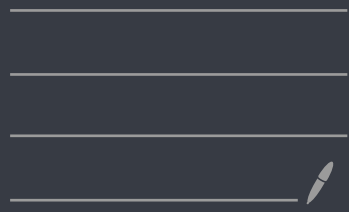
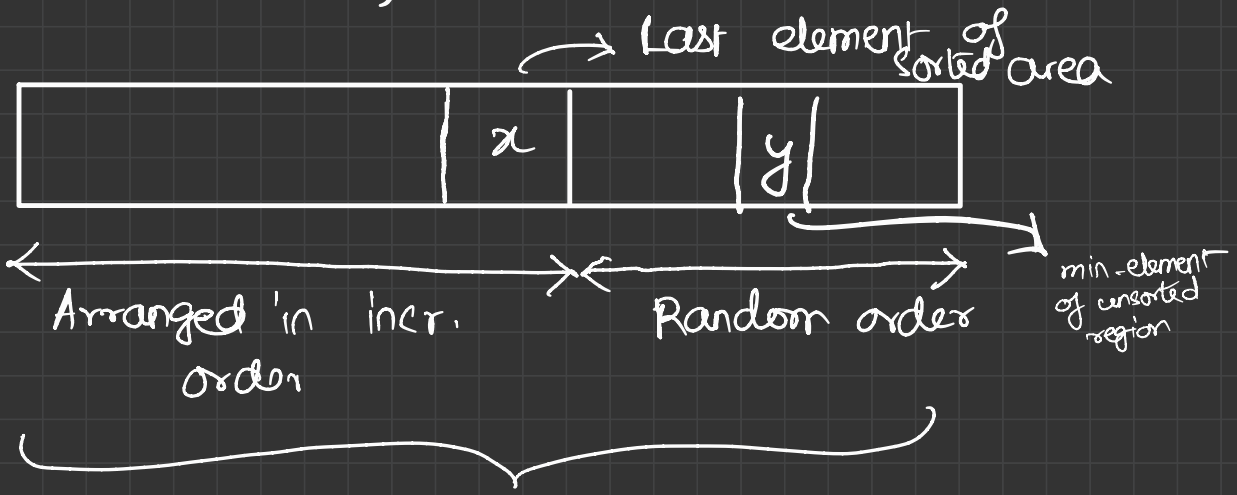
SELECTION SORT



[15, -1, 3, 8, 2, 6]



* Split into two parts, where first half is sorted, second half is not.
* Last element of sorted region x is less than or equal to the min_element of unsorted region.

$x \leq y$

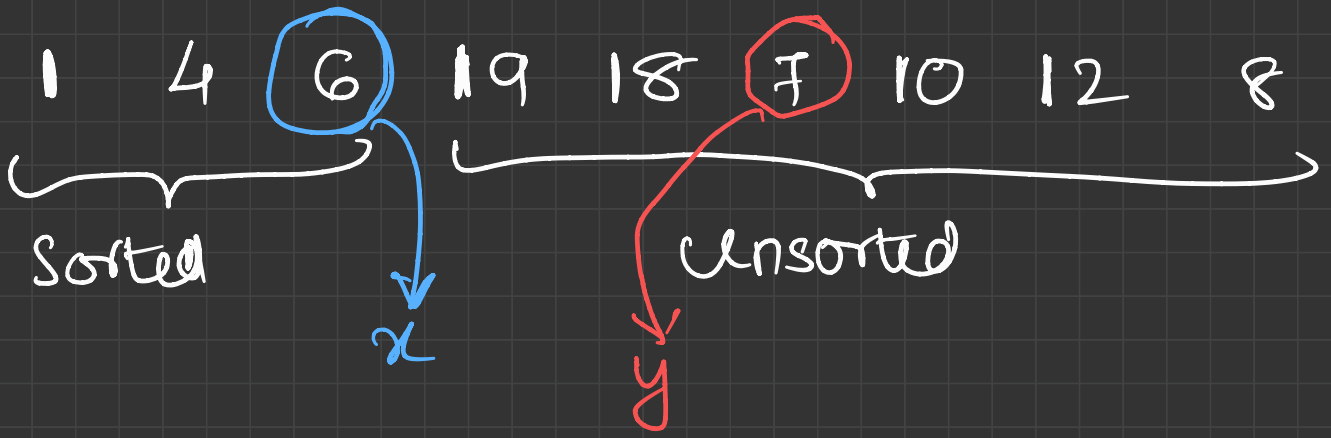
where $x \in$ sorted region,

$y \in$ unsorted region.

How can we extend the sorted region?

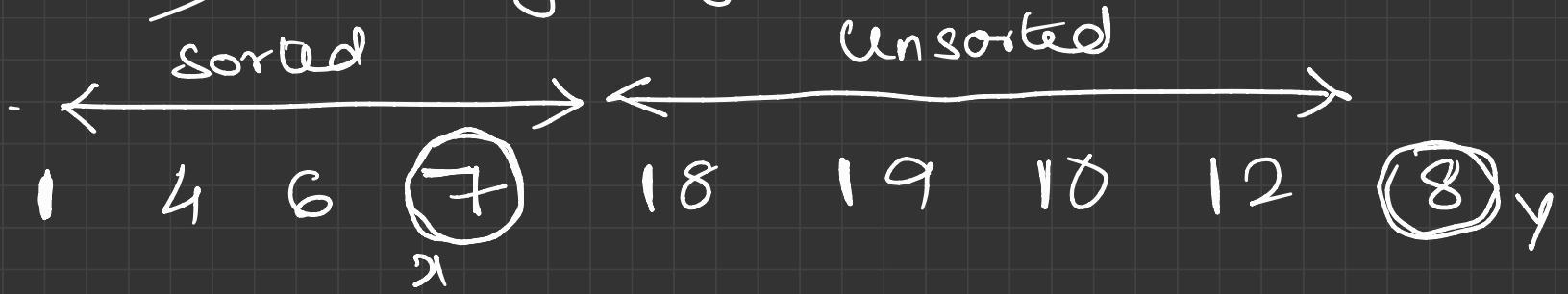
x is the largest element of sorted area.

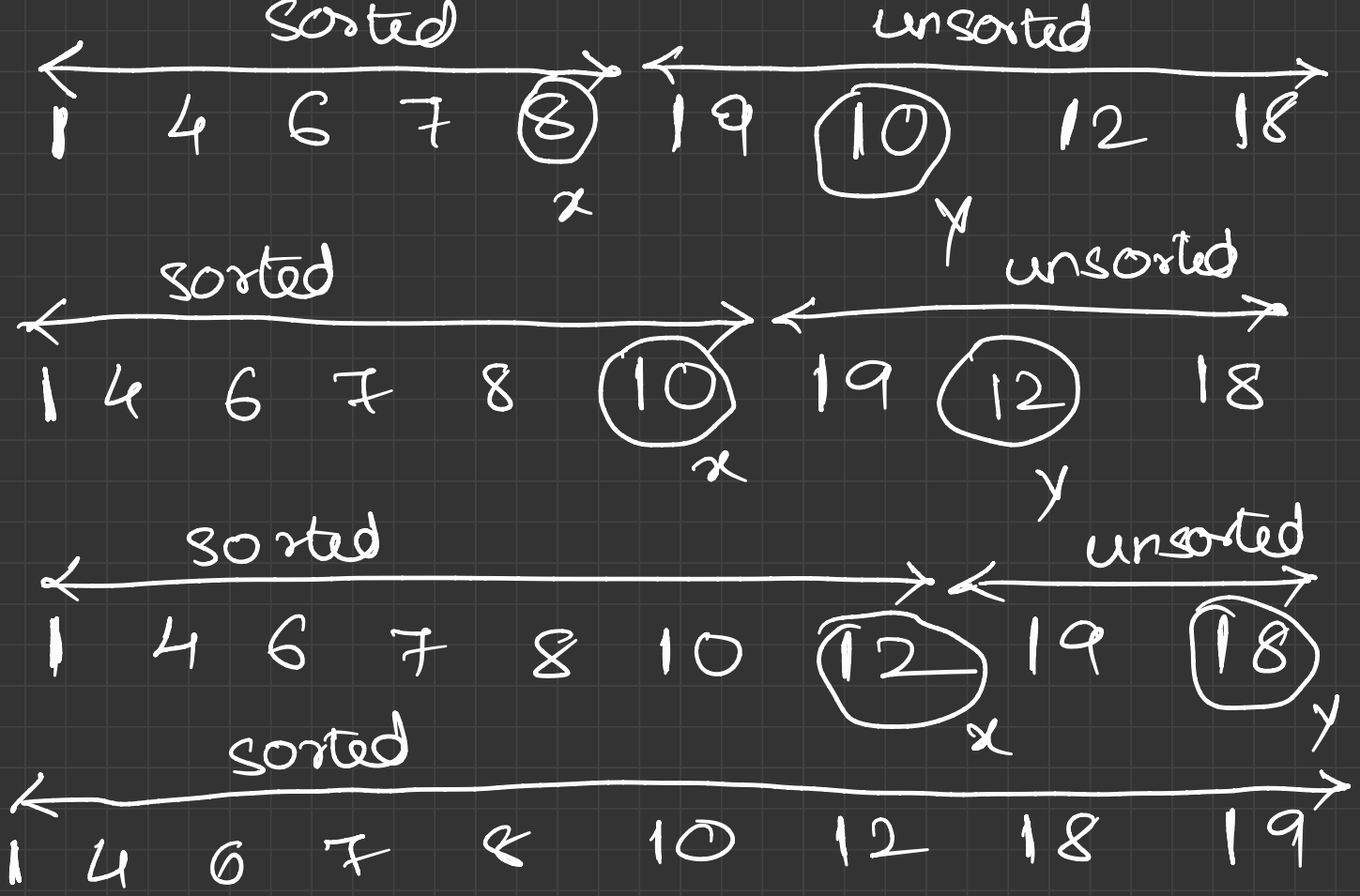
y is the min-element in unsorted region but bigger than x & all the elements in sorted region.



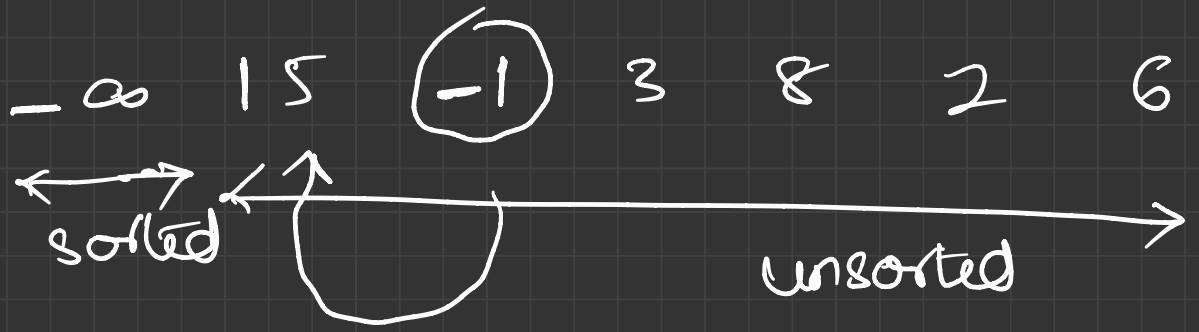
$x \leq y$

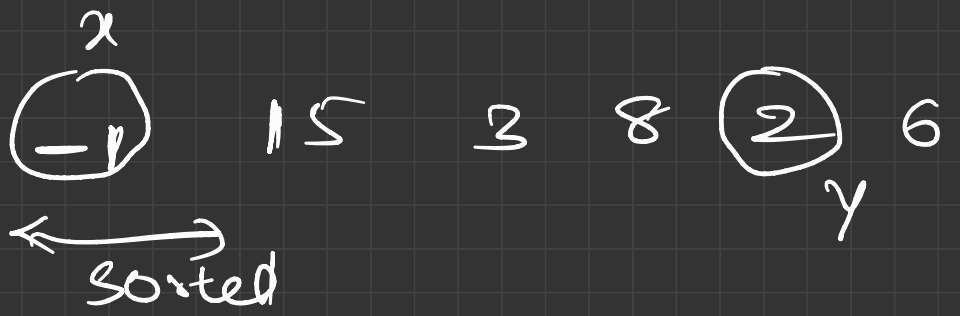
$\Rightarrow$ Put y after x.




sorted
unsorted
1 4 6 7 8 19 10 12 18
x
y
sorted
unsorted
1 4 6 7 8 10 19 12 18
x
y
sorted
unsorted
1 4 6 7 8 10 12 19 18
x
y
sorted
1 4 6 7 8 10 12 18 19

Initially,





After the first swap only you get the sorted region.

Dry Run :-

0	1	2	3	4	5	6
15	1	6	3	-1	2	10
i	j					

In every iteration, we will scan the whole unsorted region & get the min-element (index).

$i = 0$ $[i, n-1]$ → unsorted

$j \in [i+1, n-1]$ → gets the min.

If arr[j] <= arr[min_index]) {

min_index = j;

}

min_index = 4

Swap → min_index , i

0	1	2	3	4	5	6
-1	1	6	3	15	2	10
	i	j				

min_index → first element of unsorted region.

0	1	2	3	4	5	6
-1	1	6	3	15	2	10
		i	j			

Loop: min_index = 6 (i)

If (arr[j] <= arr[min_index])

min_index = j;

}

min_index = 3 ; swap → 1, min_index

0	1	2	3	4	5	6
-1	1	3	6	15	2	10

Repeat this algorithm until $i < n$.

CODE: (Pseudocode)

```
fn getMinElement(int[] arr, int start){
  //get minElementIdx from [start -> arr.length]
    int minEleIdx = arr[start];
    for(i = start+1 -> arr.length){
            if(arr[i] < arr[start]){
                minEleIdx = i;
            }
      return minEleIdx;
}
```

```
fn selectionSort(int[] arr){
//Loop through unsorted part, get MinEle & swap
  with the first unsorted element.

  for(i = 0 -> arr.length) {
      int minEleIdx = getMinElement(arr, i);
      if (i != minEleIdx) {
            swap(arr[i], arr[minEleIdx]);
      }

}
```

TIME COMPLEXITY:

$i = [0, n-1]$

getMinimumIndex → starts from i & goes to n-1.

$i=0 \rightarrow (n-1)$
$i=1 \rightarrow (n-2)$
$i=2 \rightarrow (n-3)$
$\vdots$
$i=n-1 \rightarrow 0$

$\longrightarrow \frac{n(n-1)}{2}$ (sum)

$\Rightarrow \frac{n^2}{2} - \frac{n}{2} \Rightarrow O(n^2)$

$\therefore TC = O(n^2)$

SPACE COMPLEXITY:

$O(1)$ → worst case

Stability

Two data items with the same value maintaining the same order after getting sorted.

- Selection Sort is not stable