

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB REPORT

on

# Analysis and Design of Algorithms

*Submitted by*

**Akash M (1BM20CS006)**

*in partial fulfillment for the award of the degree of*  
**BACHELOR OF ENGINEERING**  
*in*  
**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

(Autonomous Institution under VTU)

**BENGALURU-560019**

**May-2022 to July-2022**

**B. M. S. College of Engineering,  
Bull Temple Road, Bangalore 560019**  
(Affiliated To Visvesvaraya Technological University, Belgaum)  
**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled "**Analysis and Design of Algorithms**" carried out by **Akash M(1BM20CS006)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of **Analysis and Design of Algorithms - (19CS4PCADA)** work prescribed for the said degree.

**Dr. Prasad G R**  
Professor  
Department of CSE  
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**  
Professor and Head  
Department of CSE  
BMSCE, Bengaluru

## Index Sheet

<b>Sl. No.</b>	<b>Experiment Title</b>	<b>Page No.</b>
<b>1</b>	Write a recursive program to Solve <b>a)</b> Towers-of-Hanoi problem <b>b)</b> To find GCD	
<b>2</b>	Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.	
<b>3</b>	Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	
<b>4</b>	Write program to do the following: <b>a)</b> Print all the nodes reachable from a given starting node in a digraph using BFS method. <b>b)</b> Check whether a given graph is connected or not using DFS method.	
<b>5</b>	Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.	
<b>6</b>	Write program to obtain the Topological ordering of vertices in a given digraph.	
<b>7</b>	Implement Johnson Trotter algorithm to generate permutations.	
<b>8</b>	Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	
<b>9</b>	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	
<b>10</b>	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	
<b>11</b>	Implement Warshall's algorithm using dynamic programming	
<b>12</b>	Implement 0/1 Knapsack problem using dynamic programming.	
<b>13</b>	Implement All Pair Shortest paths problem using Floyd's algorithm.	
<b>14</b>	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	
<b>15</b>	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	
<b>16</b>	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	
<b>17</b>	Implement "Sum of Subsets" using Backtracking. "Sum of Subsets" problem: Find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers whose sum is equal to a given positive integer d. For example, if $S = \{1, 2, 5, 6, 8\}$ and $d = 9$ there are two solutions	

	{1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn't have a solution.	
<b>18</b>	Implement “N-Queens Problem” using Backtracking.	

## Course Outcome

<b>CO1</b>	Ability to <b>analyze</b> time complexity of Recursive and Non-Recursive algorithms using asymptotic notations.
<b>CO2</b>	Ability to <b>design</b> efficient algorithms using various design techniques.
<b>CO3</b>	Ability to <b>apply</b> the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete
<b>CO4</b>	Ability to <b>conduct</b> practical experiments to solve problems using an appropriate designing method and find time efficiency.

```
1 //1. Write a recursive program to
2 //a. Solve Towers-of-Hanoi problem
3
4 #include<stdio.h>
5
6 void toh(int n,char s,char d,char a){
7     if(n==1){
8         printf("Moving from %c to %c\n",s,d);
9         return;
10    }
11    toh(n-1,s,a,d);
12    printf("Moving from %c to %c\n",s,d);
13    toh(n-1,a,d,s);
14}
15
16 int main(){
17     int n;
18     char s,d,a,temp;
19     printf("Number of Discs :");
20     scanf("%d",&n);
21     printf("Enter the Names of Source Destination and Aux towers :");
22     scanf("%c",&temp);
23     scanf("%c %c %c",&s,&d,&a);
24     toh(n,s,d,a);
25 }
```

## **Output:**

---

Number of Discs :5

Enter the Names of Source Destination and Aux towers :a c b

Moving from a to c

Moving from a to b

Moving from c to b

Moving from a to c

Moving from b to a

Moving from b to c

Moving from a to c

Moving from a to b

Moving from c to b

Moving from c to a

Moving from b to a

Moving from c to b

Moving from a to c

Moving from a to b

Moving from c to b

Moving from a to c

Moving from b to a

Moving from b to c

Moving from a to c

Moving from b to a

Moving from c to b

Moving from c to a

Moving from b to a

Moving from c to b

Moving from c to a

Moving from b to a

Moving from b to c

Moving from a to c

Moving from a to b

Moving from c to b

Moving from a to c

Moving from b to a

Moving from b to c

Moving from a to c

```
1 //1. Write a recursive program to
2 //b. To find GCD
3 #include<stdio.h>
4
5 int gcd(int n, int m){
6     if(m!=0){
7         return gcd(m,n%m);
8     }
9     else{
10        return n;
11    }
12 }
13
14
15 int main(){
16     int n,m;
17     printf("Enter number for which GCD to be found :");
18     scanf("%d %d",&n,&m);
19     int res=gcd(n,m);
20     printf("GCD of %d and %d is %d\n",n,m,res);
21 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :366 60
GCD of 366 and 60 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :6 3
GCD of 6 and 3 is 3
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :66 6
GCD of 66 and 6 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :78 6
GCD of 78 and 6 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :78 12
GCD of 78 and 12 is 6
```

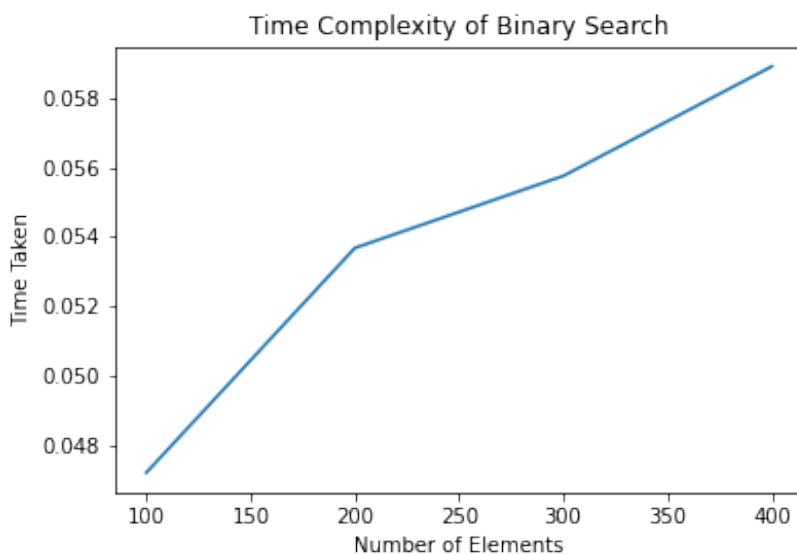
```
1 //2.a. Implement Recursive Binary search and determine the time required to
2 //search an element. Repeat the experiment for different values of N and plot a
3 //graph of the time taken versus N.
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include<time.h>
8
9 void delay(){
10    for (int i = 0; i < 5000000; ++i)
11    {
12        int d=i/33;
13    }
14
15 int binarySearch(int arr[], int k, int si, int ei){
16    if(si>ei){
17        return -1;
18    }
19    delay();
20    int mid=(si+ei)/2;
21    if(arr[mid]==k){
22        return mid;
23    }
24    else if(arr[mid]>k){
25        return binarySearch(arr,k,si,mid-1);
26    }
27    else{
28        return binarySearch(arr,k,mid+1,ei);
29    }
30
31 void swap(int arr[], int i, int j){
32    if(i==j){
33        return;
34    }
35    int temp=arr[i];
36    arr[i]=arr[j];
37    arr[j]=temp;
38 }
39
40 void bubbleSort(int arr[], int n){
41    for(int i=n-1;i>=0;i--){
42        for(int j=0;j<i;j++){
43            if(arr[j]>arr[j+1]){
44                swap(arr,j,j+1);
45            }
46        }
47    }
48 }
49
50 int main(){
51    int arr[5000],n;
52    printf("Enter the number of elements in the Array :");
53    scanf("%d",&n);
```

```
54     ... , ...
55     for(int i=0;i<n;i++){
56         arr[i]=(rand()% (1 - 5000 + 1)) + 1;
57     }
58     bubbleSort(arr,n);
59     int j=binarySearch(arr,-1,0,n-1);
60     clock_t end=clock();
61
62     printf("Time taken by Binary Search is %f", (double)(end-
63     start)/CLOCKS_PER_SEC);
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :100
Time taken by Binary Search is 0.047197%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :200
Time taken by Binary Search is 0.053674%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :300
Time taken by Binary Search is 0.055753%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :400
Time taken by Binary Search is 0.058912%
```

## **Graph:**

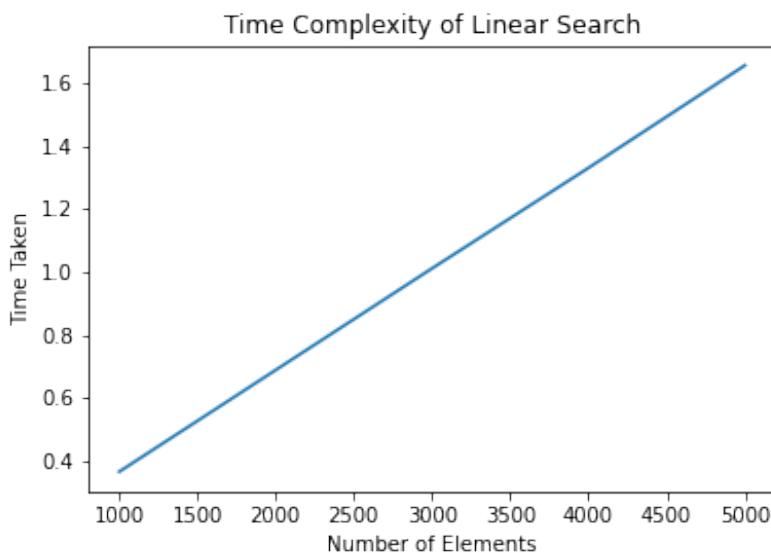


```
1 //2.b. Implement Recursive Linear search and determine the time required to
2 //search an element. Repeat the experiment for different values of N and plot a
3 //graph of the time taken versus N.
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include<time.h>
8
9 void delay(){
10     for (int i = 0; i < 500000; ++i)
11     {
12         int d=i/33;
13     }
14 }
15
16 int linearSearch(int arr[], int k,int si,int ei){
17     if(si>ei){
18         return -1;
19     }
20     if(arr[si]==k){
21         return si;
22     }
23     delay();
24     return linearSearch(arr,k,si+1,ei);
25 }
26
27 int main(){
28     int arr[5000],n;
29     printf("Enter the number of elements in the Array :");
30     scanf("%d",&n);
31     for(int i=0;i<n;i++){
32         arr[i]=(rand() % (1 - 5000 + 1)) + 1;
33     }
34     clock_t start=clock();
35     int i=linearSearch(arr,-1,0,n-1);
36     clock_t end=clock();
37
38     printf("Time taken by Linear Search is %f", (double)(end-
39     start)/CLOCKS_PER_SEC);}
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % make Lin
cc      LinearSearch.c  -o LinearSearch
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :1000
Time taken by Linear Search is 0.364711%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :2000
Time taken by Linear Search is 0.687920%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :3000
Time taken by Linear Search is 1.011175%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :4000
Time taken by Linear Search is 1.330706%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :5000
Time taken by Linear Search is 1.656666%
akashm@Akashs-MacBook-Pro ADAPrograms %
```

## **Graph:**

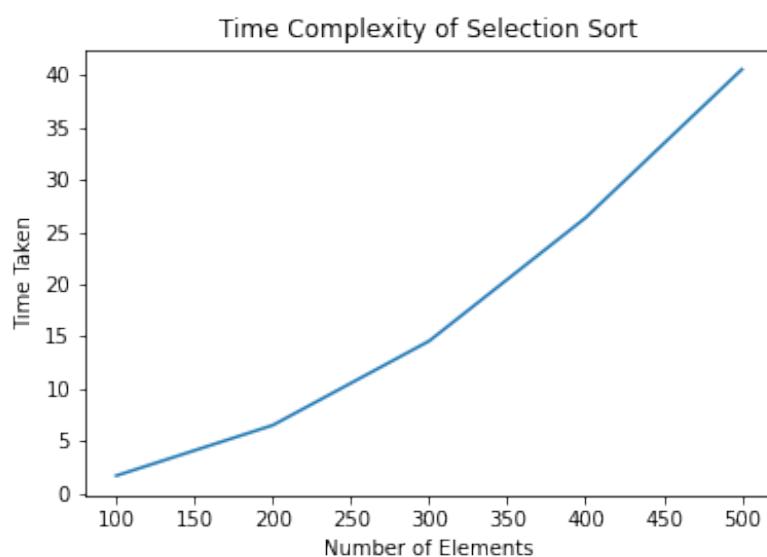


```
1 //3.Sort a given set of N integer elements using Selection Sort technique and  
2 // compute its time taken. Run the program for different values of N and record  
3 // the time taken to sort.  
4  
5 #include<stdio.h>  
6 #include<stdlib.h>  
7 #include<time.h>  
8  
9 void delay(){  
10    for (int i = 0; i < 500000; ++i)  
11    {  
12        int d=i/33;  
13    }  
14 }  
15  
16 void swap(int arr[],int i,int j){  
17    if(i==j){  
18        return;  
19    }  
20    int temp=arr[i];  
21    arr[i]=arr[j];  
22    arr[j]=temp;  
23 }  
24  
25 void selectionSort(int arr[],int n){  
26    for(int i=n-1;i>0;i--){  
27        int max=-999;  
28        int maxInd=0;  
29        for (int j = 0; j <= i; ++j)  
30        {  
31            delay();  
32            if(arr[j] > max){  
33                max=arr[j];  
34                maxInd=j;  
35            }  
36        }  
37        swap(arr,maxInd,i);  
38    }  
39 }  
40  
41 int main(){  
42    int arr[5000],n;  
43    printf("Enter the number of elements in the Array :");  
44    scanf("%d",&n);  
45    for(int i=0;i<n;i++){  
46        arr[i]=(rand() % (1 - 5000 + 1)) + 1;  
47    }  
48    clock_t start=clock();  
49    selectionSort(arr,n);  
50    clock_t end=clock();  
51  
52    printf("Time taken by Selection Sort is %f", (double)(end-  
53 start)/CLOCKS_PER_SEC);  
54 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :100
Time taken by Selection Sort is 1.663294%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :200
Time taken by Selection Sort is 6.492463%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :300
Time taken by Selection Sort is 14.554929%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :400
Time taken by Selection Sort is 26.356178%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :500
Time taken by Selection Sort is 40.571646%
akashm@Akashs-MacBook-Pro ADAPrograms % |
```

## **Graph:**



```
1 //4.Write program to do the following:  
2 //Print all the nodes reachable from a given starting node in a digraph  
3 //using BFS method.  
4  
5 #include<stdio.h>  
6 #include<stdlib.h>  
7 #include<stdbool.h>  
8  
9 int q[30];  
10 int f=-1, r=-1;  
11  
12 bool isEmpty(){  
13     if(f== -1 && r== -1){  
14         return true;  
15     }  
16     return false;  
17 }  
18  
19 void enqueue(int i){  
20     if(f== -1 && r== -1){  
21         f=0;  
22     }  
23     q[++r]=i;  
24     return;  
25 }  
26  
27 int dequeue(){  
28     if(isEmpty())  
29         return -1;  
30     }  
31     if(f==r){  
32         int temp=q[f];  
33         f=-1;  
34         r=-1;  
35         return temp;  
36     }  
37     int temp=q[f];  
38     f++;  
39     return temp;  
40 }  
41  
42  
43 void bfs(int arr[10][10], int start, int visited[], int n, int *flag){  
44     visited[start]=1;  
45     enqueue(start);  
46  
47     while(!isEmpty()){  
48         int pending=dequeue();  
49         printf("%d ", pending);  
50         for(int i=0; i<n; i++){  
51             if(arr[pending][i]==1 && visited[i]==0){  
52                 enqueue(i);  
53                 visited[i]=1;  
54                 *flag=1;
```

```
55     }
56   }
57 }
58 }

59 int main(){
60   int arr[10][10];
61   int n,m;
62   int visited[10];
63   printf("Enter number of Vertices :");
64   scanf("%d",&n);
65   printf("Enter number of Edges :");
66   scanf("%d",&m);
67   for(int i = 0; i < n; ++i){
68     for (int j = 0; j < n; ++j)
69     {
70       arr[i][j]=0;
71     }
72   }
73   for (int i = 0; i < m; ++i)
74   {
75     int a,b;
76     printf("Enter the Edge %d :",(i+1));
77     scanf("%d %d",&a,&b);
78     arr[a][b]=1;
79     arr[b][a]=1;
80   }
81   for (int i = 0; i < n; ++i)
82   {
83     visited[i]=0;
84   }
85   int start;
86   int flag=0;
87   printf("Enter the starting vertex :");
88   scanf("%d",&start);

89   bfs(arr,start,visited,n,&flag);

90   if(flag==0){
91     printf("\nNo Path Found\n");
92   }
93 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./BFS
Enter number of Vertices :5
Enter number of Edges :6
Enter the Edge 1 :1 2
Enter the Edge 2 :2 3
Enter the Edge 3 :3 4
Enter the Edge 4 :1 3
Enter the Edge 5 :0 4
Enter the Edge 6 :0 1
Enter the starting vertex :0
0 1 4 2 3 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./BFS
Enter number of Vertices :5
Enter number of Edges :4
Enter the Edge 1 :1 2
Enter the Edge 2 :2 3
Enter the Edge 3 :3 4
Enter the Edge 4 :1 3
Enter the starting vertex :0
0
No Path Found
```

```
1 //4. Write program to do the following:  
2 //b. Check whether a given graph is connected or not using DFS method.  
3 #include<stdio.h>  
4 #include<stdlib.h>  
5  
6 void dfs(int arr[10][10], int start, int visited[], int n){  
7     visited[start]=1;  
8     // printf("%d ",start);  
9     for(int i=0;i<n;i++){  
10         if(arr[start][i]==1 && visited[i]==0){  
11             dfs(arr,i,visited,n);  
12         }  
13     }  
14 }  
15  
16 int main(){  
17     int arr[10][10];  
18     int n,m;  
19     int visited[10];  
20     printf("Enter number of Vertices :");  
21     scanf("%d",&n);  
22     printf("Enter number of Edges :");  
23     scanf("%d",&m);  
24     for(int i = 0; i < n; ++i){  
25         for (int j = 0; j < n; ++j)  
26         {  
27             arr[i][j]=0;  
28         }  
29     }  
30     for (int i = 0; i < m; ++i)  
31     {  
32         int a,b;  
33         printf("Enter the Edge %d :",(i+1));  
34         scanf("%d %d",&a,&b);  
35         arr[a][b]=1;  
36         arr[b][a]=1;  
37     }  
38     for (int i = 0; i < n; ++i)  
39     {  
40         visited[i]=0;  
41     }  
42  
43     for (int i = 0; i < n; ++i)  
44     {  
45         if(i!=0 && visited[i]==0){  
46             printf("\nDisconnected Graph\n");  
47             return 0;  
48         }  
49         if(visited[i]==0)  
50             dfs(arr,i,visited,n);  
51     }  
52     printf("\nConnected Graph\n");  
53 }  
54 }
```

**Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./DFS
Enter number of Vertices :5
Enter number of Edges :3
Enter the Edge 1 :0 4
Enter the Edge 2 :3 4
Enter the Edge 3 :2 3
```

**Disconnected Graph**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./DFS
Enter number of Vertices :5
Enter number of Edges :6
Enter the Edge 1 :0 1
Enter the Edge 2 :0 4
Enter the Edge 3 :1 2
Enter the Edge 4 :2 3
Enter the Edge 5 :3 4
Enter the Edge 6 :1 3
```

**Connected Graph**

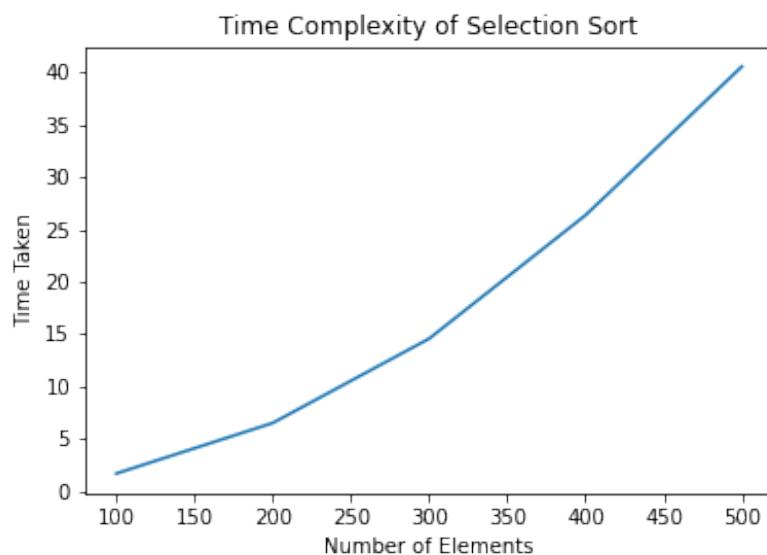
```
..... - - - - - |
```

```
1 //5. Sort a given set of N integer elements using Insertion Sort technique and  
2 // compute its time taken.  
3  
4 #include<stdio.h>  
5 #include<stdlib.h>  
6 #include<time.h>  
7  
8 void delay(){  
9     for (int i = 0; i < 500000; ++i)  
10    {  
11        int d=i/33;  
12    }  
13 }  
14  
15 void insertionSort(int arr[],int n){  
16     for (int i = 1; i < n ; ++i)  
17     {  
18         int temp=arr[i];  
19         int j;  
20         for(j=i-1;j>=0 && temp<arr[j]; j--){  
21             delay();  
22             arr[j+1]=arr[j];  
23         }  
24         arr[j+1]=temp;  
25     }  
26 }  
27  
28 int main(){  
29     int arr[5000],n;  
30     printf("Enter the number of elements in the Array :");  
31     scanf("%d",&n);  
32     for(int i=0;i<n;i++){  
33         arr[i]=(rand() % (1 - 5000 + 1)) + 1;  
34     }  
35  
36     clock_t start=clock();  
37     insertionSort(arr,n);  
38     clock_t end=clock();  
39  
40     printf("Time taken by Insertion Sort is %f", (double)(end-  
41 start)/CLOCKS_PER_SEC);  
42 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :100
Time taken by Insertion Sort is 0.896186%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :200
Time taken by Insertion Sort is 3.658319%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :300
Time taken by Insertion Sort is 8.061177%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :400
Time taken by Insertion Sort is 13.283337%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :500
Time taken by Insertion Sort is 20.731952%
```

## **Graph:**



```
1 //6. Write program to obtain the Topological ordering of vertices in a given
2 // digraph.
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<time.h>
7 #include <stdbool.h>
8
9 int q[30];
10 int f=-1, r=-1;
11
12 bool isEmpty(){
13     if(f== -1 && r== -1){
14         return true;
15     }
16     return false;
17 }
18
19 void enQueue(int i){
20     if(f== -1 && r== -1){
21         f=0;
22     }
23     q[++r]=i;
24     return;
25 }
26
27 int deQueue(){
28     if(isEmpty()){
29         return -1;
30     }
31     if(f==r){
32         int temp=q[f];
33         f=-1;
34         r=-1;
35         return temp;
36     }
37     int temp=q[f];
38     f++;
39     return temp;
40 }
41
42 void topoLogicalSorting(int arr[10][10],int n){
43     int sorted[10];
44     int k=0;
45     int indegrees[10];
46     for (int i = 0; i < n; ++i)
47     {
48         indegrees[i]=0;
49     }
50     for(int i=0;i<n;i++){
51         for(int j=0;j<n;j++){
52             if(arr[i][j]==1)
53                 indegrees[j]++;
54         }
```

```
55     }
56     for (int i = 0; i < n; ++i)
57     {
58         if(indegrees[i]==0){
59             enQueue(i);
60         }
61     }
62     int visitedNodes=0;
63     while(!isEmpty()){
64         int snode=deQueue();
65         sorted[k++]=snode;
66         for(int i=0;i<n;i++){
67             if(arr[snode][i]==1){
68                 if(--indegrees[i]==0){
69                     enQueue(i);
70                 }
71             }
72         }
73         visitedNodes++;
74     }
75     if(visitedNodes!=n){
76         printf("\nError!! Graph has a cycle\n");
77         return;
78     }
79
80     for (int i = 0; i < n; ++i)
81     {
82         printf("%d ",sorted[i]);
83     }
84 }
85
86 int main(){
87     int arr[10][10];
88     int n,m;
89     int visited[10];
90     printf("Enter number of Vertices :");
91     scanf("%d",&n);
92     printf("Enter number of Edges :");
93     scanf("%d",&m);
94     for(int i = 0; i < n; ++i){
95         for (int j = 0; j < n; ++j)
96         {
97             arr[i][j]=0;
98         }
99     }
100    for (int i = 0; i < m; ++i)
101    {
102        int a,b;
103        printf("Enter the Edge %d :",(i+1));
104        scanf("%d %d",&a,&b);
105        arr[a][b]=1;
106    }
107    topoLogicalSorting(arr,n);
108 }
```

## **Output:**

---

```
akashm@Akashs-MacBook-Pro ADAPrограмм % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :4
Enter the Edge 1 :2 0
Enter the Edge 2 :2 1
Enter the Edge 3 :1 0
Enter the Edge 4 :1 3
2 1 0 3 %

akashm@Akashs-MacBook-Pro ADAPrограмм % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :5
Enter the Edge 1 :2 0
Enter the Edge 2 :2 1
Enter the Edge 3 :1 0
Enter the Edge 4 :1 3
Enter the Edge 5 :3 0
2 1 3 0 %

akashm@Akashs-MacBook-Pro ADAPrограмм % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :5
Enter the Edge 1 :0 2
Enter the Edge 2 :2 1
Enter the Edge 3 :1 3
Enter the Edge 4 :1 0
Enter the Edge 5 :3 0

Error!! Graph has a cycle
```

```
1 //7. Implement Johnson Trotter algorithm to generate permutations
2
3 #include<stdio.h>
4 #include<stdlib.h>
5
6
7 int searchArray(int arr[], int n, int mobile){
8     for(int i = 0; i < n;i++){
9         if(arr[i]==mobile){
10             return i+1;
11         }
12     }
13     return 0;
14 }
15
16 int fact(int n){
17     if(n==0 || n==1){
18         return 1;
19     }
20     return n*fact(n-1);
21 }
22
23 int mobileSearch(int arr[], int n, int dir[]){
24     int mobile=0, mobile_prev=0;
25     for(int i=0;i<n;i++){
26         if(dir[arr[i]-1]==0 && i!=0){
27             if(arr[i]>arr[i-1] && arr[i]>mobile_prev){
28                 mobile=arr[i];
29                 mobile_prev=mobile;
30             }
31         }
32     }
33
34     for(int i = 0; i < n;i++){
35         if(dir[arr[i]-1]==1 && i!=n-1){
36             if(arr[i]>arr[i+1] && arr[i]>mobile_prev){
37                 mobile=arr[i];
38                 mobile_prev=mobile;
39             }
40         }
41     }
42
43     if(mobile != 0)
44         return mobile;
45     else
46         return 0;
47 }
48
49 void permutationPrint(int arr[], int n, int dir[]){
50     int mobile=mobileSearch(arr,n,dir);
51     int getPos=searchArray(arr,n,mobile);
52
53     if(dir[arr[getPos-1]-1]==0){
54         int temp=arr[getPos-1];
55         arr[getPos-1]=arr[getPos-2];
```

```
55     arr[getPos-2]=temp;
56 }
57 if(dir[arr[getPos-1]-1]==1){
58     int temp=arr[getPos-1];
59     arr[getPos-1]=arr[getPos];
60     arr[getPos]=temp;
61 }
62
63
64 for (int i = 0; i < n; ++i)
65 {
66     if(arr[i]>mobile && dir[arr[i]-1]==0){
67         dir[arr[i]-1]=1;
68     }
69
70     else if(arr[i]>mobile && dir[arr[i]-1]==1){
71         dir[arr[i]-1]=0;
72     }
73 }
74
75 for (int i = 0; i < n; ++i)
76 {
77     printf("%d",arr[i]);
78 }
79
80 printf(" ");
81 }
82
83 void johnsonTrotter(int n){
84     int dir[300];
85     int arr[300];
86     for(int i=0;i<n;i++){
87         arr[i]=i+1;
88     }
89
90     for (int i = 0; i < n; ++i)
91     {
92         printf("%d",arr[i]);
93     }
94
95     printf(" ");
96
97     for (int i = 0; i < n; ++i)
98     {
99         dir[i]=0;
100    }
101
102    for (int i = 1; i < fact(n); ++i)
103    {
104        permutationPrint(arr, n, dir);
105    }
106
107 }
108
109
110 }
```

```
111 int main(){
112     int n;
113     printf("Enter the value of N : ");
114     scanf("%d",&n);
115     printf("Permutations of 5 digit number is \n");
116     johnsonTrotter(n);
117 }
```

## Output:

---

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./Johnsc
Enter the value of N : 4
Permutations of 5 digit number is
1234 1243 1423 4123 4132 1432 1342 1324 3124 314
2134 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./Johnsc
Enter the value of N : 5
Permutations of 5 digit number is
12345 12354 12534 15234 51234 51243 15243 12543
1235 41325 41352 41532 45132 51432 51342 15342 1
252 31225 32125 32152 32512 35212 52312 52132 25
54 21245 21425 21452 21542 25142 52142 52412 254
2 24152 24125 21425 21452 21524 25124 52124 2512
24152 24125 42125 42152 42512 45212 54212 52412
21254 21245 21425 21452 21542 %
```

---

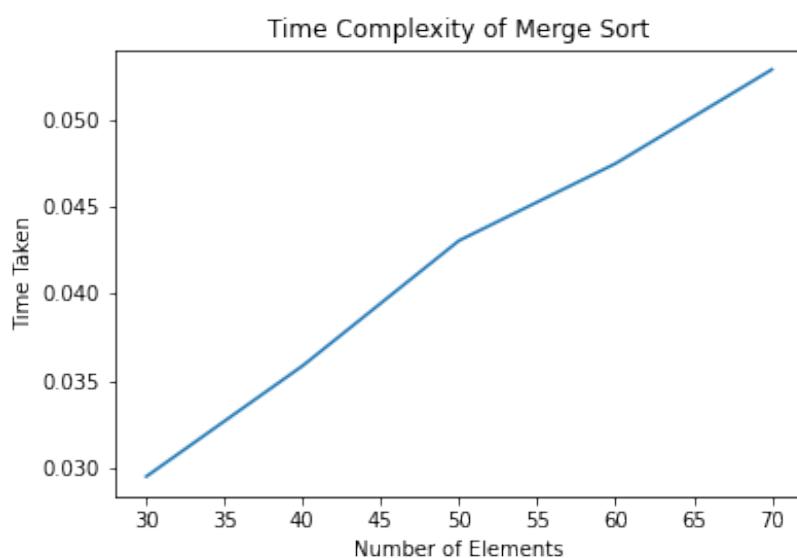
```
1 //8. Sort a given set of N integer elements using Merge Sort technique and  
2 // compute its time taken. Run the program for different values of N and record  
3 // the time taken to sort.  
4  
5 #include<stdio.h>  
6 #include<stdlib.h>  
7 #include<time.h>  
8  
9 void delay(){  
10    for (int i = 0; i < 500000; ++i)  
11    {  
12        int d=i/33;  
13    }  
14}  
15  
16 void merge(int arr[],int si,int ei,int mid){  
17    int i=si,j=mid+1,k=si;  
18    int res[ei+1];  
19    while(i<=mid && j<=ei){  
20        if(arr[i]<arr[j]){  
21            res[k]=arr[i];  
22            k++;  
23            i++;  
24        }  
25        else if(arr[i]>arr[j]){  
26            res[k]=arr[j];  
27            k++;  
28            j++;  
29        }  
30    }  
31    while(i<=mid){  
32        res[k]=arr[i];  
33        i++;  
34        k++;  
35    }  
36    while(j<=ei){  
37        res[k]=arr[j];  
38        j++;  
39        k++;  
40    }  
41    for(int i=si;i<=ei;i++){  
42        arr[i]=res[i];  
43    }  
44}  
45  
46 void mergeSort(int arr[],int si,int ei){  
47    if(si>=ei){  
48        return;  
49    }  
50    delay();  
51    int mid=(si+ei)/2;  
52    mergeSort(arr,si,mid);  
53    mergeSort(arr,mid+1,ei);
```

```
54     merge(arr,si,ei,mid);
55 }
56
57 }
58
59 int main(){
60     int arr[5000],n;
61     printf("Enter the number of elements in the Array :");
62     scanf("%d",&n);
63     for(int i=0;i<n;i++){
64         arr[i]=(rand() % (1 - 5000 + 1)) + 1;
65     }
66     clock_t start=clock();
67     mergeSort(arr,0,n-1);
68     clock_t end=clock();
69
70     printf("Time taken by Merge Sort is %f", (double)(end-start)/CLOCKS_PER_SEC);
71 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :30
Time taken by Merge Sort is 0.029481%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :40
Time taken by Merge Sort is 0.035862%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :50
Time taken by Merge Sort is 0.043071%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :60
Time taken by Merge Sort is 0.047507%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :70
Time taken by Merge Sort is 0.052922%
```

## **Graph:**



```
1 //9. Sort a given set of N integer elements using Quick Sort technique and  
2 // compute its time taken  
3  
4 #include<stdio.h>  
5 #include<stdlib.h>  
6 #include<time.h>  
7  
8 void swap(int arr[],int i,int j){  
9     if(i==j){  
10         return;  
11     }  
12     int temp = arr[i];  
13     arr[i] = arr[j];  
14     arr[j] = temp;  
15 }  
16  
17 int partition(int arr[],int si,int ei){  
18     int temp=arr[si];  
19     int count=0;  
20     for (int i = si+1; i <= ei; ++i)  
21     {  
22         if(temp>arr[i])  
23             count++;  
24     }  
25 }  
26 swap(arr,si,si+count);  
27  
28 int i=si, j=ei;  
29  
30 while(i<j){  
31     if(arr[i]<temp){  
32         i++;  
33     }  
34     else if(arr[j]>=temp){  
35         j--;  
36     }  
37     else{  
38         swap(arr,i,j);  
39         i++;  
40         j--;  
41     }  
42 }  
43 return si+count;  
44 }  
45  
46  
47  
48 void quickSort(int arr[],int si,int ei){  
49     if(si>=ei){  
50         return;  
51     }  
52  
53     int i=partition(arr,si,ei);  
54     quickSort(arr,si,i-1);
```

```
~. quickSort(...,si,ei);
55 quickSort(arr,i+1,ei);
56 }
57
58
59 int main(){
60     int arr[5000],n;
61     printf("Enter the number of elements in the Array :");
62     scanf("%d",&n);
63     for(int i=0;i<n;i++){
64         arr[i]=(rand() % (1 - 5000 + 1)) + 1;
65     }
66
67     clock_t start=clock();
68     quickSort(arr,0,n-1);
69     clock_t end=clock();
70
71     printf("Time taken by Quick Sort is %f", (double)(end-start)/CLOCKS_PER_SEC);
72 }
```

## **Output:**

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :1000
Time taken by Quick Sort is 0.000348%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :1500
Time taken by Quick Sort is 0.000720%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :2000
Time taken by Quick Sort is 0.000978%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :2500
Time taken by Quick Sort is 0.001283%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :3000
Time taken by Quick Sort is 0.001402%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :3500
Time taken by Quick Sort is 0.001820%
```

## **Graph:**

