# Selection Sort

Selection sort is conceptually the simplest sorting algorithm. This algorithm will first find the smallest element in the array and swap it with the element in the first position, then it will find the second smallest element and swap it with the element in the second position, and it will keep on doing this until the entire array is sorted.

It is called selection sort because it repeatedly selects the next-smallest element and swaps it into the right place.

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. It is also the simplest algorithm. It is an in-place comparison sorting algorithm. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right.

In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

Selection sort is generally used when –

A small array is to be sorted.

It is compulsory to check all elements.

## • Finding Smallest Element in a subarray

In selection sort, in the first step, we look for the smallest element in the array and replace it with the element at the first position. This seems doable, but

Consider that you have an array with values {3, 6, 1, 8, 4, 5}. Now as per selection sort, we will start from the first element and look for the smallest number in the array, which is 1 and we will find it at the index 2. Once the smallest number is found, it is swapped with the element at the first position.

Well, in the next iteration, we will have to look for the second smallest number in the array. How can we find the second smallest number? This one is tricky?

If you look closely, we already have the smallest number/element at the first position, which is the right position for it and we do not have to move it anywhere now. So, we can say, that the first element is sorted, but the elements to the right, starting from index 1 are not.

So, we will now look for the smallest element in the subarray, starting from index 1, to the last index.

After we have found the second smallest element and replaced it with element on index 1(which is the second position in the array), we will have the first two positions of the array sorted.

Then we will work on the subarray, starting from index 2 now, and again looking for the smallest element in this subarray.

- **How Selection Sort Works?**

Following are the steps involved in selection sort(for sorting a given array in ascending order):

Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.

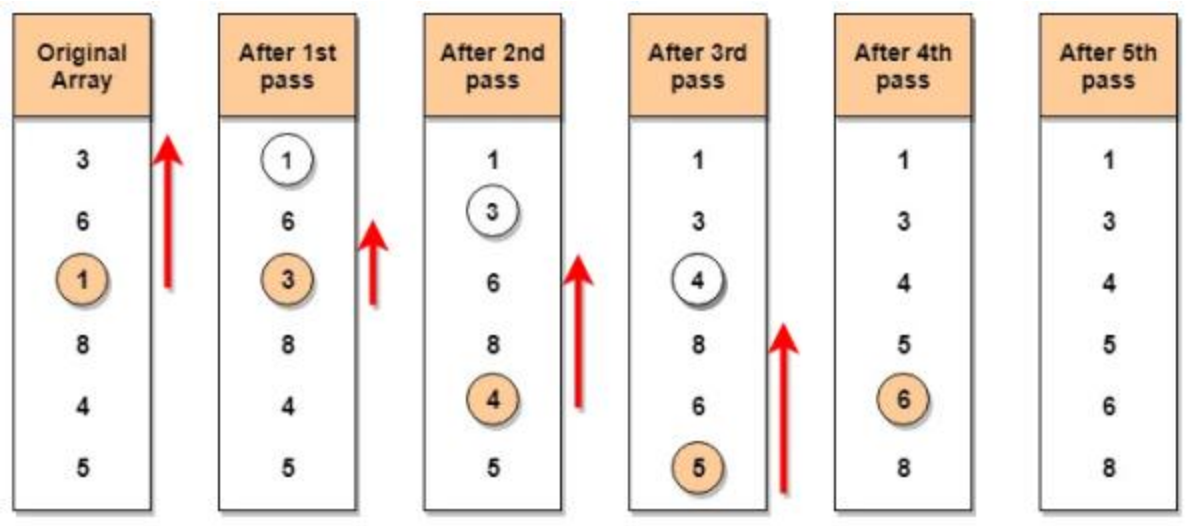
We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.

We replace the element at the second position in the original array, or we can say at the first position in the subarray, with the second smallest element.

This is repeated, until the array is completely sorted.

Let's consider an array with values {3, 6, 1, 8, 4, 5}

Below, we have a pictorial representation of how selection sort will sort the given array.



In the first pass, the smallest element will be 1, so it will be placed at the first position.

Then leaving the first element, next smallest element will be searched, from the remaining elements. We will get 3 as the smallest, so it will be then placed at the second position.

Then leaving 1 and 3(because they are at the correct position), we will search for the next smallest element from the rest of the elements and put it at third position and keep doing this until array is sorted.

- **Program of Selection sort:**

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

class Selection {
public:
   void selection(int a[], int n) {
      int i, j, small;
      for (i = 0; i < n - 1; i++) {
            small = i; // index of the minimum element in the unsorted array
            for (j = i + 1; j < n; j++) {
                if (a[j] < a[small])
                    small = j;
            }

            // Swap the minimum element with the first element
            int temp = a[small];
            a[small] = a[i];
            a[i] = temp;
        }
    }

    void printArr(int a[], int n) {
        for (int i = 0; i < n; i++)
            cout << a[i] << " ";
    }
};

int main() {
    int a[] = {91, 49, 4, 19, 10, 21};
    int n = sizeof(a) / sizeof(a[0]);

    Selection s1;
    cout << "Before sorting array elements are - " << endl;
    s1.printArr(a, n);

    s1.selection(a, n);

    cout << "\nAfter sorting array elements are - " << endl;
    s1.printArr(a, n);
    cout << endl;

    return 0;
}
```

- **Example 2:**

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

int findSmallest(int myarray[],int i)
{
  int ele_small,position,j;
  ele_small = myarray[i];
  position = i;
  for(j=i+1;j<5;j++)
  {
    if(myarray[j]<ele_small)
    {
    ele_small = myarray[j];
    position=j;
    }
  }
  return position;
}
int main ()
{
  int myarray[5] = {12,45,8,15,33};
  int pos,temp;
  cout<<"\n Input list of elements to be Sorted\n";
  for(int i=0;i<5;i++)
  {
    cout<<myarray[i]<<"\t";
  }
  for(int i=0;i<5;i++)
  {
    pos = findSmallest (myarray,i);
    temp = myarray[i];
    myarray[i]=myarray[pos];
    myarray[pos] = temp;
  }
  cout<<"\n Sorted list of elements is\n";
  for(int i=0;i<5;i++)
  {
    cout<<myarray[i]<<"\t";
  }
return 0;
}
```

**Note:** Selection sort is an unstable sort i.e it might change the occurrence of two similar elements in the list while sorting. But it can also work as a stable sort when it is implemented using linked list