ADA Laboratory Record

Dinesh K R Navada

1SI16CS034

1 Divide and conquer

Problem Statement:

Use divide and conquer method to recursively implement the following algorithms.

a. *Binary Search and Linear Search.*

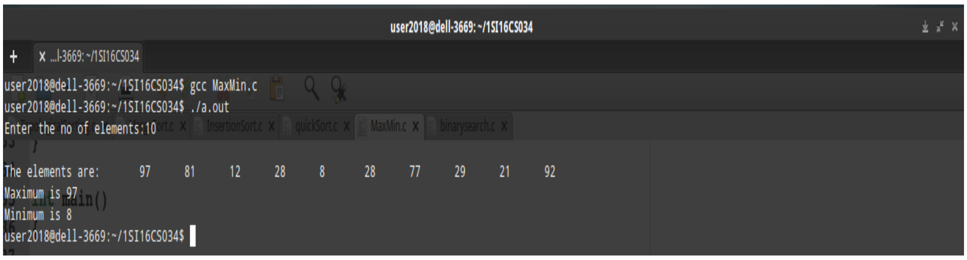
b. *To find the maximum and minimum in a given list of n elements.*

Solution

1.c.Maximum and Minimum elements

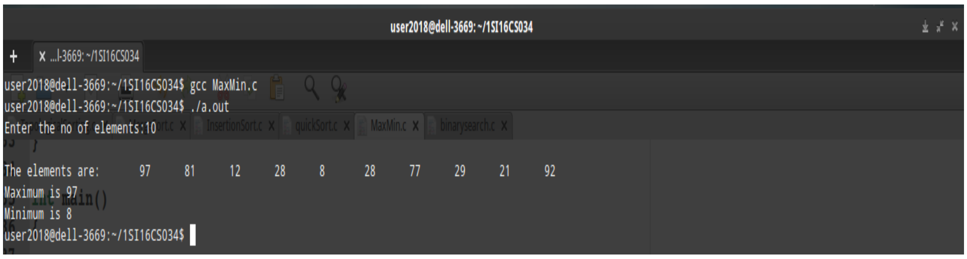
```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void RecminMax(int A[], int low, int high, int *max, int *min)
{
    if(high-low==1)
    {
    if (A[low]>A[high])
    {
        *max=A[low];
        *min=A[high];
    }

    else{
        *max=A[high];
        *min=A[low];
    }}
    else if(high==low) *max=*min=A[low];
    else if(high>low){
    int mid, max1, max2, min1, min2;
    mid=(high+low)/2;
    RecminMax(A, low, mid, &max1, &min1);
    RecminMax(A, mid+1, high, &max2, &min2);
    if(max1>max2){
    *max=max1;
    }
    else *max=max2;
    if(min1<min2){
    *min=min1;
    }
    else *min=min2;
    }
}

int main()
{

    int max, min,n,i;
    srand(time(NULL));
    printf("Enter the no of elements:");
    scanf("%d", &n);
    int arr[n];
    printf("\nThe elements are:");
    for( i=0;i<n;i++)
    {
        arr[i]=rand()%100;
        printf("\t%d", arr[i]);
    }
    RecminMax(arr, 0, n-1, &max, &min);
    printf("\nMaximum is %d\nMinimum is %d\n", max, min);

}
```

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:       97      81      12      28      8       28      77      29      21      92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$ 
```

Figure 1: Case 1



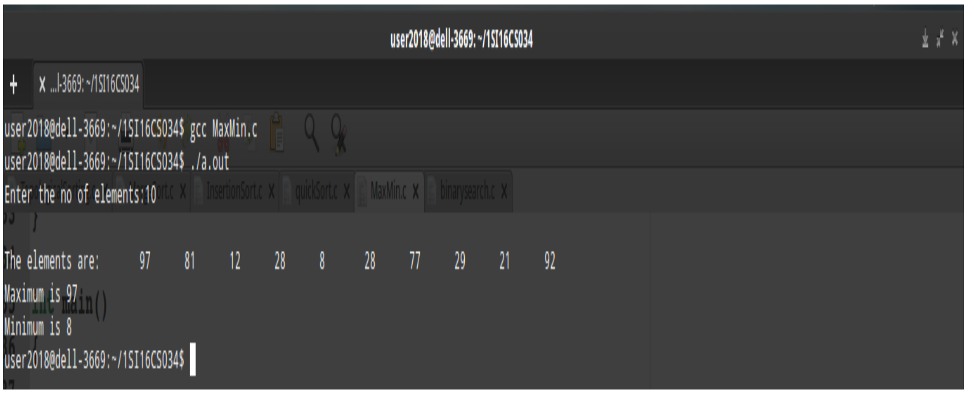
```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:       97      81      12      28      8       28      77      29      21      92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$ 
```

Figure 2: Case 2

2.Insertion Sort

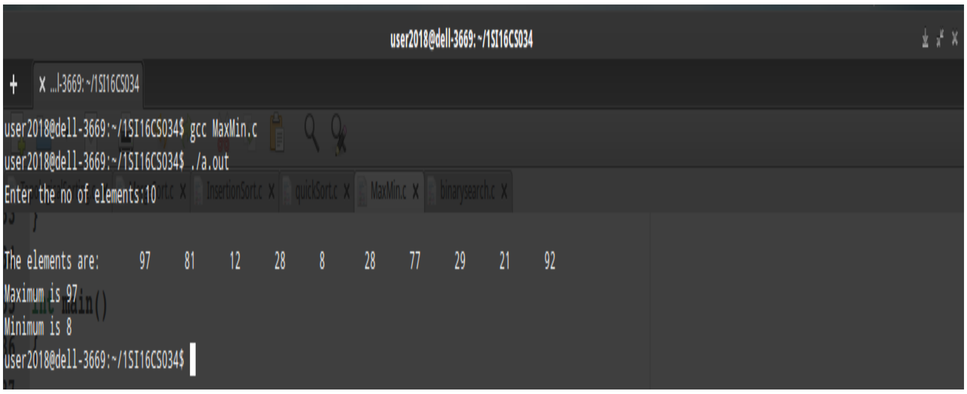
```
/*********************************
*Author:Dinesh K R Navada        *
*Date of execution:07 Sept 2018  *
*********************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/time.h>

void InsertionSort(int a[], int n){
    int i, j, v;
    for(i=1;i<n;i++){
        v=a[i];
        j=i-1;
        while(j>=0 && a[j]>v){
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=v;
    }
}
int main(){
    int i, n;
    printf("Enter the number of elements:");
    scanf("%d", &n);
    int a[n];
    srand(time(NULL));
    printf("\nThe unsorted elements are:");
    for(i=0;i<n;i++){
        a[i]=rand()%100;
        printf("%d\t", a[i]);
    }
    InsertionSort(a,n);
    printf("\nThe sorted elements are:");
    for(i=0;i<n;i++)
        printf("%d\t", a[i]);
    printf("\n");
    return 0;
}
```

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:      97      81      12      28      8       28      77      29      21      92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$
```

Figure 3: Case 1



```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:      97      81      12      28      8       28      77      29      21      92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$
```

Figure 4: Case 2

2 Merge Sort

Problem Statement:

Sort a given set of elements using the Merge sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

Solution

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

void Merge(int a[], int low, int high, int mid)
{
    int i, j, k,b[100000];
    i=k=low;
    j=mid+1;
    while(i<=mid && j<=high)
    {
        if(a[i]<a[j])
            b[k++]=a[i++];
        else
            b[k++]=a[j++];
    }
     while(i<=mid)
     {
         b[k++]=a[i++];
     }
     while(j<=high)
     {
         b[k++]=a[j++];
     }
     for(i=low;i<k;i++)
         a[i]=b[i];

}
void MergeSort(int A[], int low, int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        MergeSort(A, low, mid);
        MergeSort(A, mid+1, high);
        Merge(A, low, high, mid);
    }
}
```

```
void smallExp()
{
    int i,n;
    printf("Demonstrating for smaller size\nEnter no. of elements:");
    scanf("%d",&n);
    int a[n];
    printf("\nThe unsorted elements are:");
    for(i=0;i<n;i++)
    {
        a[i]=rand()%100;
        printf("%d\t", a[i]);
    }
    MergeSort(a, 0, n-1);
    printf("\nThe sorted elements are:");
    for(i=0;i<n;i++)
    {
        printf("%d\t", a[i]);
    }
}
int main()
{
    int n, i, j;
    srand(time(NULL));
    smallExp();
    printf("\nRepeating for larger list sizes.......");
    struct timeval tv;
    FILE *fp=fopen("mergedata.txt","w");
    double init, final;
    for(j=0;j<100000;j+=100){
    int arr[j];
    for(i=0;i<j;i++)
    {
        arr[i]=rand()%100;
        //printf("\t%d", arr[i]);
    }
    gettimeofday(&tv, NULL);
    init=tv.tv_sec+tv.tv_usec/1000000.0;
    MergeSort(arr, 0, j-1);
    gettimeofday(&tv, NULL);
    final=tv.tv_sec+tv.tv_usec/1000000.0;
    fprintf(fp, "%d\t%f\n", j, final-init);
    }
    printf("\nData file mergedata.txt generated in present working directory.\n");
}
```

Efficiency

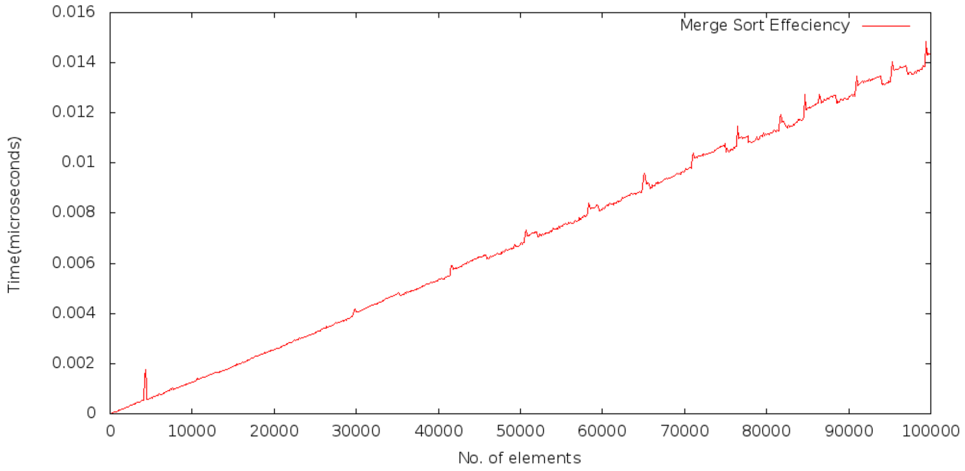
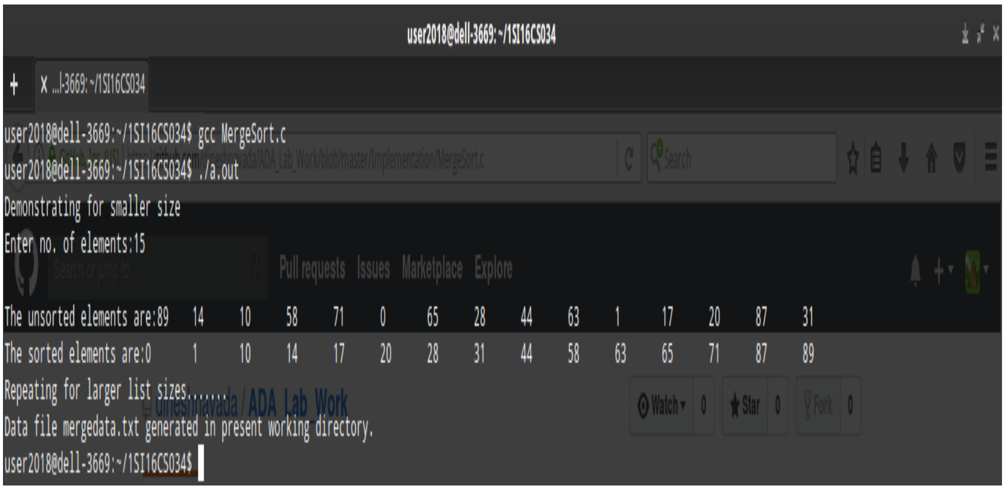


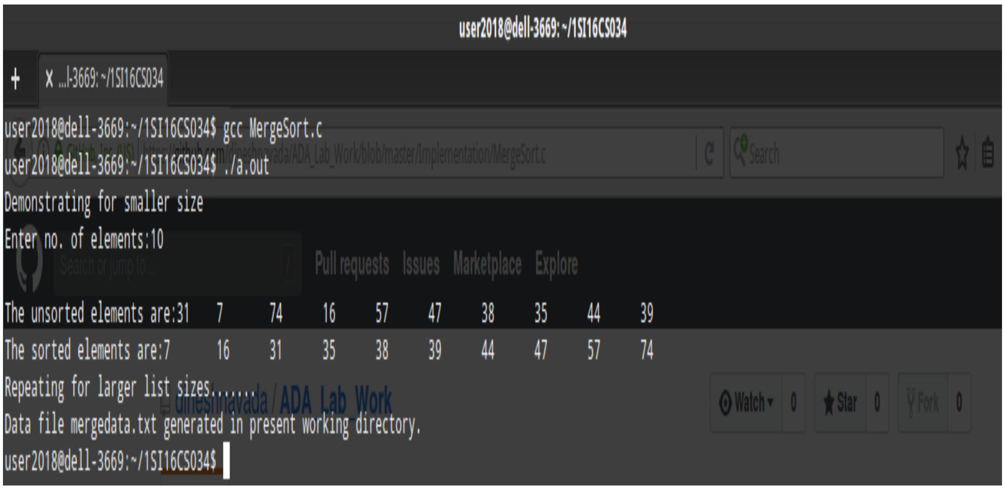
Figure 5: Plot of Time(Y-axis) v/s No. of elements(X-axis)

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc MergeSort.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Demonstrating for smaller size
Enter no. of elements:15
The unsorted elements are:89    14      10      58      71      0       65      28      44      63      1       17      20      87      31
The sorted elements are:0       1       10      14      17      20      28      31      44      58      63      65      71      87      89
Repeating for larger list sizes.......
Data file mergedata.txt generated in present working directory.
user2018@dell-3669:~/1SI16CS034$
```

Figure 6: Case 1



```
user2018@dell-3669:~/1SI16CS034$ gcc MergeSort.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Demonstrating for smaller size
Enter no. of elements:10
The unsorted elements are:31    7       74      16      57      47      38      35      44      39
The sorted elements are:7       16      31      35      38      39      44      47      57      74
Repeating for larger list sizes.......
Data file mergedata.txt generated in present working directory.
user2018@dell-3669:~/1SI16CS034$
```

Figure 7: Case 2

3 Quick Sort

Problem Statement:

Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

Solution

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/time.h>
void swap(int *a, int *b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
}
int partition(int a[], int l, int r)
{
    int p=a[l];
    int i=l;
    int j=r+1;
    do
    {
        do
        {
            i++;
        }while(a[i]<p);
        do
        {
            j--;
        }while(a[j]>p);
        swap(&a[i], &a[j]);
    }while(i<j);
    swap(&a[i], &a[j]);
    swap(&a[l], &a[j]);
    return j;
}
void quickSort(int a[], int l, int r)
{
    if(l<r)
    {
        int s=partition(a,l,r);
        quickSort(a, l, s-1);
        quickSort(a,s+1, r);
    }
}
int main()
{
    int i, j,n;
    FILE *fp;
    double init, final;
    struct timeval tv;
    //printf("%d",);

    //printf("\nEnter the no. of elements:");
    //scanf("%d", &n);
    fp=fopen("quickplot.txt", "w");
    srand(time(NULL));
    for( j=0;j<100000;j+=100){
    int a[j];

    for(i=0; i<j;i++)
    {
        a[i]=rand()%1000;
    }
    gettimeofday(&tv, NULL);
    init=tv.tv_sec+tv.tv_usec/1000000.0;
    quickSort(a,0,j-1);
    gettimeofday(&tv, NULL);
    final=tv.tv_sec+tv.tv_usec/1000000.0;
    fprintf(fp,"%d\t%f\n", j, final-init);
    }
    /*printf("\nThe elements are:");
    for(i=0; i<n;i++)
        printf("%d\t", a[i]);
    quickSort(a,0,n-1);
    printf("\nThe sorted elements are:");
    for(i=0; i<n;i++)
        printf("%d\t", a[i]);
    printf("%f",difftime(init, tv.tv_sec));
    printf("\n");*/
}
```

Efficiency

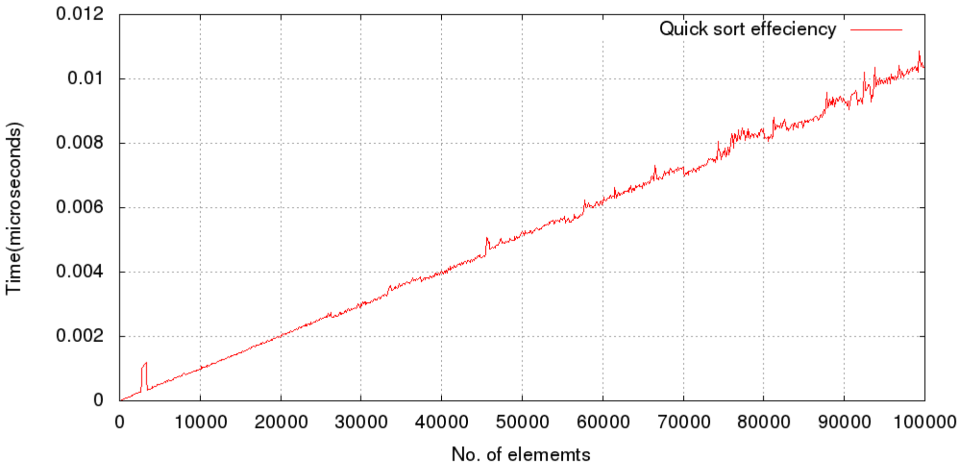
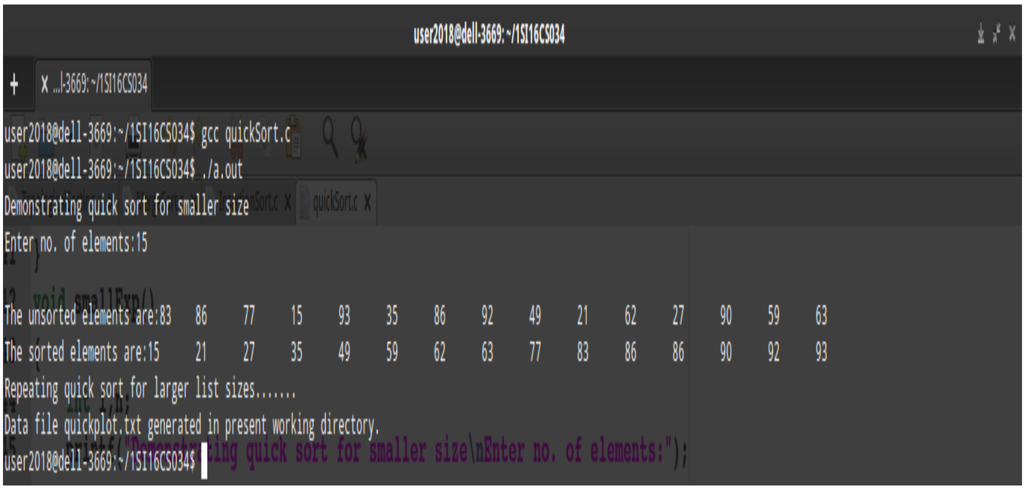


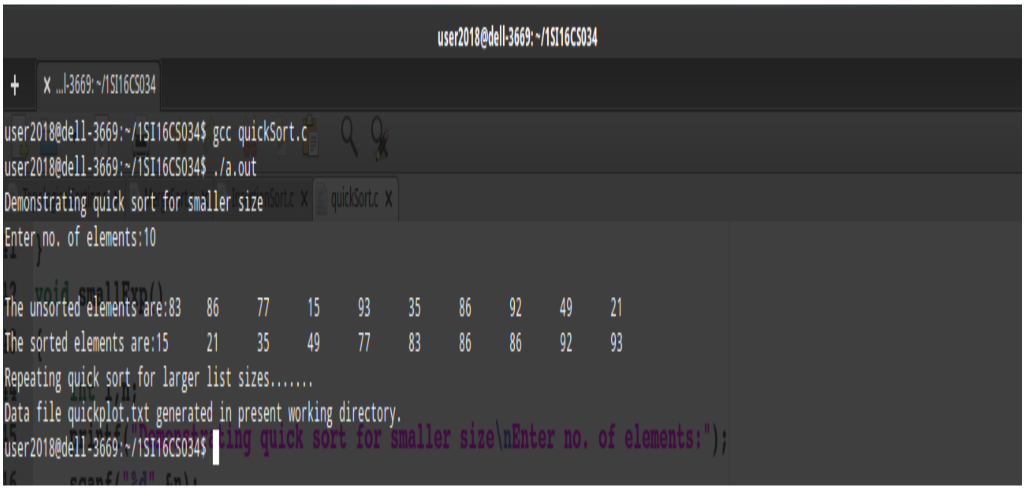
Figure 8: Plot of Time(Y-axis) v/s No. of elements(X-axis)

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc quickSort.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Demonstrating quick sort for smaller size
Enter no. of elements:15

The unsorted elements are:83	86	77	15	93	35	86	92	49	21	62	27	90	59	63
The sorted elements are:15	21	27	35	49	59	62	63	77	83	86	86	90	92	93
Repeating quick sort for larger list sizes.......
Data file quickplot.txt generated in present working directory.
user2018@dell-3669:~/1SI16CS034$
```

Figure 9: Case 1



```
user2018@dell-3669:~/1SI16CS034$ gcc quickSort.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Demonstrating quick sort for smaller size
Enter no. of elements:10

The unsorted elements are:83	86	77	15	93	35	86	92	49	21
The sorted elements are:15	21	35	49	77	83	86	86	92	93
Repeating quick sort for larger list sizes.......
Data file quickplot.txt generated in present working directory.
user2018@dell-3669:~/1SI16CS034$
```

Figure 10: Case 2

4 Decrease and conquer

Problem Statement:

a. *Obtain the Topological ordering of vertices in a given digraph.*

b. *Sort a given set of elements using Insertion sort method.*

Solution

1.Topological Ordering

```
/*********************************
*Author:Dinesh K R Navada        *
*Date of execution:07 Sept 2018  *
*********************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/time.h>
#define MAX 10

void topologicalSort(int a[MAX][MAX], int v){
    int in[MAX],out[MAX], stack[MAX];
    int i,j,k=0, top=-1;
    for(i=0;i<v;i++){
        in[i]=0;
        for(j=0;j<v;j++){
            if(a[j][i]==1)
                in[i]++;
        }
    }
    while(1){
        for(i=0;i<v;i++){
            if(in[i]==0){
                stack[++top]=i;
                in[i]=-1;
            }
        }
            if(top==-1) break;
            out[k]=stack[top--];
```

```c
            for(i=0;i<v;i++){
                if(a[out[k]][i]==1)
                    in[i]-=1;
            }
        k++;
    }
    printf("\nThe topological ordering is:");
    for(i=0;i<v;i++)
        printf("%d\t", out[i]+1);
    }
int main(){
    int a[MAX][MAX],v,i,j;
    printf("Enter number of vertices:");
    scanf("%d",&v);
    printf("\nEnter the Adjacancy matrix:\n");
    for(i=0;i<v;i++)
            for(j=0;j<v;j++)
                scanf("%d",&a[i][j]);
    topologicalSort(a,v);
    printf("\n");
    return 0;
}
```

Input Graphs

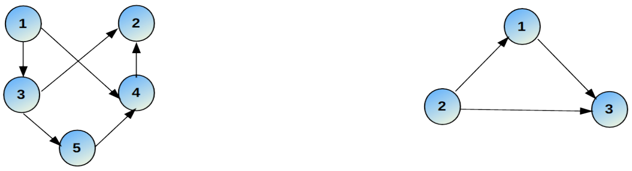
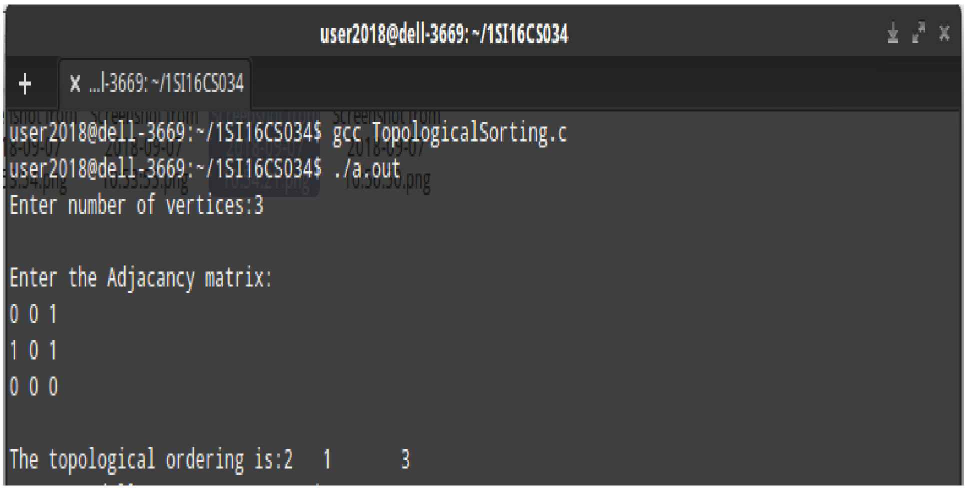


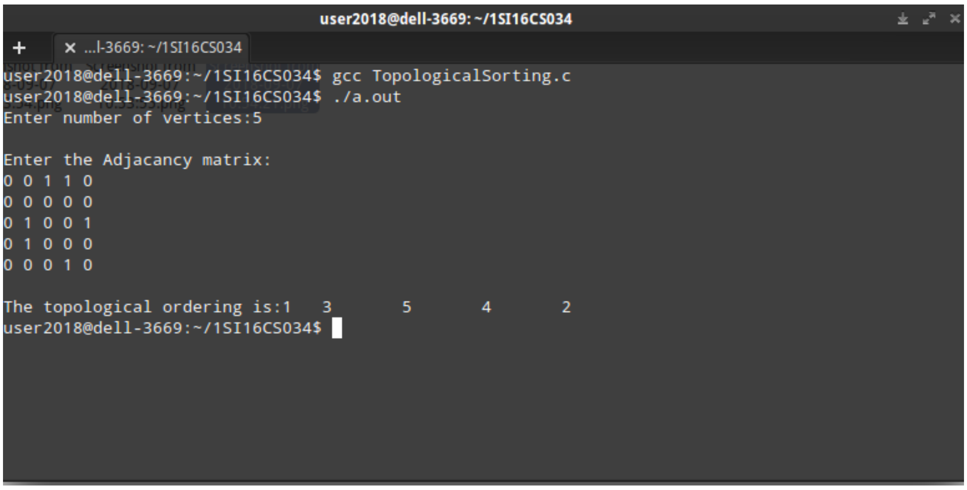
Figure 11: Input graphs

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc TopologicalSorting.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter number of vertices:3

Enter the Adjacancy matrix:
0 0 1
1 0 1
0 0 0

The topological ordering is:2   1       3
```

Figure 12: Case 1



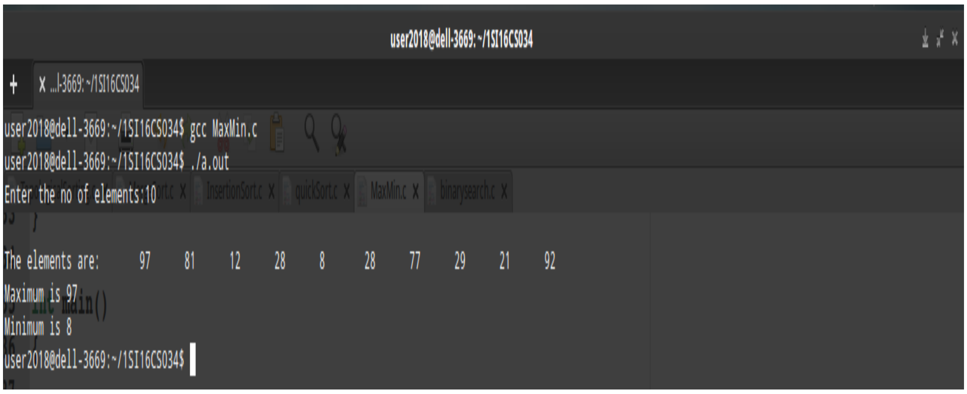
```
user2018@dell-3669:~/1SI16CS034$ gcc TopologicalSorting.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter number of vertices:5

Enter the Adjacancy matrix:
0 0 1 1 0
0 0 0 0 0
0 1 0 0 1
0 1 0 0 0
0 0 0 1 0

The topological ordering is:1   3       5       4       2
user2018@dell-3669:~/1SI16CS034$ 
```

Figure 13: Case 2

2.Insertion Sort

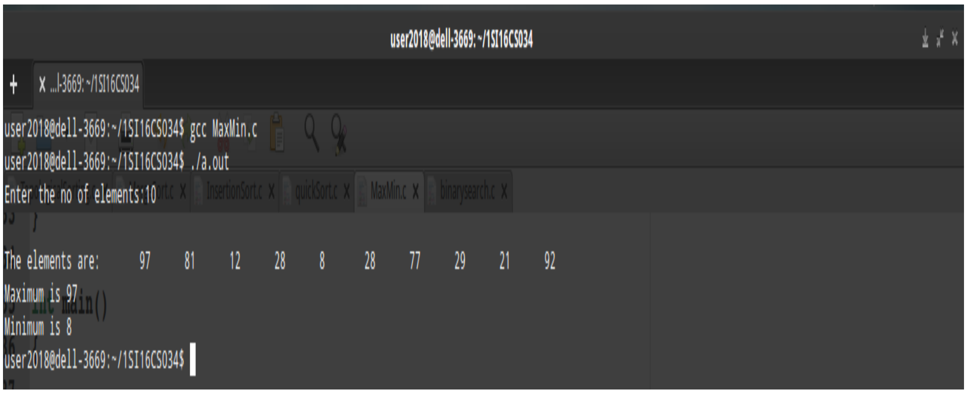
```c
/*********************************
*Author:Dinesh K R Navada        *
*Date of execution:07 Sept 2018  *
*********************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/time.h>

void InsertionSort(int a[], int n){
    int i, j, v;
    for(i=1;i<n;i++){
        v=a[i];
        j=i-1;
        while(j>=0 && a[j]>v){
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=v;
    }
}
int main(){
    int i, n;
    printf("Enter the number of elements:");
    scanf("%d", &n);
    int a[n];
    srand(time(NULL));
    printf("\nThe unsorted elements are:");
    for(i=0;i<n;i++){
        a[i]=rand()%100;
        printf("%d\t", a[i]);
    }
    InsertionSort(a,n);
    printf("\nThe sorted elements are:");
    for(i=0;i<n;i++)
        printf("%d\t", a[i]);
    printf("\n");
    return 0;
}
```

Output



```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:     97    81    12    28    8    28    77    29    21    92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$
```

Figure 14: Case 1



```
user2018@dell-3669:~/1SI16CS034$ gcc MaxMin.c
user2018@dell-3669:~/1SI16CS034$ ./a.out
Enter the no of elements:10

The elements are:     97    81    12    28    8    28    77    29    21    92
Maximum is 97
Minimum is 8
user2018@dell-3669:~/1SI16CS034$
```

Figure 15: Case 2