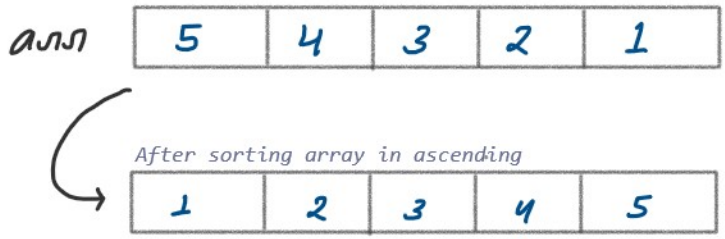
BUBBLE SORT

24 September 2023 09:22

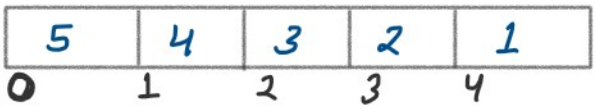
23/09/2023

BUBBLE SORT

✅**SORTINGS 01:** BUBBLE SORT
- swap the adjacent if needed till we get all the array sorted

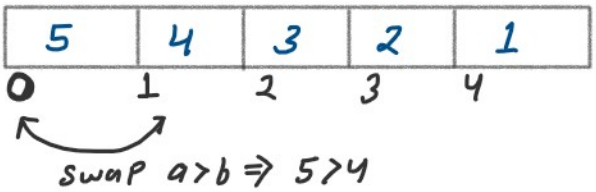


DRY RUN



1st Iteration

1st largest element at it's positions



a
b
} Adjacent value

a > b → swap

a < b → NO swap

4 5 3 2 1

swap a>b ⇒ 5>3

4 3 5 2 1

swap a>b ⇒ 5>2

4 3 2 5 1

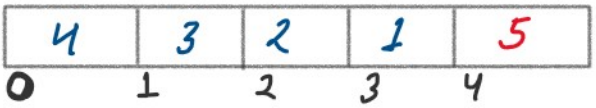
swap a>b ⇒ 5>1

4 3 2 1 5

1st largest element = 5

2nd Iteration

2nd largest element at it's positions



swap a>b => 4>3

3 4 2 1

sawap a>b => 4>2

3 2 4 1

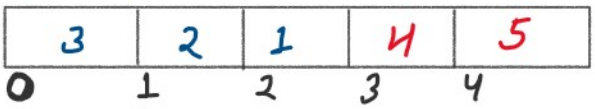
swap a>b => 4>1

3 2 1 4

2nd largest element = 4

3rd Iteration

3rd largest element at it's positions



swap a>b => 3>2

2 3 1

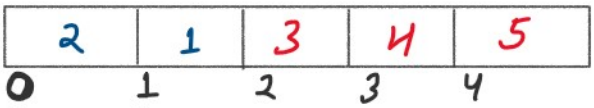
swap a>b => 3>1

2 1 3

3rd largest element = 3

4rt Iteration

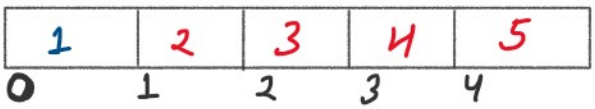
4rt largest element at it's positions



swap a>b => 2>1

1 2

4rt largest element = 2

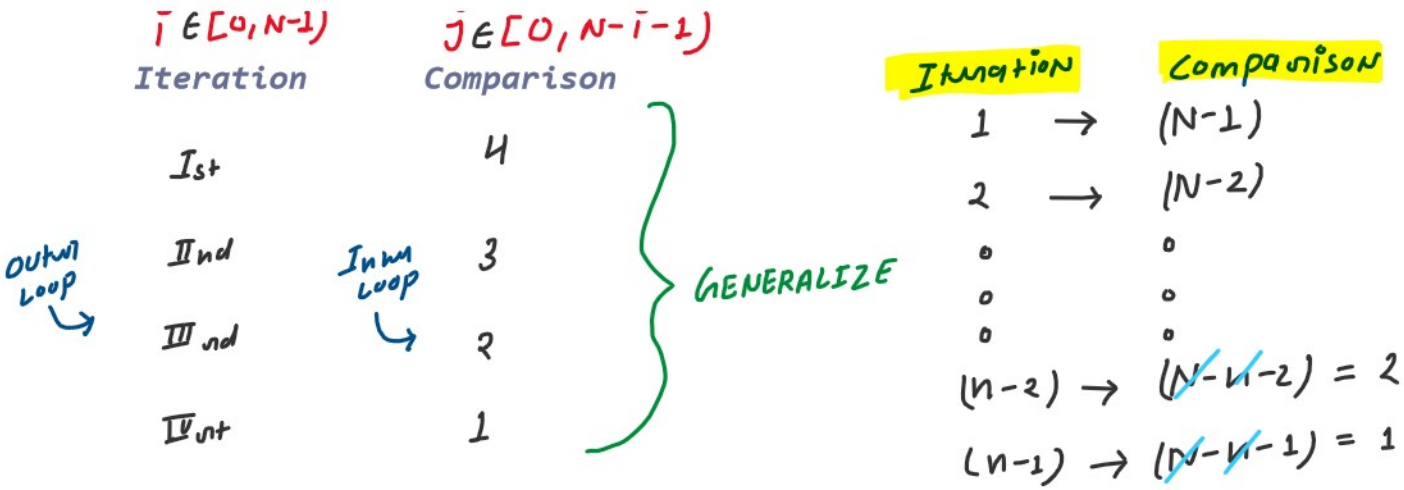


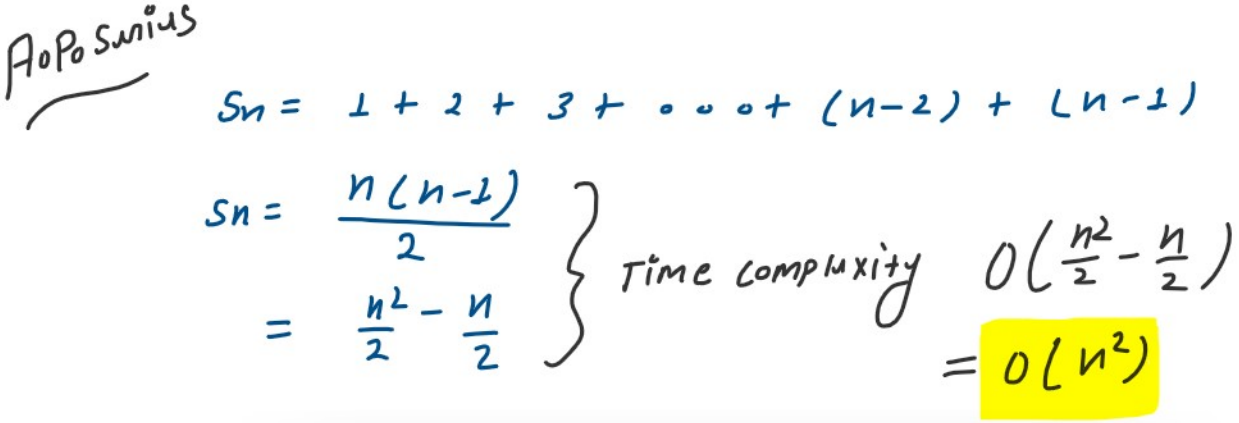
N = size = 5

$i \in [0, N-1)$ Iteration

$j \in [0, N-i-1)$ Comparison

Iteration Comparison






```cpp
// ✅SORTINGS 01: BUBBLE SORT
#include<iostream>
#include<vector>
using namespace std;

// BUBBLE SORT Function
void bubbleSort(vector<int> &arr){
    int N=arr.size();

    // Outer Loop
    for(int i=0; i < (N-1); i++){
        // Inner Loop
        for(int j=0; j < (N-i-1); j++){
            // When a>b where a=j and b=j+1 to swap kardo
            if(arr[j] > arr[j+1]){
                swap(arr[j], arr[j+1]);
            }
        }
    }
}

int main(){
    vector<int> arr{5,4,3,1,2};

    bubbleSort(arr);
    for(auto value: arr){
        cout<<value<<" ";
    }
    return 0;
}

/*
INPUT: {5,4,3,1,2}
OUTPUT: {1,2,3,4,5}
TIME COMPLEXITY: O(N^2)
SPACE COMPLEXITY: O(1)
*/
```

SELECTION SORT

24 September 2023 09:22

23/09/2023

SELECTION SORT

SORTINGS 02: *SELECTION SORT*
- *What if, I select the minimum element and put it at right position*
- *For ith iteration, pick smallest element from i to (n-1) index and swap it with ith element*

arr

5	4	3	2	1

After sorting array in ascending

1	2	3	4	5

DRY RUN

minIndex = i

Starting me first element ko smallest maan lo

5	4	3	2	1
0	1	2	3	4

N=5

i=0

j=i+1

swap

1st Iteration

5	4	3	2	1
0	1	2	3	4

4 com

Step 01: *find smallest element's index from [0,N-1]*

i=0 j=4 minIndex = j = 4

Step 02: *swap the ith and minIndex position's value*

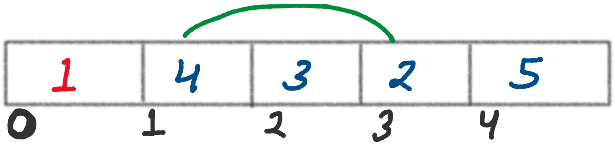
swap(arr[i], arr[minIndex])

5 1

YE Apni Right position par hai → 1 4 3 2 5

swap

2nd Iteration



3 com

Step 01: *find smallest element's index from [0,N-1]*

$i = 1$ $j = 3$ $minIndex = j = 3$

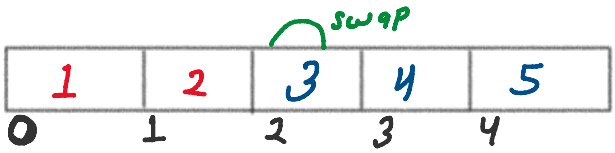
Step 02: *swap the ith and minIndex position's value*

swap(arr[i], arr[minIndex])
4 2

YE DONO APNI Right position par hai

→ 1 2 3 4 5

3rd Iteration



2 com

Step 01: *find smallest element's index from [0,N-1]*

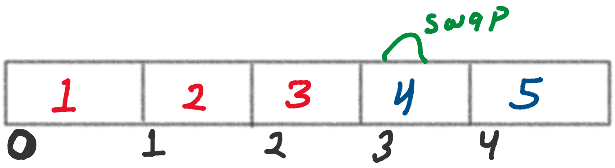
$i = 2$ $j = 2$ $minIndex = j = 2$

Step 02: *swap the ith and minIndex position's value*

swap(arr[i], arr[minIndex])
3 3

1 2 3 4 5

4rt Iteration



1 com

Step 01: *find smallest element's index from [0,N-1]*

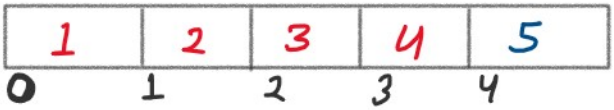
$i = 3$ $j = 3$ $minIndex = j = 3$

Step 02: swap the ith and minIndex position's value

swap(arr[i], arr[minIndex])

1 2 3 4 5

- Sorted Array



$i \in [0, N-1)$ Iteration

$j \in [i+1, N)$ Comparison

N = 5

Iteration	Comparison
Ist	4
IInd	3
IIIrd	2
IVth	1

Outer Loop

Inner Loop

GENERALIZE

Iteration		Comparison
1	→	(N-1)
2	→	(N-2)
.		.
.		.
.		.
(n-2)	→	(N-n-2) = 2
(n-1)	→	(N-n-1) = 1

$$\text{TC} \Rightarrow O\left(\frac{n^2}{2} - \frac{n}{2}\right) \Rightarrow O(N^2)$$

$$\text{SC} \Rightarrow O(1)$$

```cpp
// ✅SORTINGS 02: SELECTION SORT
#include<iostream>
#include<vector>
using namespace std;

// SELECTION SORT Function
void selectionSort(vector<int> &arr){
    int N = arr.size();

    for(int i=0; i < N-1; i++){ → O(N)

        // Lets assume ki 1st index par jo value hai wo hi smallest hai
        int minIndex = i;

        // ▶Step 01: find smallest element's index from [0,N-1]
        for(int j= i+1; j<N; j++){ → O(N)

            if(arr[j] < arr[minIndex]){
                minIndex = j;
            }
        }
        // ▶Step 02: swap the ith and minIndex position's value
        swap(arr[i], arr[minIndex]);
```

```
                minIndex = j;
            }
        }
        // ▶Step 02: swap the ith and minIndex position's value
        swap(arr[i], arr[minIndex]);
    }

}

int main(){
    vector<int> arr{5,4,3,1,2};

    selectionSort(arr);
    for(auto value: arr){
        cout<<value<<" ";
    }
    return 0;
}

/*
INPUT: {5,4,3,1,2}
OUTPUT: {1,2,3,4,5}
TIME COMPLEXITY: O(N^2)
SPACE COMPLEXITY: O(1)
*/
```

T.C. = O(N) * O(N)

= O(N * N)

= $O(N^2)$

S.C. = O(1)

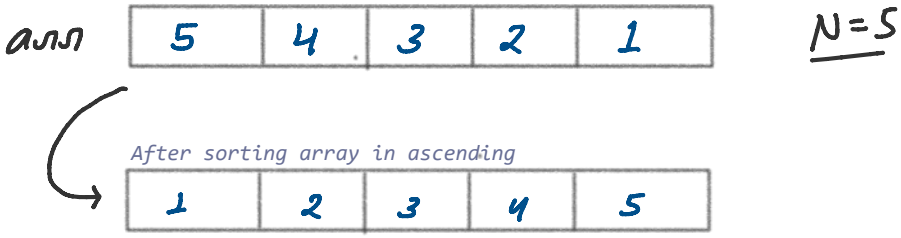
INSERTION SORT

24 September 2023 09:22

23/09/2023

INSERTION SORT

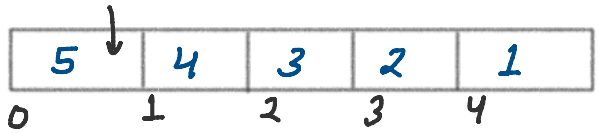
✅ *SORTINGS 03: INSERTION SORT*
- Move element of arr[0,i-1] that are greater than key to one position ahead of there current position



DRY RUN

Example:01

Always ignore the first element



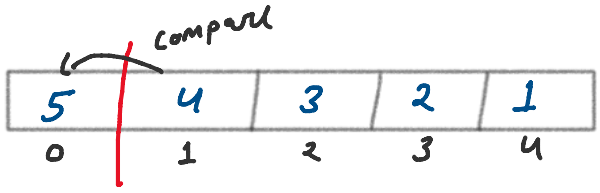
i=1 j= i-1

initially store the second element as a key value to Move element of arr[0,i-1] that are greater than key to one position ahead of there current position

key = arr[i] 4

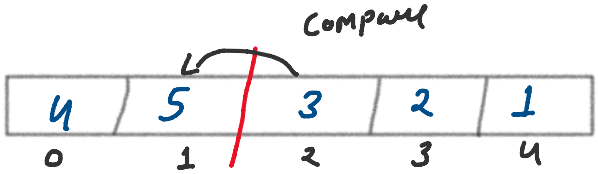
1st Iteration

key = 4
i=1 , j=0



5 > 4 ✓ → 4 5 3 2 1

2nd Iteration



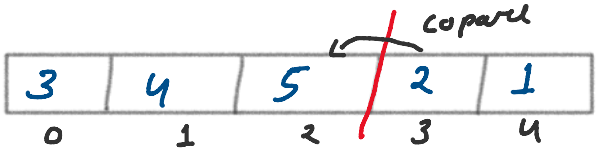
key = 3

2nd Iteration

key = 3
i = 2 , j = 1
i = 2 , j = 0

5 7 3 → 4 3 5 2 1
4 7 3 → 3 4 5 2 1

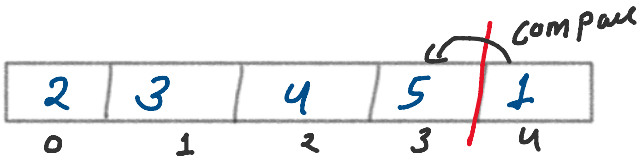
3rd Iteration



key = 2
i = 3 , j = 2
i = 3 , j = 1
i = 3 , j = 0

5 7 2 → 3 4 2 5 1
4 7 2 → 3 2 4 5 1
3 7 2 → 2 3 4 5 1

4rt Iteration



key = 1
i = 4 , j = 3
i = 4 , j = 2
i = 4 , j = 1
i = 4 , j = 0

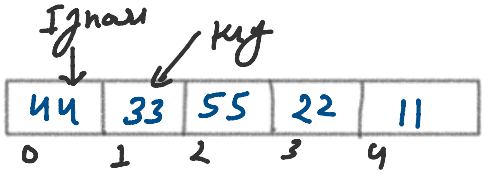
5 7 1 → 2 3 4 1 5
4 7 1 → 2 3 1 4 5
3 7 1 → 2 1 3 4 5
2 7 1 → 1 2 3 4 5

i = 5 { RUN jaaooo..... (i < 5) END }

Final output

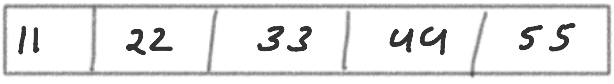
DRY RUN

Example:02 ans



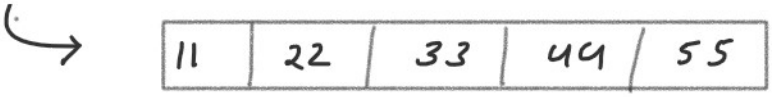
N = 5

i = 1 , j = i - 1



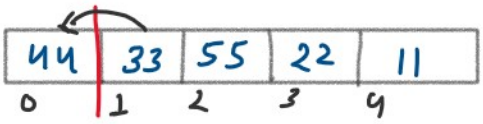
i=1 , j= i-1

key = arr[i]



SORTED ARRAY

Iteration:01

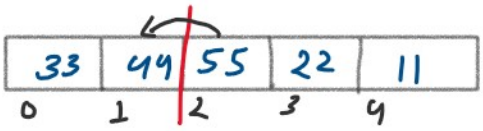


key = 33

i = 1 , j = 0

44 > 33 ✓ → 33 44 55 22 11

Iteration:02



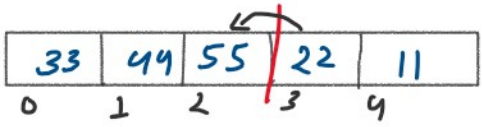
key = 55

i = 2 , j = 1

44 ? 55 ✗ → SAME ⇒ 33 44 55 22 11

BREAK inner loop

Iteration:03



key = 22

i = 3 , j = 2

i = 3 , j = 1

i = 3 , j = 0

55 ? 22 ✓ →

44 ? 22 ✓ →

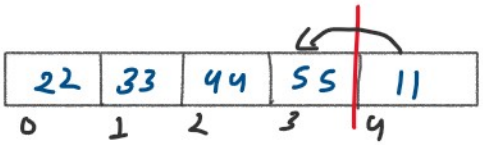
33 > 22 ✓ →

33 44 22 55 11

33 22 44 55 11

22 33 44 55 11

Iteration:04



key = 11

i = 4 , j = 3

i = 4 , j = 2

i = 4 , j = 1

i = 4 , j = 0

55 ? 11 ✓ →

44 ? 11 ✓ →

33 ? 11 ✓ →

22 ? 11 ✓ →

22 33 44 11 55

22 33 11 44 55

22 11 33 44 55

11 22 33 44 55

j = -1 ✓ END inner loop

i=4, j=0 22711

J=-1 X END Inner loop

i=5 { Ruk jaoooo.... (i<5)

↳ Final output

11	22	33	44	55

$i \in [1, N]$ Iteration

$j \in [i-1, 0]$ Comparison

Outer loop	Ist	1
	IInd	Inner loop 2
	IIIrd	3
	IVth	4

GENERALIZE

N=5

Iteration		Comparison
1	→	1
2	→	2
o		o
o		o
o		o
n-1	→	n-1
n	→	n

$S_n = 1+2+3+\dots+n$

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

Toc = $O(N^2)$

Soc = $O(1)$

```cpp
// ✅SORTINGS 03: INSERTION SORT

#include<iostream>
#include<vector>
using namespace std;

// INSERTION SORT Function
void insertionSort(vector<int> &arr){
    int N=arr.size();
    for(int i=1; i<N; i++){
        // Key value =  key value se pahle ki all values ko hum compare karte hai key value se hi
        //taki key value ko hum uske right position par rakh paye
        int key = arr[i];
        int j = i-1;

        // Move element of arr[0,i-1] that are greater than key to one position ahead of there current position
        while(j>=0 && arr[j]>key){
            arr[j+1]=arr[j];
            j--;
        }
        // Insert key at right index/position (Insertion)
        arr[j+1]=key;
    }
}

int main(){
    vector<int> arr{5,4,3,1,2};

    insertionSort(arr);
    for(auto value: arr){
        cout<<value<<" ";
    }
    return 0;
}

/*
INPUT: {5,4,3,1,2}
OUTPUT: {1,2,3,4,5}
TIME COMPLEXITY: O(N^2)
SPACE COMPLEXITY: O(1)
*/
```

CUSTOM COMPARATOR

24 September 2023 09:22

23/09/2023

Custom Comparator

✅CUTOM COMPARATOR

C++ STL Concpt

↳ 1. SORT A VECTOR: When you want to sort a vector using C++ STL sort data structure then syntax is

```cpp
// SORT A VECTOR
vector<int> v;

// Syntax of sorting
sort(v.begin(), v.end());
```

increasing order sorting by *default*

2. SORT A ARRAY: When you want to sort a vector using C++ STL sort data structure then syntax is

```cpp
// SORT A ARRAY:
int arr[];

// Syntax of sorting
sort(arr, arr+n);

/*
Where
- arr is a base address of array
- n is length of array
*/
```

increasing order sorting by *default*

1. SORT A VECTOR:

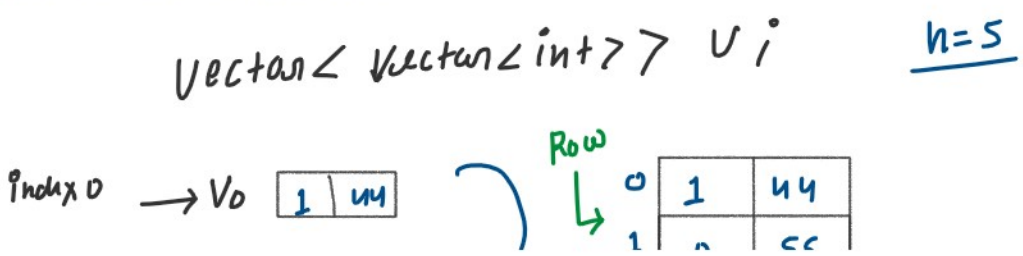
```cpp
// ✅CUTOM COMPARATOR 01: SORT A VECTOR
#include <algorithm>
```

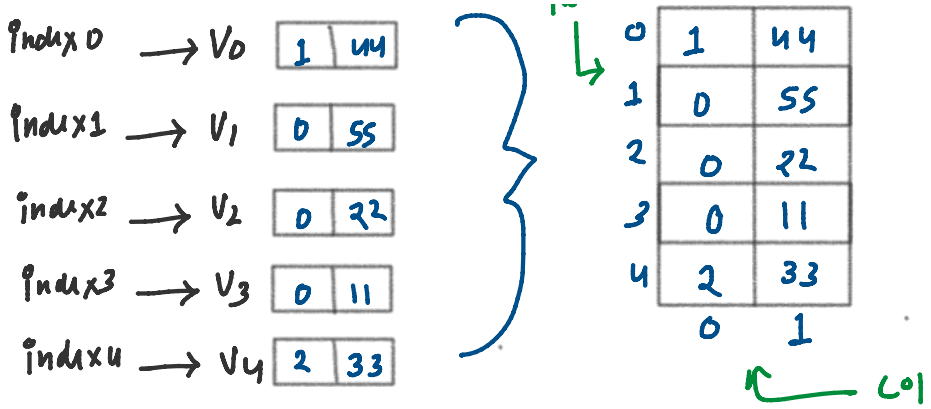
```cpp
// ✅CUTOM COMPARATOR 01: SORT A VECTOR
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

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

// Custom Comparator ki return value always true ya false hoti hai
bool myComparator1(int &a, int &b) {
    return a > b; // decreasing order sorting
}

bool myComparator2(int &a, int &b) {
    return a < b; // increasing order sorting
}

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

  cout<<"Vector"<<endl;
  print(v);

  cout<<"Increasing order sorting by default"<<endl;
  sort(v.begin(), v.end());
  print(v);

  cout<<"Decreasing order sorting by my comparator 1"<<endl;
  sort(v.begin(), v.end(), myComparator1);
  print(v);

  cout<<"Increasing order sorting by my comparator 2"<<endl;
  sort(v.begin(), v.end(), myComparator2);
  print(v);

  return 0;
}

/*
Vector
44 55 22 11 33
Increasing order sorting by default
11 22 33 44 55
Decreasing order sorting by my comparator 1
55 44 33 22 11
Increasing order sorting by my comparator 2
11 22 33 44 55
*/
```

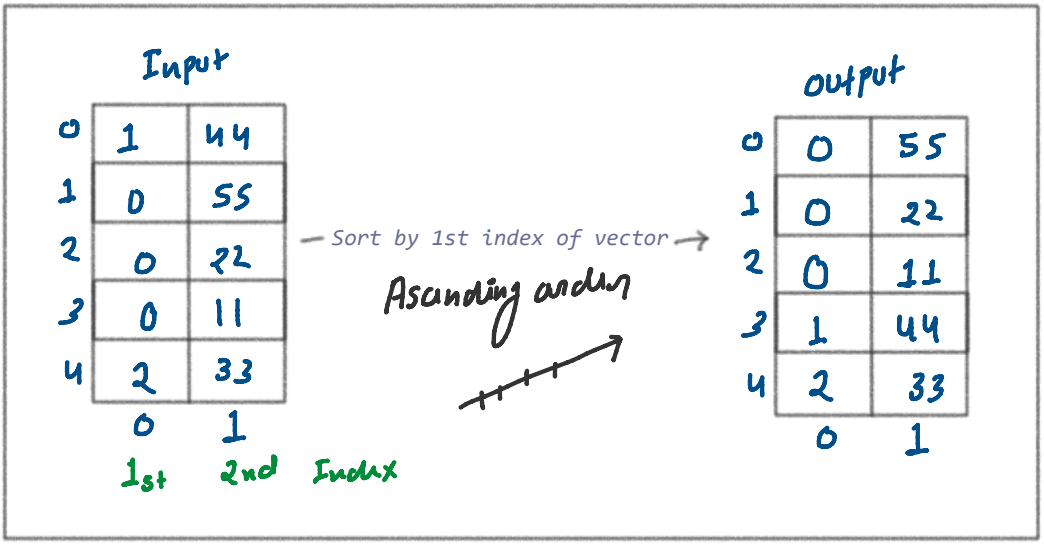
```cpp
// Custom comparator for decreasing order sorting
bool myComparator1(int &a, int &b) {
    return a > b;
}


// Custom comparator for increasing order sorting
bool myComparator2(int &a, int &b) {
    return a < b;
}
```

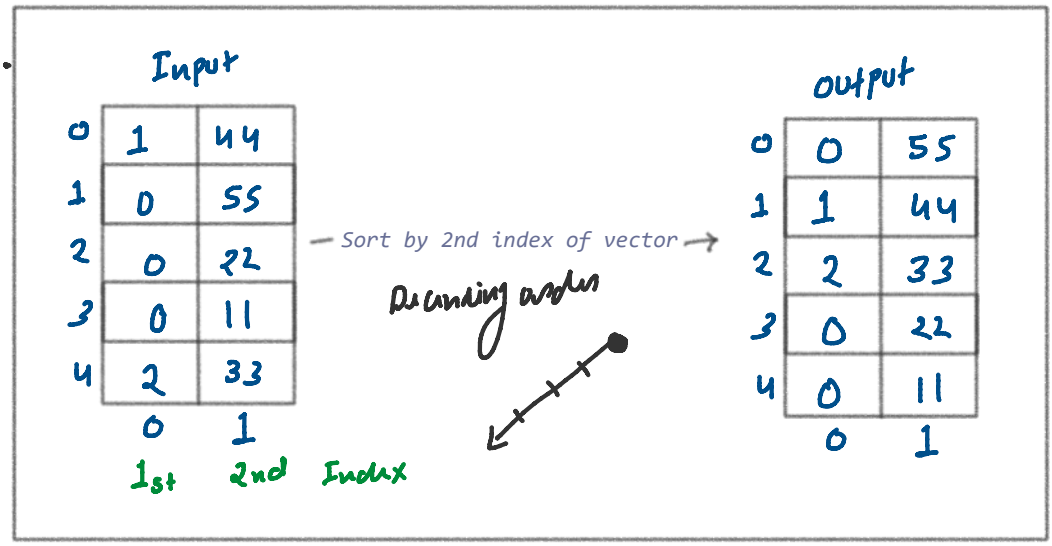
2. SORT VECTOR OF VECTOR:

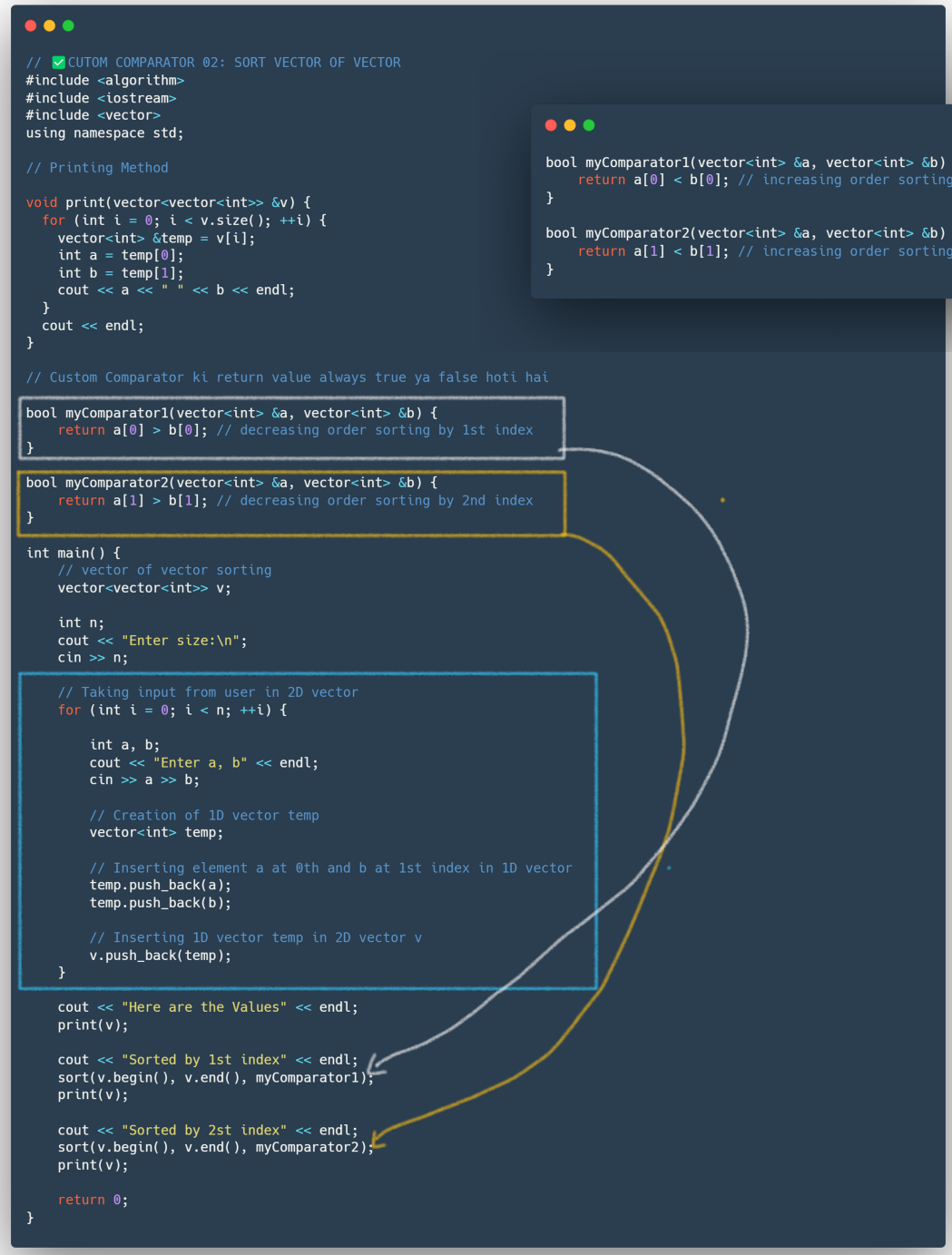
Declaration Syntax of 2D vector




index 0 → V0 | 1 | 44
index 1 → V1 | 0 | 55
index 2 → V2 | 0 | 22
index 3 → V3 | 0 | 11
index 4 → V4 | 2 | 33
0 1 4 4
1 0 5 5
2 0 2 2
3 0 1 1
4 2 3 3
0 1
Col


Input
0 1 44
1 0 55
2 0 22
3 0 11
4 2 33
0 1
1st 2nd Index
Sort by 1st index of vector
Ascending order
output
0 0 55
1 0 22
2 0 11
3 1 44
4 2 33
0 1


Input
0 1 44
1 0 55
2 0 22
3 0 11
4 2 33
0 1
1st 2nd Index
Sort by 2nd index of vector
Descending order
output
0 0 55
1 1 44
2 2 33
3 0 22
4 0 11
0 1



```cpp
// ✅CUTOM COMPARATOR 02: SORT VECTOR OF VECTOR
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

// Printing Method

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

// Custom Comparator ki return value always true ya false hoti hai

bool myComparator1(vector<int> &a, vector<int> &b) {
    return a[0] > b[0]; // decreasing order sorting by 1st index
}

bool myComparator2(vector<int> &a, vector<int> &b) {
    return a[1] > b[1]; // decreasing order sorting by 2nd index
}

int main() {
    // vector of vector sorting
    vector<vector<int>> v;

    int n;
    cout << "Enter size:\n";
    cin >> n;

    // Taking input from user in 2D vector
    for (int i = 0; i < n; ++i) {

        int a, b;
        cout << "Enter a, b" << endl;
        cin >> a >> b;

        // Creation of 1D vector temp
        vector<int> temp;

        // Inserting element a at 0th and b at 1st index in 1D vector
        temp.push_back(a);
        temp.push_back(b);

        // Inserting 1D vector temp in 2D vector v
        v.push_back(temp);
    }

    cout << "Here are the Values" << endl;
    print(v);

    cout << "Sorted by 1st index" << endl;
    sort(v.begin(), v.end(), myComparator1);
    print(v);

    cout << "Sorted by 2st index" << endl;
    sort(v.begin(), v.end(), myComparator2);
    print(v);

    return 0;
}
```

```cpp
bool myComparator1(vector<int> &a, vector<int> &b) {
    return a[0] < b[0]; // increasing order sorting by 1st index
}

bool myComparator2(vector<int> &a, vector<int> &b) {
    return a[1] < b[1]; // increasing order sorting by 2nd index
}
```