Sorting Algo

Date 24-02-2023

Selection sort :- Selection sort is the basic on the idea that, find the smallest number & put in the first place, then find the next smallest number & put it on the second place & so on.

It is named so because in each pass it selects the smallest element & keeps it in its exact place.

For example :-

82	42	49	8	25	52	36	93	59
0	1	2	3	4	5	6	7	8

First we find the smallest element in this whole array & place it in the first position.

8	42	49	82	25	52	36	93	59
0	1	2	3	4	5	6	7	8

8 is in its correct position.

now we find the smallest element from the rest of the array & place it in the right position.

Date..................

8	25	49	82	42	52	36	93	59
0	1	2	3	4	5	6	7	8

8 & 25 is in their right positions.

now find the smallest element from the rest of the array & place it in its right position.

This process will continue "n-2" times. At last we got our sorted array.

8	25	36	42	49	52	59	82	93
0	1	2	3	4	5	6	7	8

sorted array.

code:-

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

void selectionSort(int arr[], int n)
{
    for(int i = 0; i < n; i++){
        int minIndex = i;
        for(int j = i+1; j < n; j++){
            if( arr[j] < arr[minIndex])
            {
                minIndex = j;
            }
        }
        if(i != minIndex){
            swap(arr[i], arr[minIndex]);
        }
    }
}

int main()
{
    int arr[] = {82, 42, 49, 8, 25, 52 };
    int n = 6;

    selectionSort(arr, n);

    for(int i = 0; i < n; i++){
        cout << arr[i] << " ";
    }
}
```

* **Bubble sort** :- Bubble sort is a sorting algorithm, that iterates through a given array & compares each pair of adjacent elements one after the other.
If any of the adjacent pairs, if the first element is greater than the second element, then it swaps the elements & if not then it moves on to the next pair of elements.

Iteration 1,

0	1	2	3	4
8	5	2	6	12

compare these 2 elements.
8 > 5. so, they get swapped.

5	8	2	6	12

8 > 2. so, they get swapped

5	2	8	6	12

8 > 6. so they get swapped.

5	2	6	8	12

8 < 12. No need to swap.

2 5 6 8 12

Iteration 2,

5	2	6	8	12

5 > 2, so they get swapped.

2	5	6	8	12

sorted array.

After 1st iteration, the elements are not sorted. It means, we have to keep repeating the set of actions again & again until the entire array of elements are sorted. Generally, it takes n-1 iteration in order to sort a array using Bubble sort algorithm, where n is the no. of elements in the array.

code:-

```
#include<iostream>
using namespace std;
int main()
{
    int arr[] = {8, 5, 2, 6, 12};
    int n = 5;

    for(int i = 0; i < n; i++)
    {
      for(int j = 0; j < n-1; j++)
      {
        if(arr[j] > arr[j+1])
        {
          swap(arr[j], arr[j+1]);
        }
      }
    }

    cout << "Sorted: ";
    for(int i = 0; i < n; i++)
    {
      cout << arr[i] << " ";
    }
    return 0;
}
```

* Insertion sort :- Assume, first element is sorted & rest of the rest right side elements are unsorted.
It proceeds by inserting each element at the proper place in a sorted list.
We will consider our list to be divided into two parts - sorted & unsorted. Initially the sorted part contains only the first element of the list & unsorted part contains the rest of the elements.
In each pass, the first element from the unsorted part is taken & inserted into the sorted part at appropriate place.
If there are n elements in the list, then after n-1 passes, the unsorted part dissappears & our whole list becomes sorted.

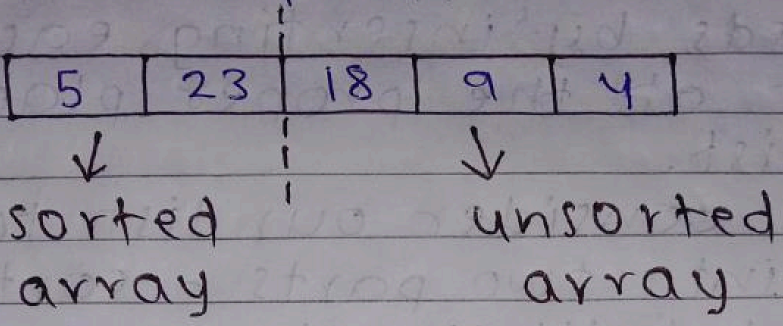
5	23	18	9	4

sorted array

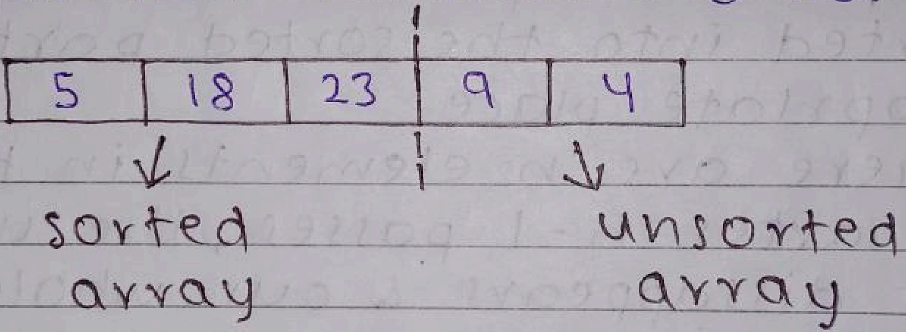
unsorted array

consider the first element, i.e., 5 as sorted array as there are no other elements on its left hand side.

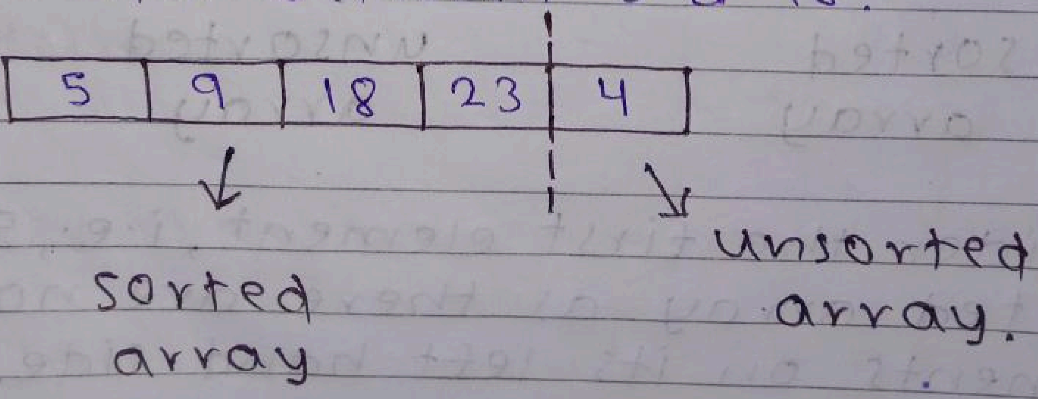
Now, we pick first element from unsorted array, i.e., 23 & compare with sorted array, 23 > 5. So, put 23 in sorted part.



Pick first element for unsorted array, i.e., 18 & compare with sorted array elements, 18 < 23, then we again compare with previous element, 18 > 5. Means, 18 lies between 5 & 23.



Pick first element from unsorted array, i.e, 9 & compare it with sorted array elements, 9 < 83, 9 < 18, 9 > 5. Means, 9 lies between 5 & 18.



Pick the element from unsorted array & compare with sorted array elements. 4 < 23, 4 < 18, 4 < 9, 4 < 5. Means 4 is placed at 0th index.

$$\underbrace{\begin{array}{|c|c|c|c|c|}\hline 4 & 5 & 9 & 18 & 23 \\ \hline\end{array}}_{\text{sorted array.}}$$

code:-

```
#include<iostream>
using namespace std;
void insertionSort(int arr[], int n)
{
    int i, j;
    for(i = 0; i < n; i++)
    {
        j = i;
        while(j > 0 && arr[j-1] >
                              arr[j])
        {
            swap(arr[j], arr[j-1]);
            j--;
        }
    }
}
int main()
{
  int arr[] = {5, 23, 18, 9, 4 };
  int n = 5;

  insertionSort(arr, n);

  cout << "Sorted:";
  for(int i = 0; i < n; i++)
  {
     cout << arr[i] << " ";
  }
  return 0;
}
```