

# 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?

	<u>Worst case</u>	<u>Average case</u>
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.

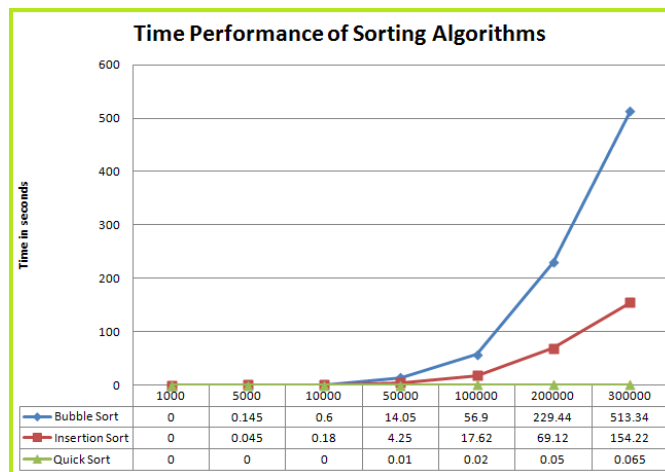
### The function name :

- 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							



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

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4
5
6  void SelectionSort(int a[], int n)
7  {
8      int i, j;
9      for (i=0; i<n-1; i++)
10     {
11         int imin=i;
12         int temp;
13         for(j=i+1; j<n; j++)
14         {
15             if(a[j]<a[imin])
16                 imin=j;
17         }
18         temp=a[i];
19         a[i]=a[imin];
20         a[imin]=temp;
21     }
22 }
23
24 #define n 10
25 int main() {
26     int tab[n];
27
28     /* C program to generate random numbers! */
29     int i, m;
30     for (i = 1; i <= n; i++) {
31         m = rand() % 100 + 1;
32         tab[i]=m;
33     }
34
35     struct timeval tv1, tv2;
36     gettimeofday(&tv1, NULL);
37
38     /* do sort! */
39     SelectionSort(tab, n);
40
41     gettimeofday(&tv2, NULL);
42
43     printf ("Total time = %f seconds\n",
44            (double) (tv2.tv_usec - tv1.tv_usec) / 1000000 +
45            (double) (tv2.tv_sec - tv1.tv_sec));
46     return 0;
47 }
48

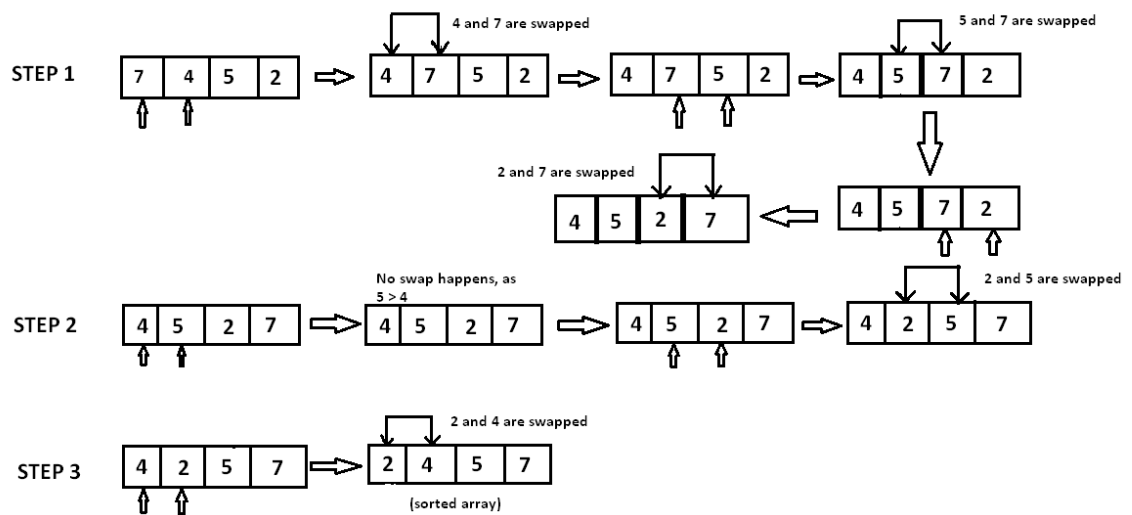
```

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

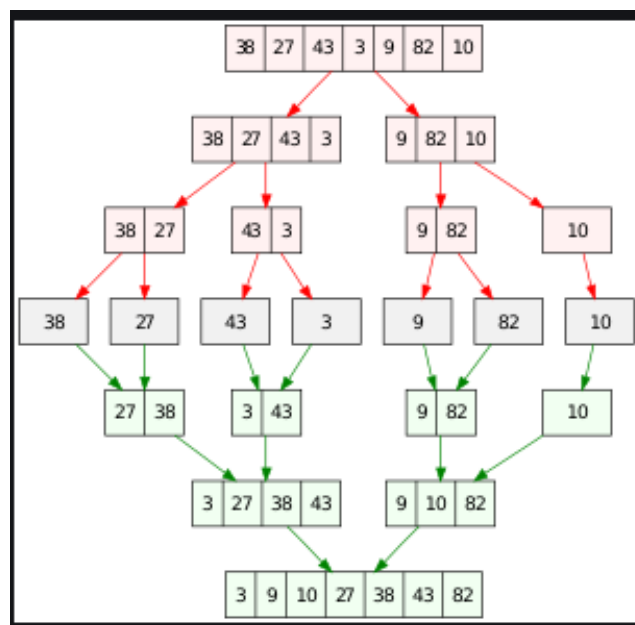
Add the C program of insertion sort :



- Bubble Sort



- MergeSort



- QuickSort

