SORTING ALGOS

Date ..../..../.....

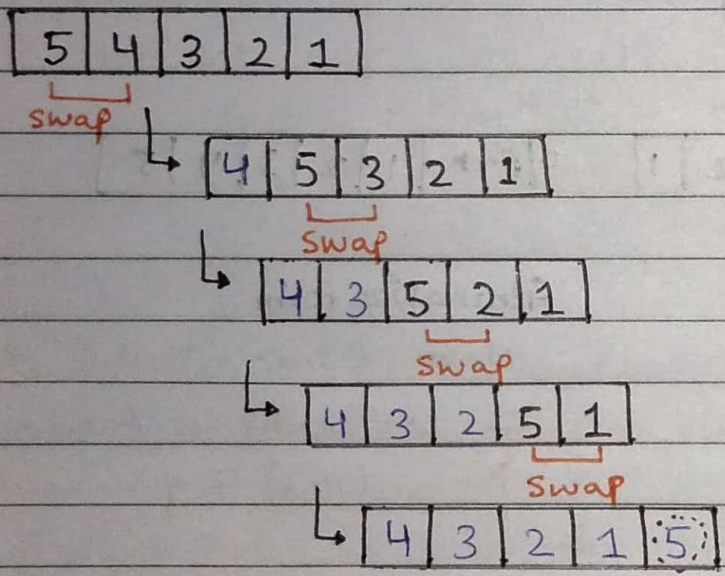
Sorting :- Arranging given elements in either strictly increasing or decreasing order.
Ex :- {5, 4, 3, 2, 1}

1. Bubble Sort :-

Algorithm :- Swap the adjacents if needed, till we get the array sorted.

Ex :- i/p → | 5 | 4 | 3 | 2 | 1 |
0 1 2 3 4

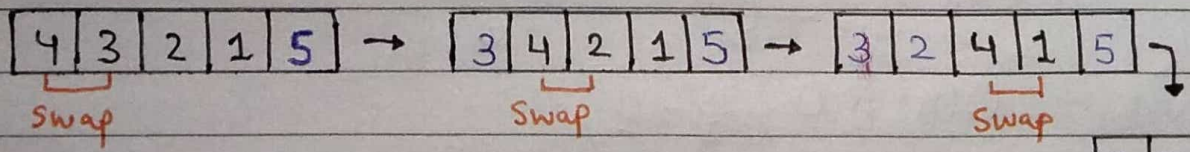
(i) I^st Iteration :



∴ It took 4 comparisons

Result of I^st Iteration :- I^st Largest element at its position.

(ii) II^nd Iteration :- It will place II^nd Largest element at its position.



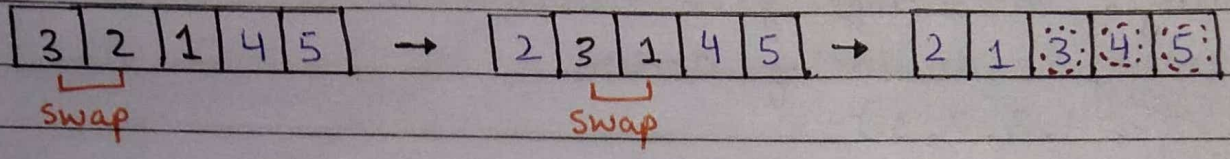
∴ It took 3 comparisons

3 2 1 4 5

Spiral
At its positions

Date .....

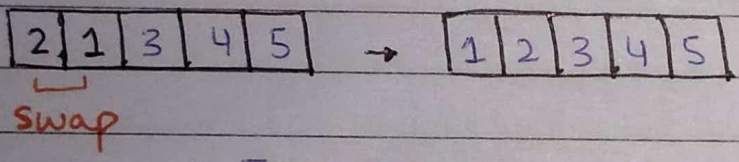
(iii) IIIrd Iteration :-



Result of IIIrd Iteration :- IIIrd Largest element at its place

∴ It takes 2 comparisons

(iv) IVth Iteration :- It will place IVth Largest element at its position.



∴ It takes 1 comparison

Summary :-

N = 5 i/p → 5 4 3 2 1 o/p → 1 2 3 4 5

Iterations		comparisons	
i=0	I	4	(n-i-1) = 4-i
i=1	II	3	
i=2	III	2	
i=3	IV	1	

$i \in [0, n-1)$ iterations

Generalize form

N = n

ITERATION
1 → (n-1) comparison
(n-2) comparison
(n-2) → 2 comparison
(n-1) → 1 comparison

Total comparisons = $S_n = 1 + 2 + 3 + \cdots + (n-2) + (n-1)$

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

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

Time Complexity = $O(n^2)$ ∴ It's the main drawback of Bubble Sort

Space " = $O(1)$

Spiral

Date ..../..../.....

Code :-

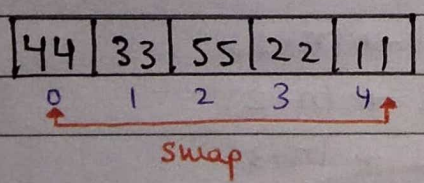
```
void bubbleSort(vector<int> &v){
    int n = v.size();
    for(int i=0; i<n-1; i++){
        for(int j=0; j<n-i-1; j++){
            if(v[j] > v[j+1]){
                swap(v[j], v[j+1]);
            }
        }
    }
}
int main(){
    vector<int> v = {5,4,3,2,1}; int n = v.size();
    bubbleSort(v);
    for(int i=0; i<n; i++){
        cout<<v[i]<<" ";
    }
}
```

2. Selection Sort :- It's a basic sorting algorithm.

Working :- Select the minimum element & Put it at right position.

Algorithm :- for i^{th} iteration, pick smallest element from i to $(n-1)$ index & swap it with i^{th} element.

Ex :- 44, 33, 55, 22, 11 $\quad N=5$; $\quad i \in [0, N-1)$



I^{st} Iteration , $[0,4) \rightarrow 0,1,2,3$

Step 1. $[0,4] \rightarrow i = 0$

minIndex = 4

Step 2. swap(v[i], v[minIndex])

:: v[0], v[4] → 11 33 55 22 44

Sorted

Date ..../..../...

IInd Iteration :- $i = 1$, $i \in [1, n-1]$, → minIndex
$[1, 4]$

11	33	55	22	44
0	1	2	3	4

swap

minIndex = 3

swap(v[i], v[3])

11	22	55	33	44

IIIrd Iteration :- $i = 2$, $i \in [2, n-1]$

11	22	55	33	44
0	1	2	3	4

swap

minIndex = 3

swap (v[i], v[minIndex])
swap (55, 33)

11	22	33	55	44

IVrd Iteration :- $i = 3$, $i \in [3, n-1]$ → $[3, 4]$

11	22	33	55	44
0	1	2	3	4

swap

minIndex = 4
swap (v[i], v[minIndex])
swap (55, 44)

11	22	33	44	55

→ Sorted

Summarise :-
N = 5

I $i = 0$ → 4 comparisons
II $i = 1$ → 3 comparisons
III $i = 2$ → 2 comparisons
IV $i = 3$ → 1 comparison

Outer loop
(n-1) Times

I → (n-1)
II → (n-2)
III → (n-3)
⋮
(n-1) → 1

Spira

Date ..../..../.....

Sum of total comparisons = $1+2+3+\cdots+(n-2)+(n-1)$

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

$O(n^2)$ → Time complexity

$O(1)$ → Space complexity

Code:-

```
void selectionSort( vector<int> &v) {
    int n = v.size();
    for( int i=0; i<n-1; i++) {
        int minIndex = i;    // ith element hi smallest hai
        for ( int j = i; j<n; j++) {
            if ( v[j] < v[minIndex]) {
                minIndex = j;
            }
        }
        //swap ith and minIndex elements
        swap ( v[i], v[minIndex]);
    }
}
int main() {
vector<int> v = { 5, 4, 3, 2, 1, 10, 200, -7, 30};
int n = v.size();
  selectionSort(v); //function call
  for( int i=0; i<n; i++){
      cout << v[i] << " ";
  }
}
```

Output:-

-7 1 2 3 4 5 10 30 200

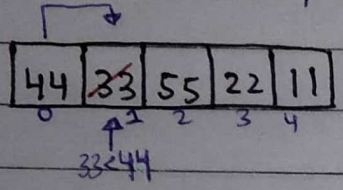
Spiral

Date ..../..../.....

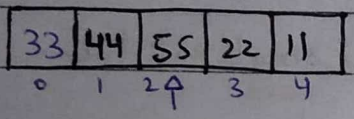
3. Insertion Sort :-

Working :- Ignore the first element and check second element if the previous element is greater than second element then simply move previous element to previous +1 position and then place the second element at first place/right place.

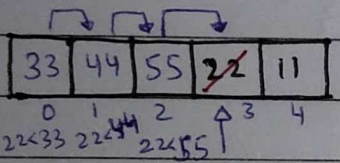
After that check 3rd element and then compare it with its previous ones and place it accordingly.

Ex :- 

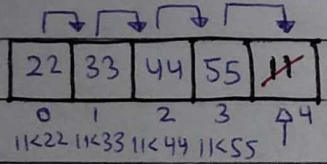
∵ Remove 33 from 1-index and put 44 and Put 33 at 0 index



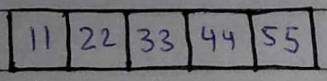
∵ Already sorted from 0 to 2 index, check next element



∵ Remove 22 from 3 index & put 55 at 3 index then increment each elements place by 1. from 0 to 2 index and Put 22 at 0 index.

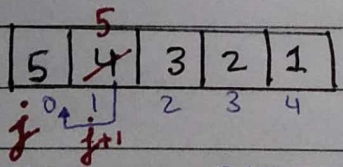


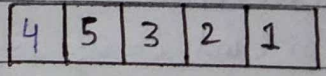
∵ Remove 11 from 4 index and move all its previous elements by 1 place ahead and Put 11 at 0 index bcz all elements are greater than 11.



sorted ✓

Ex :-

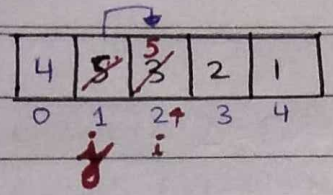





```
i=1   int key = v[i]  ⇒  key=4
      int j = i-1
      while ( j >=0  &&  v[j] > key)
      {     v[j+1] = v[j]
            j--;
      }
      v[j+1] = key
```

Date ..../..../.....

$i = 2$;



$j = i - 1 \Rightarrow j = 2 - 1 = 1$

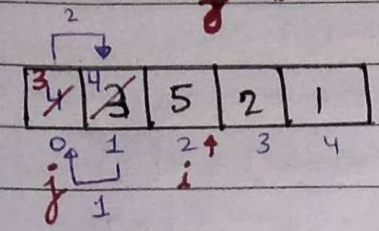
Key = v[i]

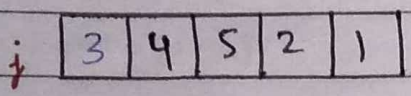
Key = 3

v[j] > Key → 5 > 3 True

v[j+1] = v[j] → 5 at 2 index

j-- → j = 0



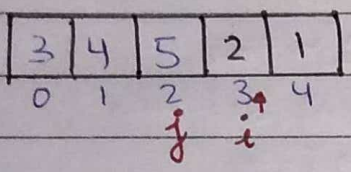


v[j] > Key → 4 > 3 True

v[j+1] = v[j] → 4 at 1 index

j-- then 3 at 0 index

$i = 3$;



Key = v[i] = 2

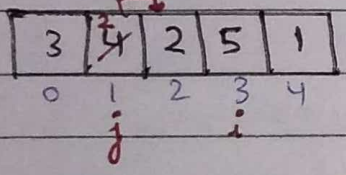
j = i - 1 ⇒ j = 2

v[j] > Key → 5 > 2 True

v[j+1] = v[j]

v[2+1] = v[2]

v[3] at v[2] → 5 replaced by 2

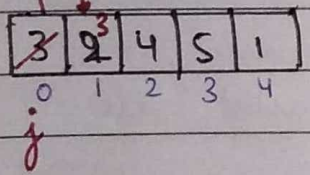


j-- ⇒ j = 1

Key = 2

v[j] > Key → 4 > 2 True

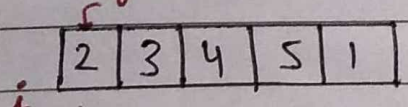
replace 2 with 4



j-- ⇒ j = 0

v[j] > Key → 3 > 2 True

Replace 3 with 2

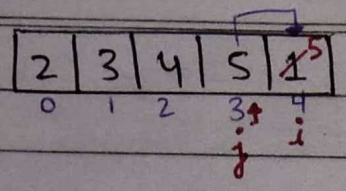


j-- ⇒ j = 0 out of bound

v[j+1] = Key

Date ..../..../.....

$i=4$



Key = v[i] = 1

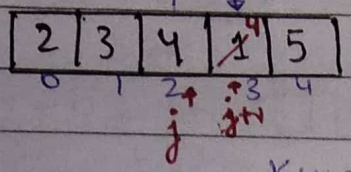
$j = i-1 = 4-1 = 3$

v[j] > Key → v[3] > Key → 5 > 1 True

↳ v[j+1] = v[j]

$j-- \Rightarrow j = 2$

↳ v[j+1] = Key



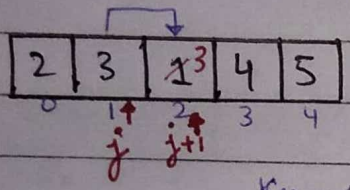
Key = 1

v[j] > Key → 4 > 1 → True

↳ Place 4 at 3rd index

$j-- \Rightarrow j = 1$

Then place 1 at 2 index



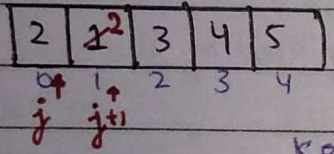
Key = 1

v[j] > Key → 3 > 1 → True

↳ Place 3 at 2 index

$j-- \Rightarrow j = 0$

Then place 1 at 1 index



Key = 1

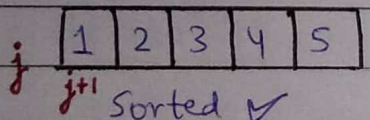
v[j] > Key → 2 > 1 → True

v[j+1] = v[j]

↳ Place 2 at 1 index

$j-- \Rightarrow j = -1$ → Invalid index

Then place 1 at 0 index



Sorted ✓

⇒ $i=0$, ignore / already sorted

$i = 1 \longrightarrow n-1$

$j = i-1 \quad > \quad j \geq 0$

Spiral

Date ..../..../.....

Code :-

```
void insertionSort( vector<int> &v){
    int n= v.size();
    // i=0; chor dete hai
    for( int i = 1; i<n; i++){
        int key = v[i];
        int j=i-1;
        // move element of arr[0 to i-1] that are greater than
        // key, to one position ahead of their current position
        while ( j>=0 && v[j] > key ){
            v[j+1] = v[j];
            j--;
        }
        v[j+1] = key ; // insertion
    }
}
int main(){
    vector<int> v = {5,4,3,2,1}; int n= v.size();
    insertionSort(v);
    for( int i=0; i<n; i++){
        cout<< v[i] << " ";
    }
}
```

Output

1 2 3 4 5

T.C :-

ITERATION			No. of comparisons
I	i=1	→	1
II	i=2	→	2
III	i=3	→	3
IV	i=4	→	4

N=5

n	
1	→ 1
2	→ 2
3	→ 3
:	
n-1	→ n-1

$S_n = 1+2+3+\dots+(n-1) = \frac{n(n-1)}{2}$

T.C = $O\left(\frac{n^2}{2} - \frac{n}{2}\right) \Rightarrow O(n^2)$

S.C = O(1)

Spiral

CUSTOM COMPARATOR

Date ..../..../.....

Custom Comparator :- By default, the sort() function of STL sorts the elements in ascending order. It generally takes 2 argument first is the beginning of array/vector and the second argument is the length upto which we want to get sorted array.
3rd parameter is optional & can be used, if we want to sort the elements according to user needs.
The comparator is a function of bool datatype which takes 2 elements/variables and compare them & retern the output.

Code :-

```
bool mycomp( int & a, int &b){
    // return a<b; // Increasing order sorting
    // return a>b; // decreasing order sorting
}
int main(){
    vector<int> v= {44, 55, 22, 11, 33}; int n = v.size();
    sort (v.begin, v.end(), myComp);
    for(int i=0; i<n; i++){
        cout<< v[i] <<" ";
    }
}
```

Output :-
55 44 33 22 11

Q Sort this 2D vector with respect to the second elements i.e 1st index of vectors [[1,44], [0,55], [0,22], [0,11], [2,33]]

Code :-

```
void print2Dvector (vector<vector<int>> &v){
    int n = v.size();
    for (int i = 0; i<n; i++){
        vector<int> & temp = v[i];
        int a = temp[0];
        int b = temp[1];
        cout<< a <<" "<< b << endl;
    }
}
```

Spiral

Date ..../..../.....

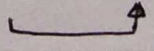
```
bool mycustomcomparatorfor1stIndex (vector<int>&a, vector<int>&b){
    // return a[1] < b[1]; // ascending order sorting
    return a[1] > b[1]; // descending order sorting
}

int main(){
    vector<vector<int>> v; int n = v.size(); cin>>n;
    for( int i=0; i<n; i++){
        vector<int> temp;
        cout<<"enter values:"<<endl;
        int a,b; cin>>a>>b;
        temp.push_back(a);
        temp.push_back(b);
        v.push_back(temp);
    } cout<<endl;
    cout<<"Here are the values:"<<endl;
    print2Dvector(v);
    cout<<"Sorted by 1st index:"<<endl;
    sort( v.begin(), v.end(), mycustomcomparatorfor1stIndex);
    print2Dvector(v);
}
```

Output :- 5

```
enter values:
1 44
enter values:
0 55
enter values:
0 22
enter values:
0 11
enter values:
2 33
Here are the values:
1 44
0 55
0 22
0 11
2 33
```

```
Sorted by 1st index:
0 55
1 44
2 33
0 22
0 11
```

∴ In Descending order



Spiral