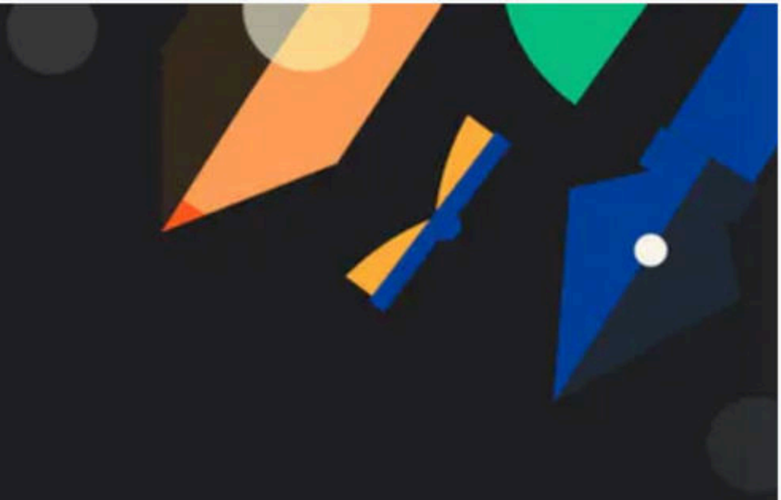


Sorting Techniques

Foundation Course on Data Structures & Algorithm - Part I

Love Babbar • Lesson 13 • May 10, 2022

→ Sorting tech :-

Sorting → what -'?

asc → 1 3 5 7 →

desc → 7 6 5 4 3 →

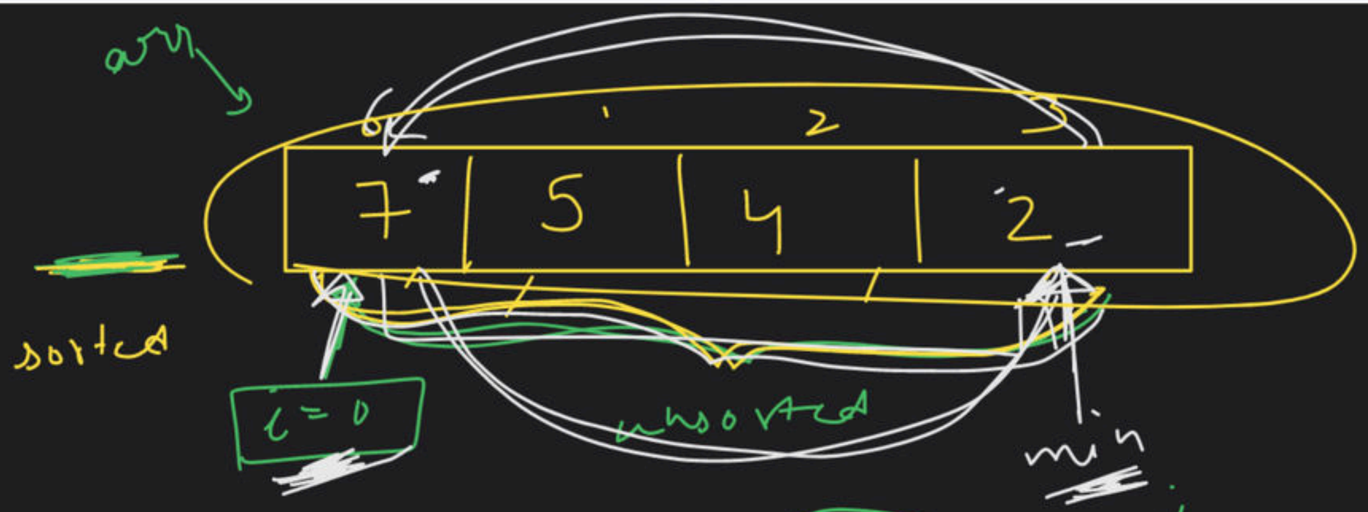
→ Selection Sort → ? → array → 2 part

sorted

left

unsorted

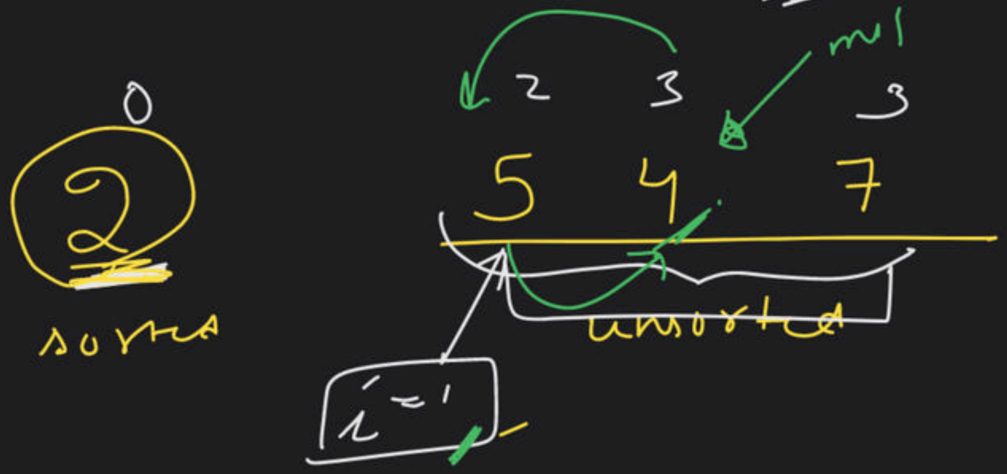
right



→ ascending order

shortest right place

Round 1



i^{th} Round

↳ i^{th} smallest

↳ right place

Round 2

2 4

sorted

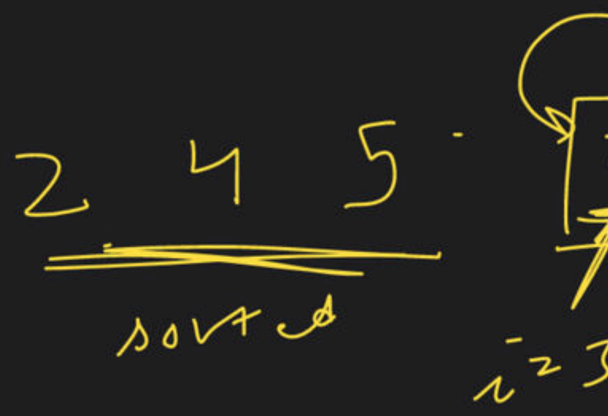
5 7

unsorted

per each value

Round 3

Round 4



2 4 5 7

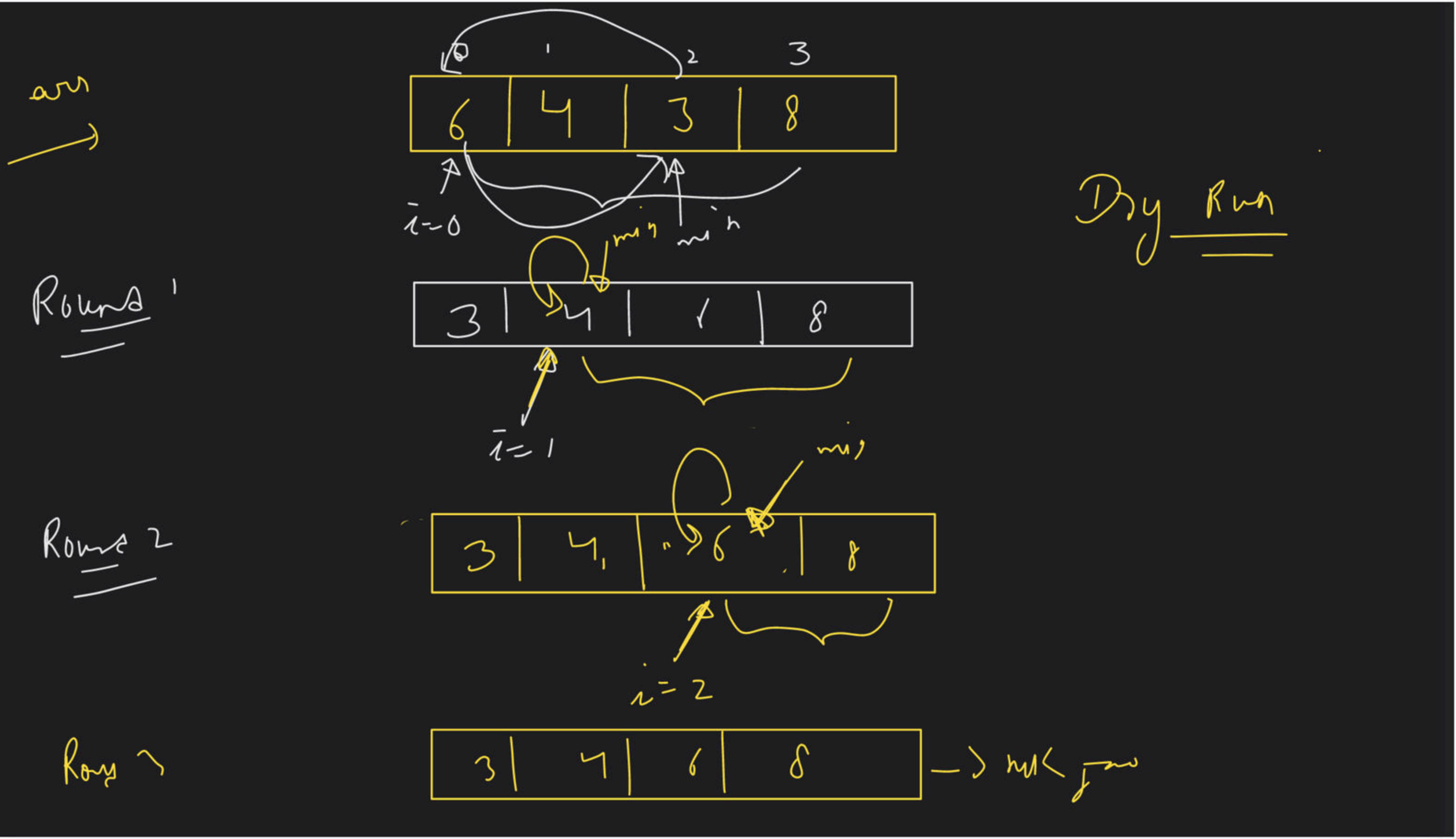
sorted

4
3
2
1

$1+2+3+\cdots+n$

$= n\left(\frac{n+1}{2}\right)$

$= \frac{n^2+n}{2} = O(n^2)$


arr
0 1 2 3
6 4 3 8
i=0
min
min
Dry Run
Round 1
3 4 6 8
i=1
min
Round 2
3 4 6 8
i=2
Round 3
3 4 6 8
-> sorted

→ Code

Outer loop

```
for ( int i = 0; i < n; i++ )
{
    int minIndex = i;
    // find min element
    for ( int j = i+1; j < n; j++ )
    {
        if ( arr[minIndex] > arr[j] )
            minIndex = j;
    }
    swap ( arr[i], arr[minIndex] );
}
```

min Element search

T.C -> $O(n^2)$

Bhatija algo

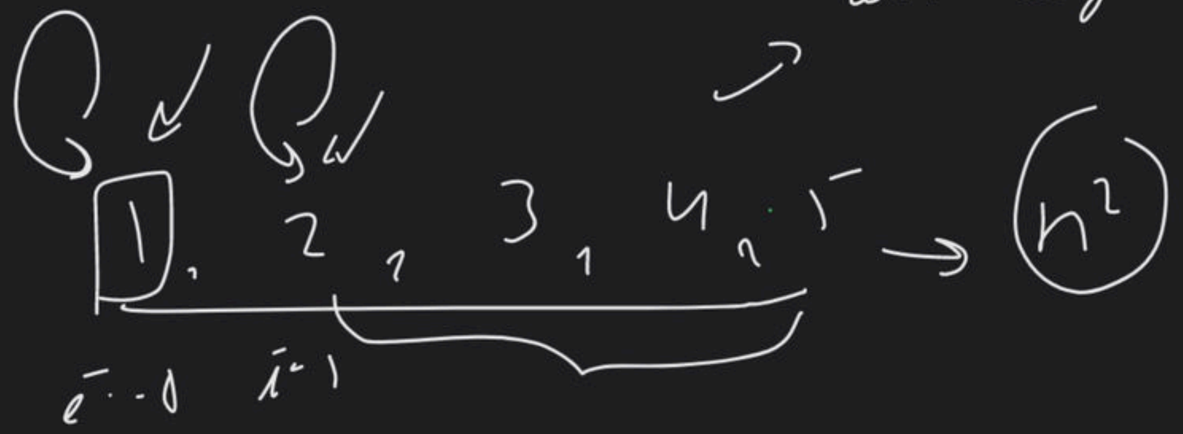
S.C -> $O(1)$

→ Space constraints → S.S

→ small array →

int

already



But Case →

Worst Case →

→ $O(n^2)$

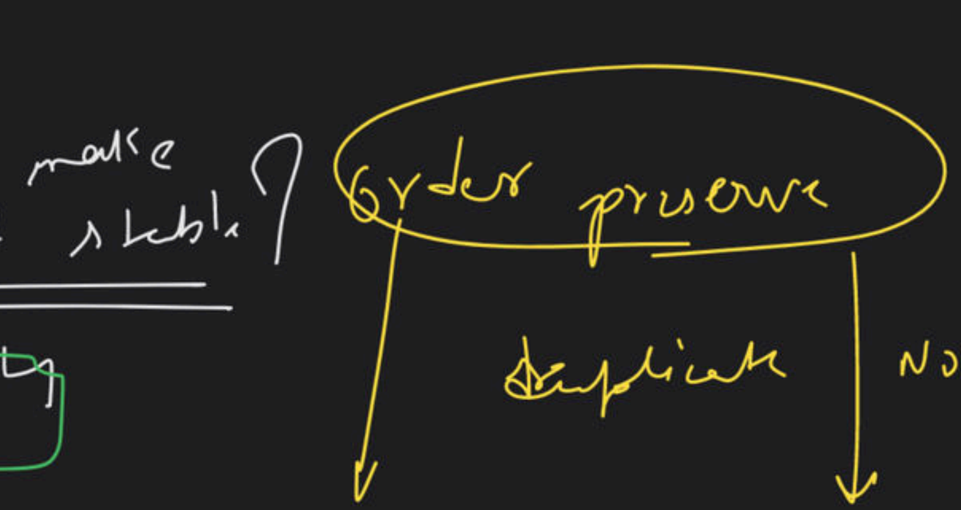
reverse sorted

H/W

↳

Selection sort → Stable (or) Not → ? → DRY RUN → Ex

→ Stable algorithm :- → How to make it stable?



i/p →

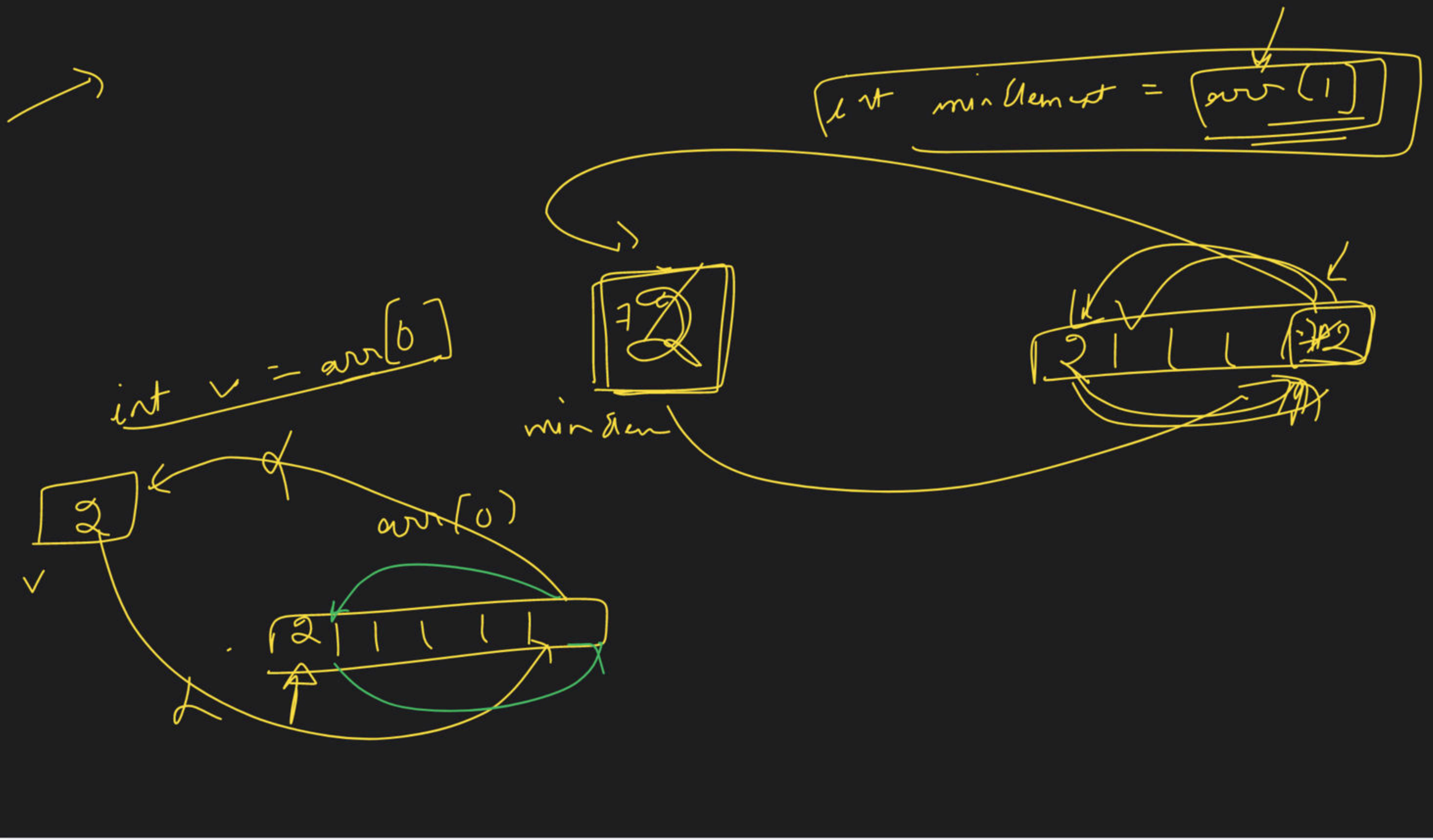
sort

2 4 5 7 7

∴

It 2 + 3 - - - - n

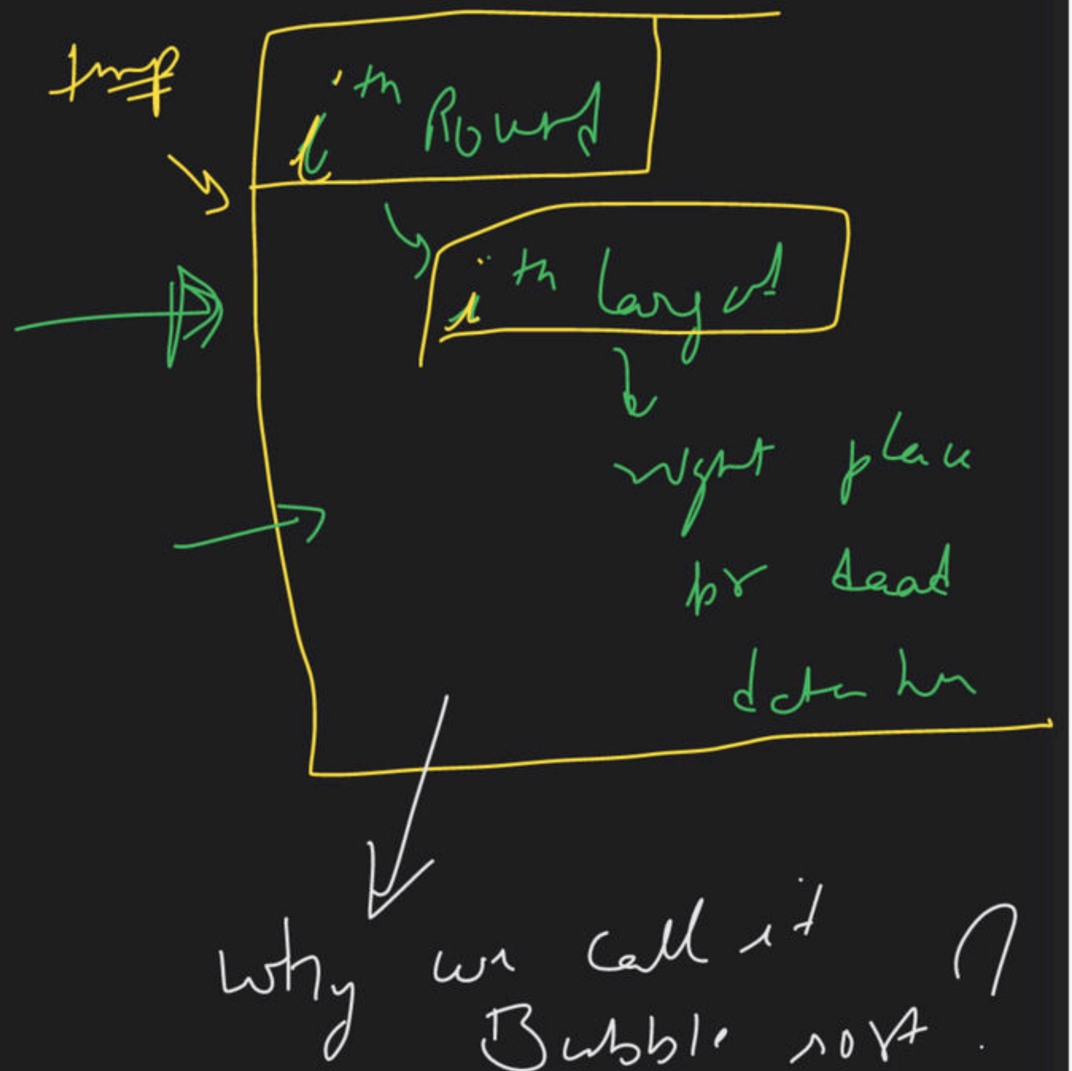
$= \frac{n \times (n+1)}{2} = \frac{n^2 + n}{2} = O(n^2)$


int minElement = arr[1]
int v = arr[0]
minElem
arr[0]
v

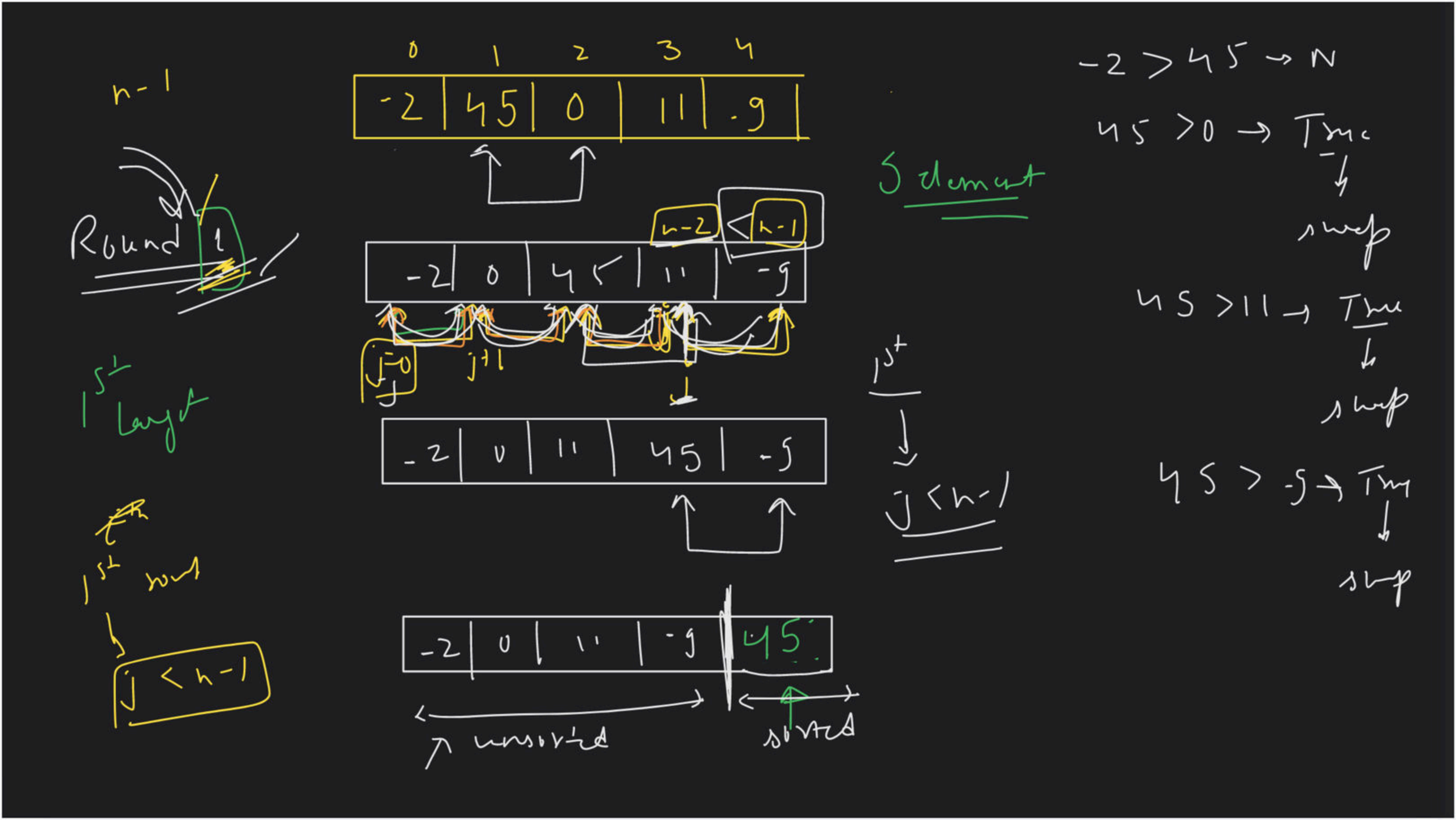
DRY-RUN

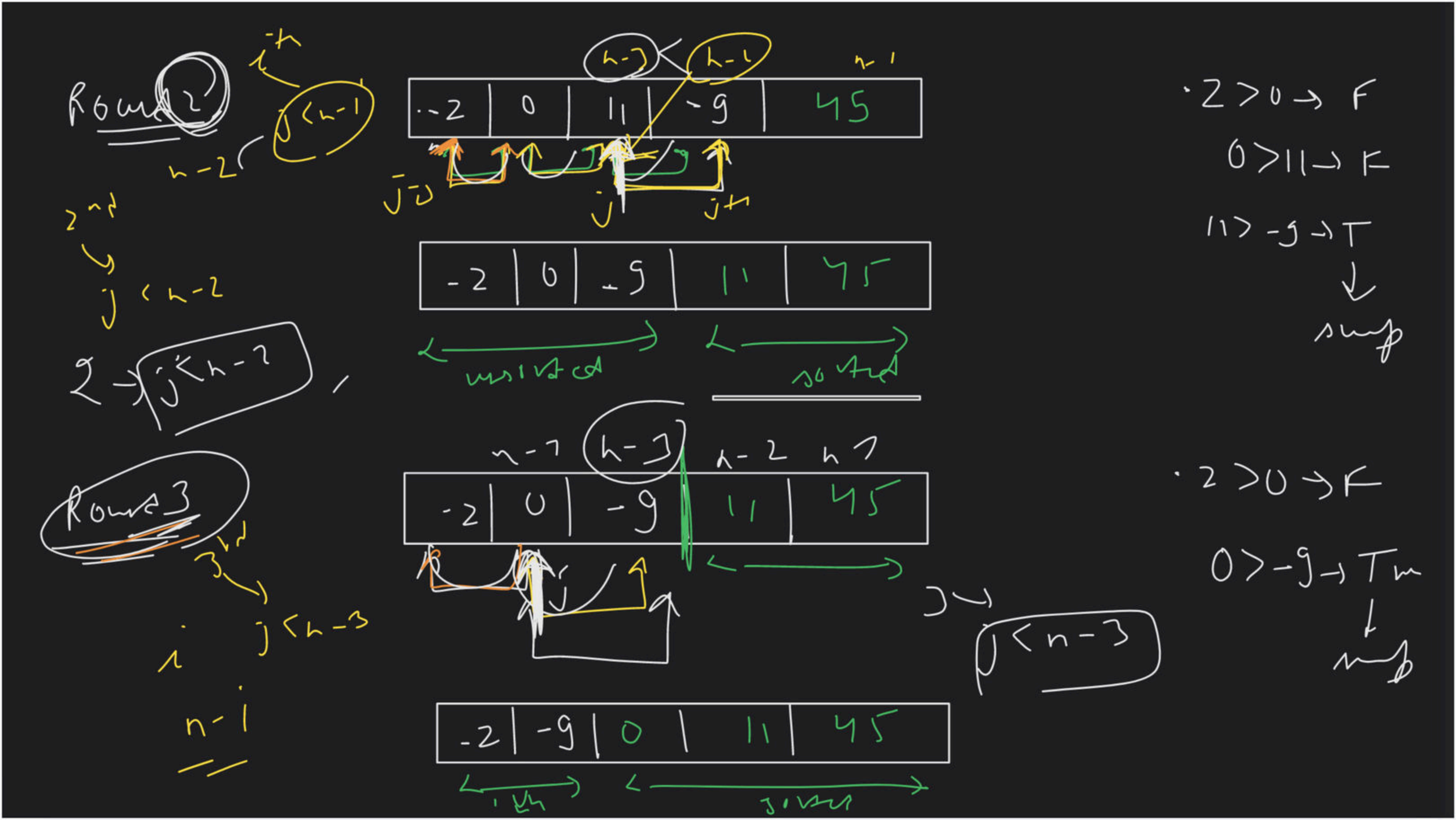
Bubble Sort → largest element → right place

i^{th} Round
↳ i^{th} largest
↳ right place
pr aajaega

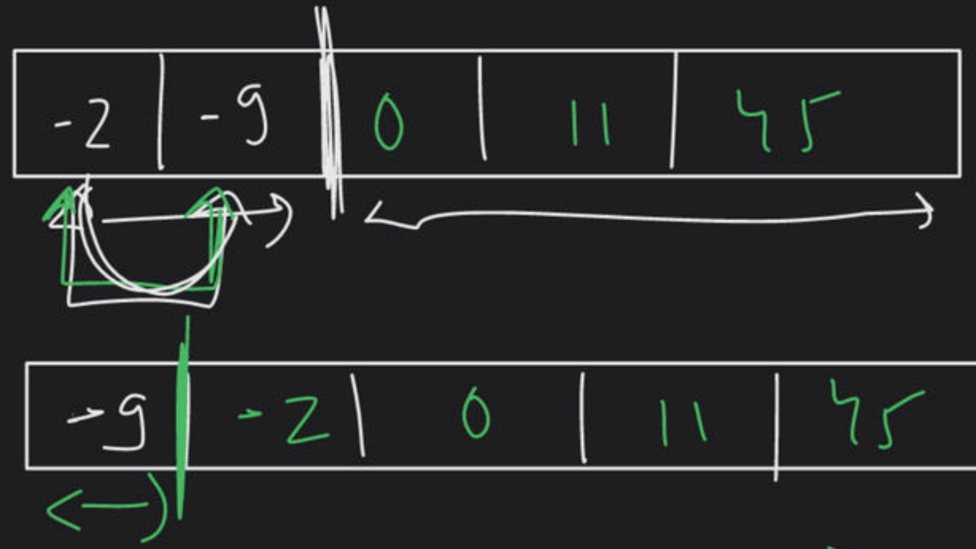


Why we call it Bubble sort?


n-1
Round 1
1st Largest
1st Round
j < n-1
0 1 2 3 4
-2 45 0 11 -9
n-2 < n-1
-2 0 45 11 -9
j=0
j+1
-2 0 11 45 -5
-2 0 11 -9 45
unsorted
sorted
5 element
1st
j < n-1
-2 > 45 → N
45 > 0 → True
swap
45 > 11 → True
swap
45 > -5 → True
swap


Round 2
Round 3
-2 | 0 | -9 | 11 | 45
-2 | 0 | -9 | 11 | 45
-2 | -9 | 0 | 11 | 45
2 > 0 → F
0 > -9 → T
-9 > 11 → F
swap
2 > 0 → F
0 > -9 → T
swap
j < n-2
j < n-3
j < n-3

Round 4



-2 > -9 → 7

sorted o/p

T.C → Best Case → already sorted → O(n) → why? → How?

1 2 3 4 5

Worst Case → reverse sorted → $O(n^2)$

code

```
for (int i=0 ; i<n-1 ; i++)
{
    bool swapped = false;
    for (int j=0 ; j<n-i-1 ; j++)
    {
        if (arr[j] > arr[j+1])
        {
            swap(arr[j], arr[j+1]);
            swapped = true;
        }
    }
    if (swapped == false)
        break;
}
```

Rounds

$j < n-i-1$

$j < n-0$

$j < n$

$j+1 <$

Trap

$j : i+1$

Optimised Bubble Sort

☆ DRY RUN

☆ Practice → syntax comfortable

$i = 0$

5

j

$j < n-i-1$

$j < n-0-1$

$j < n-1$

$j <= n-i$

$j <= 5-0$

$j <= 5$

1st → $j < n-1$

2nd → $j < n-2$

3rd → $j < n-3$

i^{th} → $j < n-i-1$ → 0-based indexing

Stable ⇒ ? → YES → How → DRY RUN

In-place → not using extra space

algo →

If ✗

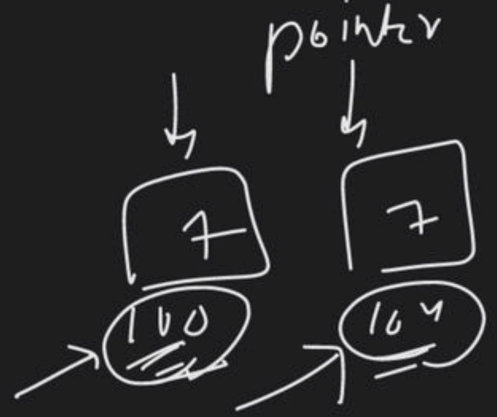
Yes

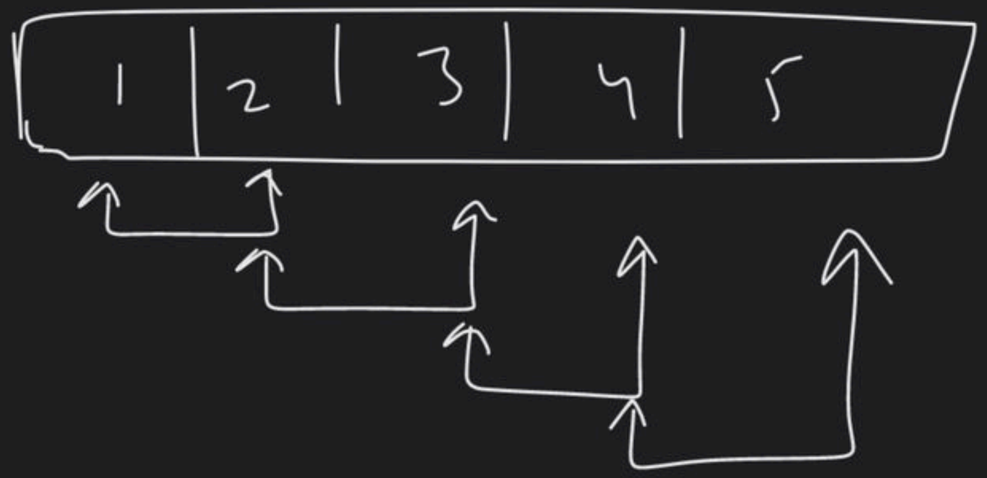
flag

bool =

B.S > S.S

B.C → O(n) O(n²)





Round 1

(n-1) comparision

O(n)

swapped = false

break

→ S.S

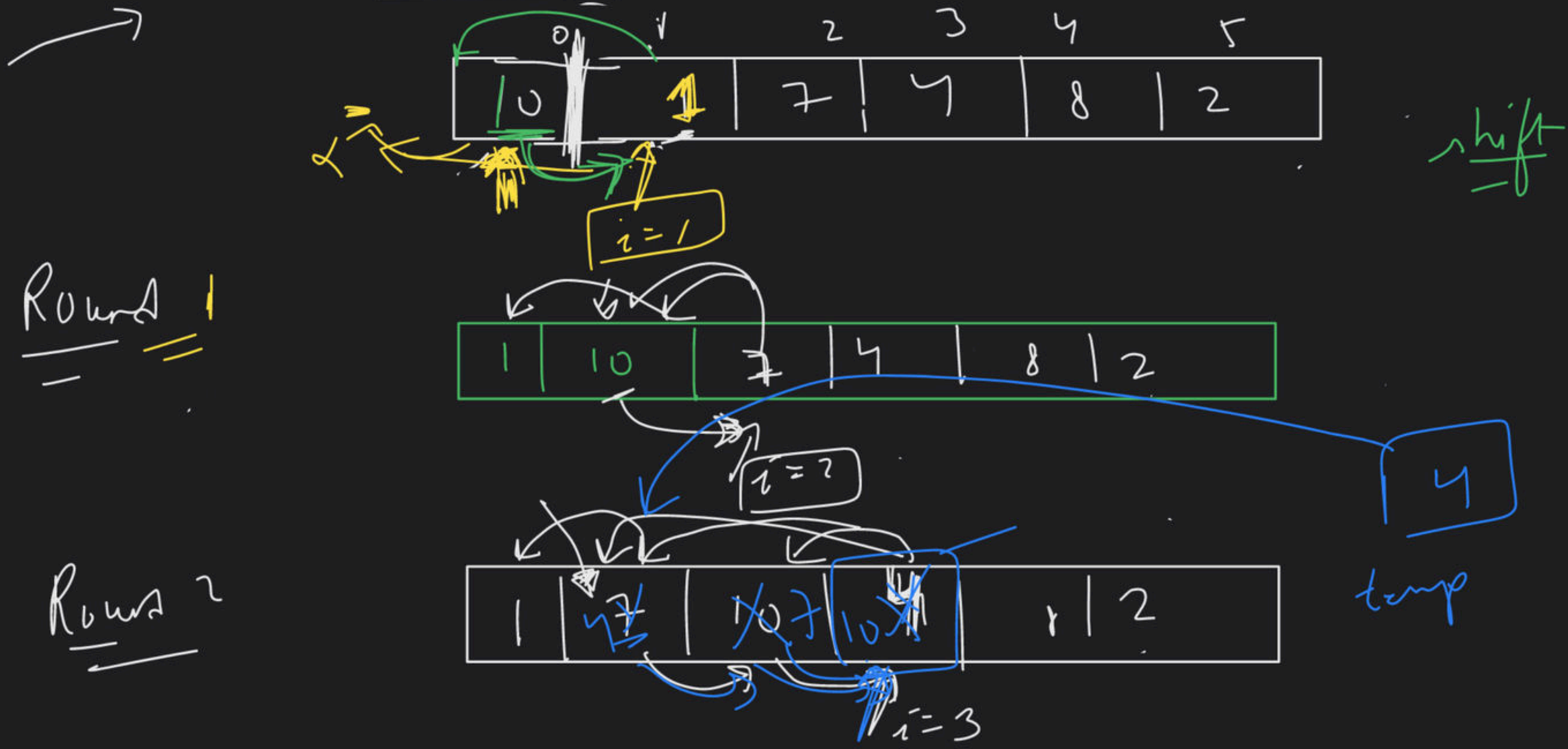
→ B.S

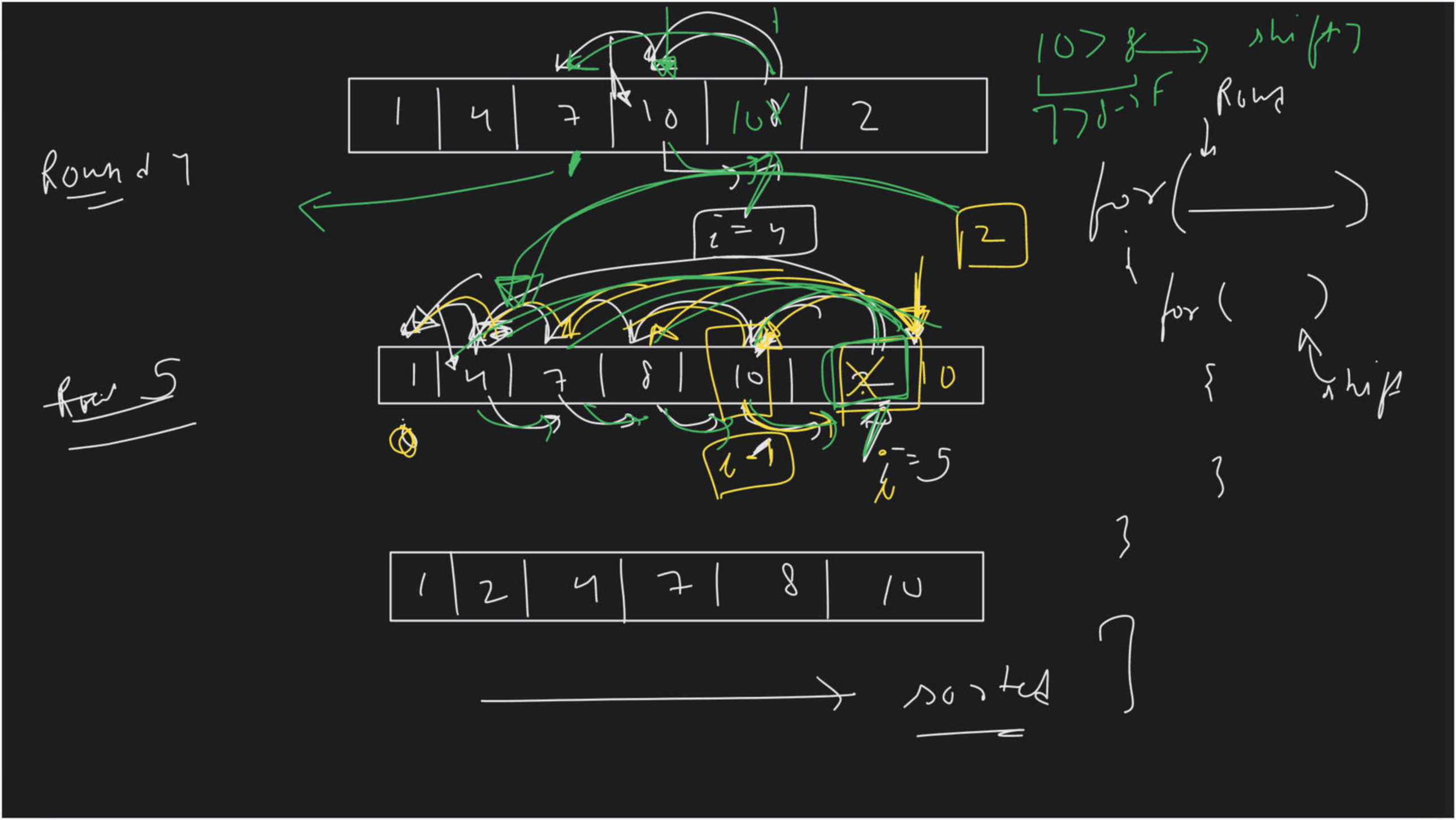
→ Yes or NO

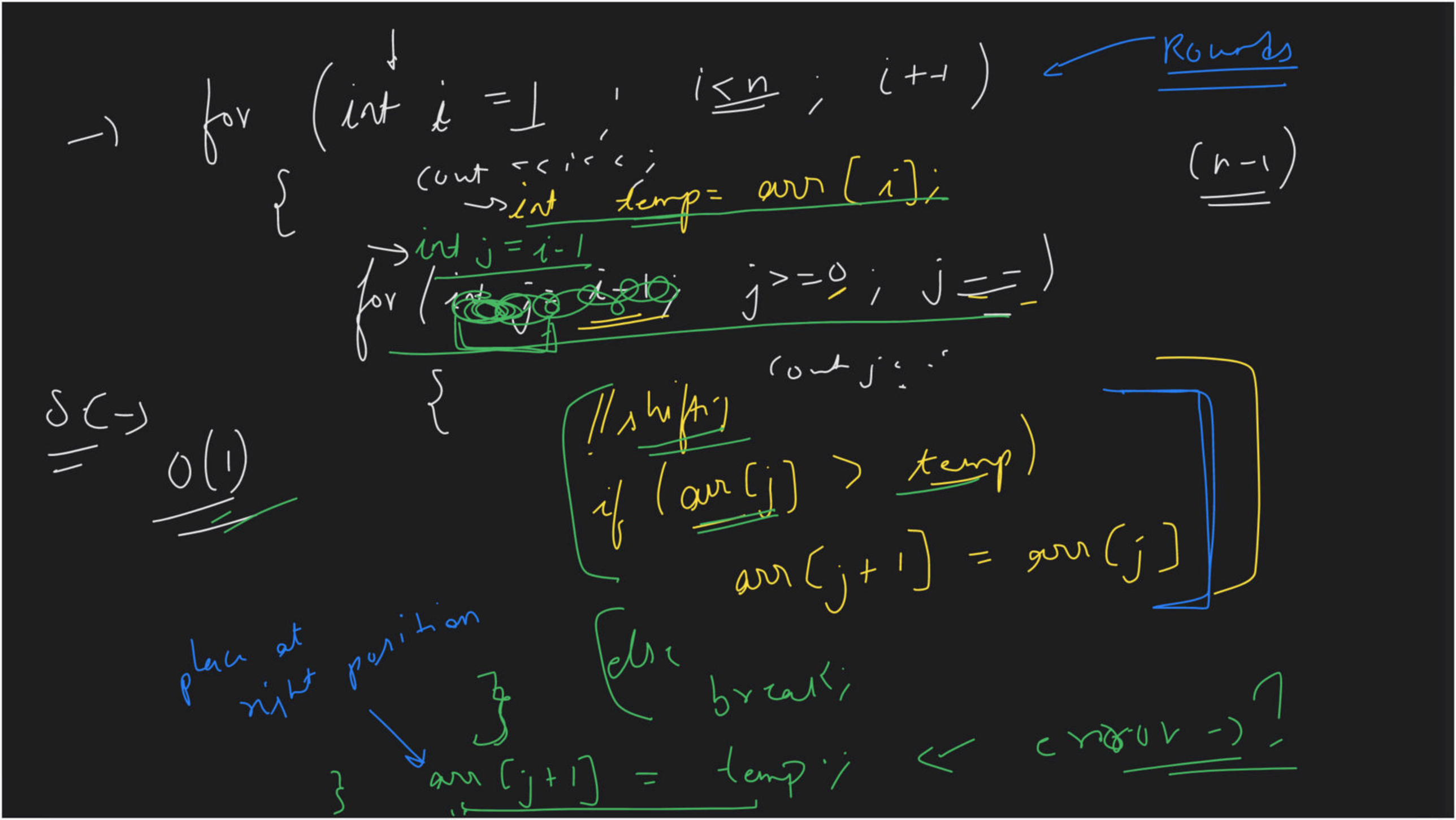
→ Repeat

→ Day Run → 2/3 La

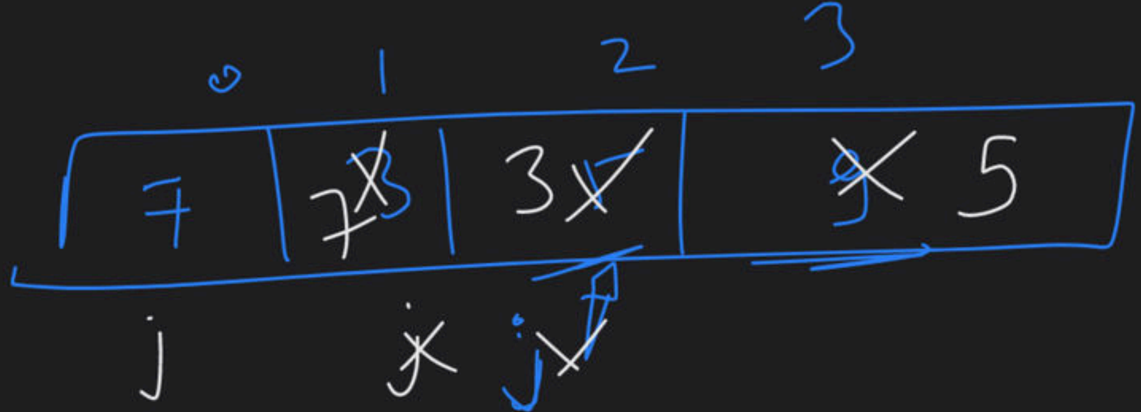
Insertion Sort :->




Round 7
for 5
1 | 4 | 7 | 10 | 108 | 2
i = 4
2
1 | 4 | 7 | 8 | 10 | 10
i - 1
i = 5
1 | 2 | 4 | 7 | 8 | 10
sorted
10 > 8 → shift
Round
for ( )
for ( )
{
}
shift
}


for (int i = 1 ; i<=n ; i++)
Rounds
(n-1)
{
cout << i << ;
int temp = arr[i];
int j = i-1
for (; j>=0 ; j--)
{
cout j ;
//shifting
if (arr[j] > temp)
arr[j+1] = arr[j]
else
break;
}
arr[j+1] = temp;
error ?
O(1)
place at right position

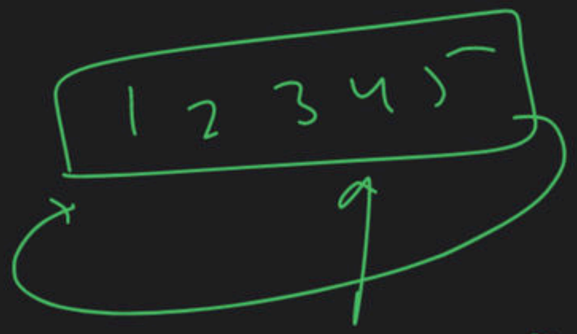
shift

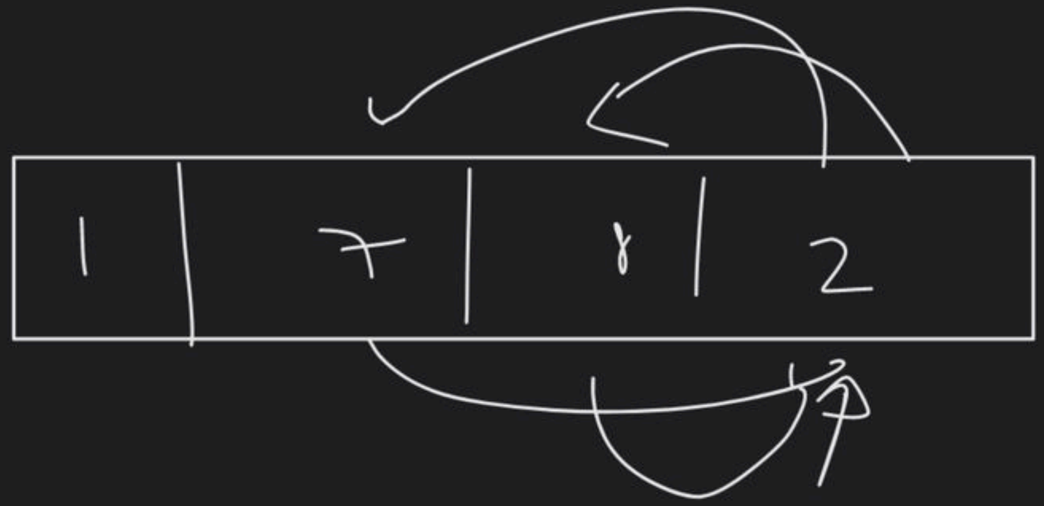


why not swap

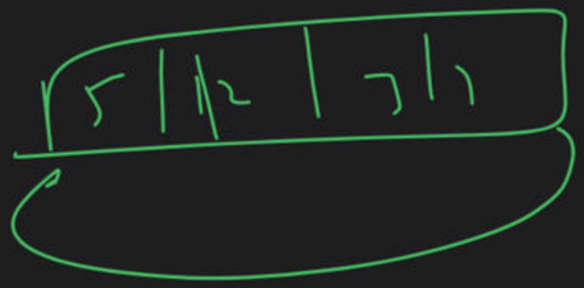
arr[j+1] = arr[j]

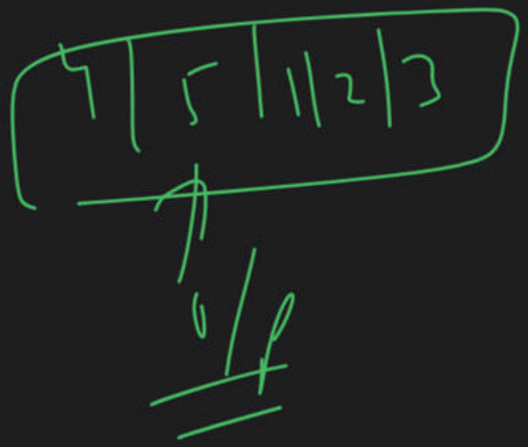
j--





B:K





1 2 8 7

1 7 2 8

cyclically rotate arr by 2

$i = 1$

$j = (i-1) \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow 2$

$3 \rightarrow 3$

$(n-1) \rightarrow (n-1)$

$O(n^2)$

T.C

Java in 1 Video

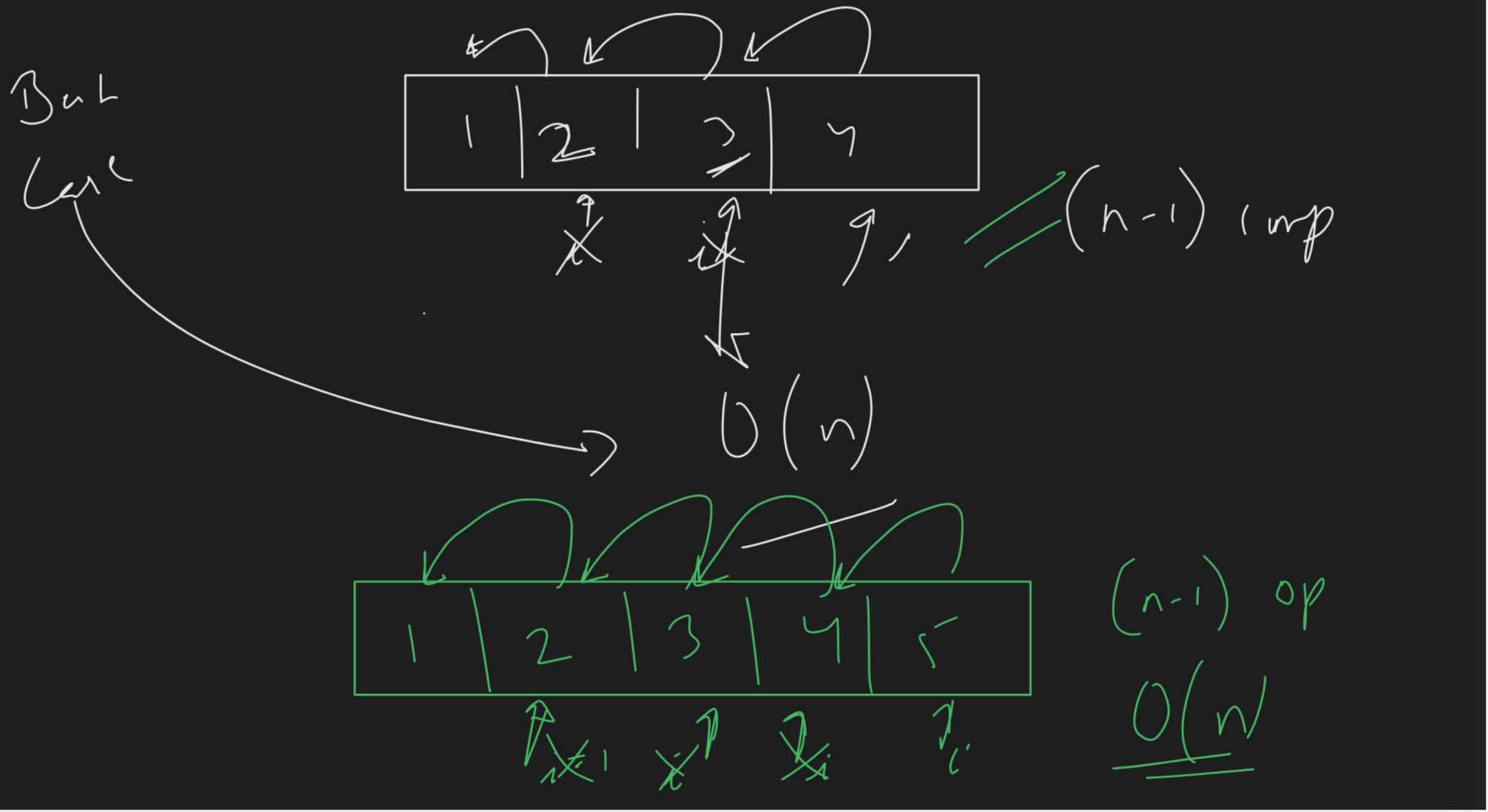
Search sorting

La

STL

IDE

Collection framework


But
Can't
1 2 1 3 4
O(n)
(n-1) comp
1 2 3 4 5
(n-1) op
O(n)

T.C ⟶ Best Case -> O(n)

Worst Case -> $O(n^2)$

↳ Junk

1900

1900 → MF

S.C -> O(1)

Stable → ? Y → How (Dry Run)

↳ 19
15 11

In-place -> ? → How → Yes

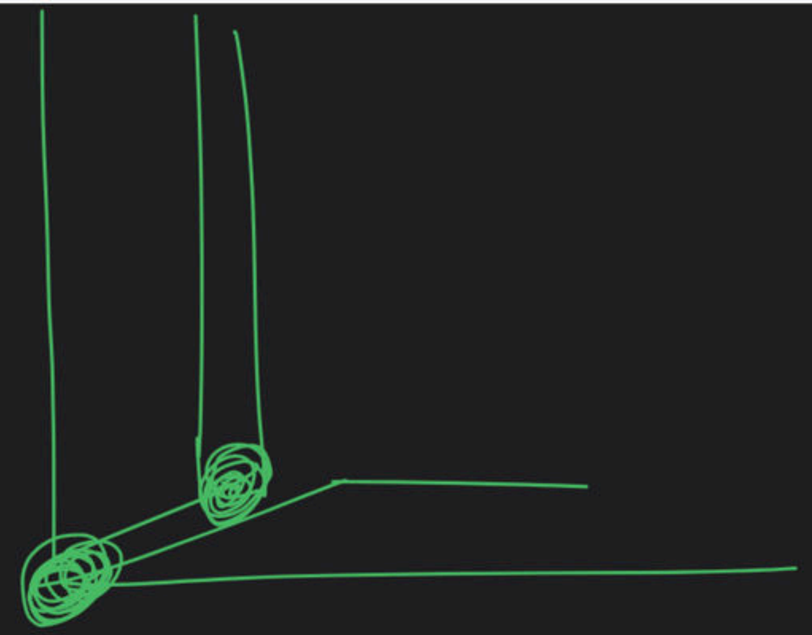
191

KRUD

(compensati

T-C

	B.C	W.C
S.S	$O(n^2)$	$O(n^2)$
B.S	$O(n)$	$O(n^2)$
I.S	$O(n)$	$O(n^2)$



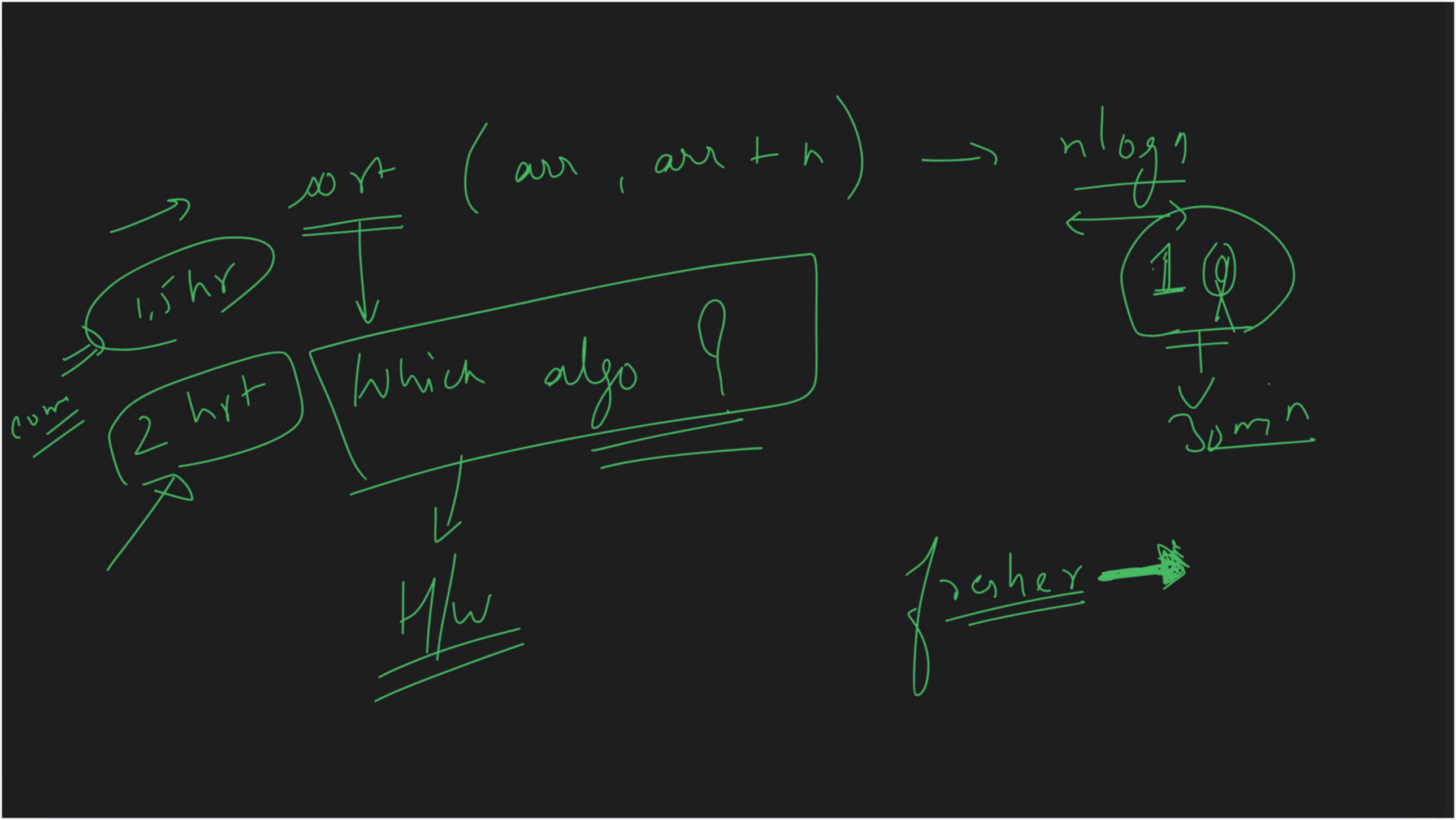
B.S. → Quant.

Time pass

Get job

DSA + Project → Accha hai → Daily Tick ✗

Devguy


sort (arr, arr + n) → n log n
1,5 hr
2 hrs
com
which algo ?
H/w
1
0
3 sum n
fresher