

Les algorithmes de tri

- ② { : Sélection
- ③ { : Insertion
- ④ { : bulle

<
>

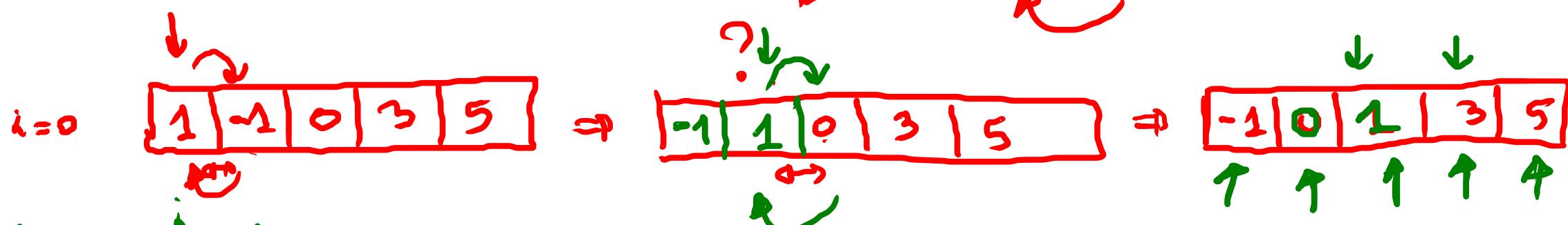
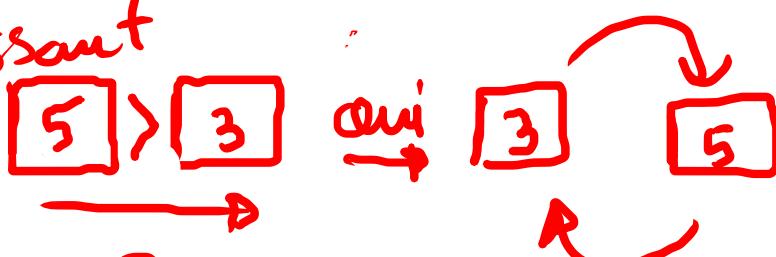
1	-1	0	3	5
---	----	---	---	---

↓ tri par ordre Croissant

-1	0	1	3	5
----	---	---	---	---

- parcour
- comparaison
- permutation / échange

le tri à bulle : (ordre croissant)



il y avait échange.
répéter encore

i	0	1	2	3	4
	-2	3	-4	10	8
-2	-4	3	10	8	4
-2	-4	3	8	10	

i	0	1	2	3	4
	-4	-2	3	8	10

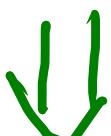
echange	i	$T[i]$	$T[i+1]$
faux	0	-2	-4
vrai	1	-5	3
"	2	3	8
"	3	8	10
"	4		

↗

echange	i	$T[i]$	$T[i+1]$
faux	0	-2	3
"	1	3	-4
vrai	2	3	10
"	3	10	8
"	4	8	10

Il y a echange

Il y avait echange



Le tableau est trié \Leftrightarrow il n'y avait pas d'echange \Rightarrow

echange	i	$T[i]$	$T[i+1]$
faux	0	-4	-2
	1	-2	3
	2	3	8
	3	8	10
faux			

Algorithme : triBulle

échange : Booléen ;

{m, i : entier.

T[0..N-1] : dévier plein

debout // affichage avant le tri

① Répéter :

échange \leftarrow faux ; //

② Pour i = 0 à N-2

si ($T[i] > T[i+1]$) alors :

// échange

échange \leftarrow vrai ; //

tmp $\leftarrow T[i]$;

$T[i] \leftarrow T[i+1]$;

$T[i+1] \leftarrow tmp$;

fin si

Fin Pour

Jusqu'à (échange = faux)

// affichage après le tri

fin

do

do

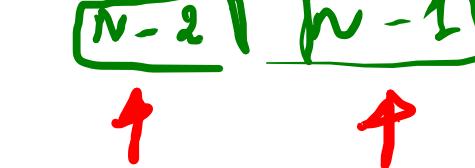
N elts

N-1

[N-2] [N-1]

N

↑ ↑



échange \leftarrow vrai ;

Tant que (échange = vrai)

échange \leftarrow faux ;

{

fin Tant que

while (échange = faux)
while (échange = vrai)

Algorithme : triBulle

échange : Booléen ;

lm, i : entier.

T[0..N-1] : débute plein

debout // affichage avant le tri

① Répéter :

échange \leftarrow faux ;

② Pour i de 0 à N-2

Si ($T[i] > T[i+1]$) alors :

//échange échange \leftarrow vrai ;

tmp $\leftarrow T[i]$;

$T[i] \leftarrow T[i+1]$;

$T[i+1] \leftarrow tmp$;

fin si

fin Pour

Jusqu'à (échange = faux)

// affichage après le tri

fin

```
#include <stdio.h>
```

```
void main( ) {
```

```
int t[] = {-1, 0, 1, 2, -4}
```

```
int taille = 5;
```

```
int tmp, i;
```

```
int echange;
```

```
do {
```

```
    echange = 0;
```

```
    for(i = 0, i <= taille-2; i++)
```

```
    { if( t[i] > t[i+1] ) {
```

```
        echange = 1;
```

```
        tmp = t[i];
```

```
        t[i] = t[i+1];
```

```
        t[i+1] = tmp;
```

```
    } while ( echange );
```

```
// affichage après échange
```

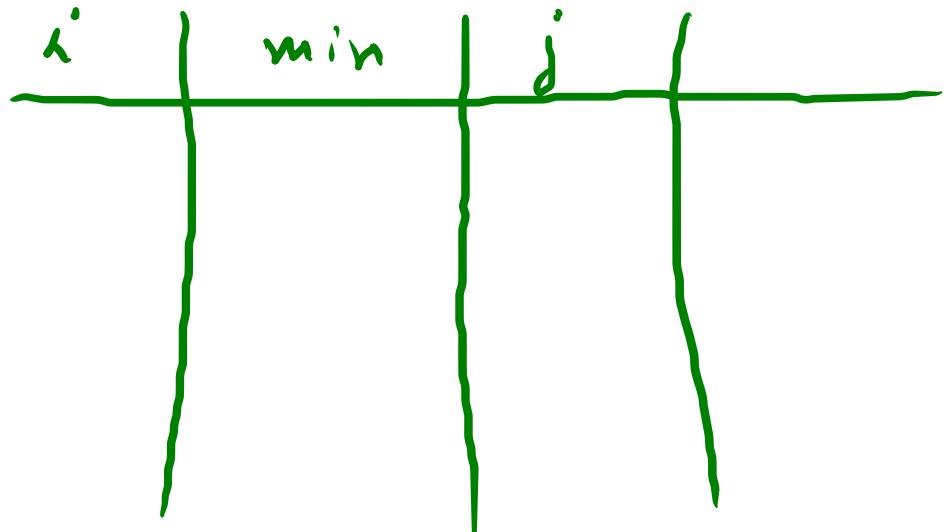
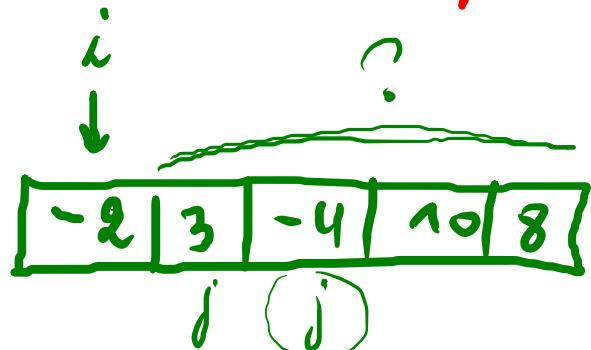
```
for(i = 0; i < taille; i++)
```

```
printf (" %d ", t[i]);
```

```
} // fin main
```

Tri par sélection

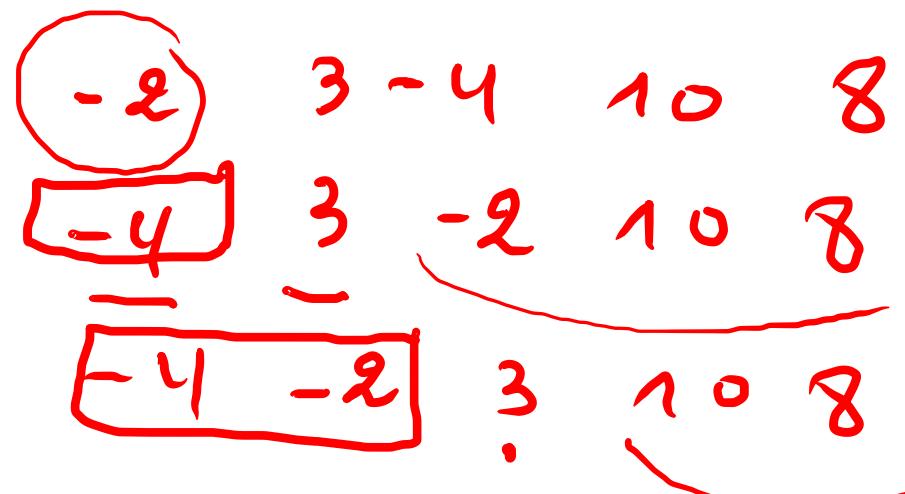
min.



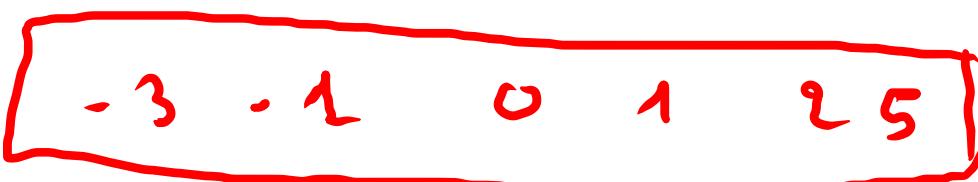
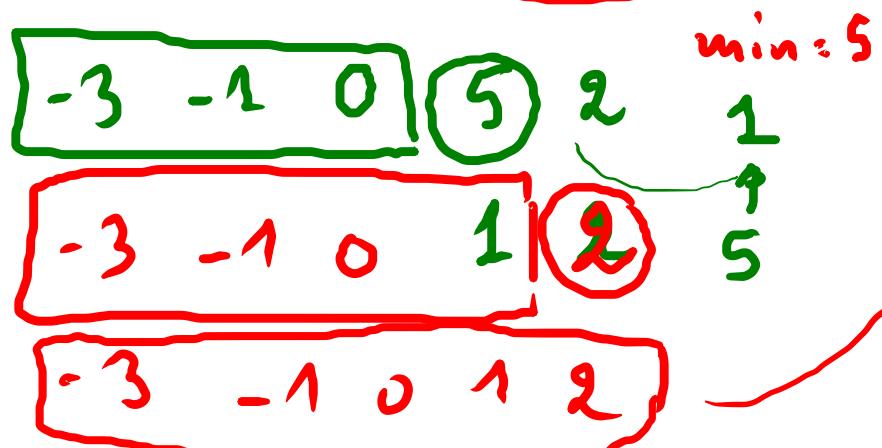
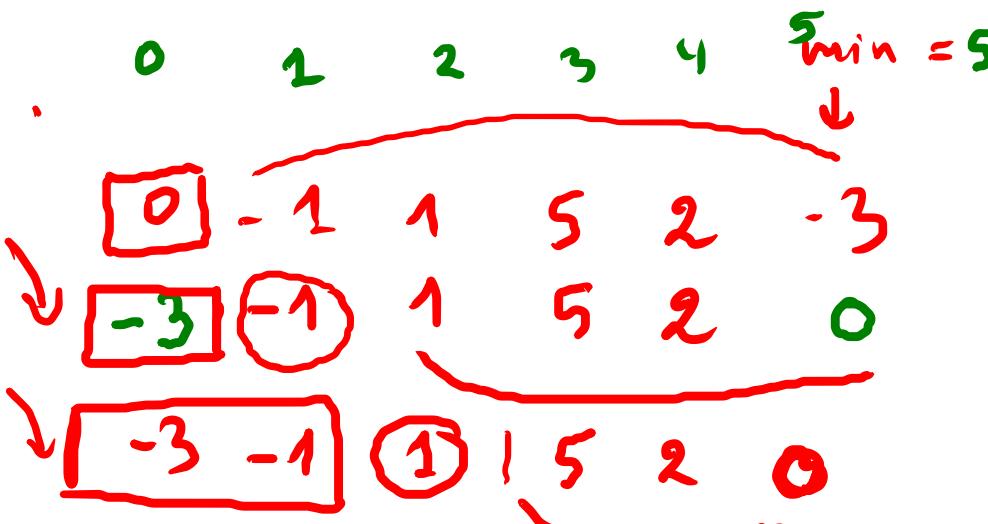
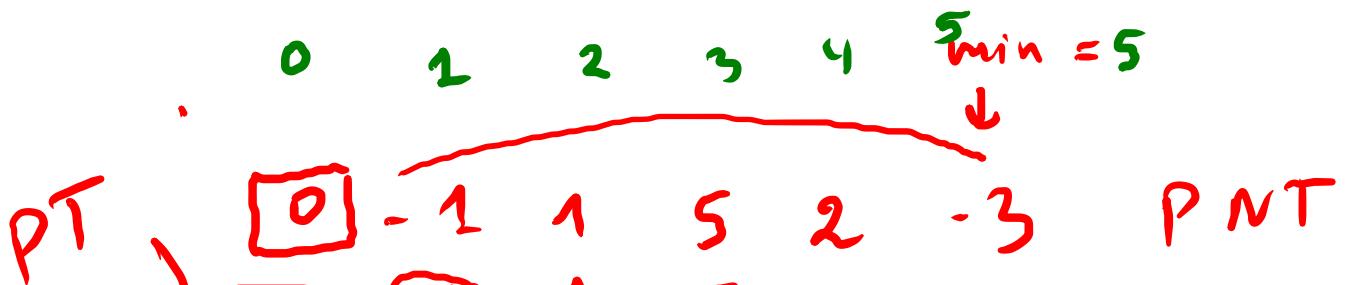
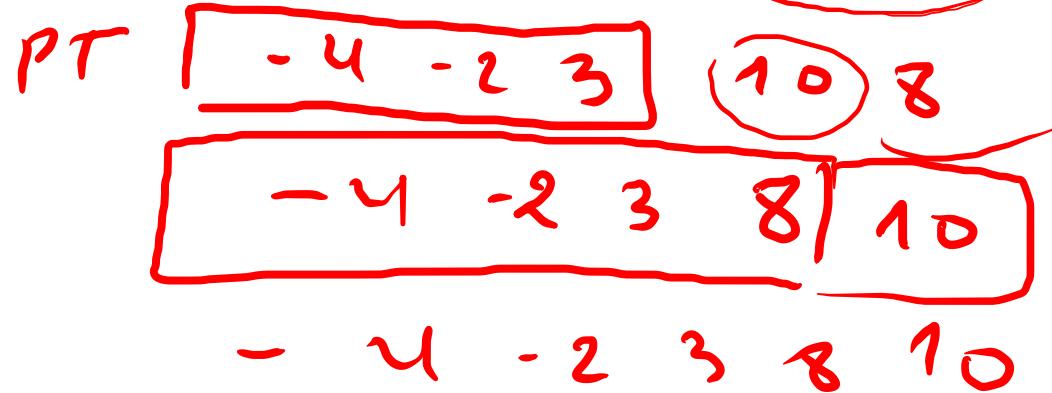
$i = 0$

$\min a[i] \Rightarrow \min = 0$

|
 | sélection
 | échange



partie triée



Tri par sélection

0	1	2	3	4
-2	3	-4	10	8

$$\boxed{-4} \quad 3 \quad \boxed{-2} \quad 10 \quad 8$$

$$-4 \quad -2 \textcolor{red}{|} 3 \quad 10 \ 8$$

-4 -2 1 10

$$-4 -2 + 3 \quad) 8$$

(min +)

min J 1 min

2 3 3

2 4 3

トナリ

+ ^ w w p w c

3 4 8

4

4 5

i) exchange

卷之三

— 1 —

i ← min - i
j ← i + 1
selection du mini

selection &
change

exchange $i = 2$ $j = 3$ $T[i] = 2$ $T[j] = 3$
 $i \quad \text{min} \quad j \quad T[\text{min}] \quad T[j]$
 2 2 3 3 > 10
 ? 4 3 > 8
 $\min \neq i$ no exchange

3 3 4 10 > 8

(min \neq i) echange

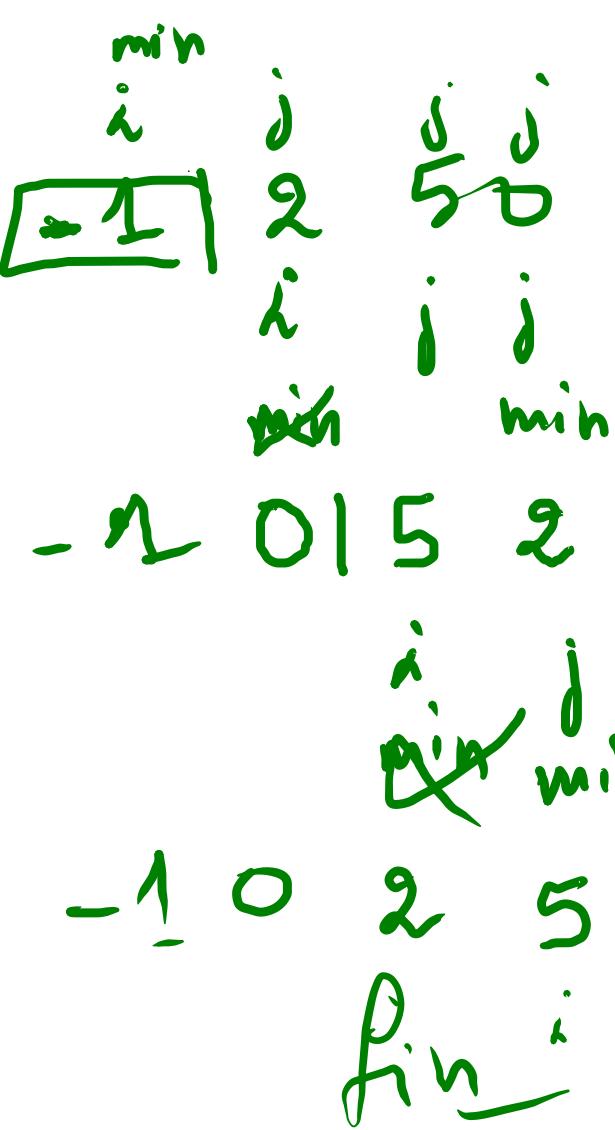
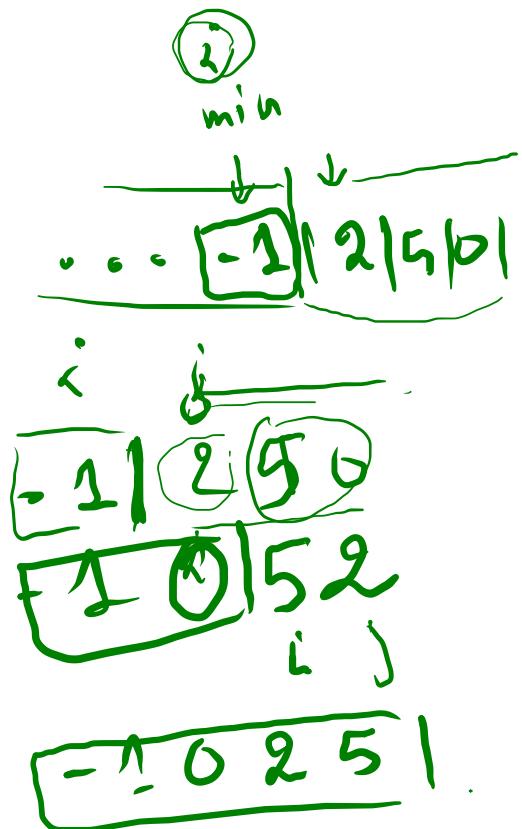
4
↓ fin

\min_i	j	$T[\min]$	$T[j]$
0	1	-2	3
2	2	-2	-4
2	3	-4	10
4	4	-4	8

exchange ($\min \neq i$)
 $\text{tmp} \leftarrow T[i]$
 $T[i] \leftarrow T[\min]$
 $T[\min] \leftarrow \text{tmp}$

```

    graph LR
      A[1 > -2] --> B[2 < 3]
      B --> C[3 > -2]
      C --> D[-2 > 10]
      D --> E[4 > -2 > 8]
    
```



Tri par sélection
 Algorithme : triSelection
 $\text{tmp}, i, \text{min}, j$: entier
 $T[0..N-1]$: entier plain
 , , affichage avant
 Debut

```

    Pour  $i \leftarrow 0$  à  $N-2$  faire :
       $\text{min} \leftarrow i;$ 
      Pour  $j \leftarrow i+1$  à  $N-1$  faire :
        Si  $T[j] < T[\text{min}]$  alors
           $\text{min} \leftarrow j;$ 
        FinPour
      FinSi
      Si ( $\text{min} \neq i$ ) alors :
         $\text{tmp} \leftarrow T[i];$ 
         $T[i] \leftarrow T[\text{min}];$ 
         $T[\text{min}] \leftarrow \text{tmp};$ 
      FinSi
    FinPour
    Fin "affichage après"
  
```

Tri par sélection

Algorithm : triSelection
tmp, i, min, j : entier

T[0...N-1] : entier plein
,, affichage avant

Début Pour i = 0 à N-2 faire:
 min ← i;
 Pour j ← i+1 à N-1 faire:
 Si T[j] < T[min] alors
 min ← j;
 Fin Pour fin si
 Si (min ≠ i) alors:
 tmp ← T[i];
 T[i] ← T[min];
 T[min] ← tmp;
 Fin Si
Fin Pour "affichage après"

"affichage après tri"
for (i = 0; i < taille; i++) {
 printf("%d ", t[i]);
}
} // fin de main

```
void main() {  
    int tmp, i, min, j;  
    int t[] = {-5, 3, -8, 10, 1};  
    int taille = 5;  
  
    for (i = 0; i <= taille - 2; i++) {  
        min = i;  
        for (j = i + 1; j < taille; j++) {  
            if (t[j] < t[min])  
                min = j;  
        }  
        // exchange  
        if (min != i) {  
            tmp = t[i];  
            t[i] = t[min];  
            t[min] = tmp;  
        }  
    }  
}
```

Appliquer les algorithmes de tri
• tri à bulle
• tri par sélection
au tablau suivant :

-5	0	-7	10	5
----	---	----	----	---

→ tri à bulle

[echange?
[parcour
• permutation

1/ change = faux

-5	0	-7	10	5
-7	0	10	5	?

⇒ change = vrai

2/

change = faux

-5	-7	0	5	10
-7	-5	0	5	10

⇒ change = vrai

3/

change = faux

-7	-5	0	5	10
-7	-5	0	5	10

⇒ change = faux

Appliquer les algorithmes de tri

- tri à bulle

- tri par sélection

au tabl leau suivant :

0	1	2	3	n
-5	0	-7	10	5

→ tri par sélection

$i = 0$, $\min = 0$ (indice du premier élé)

• sélectionner (chercher) le min des élé de $i+1$ à $n-1$



-5	0	-7	10	5
----	---	----	----	---

$$(11 \rightarrow 4) \\ N = 5$$

$\min = 2$

• $\min \neq i$ si oui \Rightarrow échange

-7	0	-5	10	5
----	---	----	----	---

↓ select

$\min = 2$

* $i = 1$, $\min = 1$

$i < N-2$

-7	0
----	---

• $\min \neq i$

-7	-5	0	10	5
----	----	---	----	---

* $i = 2$, $\min = 2$

$i < N-2$

-7	-5
----	----

• $\min \neq i$ nm pas d'échange

* $i = 3$, $\min = 3$

-7	-5	0
----	----	---

10 5

-7	-5	0
----	----	---

\min

i

10

5

↑
 \min

• $\min \neq i$ oui

-7	-5	0	5	10
----	----	---	---	----

* $i = 4$ ($N-2 = 3$)

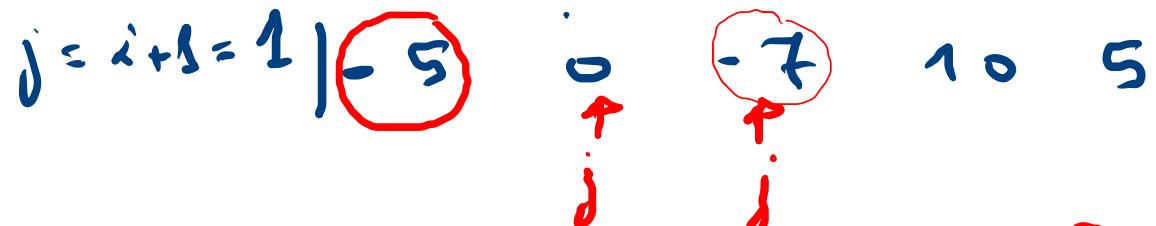
fin

-7	-5	0	5	10
----	----	---	---	----

Tri par insertion



- i = 0 trouver l'elt $T[j]$ tel que $T[j] < T[i]$

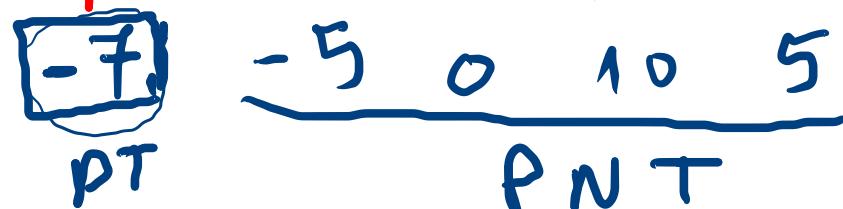


oui ($T[j] < T[i]$)
 $\text{tmp} = -7 = T[j]$

Décaler jusqu'à j



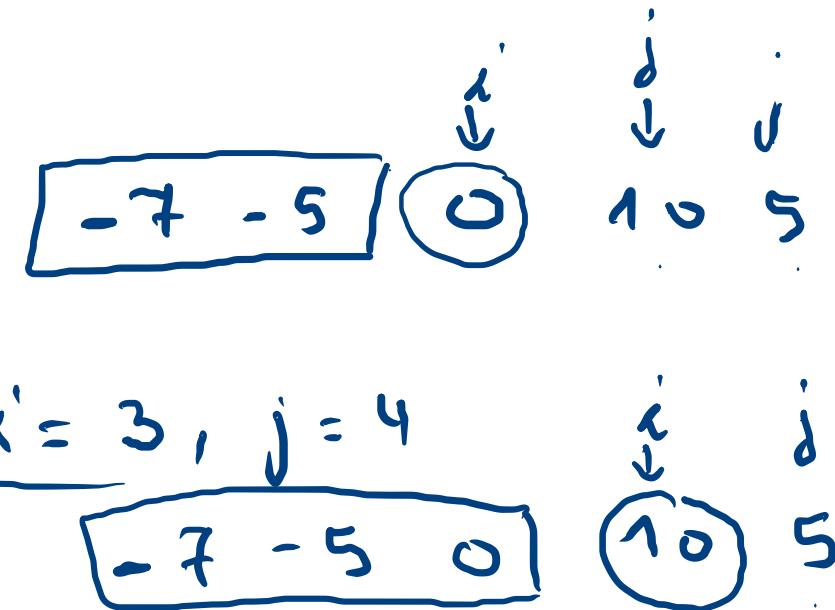
replacer $T[i]$ par tmp



- $i = 1, j = i + 1 = 2$



$i = 2, j = i + 1 = 3$



$$i = 3, j = 4$$

$\text{tmp} = 5 = T[j]$

Décalage de i à j



$T[i] \leftarrow \text{tmp}$



$i = 4 > (N - 2 = 3)$

fin

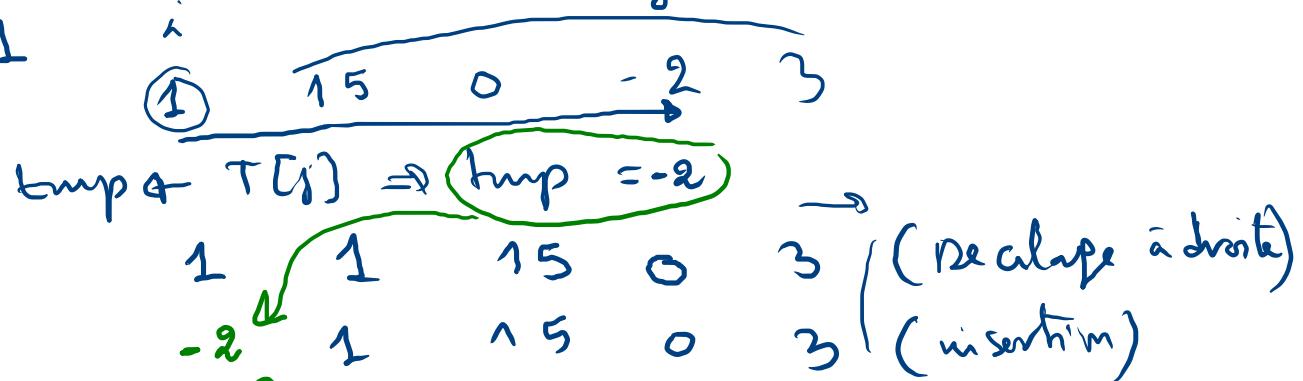
-7 | -5 | 0 | 5 | 10

0	1	2	3	4
1	15	0	-2	3

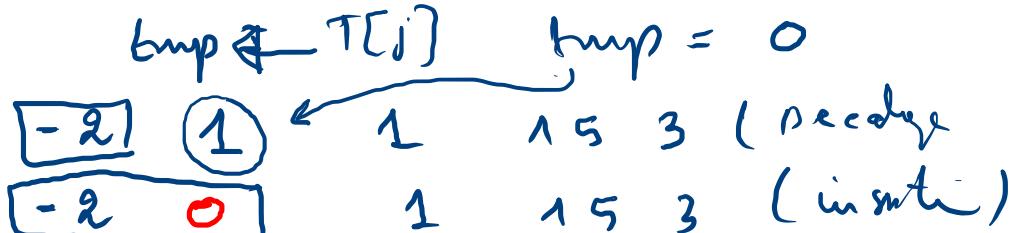
$N = 5$ (elts)

$$i \leftarrow 0 \Leftarrow (N-2 = 3) \quad j \leq N-1$$

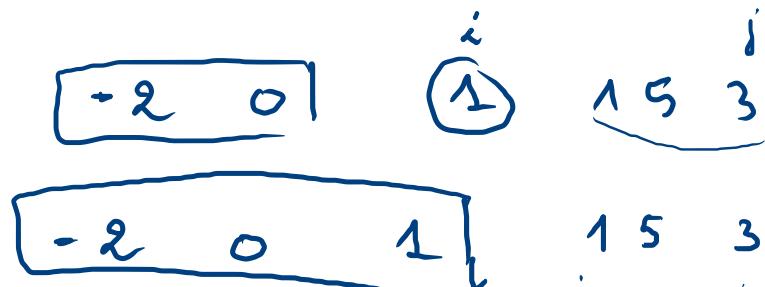
$$\begin{array}{l} i \leftarrow 0 \\ j \leftarrow 1 \end{array}$$



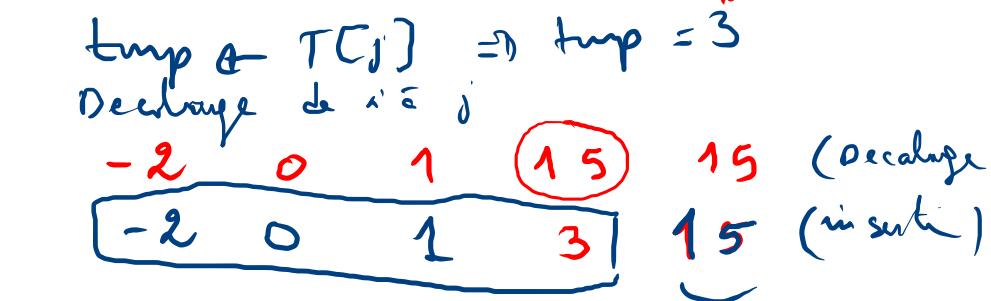
$$\begin{array}{l} i \leftarrow 1 \\ j \leftarrow 2 \end{array}$$



$$\begin{array}{l} i \leftarrow 2 \\ j = 3 \end{array}$$



$$\begin{array}{l} i \leftarrow 3 \\ j = 4 \end{array}$$



$$i \leftarrow 4 > (N-2) \Rightarrow \text{fin}$$



Algorithme : hi-par-insertri

r, k, j, i, N : entier

$T[0 \dots N-1]$: tableau d'entier plein

tmp : entier

pour $i \leftarrow 0$ à $n-1$ faire

$k \leftarrow i$; (indice du min)

 pour $j \leftarrow i+1$ à $n-1$ faire

 si ($T[j] < T[i]$) alors

$\text{tmp} \leftarrow T[j]$;

$k \leftarrow j$;

 fin si

 fin pour

 si ($k \neq i$) alors ($K > i$)

 pour $r \leftarrow k$ à $k+1$ faire

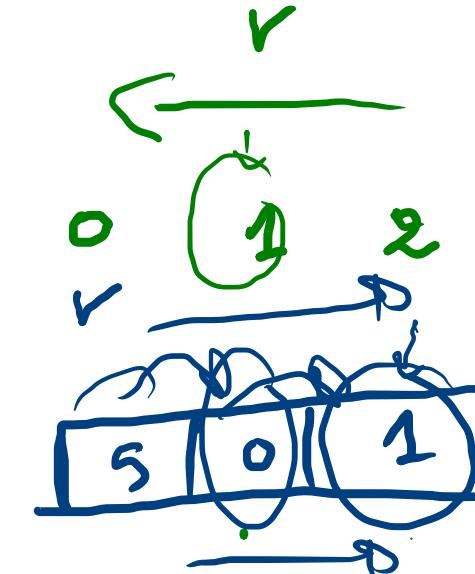
$T[r] \leftarrow T[r+1]$

 fin pour

$t[i] = \text{tmp}$

 fin si

fin pour



$r \leftarrow r - 1$

(r se décrémente
 r (sup → min))

$\underline{r \leftarrow s \bar{a} 0}$

$\Rightarrow r$ se décrémente

0 1 2 3 4
 5 6 7 8 9 → 5 6 6 7 8
 Decaller ↓ 1 1 1 1
 pour $i \leftarrow 4 \text{ à } 2$
 $T[i] \leftarrow T[i-1]$
 fin pour

pour $i \leftarrow 1 \text{ à } 4$
 $T[i+1] \leftarrow T[i]$
 fin pour

$i = 4$
 $T[4] \leftarrow T[3]$
 5 6 7 8 8
 $i = 3$
 $T[3] \leftarrow T[2]$
 5 6 7 7 8
 $i = 2$
 $T[2] \leftarrow T[1]$
 5 6 6 7 8
 $i = 1$
 5 6 6 7 8

$i = 1$
 $T[2] \leftarrow T[1]$
 5 6 6 8 9
 $T[3] \leftarrow T[2]$
 5 6 6 6 9
 $i = 2$
 $T[2] \leftarrow T[1]$
 5 6 6 6 8
 $i = 1$
 5 6 6 6 7

Algorithme : Tri-Insertion

i, j, tmp : entier

T[0..N-1] : entier (plein) (N : nombre d'elts)

Début

Pour $i \leftarrow 1$ à $N-1$ faire

 tmp $\leftarrow T[i];$

 j $\leftarrow i;$

 Tant que ($j > 0$ et $T[j-1] > tmp$)

$T[j] \leftarrow T[j-1];$

 j $\leftarrow j - 1;$

 fin Tant que

$T[j] = tmp;$

fin pour

fin

Algorithme : Tri-Insertion

i, j, tmp : entier

$T[0..N-1]$: entier (plein) (N : nombre d'elts)

Début

Pour $i \leftarrow 1$ à $N-1$ faire

$\text{tmp} \leftarrow T[i];$

$j \leftarrow i - 1;$

 Tant que ($j > 0$ et $T[j-1] > \text{tmp}$)

$T[j] \leftarrow T[j-1];$

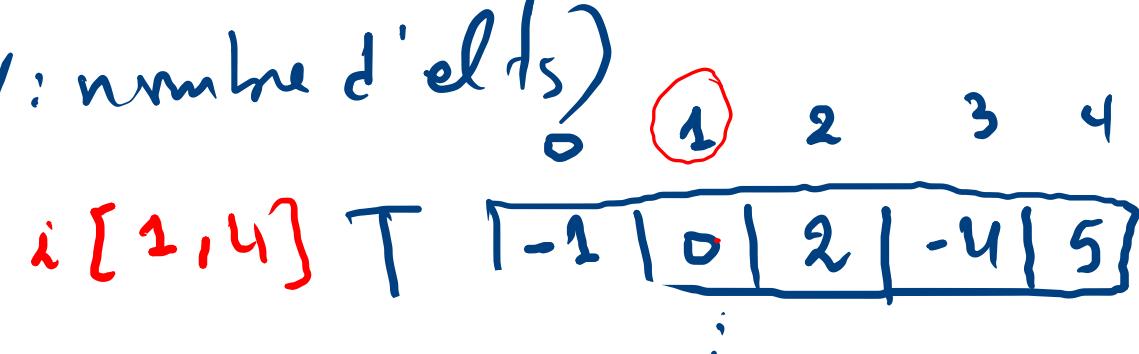
$j \leftarrow j - 1;$

 fin Tant que

$T[j] \leftarrow \text{tmp};$

fin pour

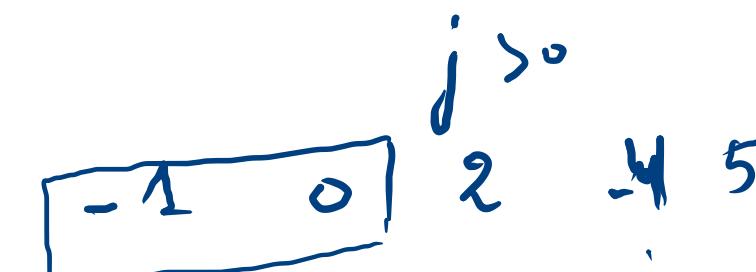
fin



$\text{tmp} = 0$
 $j \geq 0$ mais $T[0] < T[1]$



i
 $\text{tmp} = 2$



i
 $j > 0$



$\text{tmp} = -4$

Recherche d'un élément dans Tableau

- sequenziel
- Dichotomique (le tableau doit être trié)
algorithme:
trouve \leftarrow faux

Pour $i \leftarrow 0 \rightarrow N-1$ faire

 si ($T[i] = x$) alors

 trouve \leftarrow vrai

 fin si

fin pour

sequenziel