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?

	Worst case	Average case
Selection sort	n^2	n^2
Bubble sort	n^2	n^2
Insertion sort	n^2	n^2
Mergesort	n * log n	n * log n
Quicksort	n^2	n * log n
Radix sort	n	n
Treesort	n^2	n * log n
Heapsort	n * log n	n * log n

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.

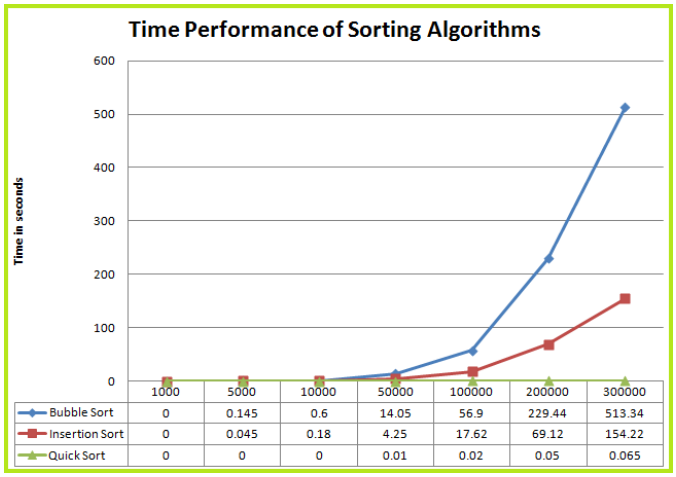
The function name :

- 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							



	1000	5000	10000	50000	100000	200000	300000
Bubble Sort	0	0.145	0.6	14.05	56.9	229.44	513.34
Insertion Sort	0	0.045	0.18	4.25	17.62	69.12	154.22
Quick Sort	0	0	0	0.01	0.02	0.05	0.065

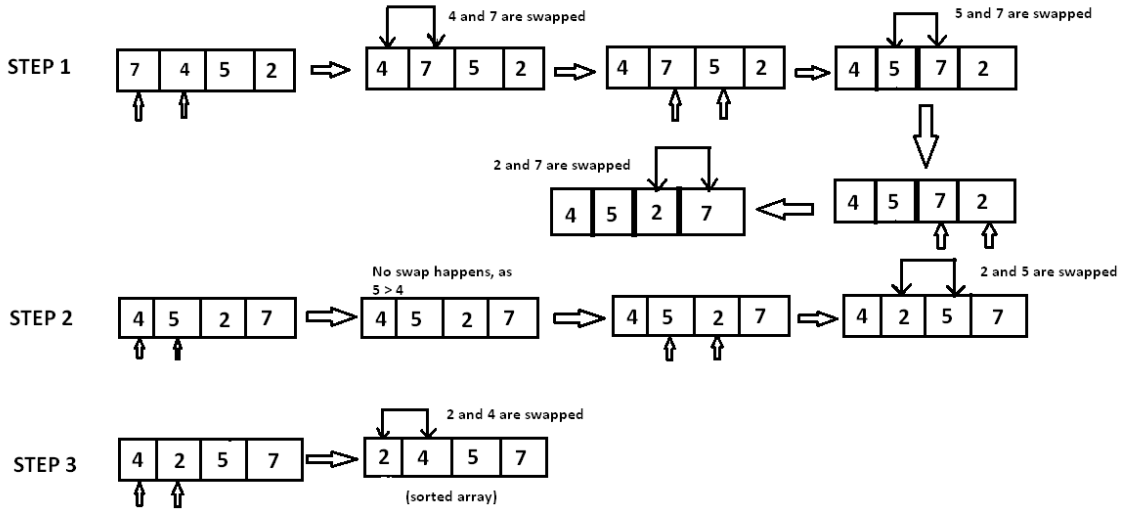
Here is a program that calculates the time required to sort an array with the Selection sort algorithm

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>


void SelectionSort(int a[], int n)
    {int i,j;
        for (i=0;i<n-1;i++)
        {
            int imin=i;
            int temp;
            for(j=i+1;j<n;j++)
            {
                if(a[j]<a[imin])
                    imin=j;
            }
                    temp=a[i];
                    a[i]=a[imin];
                    a[imin]=temp;
        }
    }

#define n 10
int main() {

int tab[n];

/* C program to generate random numbers! */
int i,m;
  for (i = 1; i <= n; i++) {
    m = rand() % 100 + 1;
    tab[i]=m;
  }

struct timeval  tv1, tv2;
gettimeofday(&tv1, NULL);

/* do sort! */
SelectionSort(tab,n);

gettimeofday(&tv2, NULL);

printf ("Total time = %f seconds\n",
         (double) (tv2.tv_usec - tv1.tv_usec) / 1000000 +
         (double) (tv2.tv_sec - tv1.tv_sec));
return 0;
}
```

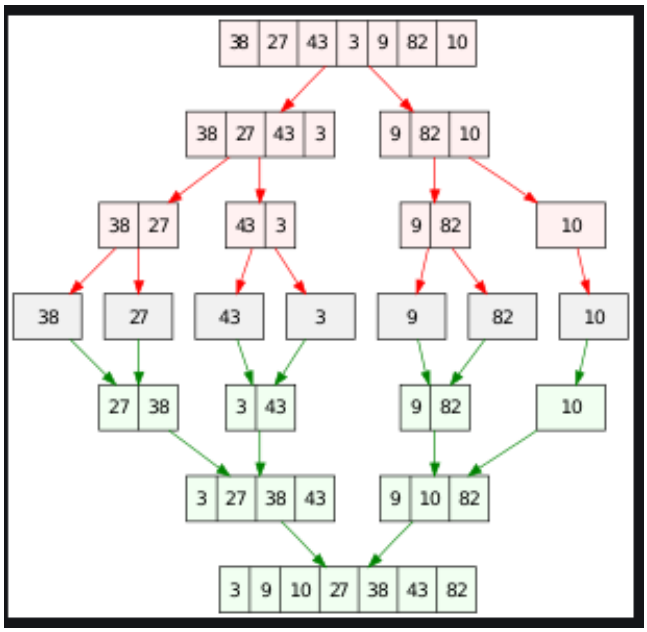
void InsertionSort(**int** a[],**int** n)
{ ……. }

Add the C program of insertion sort :

- **Bubble Sort**



- **MergeSort**



- **QuickSort**