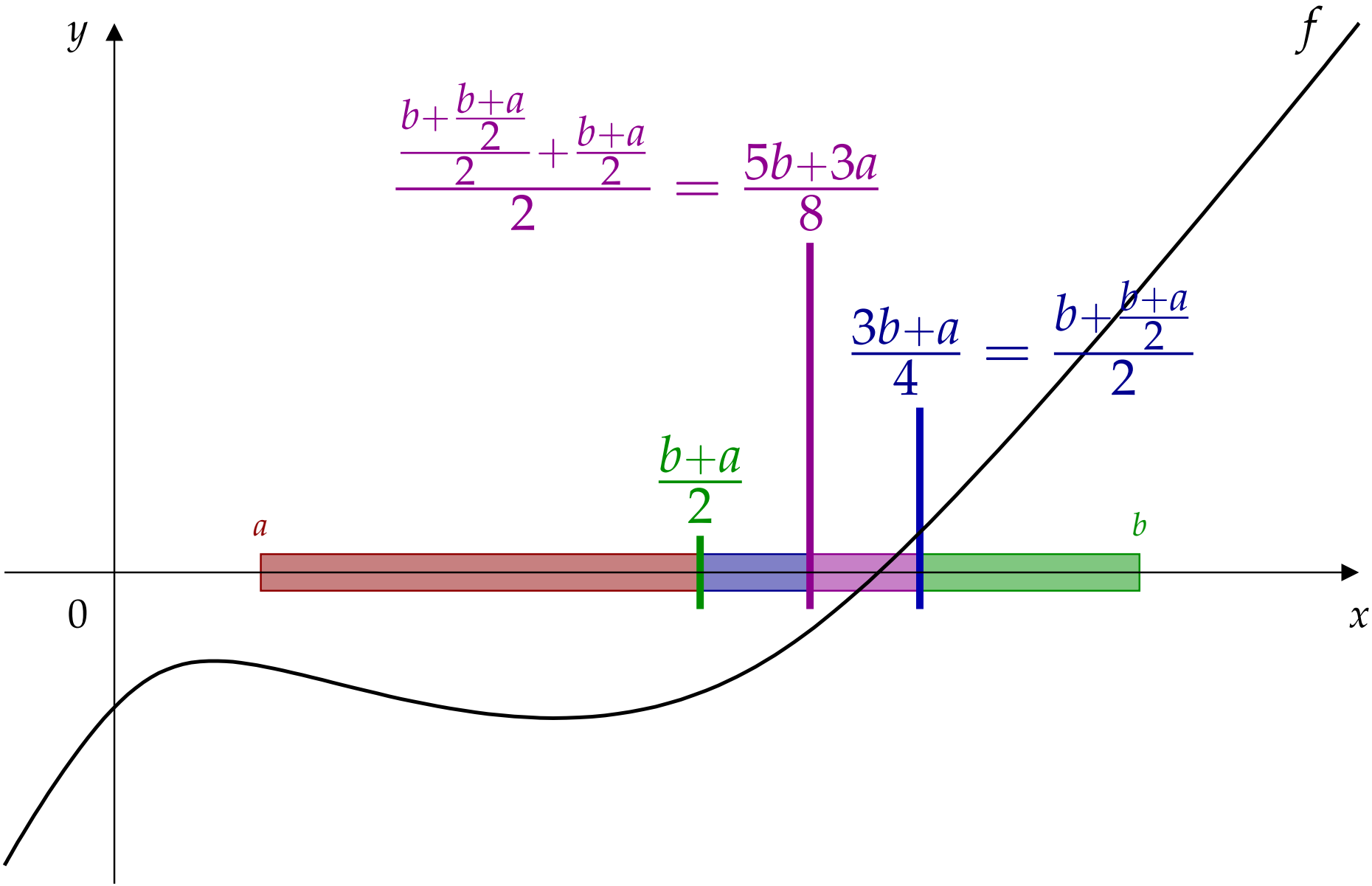
Recherche dichotomique



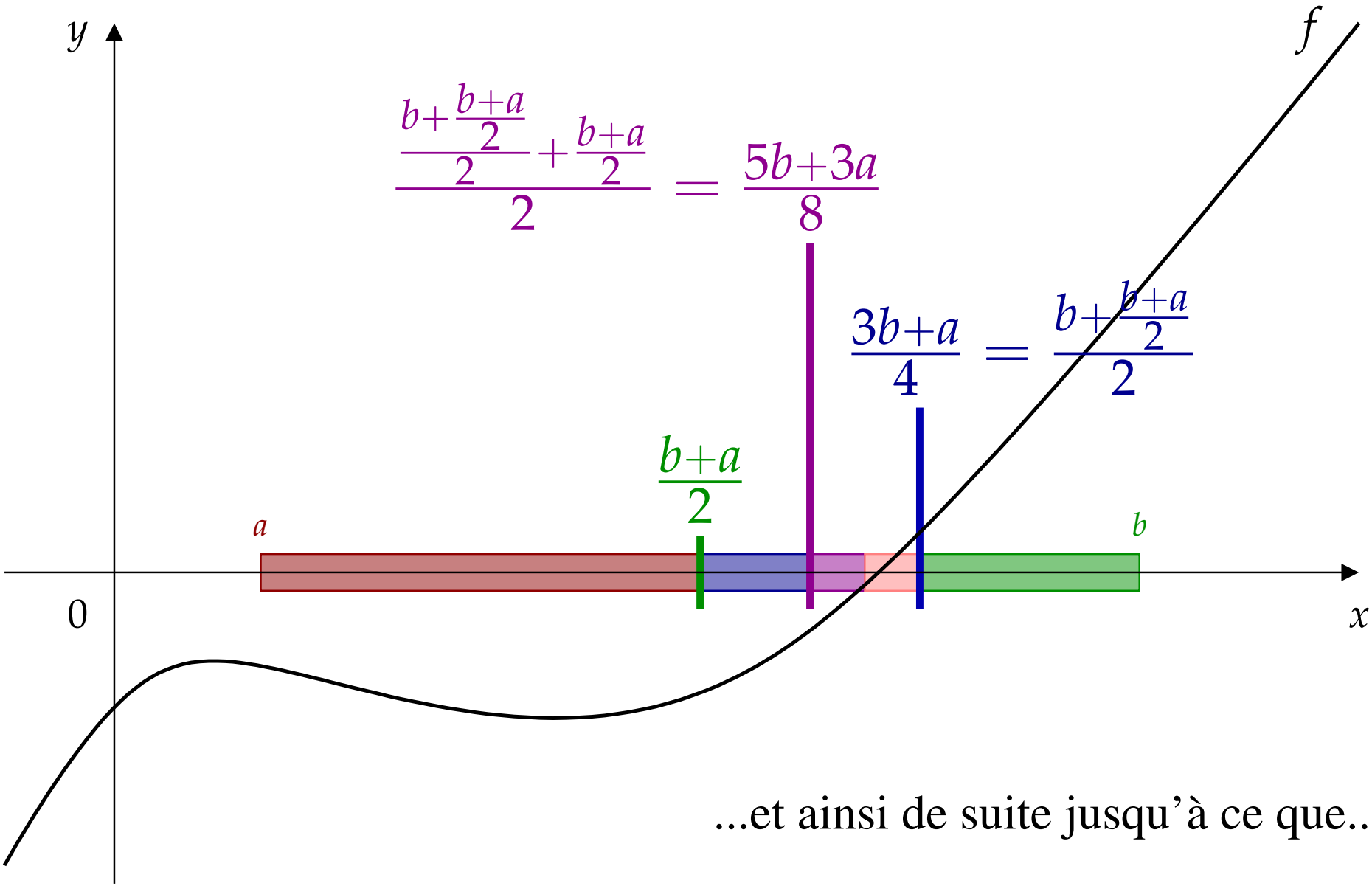
Recherche dichotomique



Recherche dichotomique

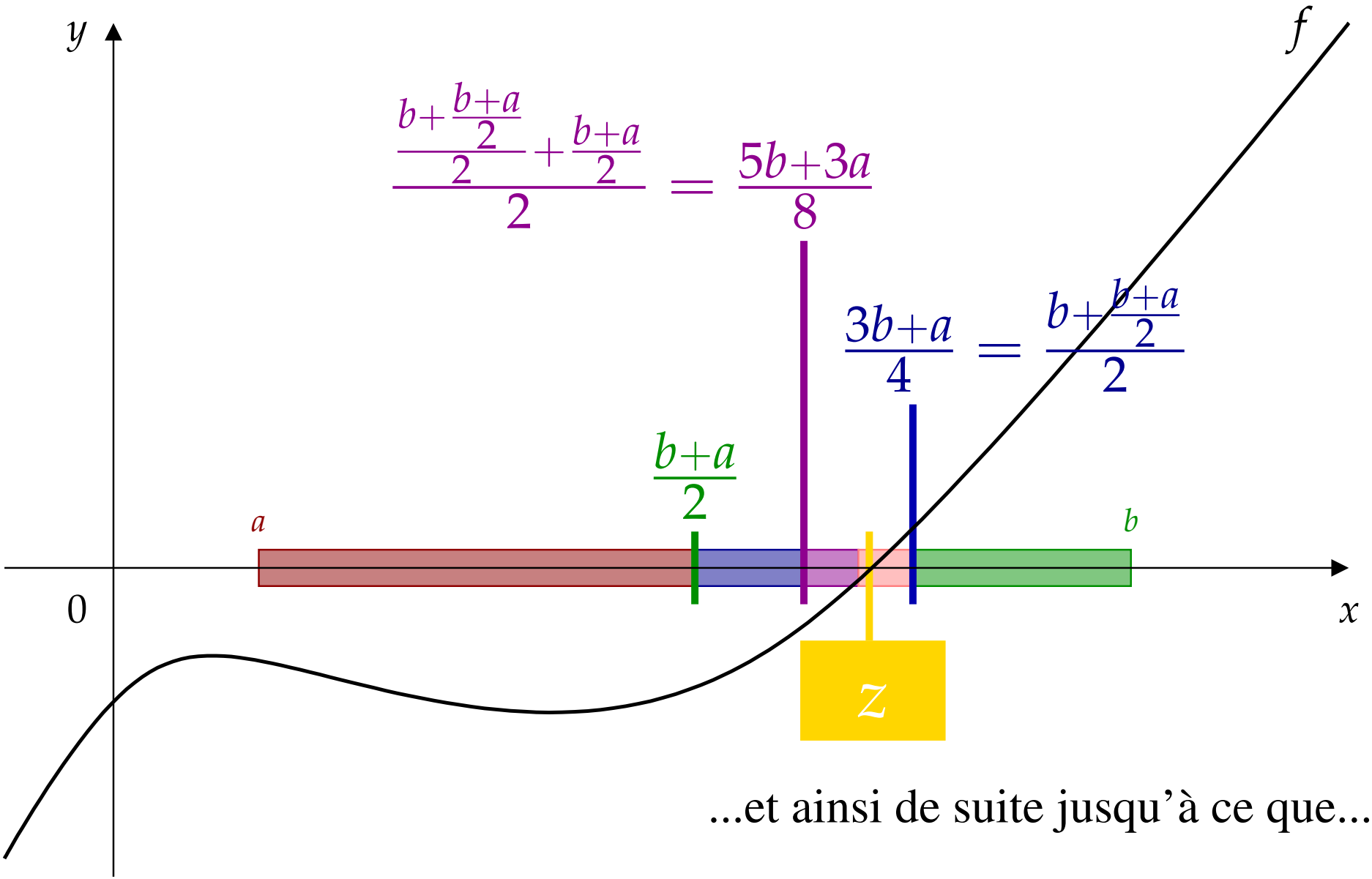


Recherche dichotomique



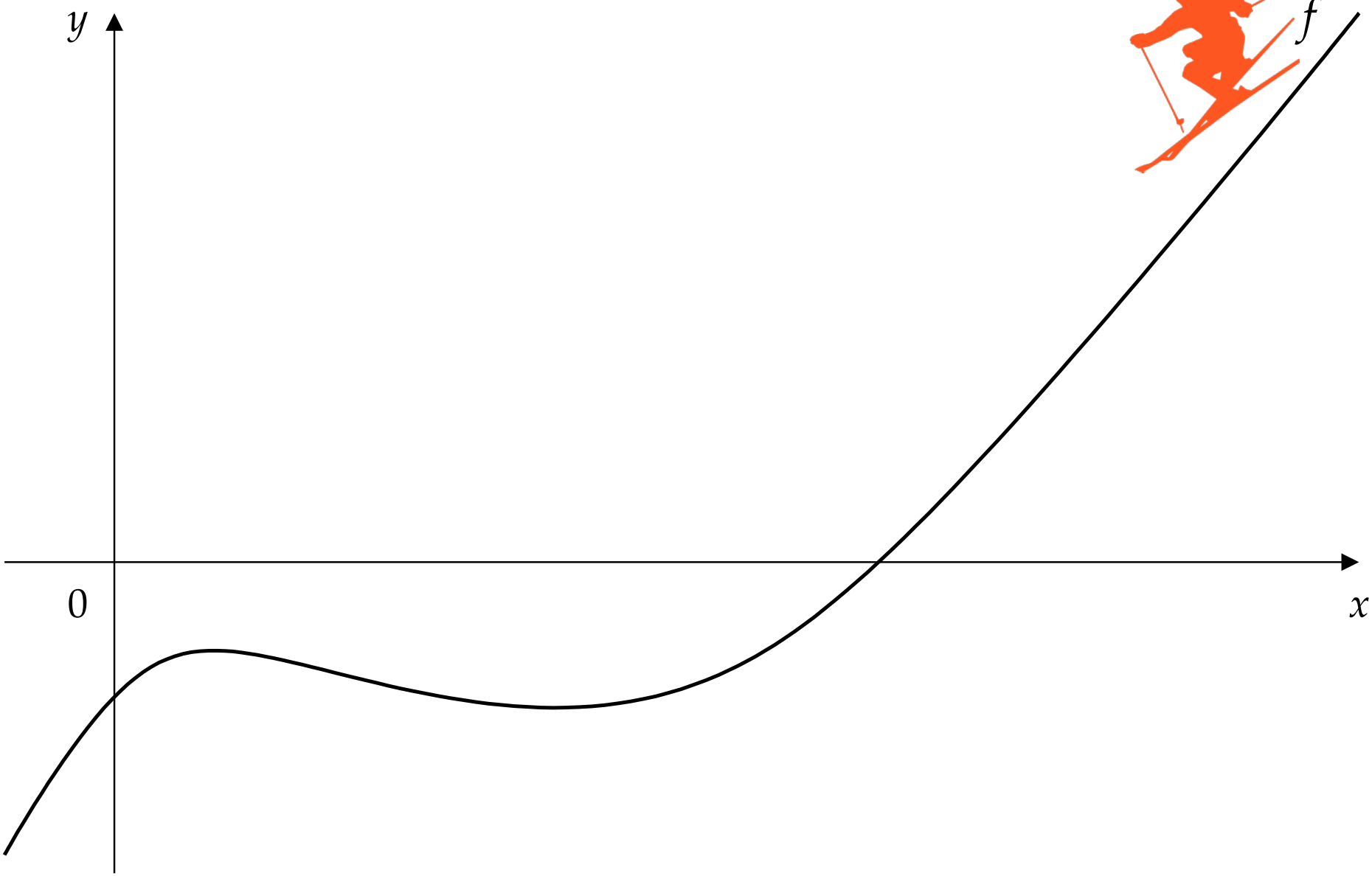
...et ainsi de suite jusqu'à ce que...

Recherche dichotomique

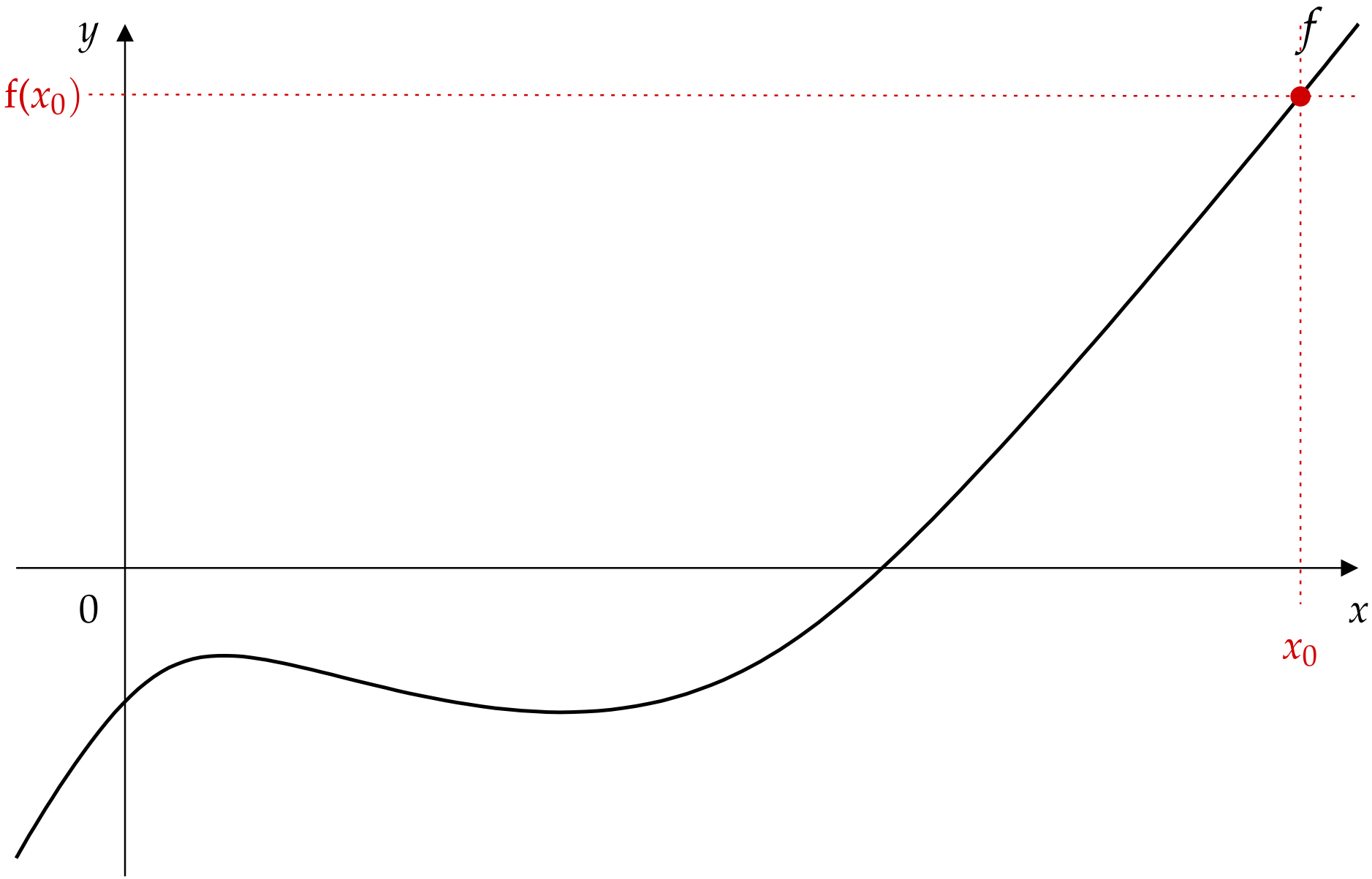


...et ainsi de suite jusqu'à ce que...

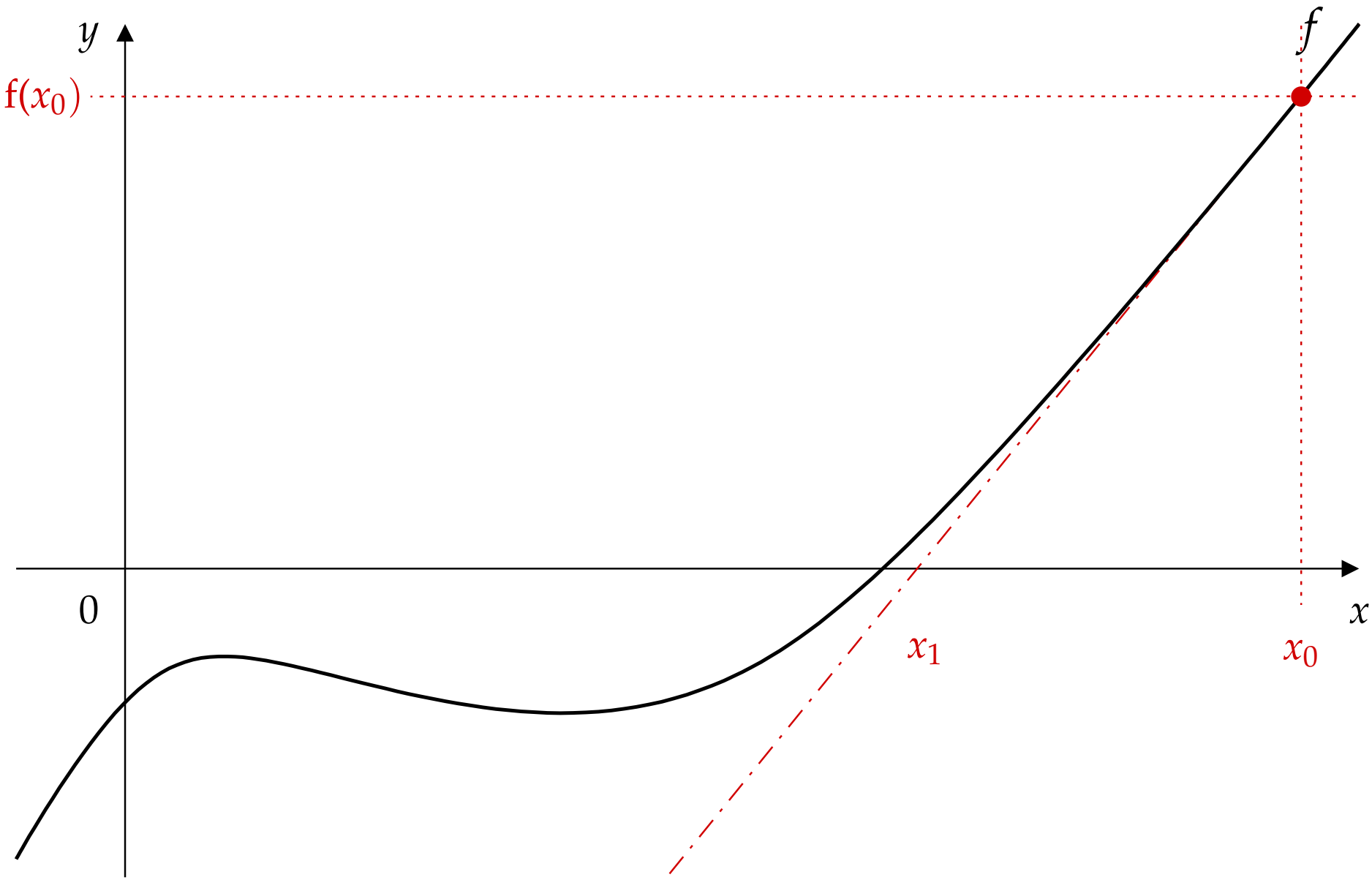
Une autre méthode : la descente



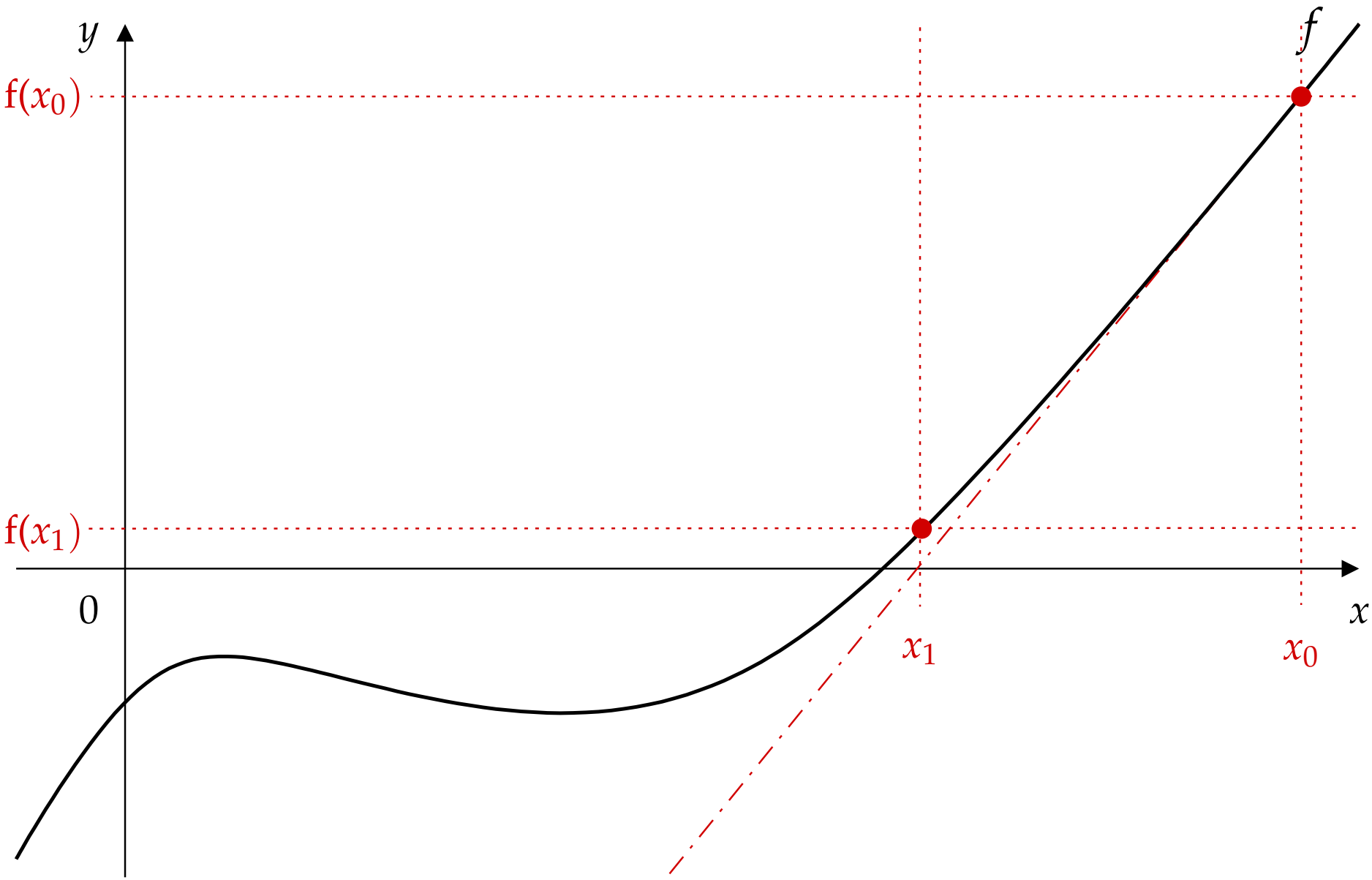
Une autre méthode : la descente



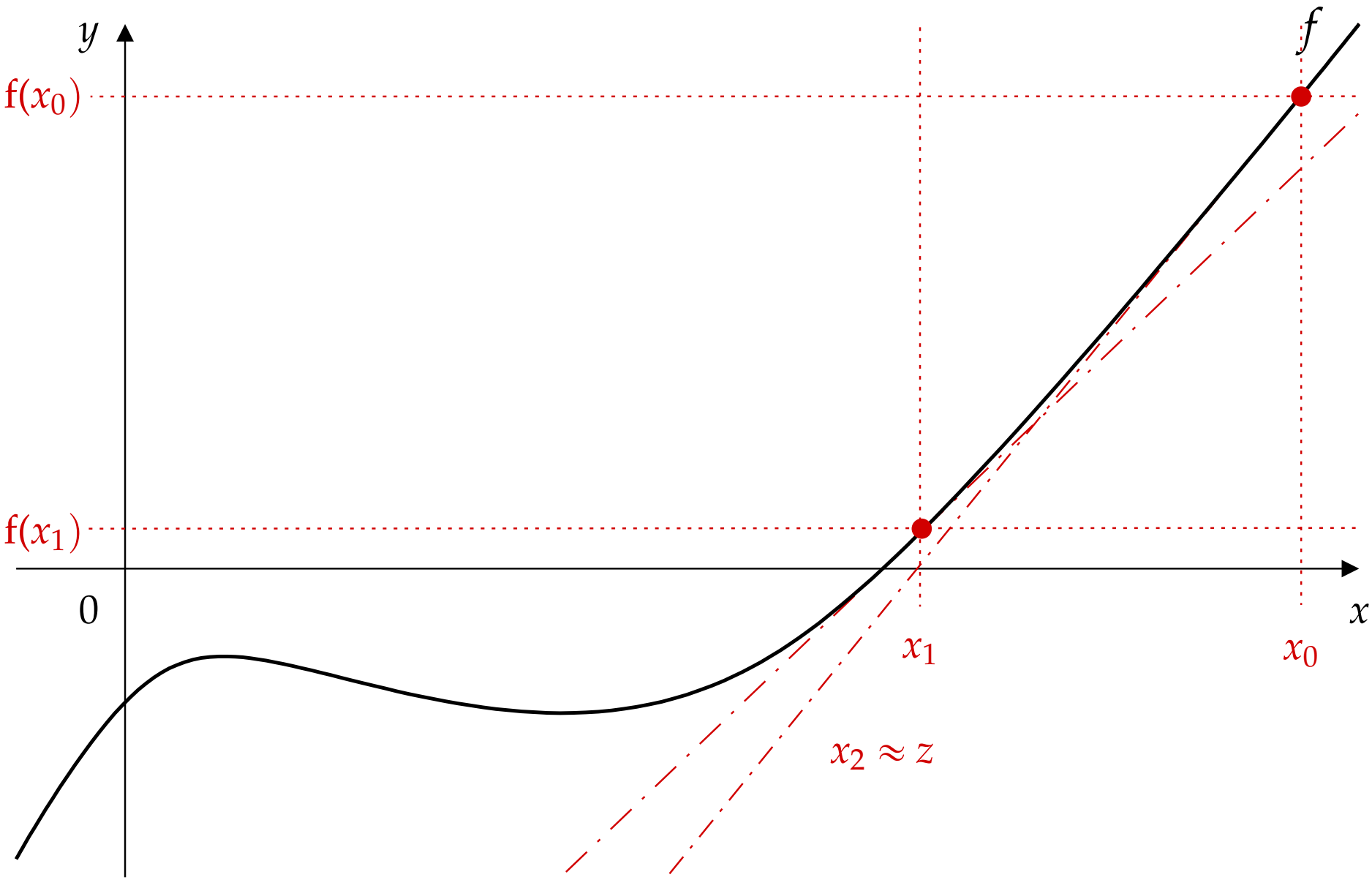
Une autre méthode : la descente



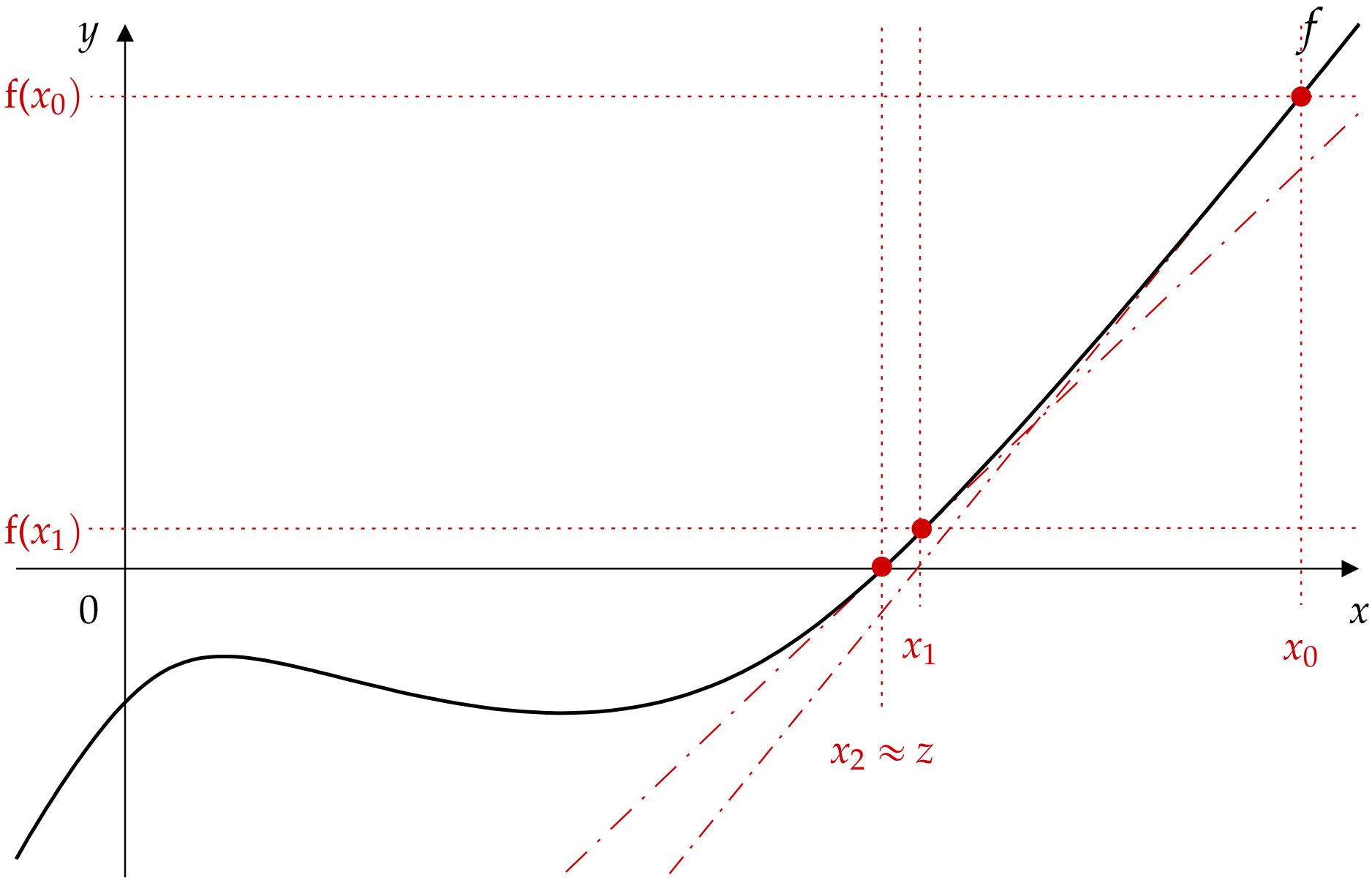
Une autre méthode : la descente



Une autre méthode : la descente



Une autre méthode : la descente



Algorithme de Newton

Fonction approx_zero(d f, x_0 : **réel**) : **réel**
    x_{tmp} : **réel** ;
Début
    $x_{\text{tmp}} := x_0$;
    Tant que $(f(x_{\text{tmp}}) \neq 0)$ **faire**
        "Tracer" la tangente t de f en x_{tmp} ;
        x_{tmp} := abscisse de l'intersection de t et $(0, x)$;
    Fin faire
    retour x_{tmp} ;
Fin

Algorithme de Newton

Fonction approx_zero(d f, x_0 : **réel**) : **réel**
    x_{tmp} : **réel** ;
Début
    $x_{\text{tmp}} := x_0$;
    Tant que $(f(x_{\text{tmp}}) \neq 0)$ **faire**
        "Tracer" la tangente t de f en x_{tmp} ;
        x_{tmp} := abscisse de l'intersection de t et $(0, x)$;
    Fin faire
    retour x_{tmp} ;
Fin

- Comment "tracer"/calculer la tangente de f en a ?
- La tangente de f en a est la droite d'équation :

$$t = f(a) + (x - a) \times f'(a)$$

- L'abscisse du point d'intersection de t et $(0, x)$ est :

$$f(a) + (x - a) \times f'(a) = 0 \iff x = a - \frac{f(a)}{f'(a)}$$

Algorithme de Newton

Fonction approx_zero(d f, f', x_0 : **réel**) : **réel**
    x_{tmp} : **réel** ;
Début
    $x_{\text{tmp}} := x_0$;
    Tant que $(f(x_{\text{tmp}}) \neq 0)$ **faire**
        $x_{\text{tmp}} := x_{\text{tmp}} - f(x_{\text{tmp}})/f'(x_{\text{tmp}})$;
    Fin faire
    retour x_{tmp} ;
Fin

- Calcul de f' ?
  $\rightsquigarrow$ on l'ajoute en argument !

- $f(x) = e^x - 14$
  $\rightsquigarrow f'(x) = e^x$

- $f(x) = 2^x - 152$
  $\rightsquigarrow f'(x) = \ln(2) \times e^{x \times \ln(2)}$

Algorithme de Newton

Fonction approx_zero(d f, f', x_0 : **réel**) : **réel**
    x_{tmp} : **réel** ;
Début
    $x_{\text{tmp}} := x_0$;
    Tant que $(f(x_{\text{tmp}}) \neq 0)$ **faire**
        $x_{\text{tmp}} := x_{\text{tmp}} - f(x_{\text{tmp}})/f'(x_{\text{tmp}})$;
    Fin faire
    retour x_{tmp} ;
Fin

- Cet algorithme termine-t-il ?
- Non !
- On s'arrête quand on pense être suffisamment proche

Algorithme de Newton

Fonction approx_zero(d f, f', x_0, err : **réel**) : **réel**
    x_{tmp} : **réel** ;
Début
    $x_{\text{tmp}} := x_0$;
    Tant que ($|f(x_{\text{tmp}})| >$ err) **faire**
        $x_{\text{tmp}} := x_{\text{tmp}} - f(x_{\text{tmp}})/f'(x_{\text{tmp}})$;
    Fin faire
    retour x_{tmp} ;
Fin

- Cet algorithme termine-t-il ?
- Non !
- On s'arrête quand on pense être suffisamment proche