Selection Sort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.

2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the end of sorted subarray.

```
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

/* In every pass the Current Minimum element will be placed in its
appropriate position. */

void selectionSort(int *arr, int n)
{
    int i, j, min_idx;

/* i is the position index where the Current Minimum element will be
placed.Initial value of i = 0 and final value of i = n-2. */

/* j is used to travel the remaining portion of the array from which the
Current Minimum element will be searched. Initial value of j = i+1 and
final value of j = n-1. */

    for (i = 0; i < n-1; i++)
    {
/* Find the minimum element from the remaining unsorted portion of the
array.*/
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;
/*min_index : position index of the Current Minimum element */
/* Swap the found minimum element with the i th element. */
        swap(&arr[min_idx], &arr[i]); /* swap(arr+min_idx, arr+i); */

/*Size of the remaining portion of the array will be reduced by one after
the completion of a pass.*/
    }
}
```

```
/* Function to print an array */
void printArray(int *arr, int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}



// Driver program to test above functions

int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Complexity Analysis:

No. of comparisons required to find the 1st minimum element from the unsorted part in the 1st pass = n-1

No. of comparisons required to find the 2nd minimum element from the unsorted part in the 2nd pass = n-2
…

No. of comparisons required to find the i^{th} minimum element from the unsorted part in the i^{th} pass = n-i
…

No. of comparisons required to find the $(n-1)^{th}$ minimum element from the unsorted part in the $(n-1)^{th}$ pass = 1

Total no. of passes required = n-1
(Because, if (n-1) elements are placed in their appropriate position then the remaining element will be automatically placed in its correct position)

$$\therefore \textit{Total no. of comparisons required} = (n-1) + (\mathrm{n}-2) + \ldots + 2 + 1$$

$$= \frac{n}{2}\left(n-1\right) = O(n^2)$$