SORTING

25/09/2023
Monday

* BUBBLE SORT →

Swap the adjacents if needed till we get a sorted array.

let an array → 9 4 8 2 6 7

9 ↔ 4 8 2 6 7
4 9 ↔ 8 2 6 7
4 8 9 ↔ 2 6 7
4 8 2 9 ↔ 6 7
4 8 2 6 9 ↔ 7
4 8 2 6 7 9

4 8 ↔ 2 6 7 9
4 2 8 ↔ 6 7 9
4 2 6 8 ↔ 7 9
4 2 6 7 8 9

4 ↔ 2 6 7 8 9
2 4 6 7 8 9 → Sorted Array

Logic → Swap the adjacents if $v[j] < v[j+1]$

- For every n^{th} iteration → Number of comparisons would be $n-i$.

Here, $n=5$ and i starts from 1.

Iteration	comparisons
1	4
2	3
3	2
4	1

If $n=5$, outer loop will run $n-1$ i.e. 4 times.
for $[0,4) \longrightarrow 0,1,2,3$. That's why $i<n-1$ kiya hai.

```
for (int i=0; i<n-1; i++) {  ⟶  i ∈ [0, n-1)
    for (int J=0; J<n-i-1; J++) {→ J ∈ [0, n-i-1)
        if (v[J] > v[J+1]) {
            swap(v[J], v[J+1]);
        }
    }
}
```

Outer loop will tell the number of iteration and inner loop will tell the number of comparisons would occur for any particular iteration. Then, swapping.

- **Time Complexity of Bubble sort →**

for every iteration, number of comparisons is reducing by 1.

Iteration	Comparisons
1	$n-1$
2	$n-2$
3	$n-3$
⋮	⋮
$n-2$	2
$n-1$	1

Total comparisons $(S_n) = 1+2+3+4+\_\_\_+(n-2)+(n-1)$

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

$\therefore S_n = n(n+1)/2$
Here, $n = n-1$

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

Time complexity $= O(n^2)$

Space complexity → O(1)

Code →

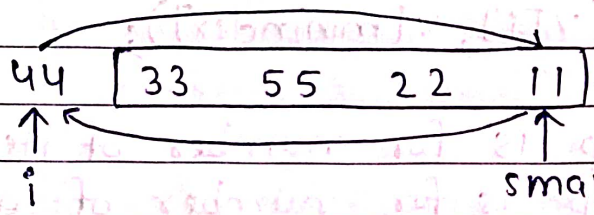
```
#include <bits/stdc++.h>
using namespace std;

void print(vector<int> &v){
    int n=v.size();
    for(int i=0; i<n; i++){
        cout<< v[i]<<" ";
    }
}

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{9,4,8,2,6,7};

    bubbleSort(v);
    print(v);

}
```

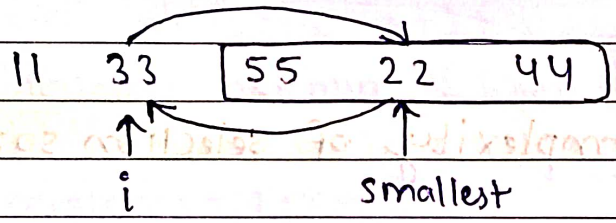
Output → 2 4 6 7 8 9

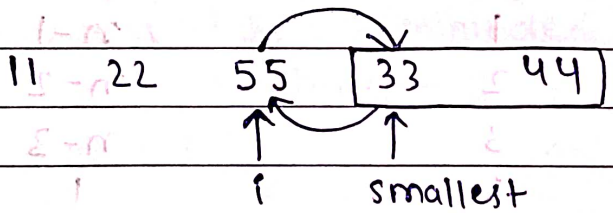
* SELECTION SORT →

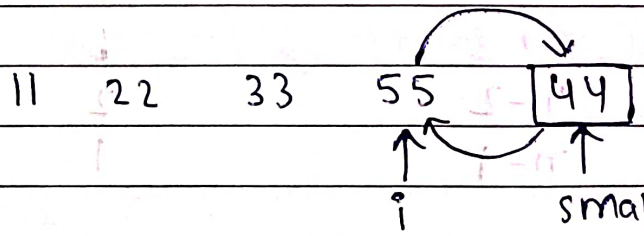
Swap the smallest element in range [i, n-1] with the ith element.

Let an array - 44 33 55 22 11

① 

② 

③ 

④ 

11	22	33	44	55	→ Sorted array

__/__/__

logic→
```
for (int i=0; i<n-1; i++){ ⟶ i ∈ [0, n-1)
    int minIndex = i;
    for (int J=i+1; J<n; J++){ ⟶ J ∈ (i, n)
        if (v[J] < v[minIndex]){
            minIndex = J;
        }
    }
swap (v[i], v[minIndex]);
```

Outer loop is for number of iterations and inner loop is for number of comparisons per iteration.

Time Complexity of Selection sort →

Iteration	Comparison / iteration
1	n-1
2	n-2
3	n-3
⋮	⋮
n-2	2
n-1	1

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

Here, last term is $n-1$. So, S_n is →

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

Time complexity = $O(n^2)$

Space complexity = O(1)

__/__/____

Code ->

```
#include <bits/stdc++.h>
using namespace std;

void print(vector<int>&v){
    int n= v.size();
    for(int i=0; i<n; i++){
        cout<< v[i]<<" ";
    }
}

int selectionSort(vector<int>&v){
    int n=v.size();
    for(int i=0; i<n-1; i++){
        // minIndex mai hum let krke chl rhe hai
        // ki ith index pr jo element hai vo smallest hai
        int minIndex = i;
        for(int j=i+1; j<n; j++){
            if(v[j]<v[minIndex]){
                minIndex = j;
            }
        }
        // swapping
        swap(v[i], v[minIndex]);
    }
}
int main(){
    vector<int>v{44,33,55,22,11};
    selectionSort(v);
    print(v);
}
```

Output = 11 22 33 44 55

__/__/__

* Insertion Sort →

Shift the elements in array until we get a sorted array.

let an array → 5 4 3 2 1

- let the element at 0th index be sorted.
- Initialise i from Ist index.
- As we have to compare all the elements before i with ith element. That's why J is initialised with i-1. loop breaks when J gets out of bound.

(1)

```
      4  5
5  4  3  2  1   →   4  5  3  2  1
↑  ↑  ↑
J  J  i
```

(2)

```
      3  4  5
4  5  3  2  1   →   3  4  5  2  1
↑  ↑  ✱  ↑
J  J  J  i
```

(3)

```
      2  3  4  5
3  4  5  2  1   →   2  3  4  5  1
↑  ✱  ✱  ✱  ↑
J  J  J  J  i
```

(4)

```
      1  2  3  4  5
2  3  4  5  1   →   1  2  3  4  5
↑  ✱  ✱  ✱  ✱  ↑
J  J  J  J  J  i
```

Sorted Array

logic →

```
for (int i=1; i<n; i++){ → i ∈ [1,n)
    int key= v[i];
    int J = i-1;
    while (J>=0 && v[J]>key){ → J ∈ [i-1,0]
        v[J+1] = v[J];
        J--;
    }
    v[J+1] = key;
}
```

- key variable is initialised to store element at ith index otherwise it will get lost during shifting.
- Compare all the elements placed at indices less than i with the ith element.
- Shift if the condition inside while is true.
- Decrement J.
- Store the value of key in v[J+1].

Time complexity of insertion sort →

No. of iteration	No. of comparison
1	1
2	2
3	3
!	!
!	!
!	!
n-2	n-2
n-1	n-1

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

Time complexity = $O(n^2)$

Space complexity = $O(1)$

Code →

```
#include <bits/stdc++.h>
using namespace std;

void print (vector<int> &v){
    int n=v.size();
  for(int i=0; i<n; i++){
      cout<<v[i]<<" ";
    }
  }

int insertionSort (vector<int> &v){
    int n=v.size();
  for(int i=1; i<n; i++){
    int key = v[i];
    int j= i-1;
    while (j>=0 && v[j]>key){
      v[j+1] = v[j];
      j--;
    }
    v[j+1] = key;
  }
}

int main( ){
    vector<int> v {44, 33, 55, 11, 22};
    insertionSort(v);
    print(v);
}
```

Input = 44 33 55 11 22

Output = 11 22 33 44 55

__/__/____

* CUSTOM COMPARATOR →

Using custom comparator for sorting →

Code →

```
#include<bits/stdc++.h>
using namespace std;
// custom comparator function
// This will return true or false
// That's why its return-type is bool
    bool myComp (vector<int>&a, vector<int>&b){
        return a[1]<b[1];
    }

// Printing function
    void printVv (vector<vector<int>>&v){
        for (int i= 0; i<v.size(); i++) {
            for (int j=0; j<v[i].size; j++){
                cout<< v[i][j] <<"  ";
            }
        cout<<endl;
    }
}
int main(){
vector<vector<int>> v {{1,44},{2,33},{0,55},{0,22},{0,11}};
    vector<int>a,b;
    // Sort function of STL
    sort (v.begin(), v.end(), myComp);
    cout<<"sorting 1st index of vector"<<endl;
    printVv (v);
}
```

Output - Sorting 1st index of vector

```
0 11
0 22
2 33
1 44
0 55
```