SELECTION SORT

- Selection sort is conceptually the most 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.

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

SELECTION SORT

- Selection sort is one of the easiest approaches to sorting.

- It is inspired from the way in which we sort things out in day to day life.

- It is an in-place sorting algorithm because it uses no auxiliary data structures while sorting.

SELECTION SORT

Selection sort works as:-

- It finds the first smallest element.
- It swaps it with the first element of the unordered list.
- It finds the second smallest element.
- It swaps it with the second element of the unordered list.
- Similarly, it continues to sort the given elements.

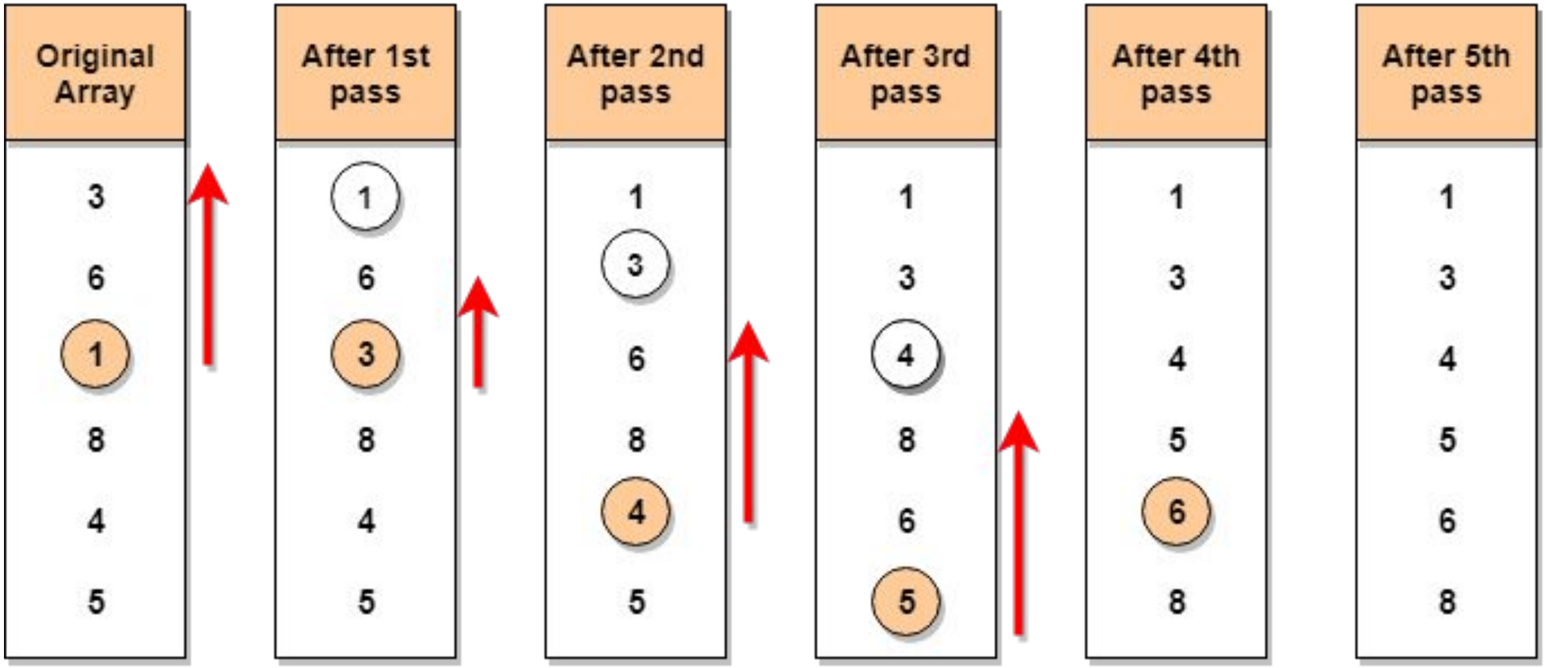
SELECTION SORT

Algorithm :-

```
selectionSort(array, size)
  repeat (size - 1) times
  set the first unsorted element as the minimum
  for each of the unsorted elements
    if element < currentMinimum
      set element as new minimum
  swap minimum with first unsorted position
end selectionSort
```

SELECTION SORT

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

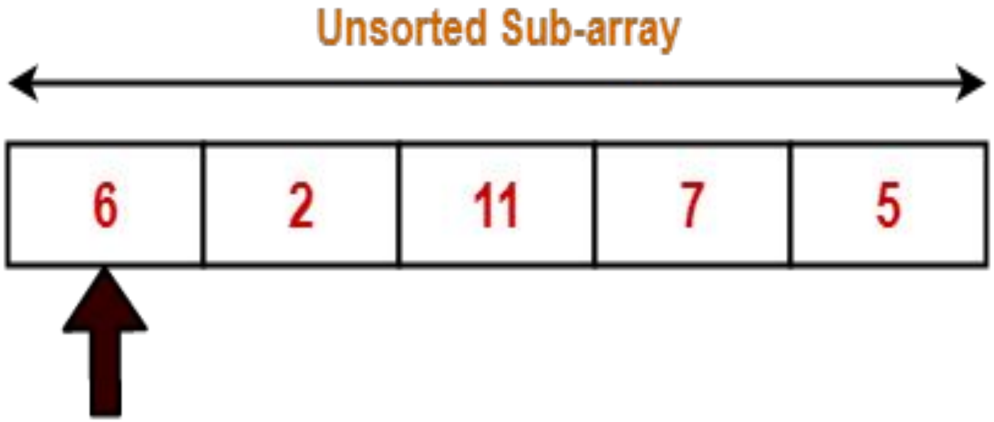


Original Array	After 1st pass	After 2nd pass	After 3rd pass	After 4th pass	After 5th pass
3	1	1	1	1	1
6	6	3	3	3	3
1	3	6	4	4	4
8	8	8	8	5	5
4	4	4	6	6	6
5	5	5	5	8	8

SELECTION SORT

Ex-2 :- 6, 2, 11, 7, 5

Step-01: For i = 0



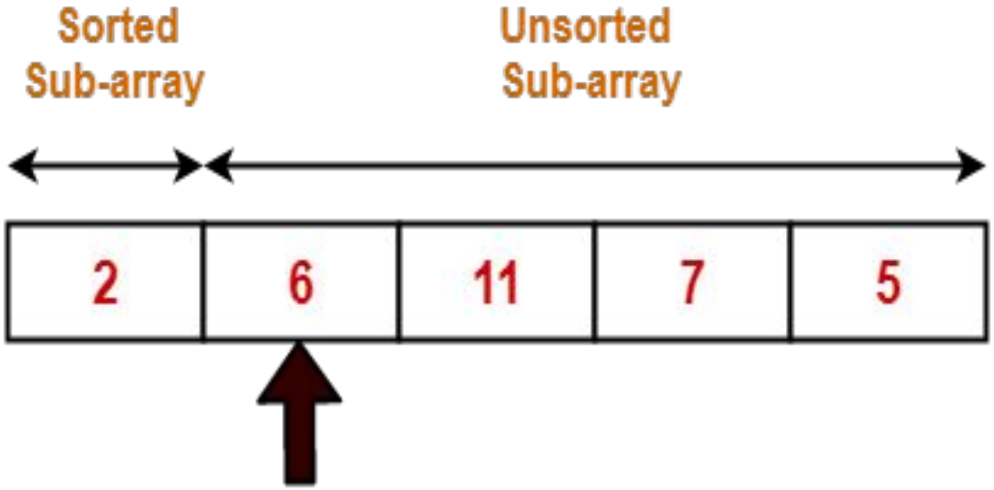
Unsorted Sub-array

6	2	11	7	5

We start here, find the minimum element and swap it with the 1st element of array

SELECTION SORT

Step-02: For i = 1

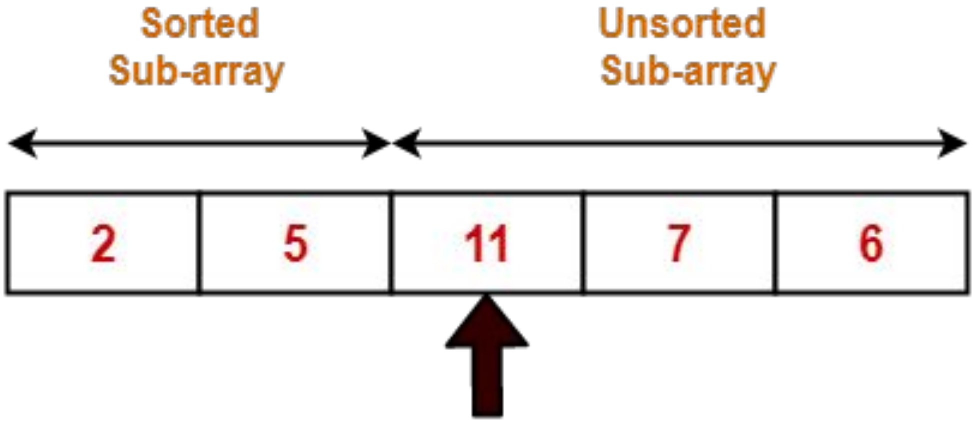


2	6	11	7	5

We start here, find the minimum element and swap it with the 2nd element of array

SELECTION SORT

Step-03: For i = 2

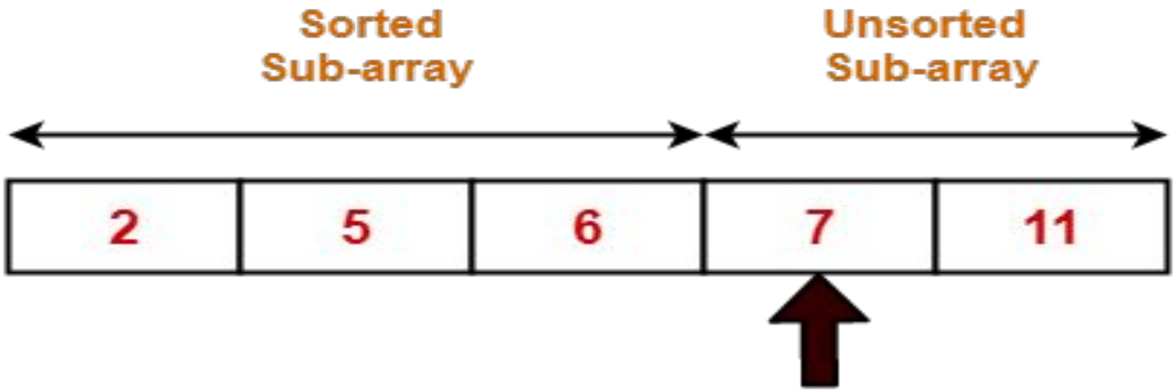


2	5	11	7	6

We start here, find the minimum element and swap it with the 3rd element of array

SELECTION SORT

Step-04: For i = 3



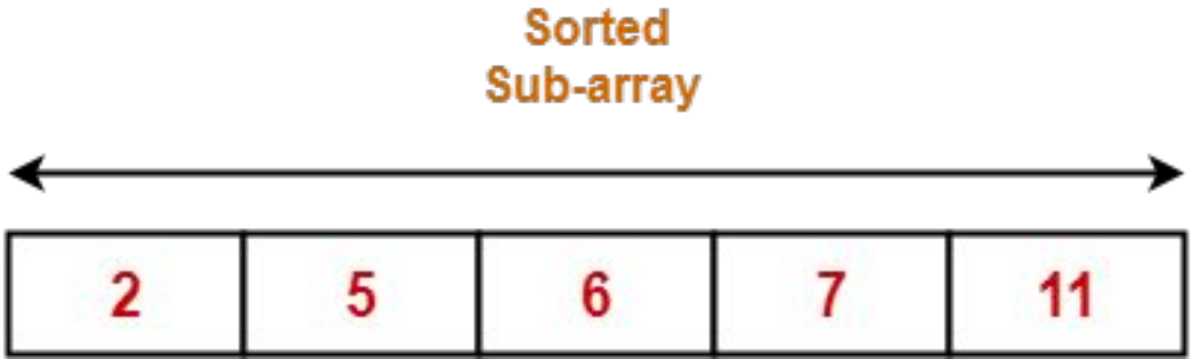
2	5	6	7	11

We start here, find the minimum element but there is no need to swap
(4th element is itself the minimum)

SELECTION SORT

Step-05: For i = 4

- Loop gets terminated as ‘i’ becomes 4.
- The state of array after the loops are finished is as shown-



2	5	6	7	11

SELECTION SORT

Time Complexity :-

- Selection sort algorithm consists of two nested loops.

- Owing to the two nested loops, it has O(n^2) time complexity.

	Time Complexity
Best Case	n^2
Average Case	n^2
Worst Case	n^2

SELECTION SORT

Selection Sort Logic for implementation :-

```
for (i = 0 ; i < n-1 ; i++)
{
    index = i;
    for(j = i+1 ; j < n ; j++)
    {
        if(A[j] < A[index])
        index = j;
    }
    temp = A[i];
    A[i] = A[index];
    A[index] = temp;
}
```

Here,
i = variable to traverse the array A
index = variable to store the index of minimum element
j = variable to traverse the unsorted sub-array
temp = temporary variable used for swapping

Selection Sort implementation

```
#include <stdio.h>
int main()
{
    int a[100], n, i, j, position, swap;
    printf("Enter number of elementsn");
    scanf("%d", &n);
    printf("Enter %d Numbersn", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n - 1; i++)
    {
        position=i;
        for(j = i + 1; j < n; j++)
        {
            if(a[position] > a[j])
            position=j;
        }
        if(position != i)
        {
            swap=a[i];
            a[i]=a[position];
            a[position=swap;
        }
    }
    printf("Sorted Array:n");
    for(i = 0; i < n; i++)
         printf("%dn", a[i]);
    return 0;
}
```

Selection Sort implementation

```
#include <stdio.h>
 void swap(int *a, int *b)
 {
        int tmp = *a;
        *a = *b;
        *b = tmp;
}
//Selection sort function
void selectionSort(int arr[], int n)
{
         for (int j = 0; j< n - 1; j++)
        {
             int min = j;
             for (int i = j + 1; i < n; i++)
             {
                  if (arr[i] < arr[min])
                   min = i;
             }
             swap(&arr[min], &arr[j]);
         }
}
```

```
void display(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
 }

int main()
{
    int arr[] = {20, 12, 10, 15, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Elements before sorting: \n");
    selectionSort(arr, n);
    printf("Elements after sorting:\n");
    display(arr, n);
}
```