# C++
# BUBBLE SORT

## Lecture-21

Raghav Garg

# Today's checklist

1) Sorting
2) Bubble sort Algorithm
3) Time complexity and space complexity
4) Bubble sort optimization
5) Stable and unstable sort
6) 2 Questions

# What is sorting?

arr =


To Sort = put in ascending order → 1 2 3 4 5

Increasing order

Non-Decreasing

Sort in decreasing order → put the elements in descending order → 5 4 3 2 1

# Bubble sort algorithm

1st Pass

arr = 5 1 4 3 2

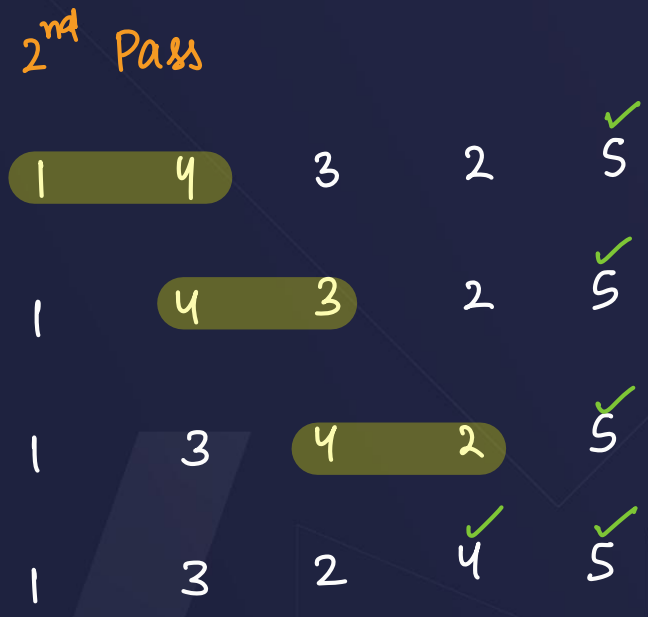
1 5 4 3 2

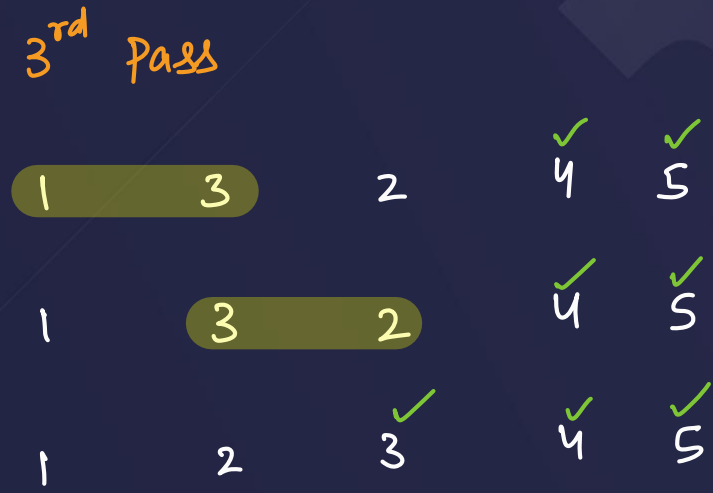
1 4 5 3 2

1 4 3 5 2

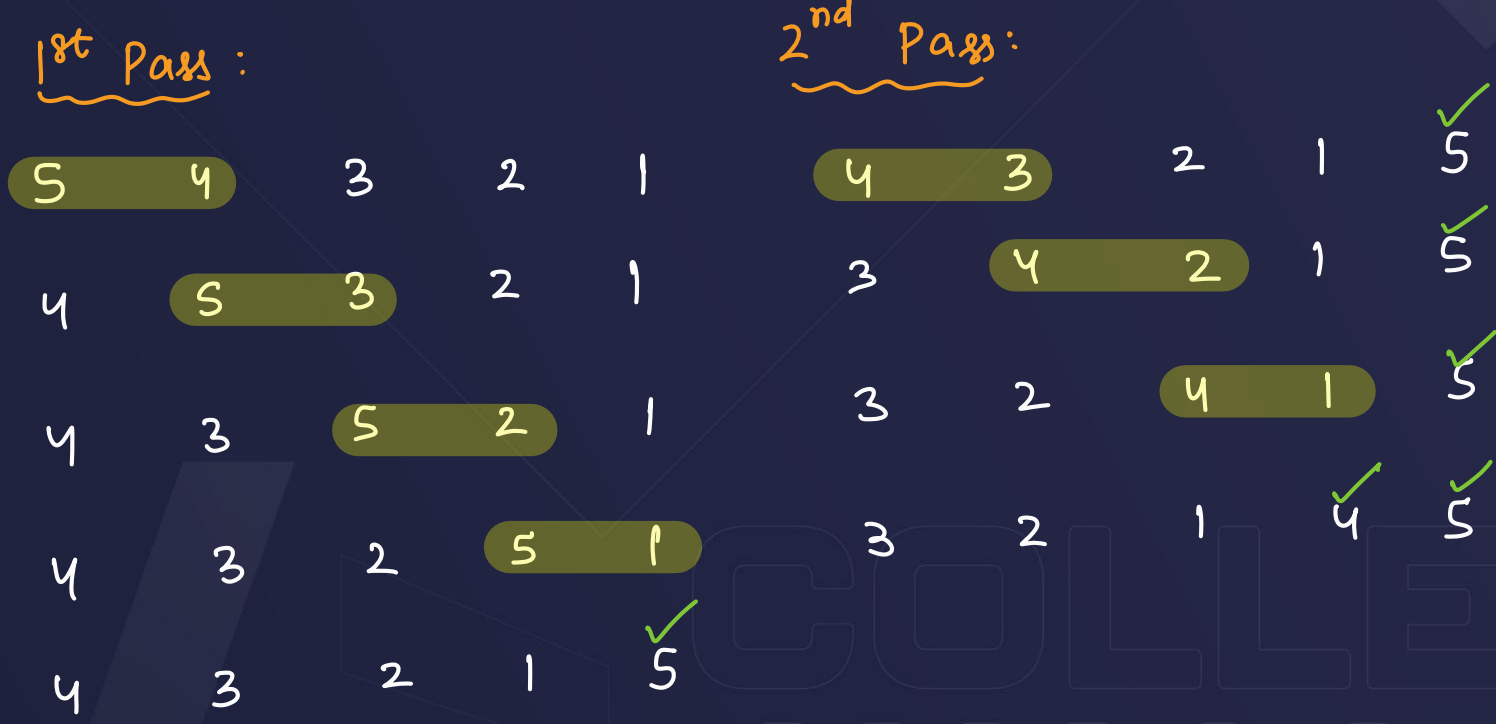
1 4 3 2 5 ✓

# Bubble sort algorithm



# Bubble sort algorithm



# Bubble sort algorithm

3rd Pass :

3 2 1 4 5

2 3 1 4 5

2 1 3 4 5

4th pass :

2 1 3 4 5

1 2 3 4 5

Formula → $\frac{n(n-1)}{2}$ → $\frac{5(4)}{2} = 10$

Observations :

- In each pass the $n^{th}$ max element goes to its right position
- If there are 'n' elements, then we require almost 'n-1' passes to sort.

Algorithm : In each pass swap 2 adjacent elements if arr[i] > arr[i+1]

Iteration in each pass also reduces.

```java
// bubble sort
for(int i=0;i<n-1;i++){ // n-1 passes
    // traverse
    for(int j=0;j<n-1;j++){
        if(arr[j]>arr[j+1]){ //swap
            swap(arr[j],arr[j+1]);
        }
    }
}
```

T.C. → $O(n^2)$

arr = 4 3 2 1 (0 1 2 3) n = 4

3 4 2 1

3 2 4 1

3 2 1 4 ✓

2nd pass

3 2 1 4

2 3 1 4

2 1 3 4 ✗

2 1 3 4

3rd pass

2 1 3 4

1 2 3 4 ✗

1 2 3 4

1 2 3 4

T.n.O → (n-1)*(n-1)

# Time and Space complexity

```
// bubble sort
for(int i=0;i<n-1;i++){ // n-1 passes
    // traverse
    for(int j=0;j<n-1-i;j++){
        if(arr[j]>arr[j+1]){ //swap
            swap(arr[j],arr[j+1]);
        }
    }
}
```

$i = 0, 1, 2, \ldots n-2$

$i=0$, $j = 0, 1, 2, \ldots n-2$ → $n-1$

$i=1$, $j = 0, 1, 2, \ldots n-3$ → $n-2$

$i=2$, $j = 0, 1, 2 \ldots n-4$ → $n-3$

⋮

→ 2

$i=n-2$, $j = 0$ → 1

$T.C.0 \rightarrow 1+2+3+\cdots n-1$

$= \frac{(n-1)(n-1+1)}{2} = \frac{n(n-1)}{2} \rightarrow T.C. \rightarrow O(n^2)$

# Time and Space complexity

Time Complexity → $O(n^2)$

Space Complexity → $O(1)$

# Can we optimize it further ?

# Can we optimize it further ?

```
// bubble sort optimised
for(int i=0;i<n-1;i++){ // n-1 passes
    // traverse
    bool flag = true;
    for(int j=0;j<n-1-i;j++){
        if(arr[j]>arr[j+1]){ //swap
            swap(arr[j],arr[j+1]);
            flag = false;
        }
    }
    if(flag==true){ // swap didn't happen
        break;
    }
}
```

Time Complexity :

Best Case : $O(n)$

Avg. Case : $O(n^2)$

Worst Case : $O(n^2)$

**Ques** : Given an array, find if it is sorted or not

→ arr = { 1, 2, 3, 4 }  O(n)

```
bool flag = true; // sorted
for (int i = 0; i < n-1; i++){
    if (arr[i] > arr[i+1]){
        flag = false;
        break;
    }
}
if (flag == true) → sorted
else → unsorted
```

# Stable and Unstable sort

arr → 1 2 3 $5_1$ $5_2$

Conclusion : Bubble Sort is a stable sort

after sort → 1 2 3 $5_1$ $5_2$ → Stable

→ 1 2 3 $5_2$ $5_1$ → Unstable

**Ques : How much maximum swaps are needed to sort array of length 6 ?**

→ arr = 6 5 4 3 2 1

↓

total no. of swap = total no. of operations

$= \frac{n(n-1)}{2} = \frac{6 \cdot 5}{2} =$ 15 swaps

**Ques :** Sort a String in decreasing order of values associated after removal of values smaller than X.

String S = "AZYZXBDJKX";

Str = "ZYZXX";

Sort G = ZZYXX

**Ques :** Push zeroes to end while maintaining the relative order of other elements.

'Bubble Sort Ki Importance'

arr = { 5, 0, 1, 2, 0, 0, 4, 0, 3 }

arr = { 5, 1, 2, 4, 3, 0, 0, 0, 0 }

Summary : Bubble Sort → T.C. → $O(n^2)$

Inbuilt Sort → T.C. → $O(n \cdot \log n)$

# THANK YOU