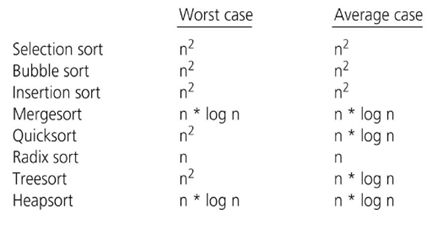
## **Time Complexities of Sorting Algorithms**

## **Why sorting algorithms?**

1. Sorting is a simple and well-defined problem, which makes it perfect for studying.
2. Sorting algorithms cover important concepts like **divide-and-conquer**, **data structures**, and **randomized algorithms**.
3. Sorting is a common computer task. At one point a quarter of all mainframe cycles were spent sorting.

**Comparison of Sort Algorithms :**

People ask the ageless question: Which sorting algorithm is the fastest?



**The sorting functions have the following prototype:**

void sort(int arr[], int n)

• arr is the array to be sorted.

• n is the number of elements in arr.

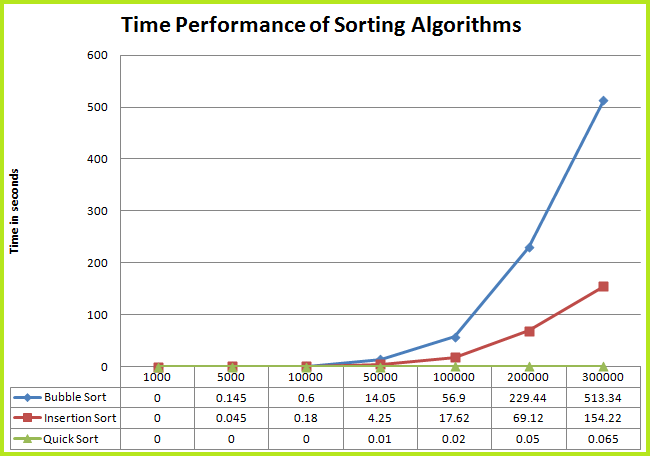
**The function name :**

* void BubbleSort(int arr[],int n) { }
* void SelectionSort(int arr[], int n) { }
* void InsertionSort(int arr[],int n) { }
* void MergeSort(int arr[],int n) { }
* void Merge(int left[],int right[],int arr[]) { }
* void QuickSort(int arr[],int start,int end) { }
* int QuickPartition(int arr[],int start,int end) {}

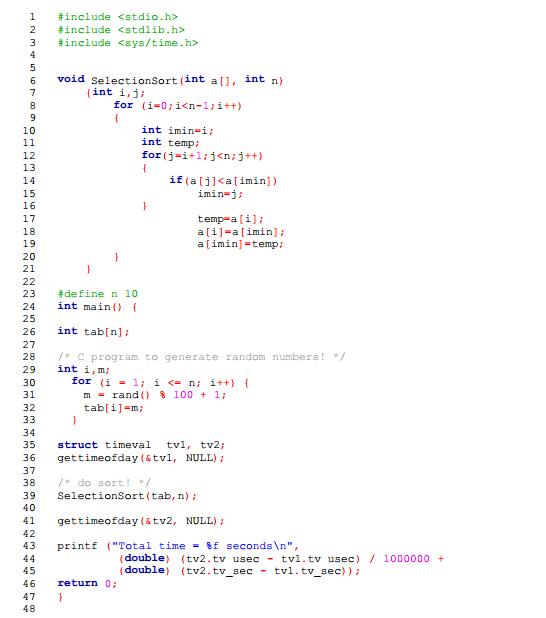
**Algorithm analysis :**

To defines the time complexity of the algorithm we use the execution time of C program under varying the variable n (n=1000, n=5000, n=10000, n= 50000, n=1000000, n=2000000, n=3000000)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1000 | 5000 | 10000 | 50000 | 1000000 | 2000000 | 3000000 |
| BubbleSort |  |  |  |  |  |  |  |
| SelectionSort |  |  |  |  |  |  |  |
| InsertionSort |  |  |  |  |  |  |  |



Here is a program that calculates the time required to sort an array with the Selection sort algorithm



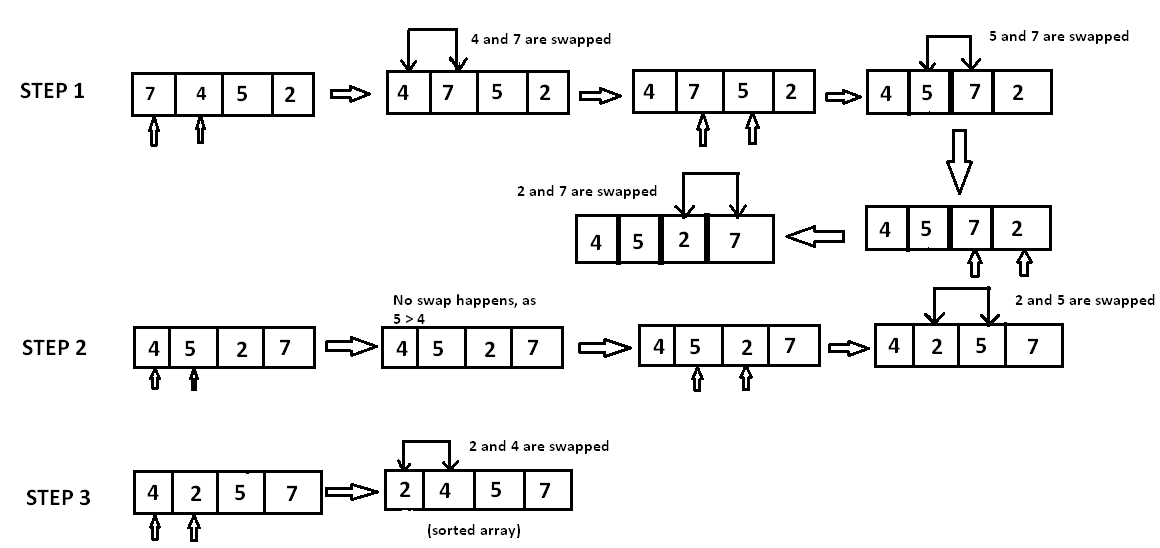
**void** InsertionSort(**int** a[],**int** n)

{ ……. }

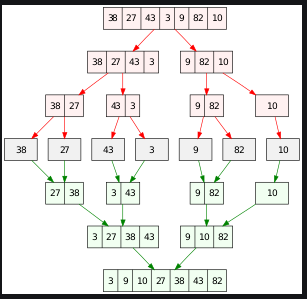
**Add the C program of insertion sort :**

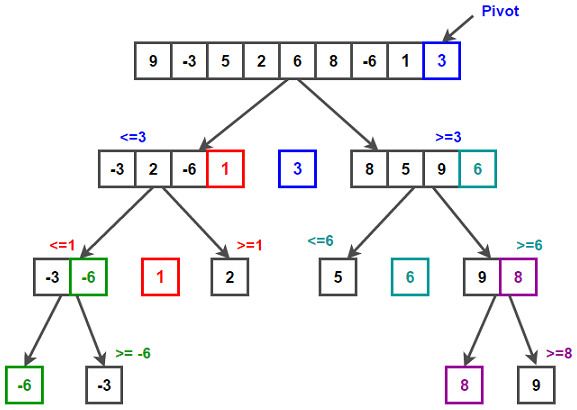
|  |
| --- |
|  |

* **Bubble Sort**



* **MergeSort**



* **QuickSort**