Bubble Sorting

→ Real world Application is not there

e.g ⇒ 5 | 4 | 3 | 2 | 1

1 | 2 | 3 | 4 | 5

Generally sort means ascending.

Algorithm

1. swap the adjacents if needed till we get the all the array sorted.

e.g ⇒

5 | 4 | 3 | 2 | 1 |
0 1 2 3 4

I largest at its position.

(1) iteration 5 | 4 | 3 | 2 | 1

4 | 5 | 3 | 2 | 1

4 | 3 | 5 | 2 | 1

4 | 3 | 2 | 5 | 1

4 | 3 | 2 | 1 | 5

[4 comparisons]

observation: 1st largest gone to last.

② second Largest at its position

4 | 3 | 2 | 1 | 5

3 | 4 | 2 | 1 | 5

3 | 2 | 4 | 1 | 5

3 | 2 | 1 | 4 | 5

NO

~~3 | 2 | 1 | 5 | 4~~

[3 comparisons]

③ 3rd Largest at position

3 | 2 | 1 | 4 | 5

2 | 3 | 1 | 4 | 5

2 | 1 | 3 | 4 | 5

[2 comparisons]

④ 4th largest

2 | 1 | 3 | 4 | 5

1 | 2 | 3 | 4 | 5

[1 comparison]

N=5

4 iterations

I , II , III , IV

↓ ↓ ↓ ↓

4 3 2 1

$S_n = 1+2+3.... (n-2)+(n-1)$

$S_n = \frac{n(n-1)}{2}$

$= \frac{n^2-n}{2}$

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

$O(n^2)$

Time complexity

Code:

```
void bubbleSort(vector<int> &v){
    int n = v.size();
  for(int i=0; i<n-1; i++){
    for(int j=0; j<n-i-1; j++){
      if(v[j] > v[j+1]){
        swap(v[j], v[j+1]);
      }
    }
  }
}
```

Selection Sort

selecting minimum element and putting at right position.

44 | 33 | 55 | 22 | 11

① 11 | 33 | 55 | ~~44~~ 22 | 44

② 11 | 22 | 55 | 33 | 44

③ 11 | 22 | 33 | 55 | 44

④ 11 | 22 | 33 | 44 | 55

Note: for ith iteration, pick smallest element from i to (n-1) index & swap it with ith element.

Time complexity → O(n²)

code:

```
void selectionSort(vector<int>& v){
    int n = v.size();
    for(int i=0; i<n-1; i++){
        ~~for~~ int minIndex = i;
        // ith element hi smallest hai
        for(int j = i+1; j<n; j++){
            if(v[j] < v[minIndex]){
                minIndex = j;
            }
        }
        //swap ith and minIndex elements;
        swap(v[i], v[minIndex]);
    }
}
```

Insertion Sort.

① | 5 | 4 | 3 | 2 | 1 |

② | 4 | 5 | 3 | 2 | 1 |

③ | 3 | 4 | 5 | 2 | 1 |

④ | 2 | 3 | 4 | 5 | 1 |

⑤ | 1 | 2 | 3 | 4 | 5 |

Time complexity ⇒ $O(n^2)$

Space Complexity ⇒ O(1)

Code:-

```
void insertionSort(vector<int> &v){
    int n= v.size();
    //i=0, chhodh deta hu
for(int i=1;i<n;++i){
    int key= v[i];
    int j =i-1;
//move elements of arr[0...i-1] that are greater than
// key to one position ahead of the current position
while(j>=0 && v[j]>key){
    v[j+1] =v[j];
    j--;
}
}v[j+1] = key; // insertion
}
```

Custom comparator

Custom camparator for ~~as~~ descending Sorting

```
bool myComp(int &a, int &b){
    return a>b;
}
```

e.g NO 2

```
bool myCompfor1stIndex(vector<int>& a,
                       vector<int>&b){
    return a[1] < b[1];
}
```