


Bubble sort

1st round

largest element at right place

Round 1

i=0	1	2	3	4	5
10	1	7	6	14	9

↓
1st largest element

↓
sort

↓
placed at end

5 comparison

1 10 7 6 14 9

1 7 10 6 14 9

1 7 6 10 14 9

1 7 6 10 14 9

1 7 6 10 9 14

right place

Round 2

2nd largest element

↓
sort

4 comparison

1 7 6 10 9 14

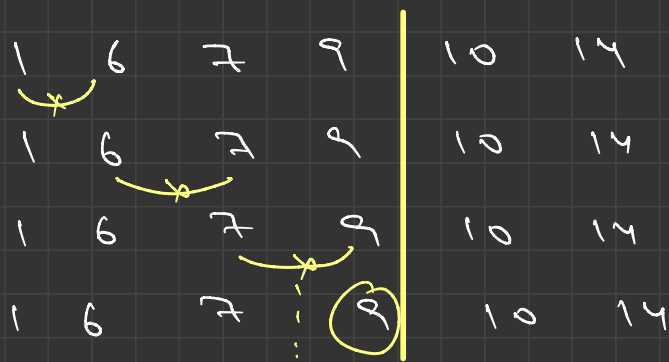
1 6 7 10 9 14

1 6 7 10 9 14

1 6 7 9 10 14

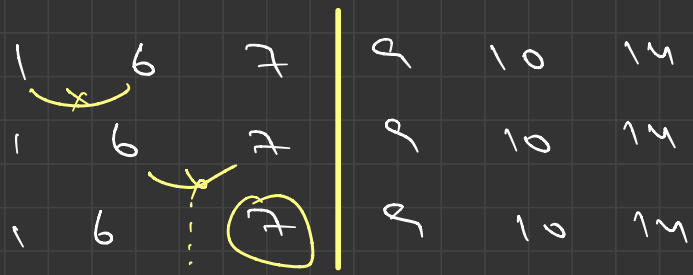
Round 3

3rd largest element
↓
sort
3 comparison



Round 4

4th largest element
↓
sort
2 comparison



Round 5

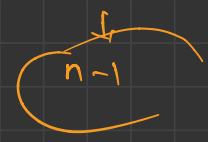
5th largest element
↓
sort
1 comparison



Round 6

→ single element left

↳ no need → already sorted in 5 rounds



total comparison = $5 + 4 + 3 + 2 + 1 = \frac{n(n-1)}{2}$
 $O(n^2)$

Entire array is sorted

Condition

if $a < b$ X

$a > b$ (Sweep)

Globals = $n-1$

i

0 \rightarrow $< (n-1)$

0 \rightarrow $< (n-2)$

0 \rightarrow $< (n-3)$

\vdots

u

1 \rightarrow $< (n-1)$

\vdots

sorted

$O(n)$

Time complexity $\Rightarrow O(n^2)$

Space complexity $\Rightarrow O(1)$

optimisation

a b c d e \Rightarrow no sweep

$a < b < c < d < e$ \Rightarrow sorted

$6n \rightarrow 5 \text{ rounds}$

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

becoz we access j+1 element for comparison & round last ele sent in every round

Round 1 $\rightarrow j = 4$
Round 2 $\rightarrow j = 3$
Round 3 $\rightarrow j = 2$
Round 4 $\rightarrow j = 1$
Round 5 $\rightarrow j = 0$

```
void bubbleSort(int arr[], int n)  
{  
    for(int i = 0; i < n-1; i++){  
        for(int j = i+1; j < n; j++){  
            if(arr[j] < arr[i]){  
                swap(arr[i], arr[j]);  
            }  
        }  
    }  
}
```