

Les algorithmes de tri

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

<
>

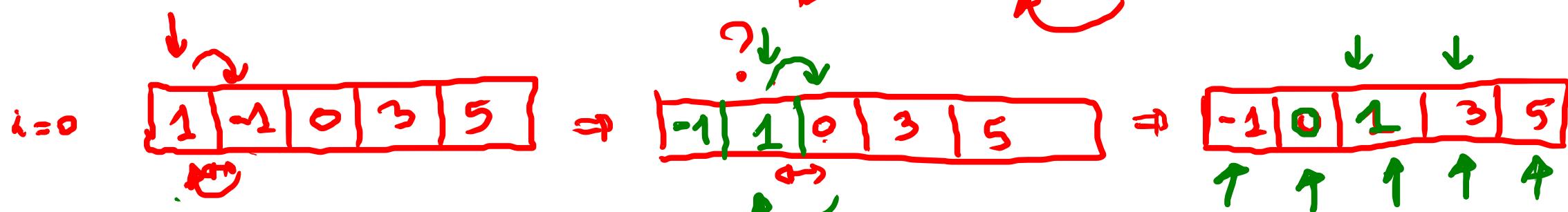
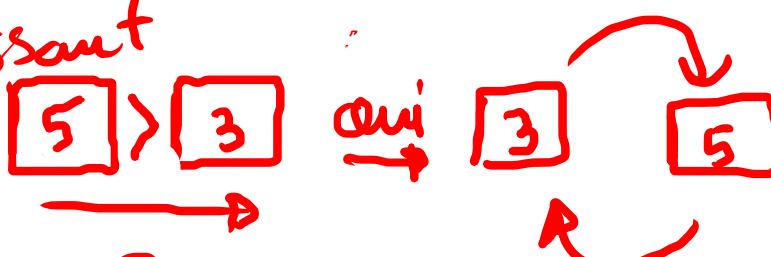
1	-1	0	3	5
---	----	---	---	---

↓ tri par ordre Croissant

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

- parcour
- comparaison
- permutation / échange

le tri à bulle : (ordre croissant)



il y avait ⁱ 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

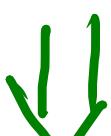
echange	i	$T[i]$	$T[i+1]$
faux	0	-2	-4
vrai	1	-5	3
"	2	3	8
"	3	8	10
"	4		

↗

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 \neg

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

until ($\xrightarrow{\text{échange = faux}}$)

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 échange;

do {

échange = 0;

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

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

échange = 1;

tmp = t[i];

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

t[i+1] = tmp;

} while (échange);

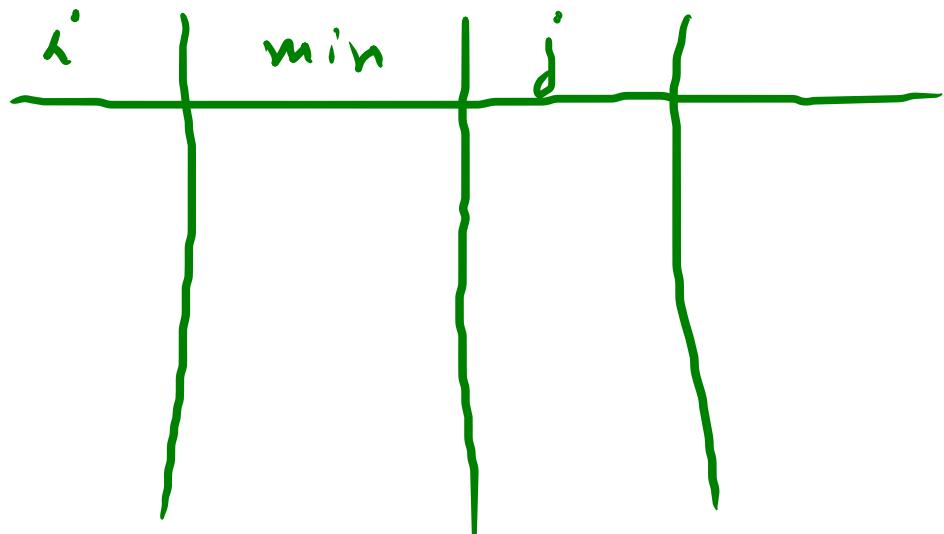
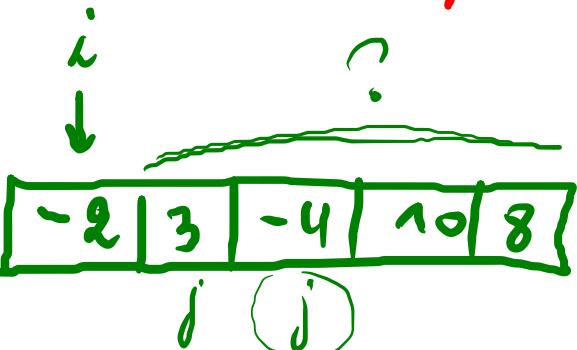
// affichage après échange

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

printf ("%d", T[i]);

Tri par sélection

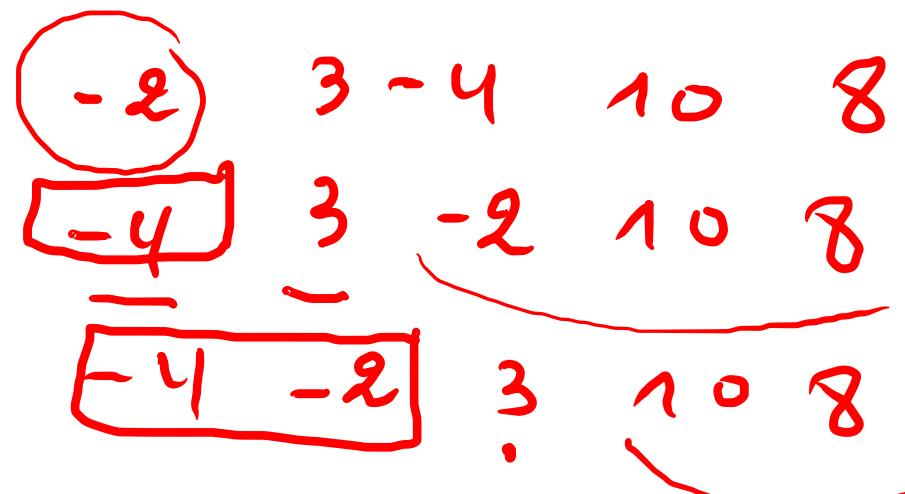
min.



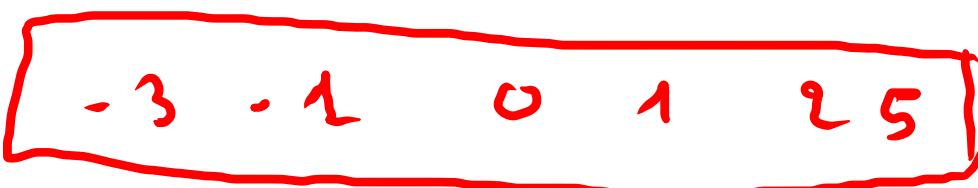
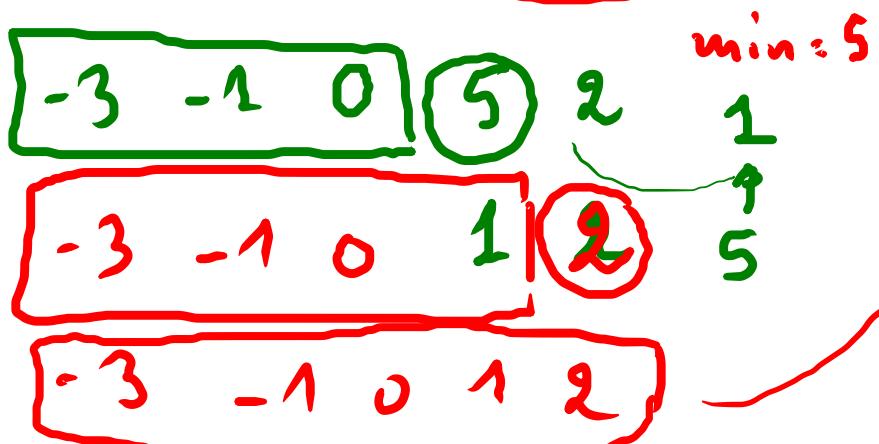
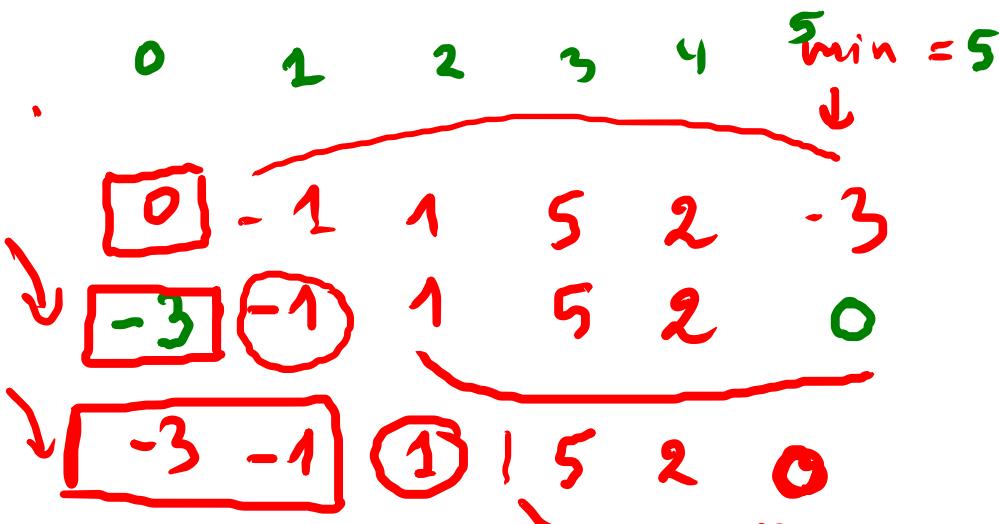
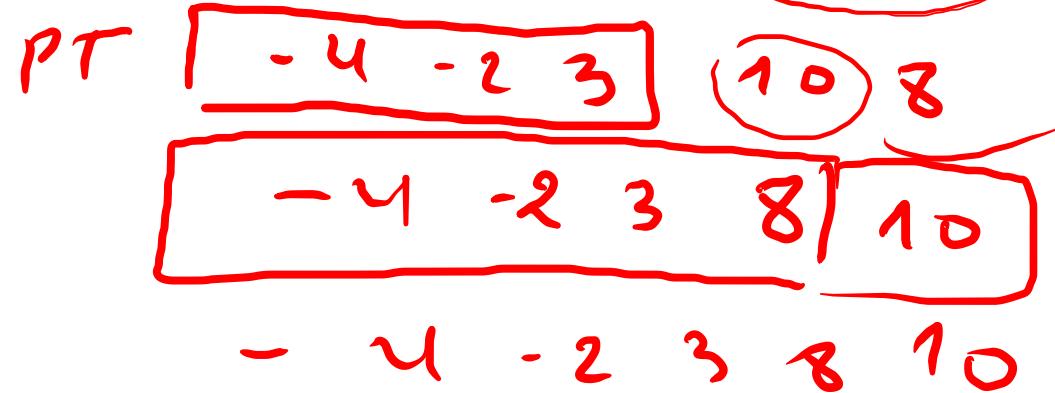
$i = 0$

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

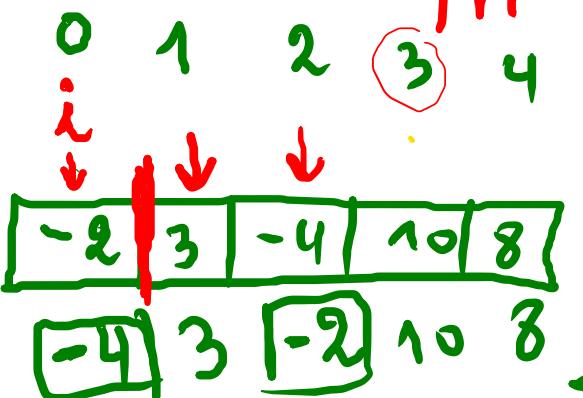
|
 | sélection
 | échange



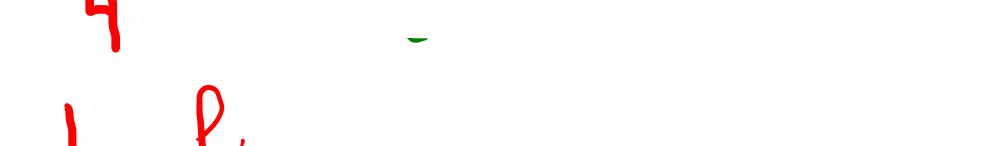
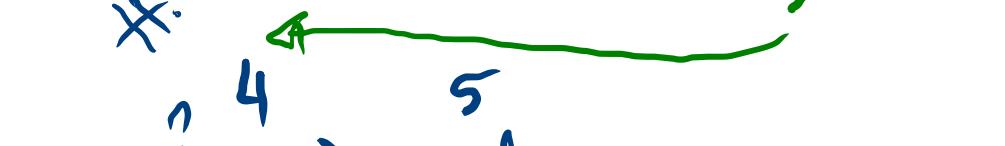
partie triée



Tri par sélection



min ≠ i → no swap

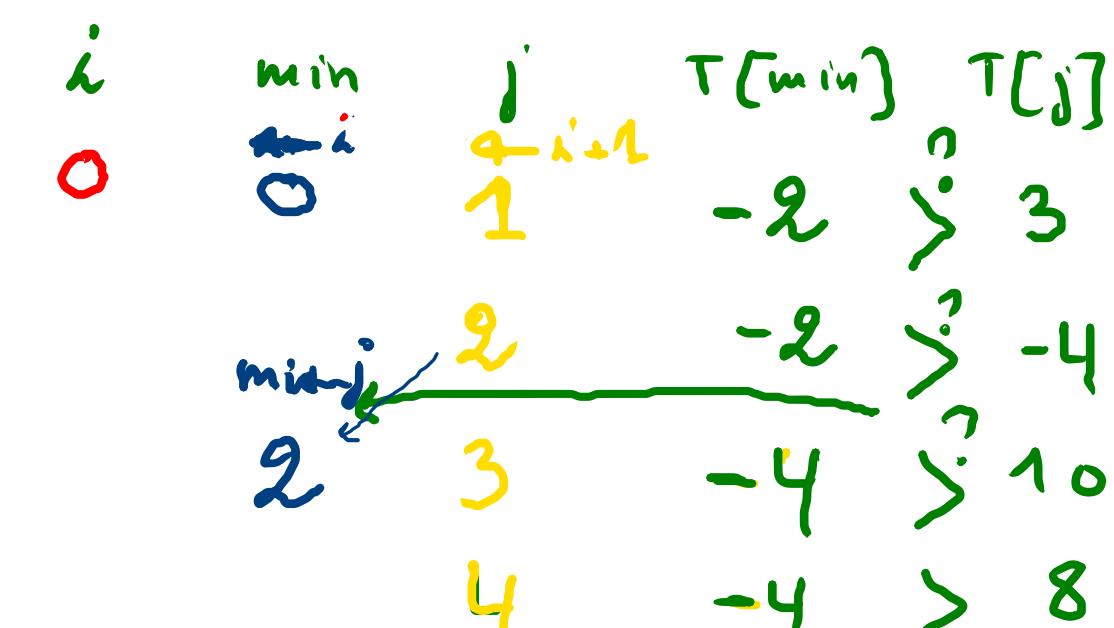


Algorithm steps:

- $i \leftarrow \text{min} - i$
- $j \leftarrow i + 1$
- selection du min
- échange

sélection *

échange *



swap ($\min \neq i$)

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

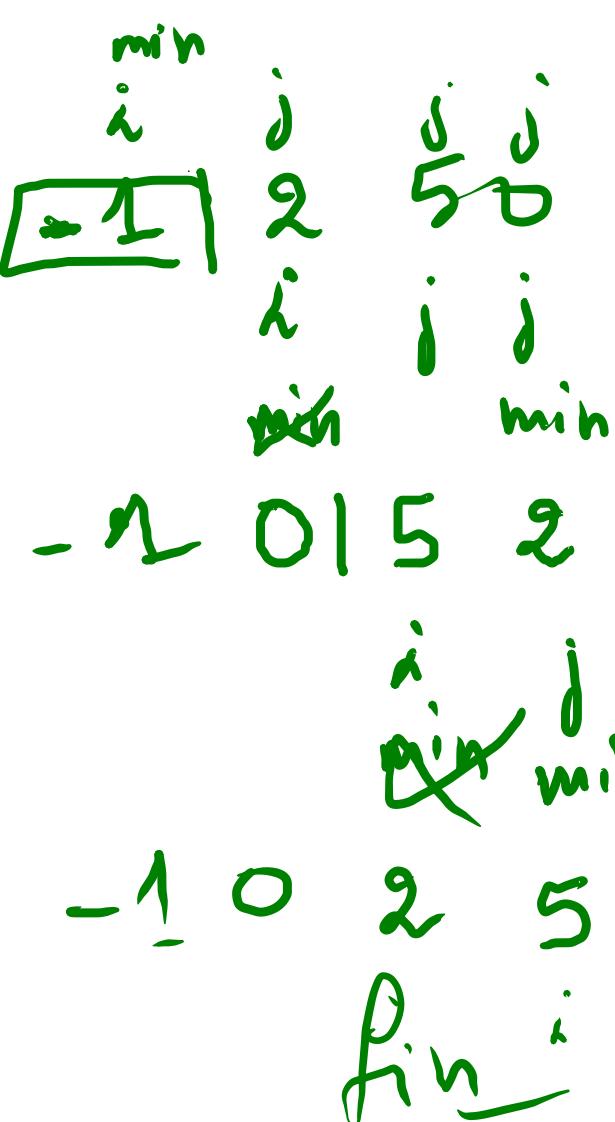
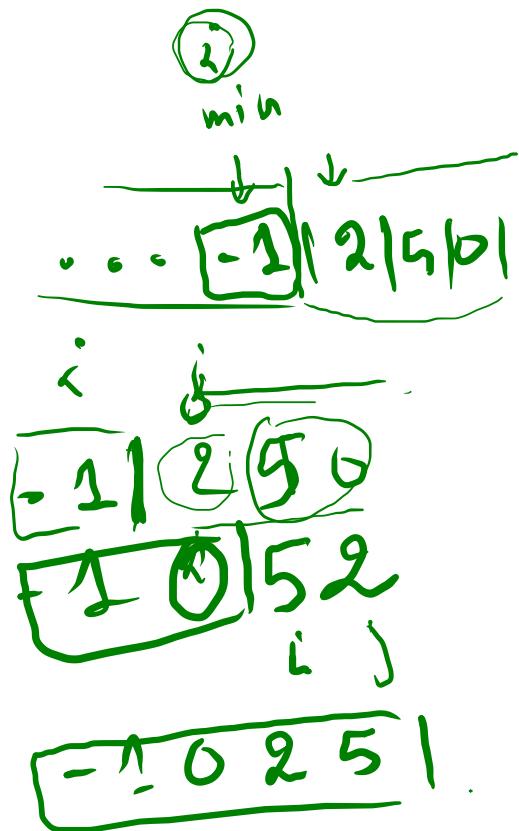
$T[i] \leftarrow T[\min]$

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

1 2 3 > -2

2 3 -2 > 10

4 -2 > 8



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

Début
 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 fin si
 Si ($\text{min} \neq i$) alors:
 $\text{tmp} \leftarrow T[i];$
 $T[i] \leftarrow T[\text{min}];$
 $T[\text{min}] \leftarrow \text{tmp};$
 fin si
 Fin Pour
 Fin "affichage après"

Tri par sélection

Algorithm : triSelection
temp, 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:
 temp ← T[i];
 T[i] ← T[min];
 T[min] ← temp;
 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 temp, 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) {  
            temp = t[i];  
            t[i] = t[min];  
            t[min] = temp;  
        }  
    }  
}
```