PRACTICAL FILE

Session : 2018-19

B.Tech. 2nd Year 4th Semester

Core Lab – CSB 252

(Design & Analysis of Algorithm LAB)

Submitted To:	**Submitted By:**
Dr. Arun Dept. of Computer Science & Engineering NIT, Delhi	Hitendra Singh 171210028 Date : / /2019

INDEX

No.	Program Title	Date	Sign

No.	Program Title	Date	Sign

PRACTICAL - 1

Problem Statement :

Implement linear search, bubble sort and then binary search on an array.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

int linear_search(int s,int arr[],int item)
{
    int i=0,pos=-1;
    while(i<s&&pos==-1)
    {
        if(arr[i]==item)
            pos=i+1;
        i++;
    }
    return pos;
}

void bubble_sort(int s,int arr[])
{
    int i,j,t;
    for(i=0;i<s;i++)
    {
        for(j=0;j<s-1-i;j++)
        {
            if(arr[j]>arr[j+1])
            {
                t=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=t;
            }
        }
    }
}

void print_arr(int s,int arr[])
{
  for(int i=0;i<s;i++)
  cout<<" "<<arr[i];
  cout<<endl;
```

```
}

int bin_search(int n,int item,int arr[])
{
    int beg=0,last=n-1,pos=-1;
    int mid=(beg+last)/2;
    while(beg<=last&&pos==-1)
    {
        mid=(beg+last)/2;
        if(arr[mid]==item)
            pos=mid+1;
        else if(arr[mid]<item)
            beg=mid+1;
        else
            last=mid-1;
    }
    return pos;
}

int main()
{
     int s,i;
     cout<<"\n Enter the size : ";
     cin>>s;
     int arr[s];
     cout<<"\n Enter the "<<s<<" elements : ";
     for(i=0;i<s;i++)
     cin>>arr[i];
     cout<<"\n The array is : ";
     print_arr(s,arr);
     int ch;
     char c;
     do
     {
        system("cls");
        cout<<"\n The array is : ";
        print_arr(s,arr);
        cout<<"\n-------------------------------------+\n";
        cout<<"\t      MENU                          |\n";
        cout<<"-------------------------------------+";
        cout<<"\n\t 1. Linear Search                  |";
        cout<<"\n\t 2. Bubble Sort                    |";
        cout<<"\n\t 3. Binary Search                  |";
        cout<<"\n\t 4. EXIT                           |";
        cout<<"\n-------------------------------------+\n";
        cout<<"\n\n Enter your choice :";
        cin>>ch;
        switch(ch)
        {
```

```
        case 1:{
                  int ele;
                  cout<<"\n Enter the search element :";
                  cin>>ele;
                  int p=linear_search(s,arr,ele);
                  if(p==-1)
                      cout<<"\n "<<ele<<" Not found !!!\n";
                  else
                     cout<<"\n "<<ele<<" found at position "<<p<<"\n";
                  cin>>c;
                        break;
                 }
        case 3:{
                  int ele;
                  cout<<"\n Enter the search element :";
                  cin>>ele;
                  int p=bin_search(s,ele,arr);
                  if(p==-1)
                      cout<<"\n "<<ele<<" Not found !!!\n";
                  else
                     cout<<"\n "<<ele<<" found at position "<<p<<"\n";
                  cin>>c;
                        break;
                 }
        case 2:{
                    cout<<"\n The sorted array is : ";
                    bubble_sort(s,arr);
                    print_arr(s,arr);
                    cin>>c;
                        break;
                 }
         case 4:break;
        default : cout<<"\nINVALID\n";cin>>c;break;
       }}
     while(ch!=4);
    return 0;
}
```

Output :

```
Enter the size : 5

Enter the 5 elements : 5 4 1 3 2
```

```
The array is :  5 4 1 3 2

----------------------------------------+
                MENU                    ¦
----------------------------------------+
          1. Linear Search              ¦
          2. Bubble Sort                ¦
          3. Binary Search              ¦
          4. EXIT                       ¦
----------------------------------------+

Enter your choice :1

Enter the search element :4

4 found at position 2
```

```
The array is :  5 4 1 3 2

----------------------------------------+
                MENU                    ¦
----------------------------------------+
          1. Linear Search              ¦
          2. Bubble Sort                ¦
          3. Binary Search              ¦
          4. EXIT                       ¦
----------------------------------------+

Enter your choice :2

The sorted array is :  1 2 3 4 5
```

```
The array is :  1 2 3 4 5

----------------------------------------+
                MENU                    ¦
----------------------------------------+
          1. Linear Search              ¦
          2. Bubble Sort                ¦
          3. Binary Search              ¦
          4. EXIT                       ¦
----------------------------------------+

Enter your choice :3

Enter the search element :4

4 found at position 4
```

PRACTICAL - 2

Problem Statement :

Implement Selection Sort,Insertion sort And Quick sort.

Source Code :

// SELECTION SORT

```
#include <iostream>
using namespace std;

void print_arr(int s,int arr[])
{
  for(int i=0;i<s;i++)
  cout<<" "<<arr[i];
  cout<<endl;
}

int selection_sort(int n,int arr[])
{
    int i,j,t,c=1;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if((c++)&&arr[j]<=arr[i])
            {
                t=arr[j]; arr[j]=arr[i]; arr[i]=t;
            }
        }
    }
    return c-1;
}

int main()
{
    int i,j,n=5,cnt,k;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"\n---------------------------------------\n";
    cout<<"\n USING SELECTION SORT \n";
    cout<<"\n BEST CASE : \n";
```

```
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is : ";
    cnt=selection_sort(n,arr);
    print_arr(n,arr);
    cout<<"\n Number of Comparisons : "<<cnt<<endl;
    cout<<"\n WORST CASE : \n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is : ";
    cnt=selection_sort(n,arr);
    print_arr(n,arr);
    cout<<"\n Number of Comparisons : "<<cnt<<endl;
    cout<<"\n--------------------------------------\n";
    return 0;
}
```

Output :

```
Enter the size of array : 5
--------------------------------------

USING SELECTION SORT
BEST CASE :
Enter 5 elements : 1 2 3 4 5
The unsorted array is :  1 2 3 4 5
The sorted array is :  1 2 3 4 5
Number of Comparisons : 10
WORST CASE :
Enter 5 elements : 5 4 3 2 1
The unsorted array is :  5 4 3 2 1
The sorted array is :  1 2 3 4 5
Number of Comparisons : 10
--------------------------------------
```

// INSERTION SORT

```
#include <iostream>
using namespace std;
void print_arr(int s,int arr[])
{
  for(int i=0;i<s;i++)
  cout<<" "<<arr[i];
  cout<<endl;
}

int insertion_sort(int n,int arr[])
{
    int i,j,k,c=1;
    for(i=1;i<n;i++)
    {
        k=arr[i];
        j=i-1;
        while(j>=0&&(c++)&&arr[j]>k)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=k;
    }
    return c-1;
}

int main()
{
    int i,j,n=5,cnt,k;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"\n---------------------------------------\n";
    cout<<"\n USING INSERTION SORT \n";
    cout<<"\n BEST CASE : \n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is : ";
    cnt=insertion_sort(n,arr);
    print_arr(n,arr);
    cout<<"\n Number of Comparisons : "<<cnt<<endl;
    cout<<"\n WORST CASE : \n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
```

```
    cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is : ";
    cnt=insertion_sort(n,arr);
    print_arr(n,arr);
    cout<<"\n Number of Comparisons : "<<cnt<<endl;
    cout<<"\n-------------------------------------\n";
    return 0;
}
```

Output :

```
Enter the size of array : 5
-------------------------------------
USING INSERTION SORT
BEST CASE :
Enter 5 elements : 1 2 3 4 5
The unsorted array is :  1 2 3 4 5
The sorted array is :  1 2 3 4 5
Number of Comparisons : 4
WORST CASE :
Enter 5 elements : 5 4 3 2 1
The unsorted array is :  5 4 3 2 1
The sorted array is :  1 2 3 4 5
Number of Comparisons : 10
-------------------------------------
```

// QUICK SORT

```
#include <iostream>
using namespace std;
int cnt=0;
int divide(int l, int u,int arr[])
{
    cnt++;
    int p = arr[u],i =(l - 1),t;
    for (int j = l; j <= u- 1; j++)
    {
        if (arr[j] <= p)
        {
            i++;
            t=arr[i];arr[i]=arr[j];arr[j]=t;
        }
    }
    t=arr[i+1];arr[i+1]=arr[u];arr[u]=t;
    return (i + 1);
}

void quick_sort(int l, int u,int arr[])
{
    if (l < u)
    {
        int pi = divide(l, u,arr);
        quick_sort(l, pi - 1,arr);
        quick_sort( pi + 1, u,arr);
    }
}

void print_arr(int n,int arr[])
{
    int i;
    for (i=0; i < n; i++)
    cout<<"  "<<arr[i];
    cout<<endl;
}

int main()
{
    int i,j,n;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
```

```
    cout<<"\n---------------------------------------\n";
    cout<<"\n USING QUICK SORT \n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is   : ";
    quick_sort(0,n-1,arr);
    print_arr(n,arr);
    cout<<"\n---------------------------------------\n";
    cout<<"\n The number of comparisons are : "<<cnt;
    return 0;
}
```

Output :

```
Enter the size of array : 5
---------------------------------------

USING QUICK SORT

Enter 5 elements : 5 3 2 6 1

The unsorted array is :   5  3  2  6  1

The sorted array is   :   1  2  3  5  6

---------------------------------------
```

PRACTICAL - 3

Problem Statement :

Implement Merge Sort on an array.

Source Code :

```
#include <iostream>
using namespace std;
int cnt=0;

void print_arr(int n,int arr[])
{
    int i;
    for (i=0; i < n; i++)
    cout<<"  "<<arr[i];
    cout<<endl;
}

void merge_arr(int l,int m,int u,int arr[])
{
    int i,j,k,size1,size2;
    size1=m-l+1;
    size2=u-m;
    int left[size1],right[size2];
    for(i=0;i<size1;i++)
       left[i]=arr[l+i];
    for(i=0;i<size2;i++)
       right[i]=arr[m+i+1];

    i=0;j=0;k=l;

    while(i<size1&&j<size2)
    {
        if(left[i]<=right[j])
        {
            arr[k]=left[i];
            i++;
        }
        else
        {
```

```
            arr[k]=right[j];
            j++;
        }
        k++;
    }
    while(i<size1)
    {
        arr[k]=left[i];
        i++;
        k++;
    }
    while(j<size2)
    {
        arr[k]=right[j];
        j++;
        k++;
    }
}

void merge_sort(int arr[],int n)
{
    int temp=1;
    for(temp=1;temp<=n-1;temp*=2)
    {   int i;
        cnt++;
        for(i=0;i<n-1;i+=2*temp)
        {
            int l,m,u,t;
            l=i;
            m= i+temp-1;
            u=((n-1)<=(m+temp))?(n-1):(m+temp);
            merge_arr(l,m,u,arr);
        }
    }
}

int main()
{
 int i,j,n;

    cout<<"\n Enter the size of array : ";
    cin>>n;
     int arr[n];
    cout<<"\n-------------------------------------\n";
    cout<<"\n USING MERGE SORT \n";
    cout<<"\n Enter "<<n<<" elements : ";
```

```
    for(i=0;i<n;i++)
        cin>>arr[i];
    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is   : ";
    merge_sort(arr,n);
    print_arr(n,arr);
    cout<<"\n---------------------------------------\n";
    cout<<"\n\n count : "<<cnt;
    return 0;
}
```

Output :

```
Enter the size of array : 6
-------------------------------------------

USING MERGE SORT
Enter 6 elements : 2 1 9 4 5 6
The unsorted array is :   2  1  9  4  5  6
The sorted array is   :   1  2  4  5  6  9
-------------------------------------------

Count : 3
-------------------------------------------
```

PRACTICAL - 4

Problem Statement :

Implement Heap Sort on an array.

Source Code :

```
#include <iostream>
using namespace std;

void print_arr(int n,int arr[])
{
    int i;
    for (i=0; i < n; i++)
    cout<<"  "<<arr[i];
    cout<<endl;
}

void max_heapify(int n,int r,int arr[])
{
    int i,p=r,x;
    int lchild=2*p+1;
    int rchild=2*p+2;

    if(lchild<n)
      p=(arr[lchild]>arr[p]?lchild:p);
    if(rchild<n)
      p=(arr[rchild]>arr[p]?rchild:p);
    swap(arr[p],arr[r]);
    if(p!=r)
    max_heapify(n,p,arr);
}

int main()
{
    int i,j,n;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"\n---------------------------------------\n";
    cout<<"\n USING HEAP SORT \n";
    cout<<"\n Enter "<<n<<" elements : ";

    for(i=0;i<n;i++)
```

```
        cin>>arr[i];

    cout<<"\n The unsorted array is : ";
    print_arr(n,arr);
    cout<<"\n The sorted array is   : ";

    for(i=n/2-1;i>=0;i--)
    max_heapify(n,i,arr);

    for(i=n-1;i>0;i--)
    {
        swap(arr[0],arr[i]);
        max_heapify(i,0,arr);
    }

    print_arr(n,arr);
    cout<<"\n--------------------------------------\n";
    return 0;
}
```

Output :

```
Enter the size of array : 6
-------------------------------------------

USING HEAP SORT
Enter 6 elements : -1 8 0 2 6 4
The unsorted array is :   -1  8  0  2  6  4
The sorted array is   :   -1  0  2  4  6  8
-------------------------------------------
```

PRACTICAL - 5

Problem Statement-1 :

Find the maximum and minimum element of an array using divide and conquer.

Source Code :

```
#include <iostream>
using namespace std;
void print_arr(int n,int arr[])
{
    int i;
    for (i=0; i < n; i++)
    cout<<"  "<<arr[i];
    cout<<endl;
}

int minimum(int arr[],int l,int r)
{
    if(l==r)
        return arr[l];
    else if(r-l==1)
        return (arr[l]<arr[r])?arr[l]:arr[r];
    else
    {
      int mid=(l+r)/2;
      int minl,minr;
      minl=minimum(arr,l,mid);
      minr=minimum(arr,mid+1,r);
      return (minl<minr)?minl:minr;
    }
}

int maximum(int arr[],int l,int r)
{
    if(l==r)
        return arr[l];
    else if(r-l==1)
        return (arr[l]>arr[r])?arr[l]:arr[r];
    else
    {
      int mid=(l+r)/2;
```

```
      int maxl,maxr;
      maxl=maximum(arr,l,mid);
      maxr=maximum(arr,mid+1,r);
      return (maxl>maxr)?maxl:maxr;
    }
}

int main()
{
    int i,j,n,mini,maxi;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"\n---------------------------------------\n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The array is : ";
    print_arr(n,arr);
    cout<<"\n Minimum element : "<<minimum(arr,0,n-1);
    cout<<"\n Maximum element : "<<maximum(arr,0,n-1);
    cout<<"\n---------------------------------------\n";
    return 0;
}
```

Output :

```
Enter the size of array : 6
---------------------------------------

Enter 6 elements : -2 0 5 55 3 11
The array is :   -2  0  5  55  3  11
Minimum element : -2
Maximum element : 55
---------------------------------------
```

Problem Statement-2 :

Find the k^{th} smallest element in an array.

Source Code :

```
#include <iostream>
using namespace std;

int divide(int l, int u,int arr[])
{
    int p = arr[u],i =(l - 1),t;
    for (int j = l; j <= u- 1; j++)
    {
        if (arr[j] <=  p)
        {
            i++;
            t=arr[i];arr[i]=arr[j];arr[j]=t;
        }
    }
    t=arr[i+1];arr[i+1]=arr[u];arr[u]=t;
    return (i + 1);
}

int find_k(int k,int l, int u,int arr[])
{
        if(k<=u-l+1)
        { int pi = divide(l, u,arr),x;
        if (pi+1==k)
        x=arr[pi];
        if(pi+1>k)
        x=find_k(k,l, pi - 1,arr);
        else
        x=find_k(k, pi + 1, u,arr);
        return x;
        }
}

void print_arr(int n,int arr[])
{
    int i;
    for (i=0; i < n; i++)
    cout<<"  "<<arr[i];
```

```
    cout<<endl;
}

int main()
{
    int i,j,n,k;
    cout<<"\n Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"\n---------------------------------------\n";
    cout<<"\n Enter "<<n<<" elements : ";
    for(i=0;i<n;i++)
    cin>>arr[i];
    cout<<"\n The array is : ";
    print_arr(n,arr);
    cout<<"\n Enter k : ";
    cin>>k;
    cout<<"\n The "<<k<<" smallest element is : "<<find_k(k+1,0,n-1,arr);
    cout<<"\n---------------------------------------\n";
    return 0;
}
```

Output :

```
Enter the size of array : 5
--------------------------------

Enter 5 elements : 5 4 3 2 1
The array is :   5  4  3  2  1
Enter k : 3
The 3 smallest element is : 3
--------------------------------
```

PRACTICAL - 6

Problem Statement-1 :

Implement the adjacency matrix representation of a graph.

Source Code :

```
#include <iostream>
using namespace std;
int main()
{
    int e,v,i,s,d,j;
    cout<<"\n Enter number of vertex : ";
    cin>>v;
    int ver[v];
    cout<<"\n Enter number of edges : ";
    cin>>e;
    int edge[e];
    int am[v][v];
    for(i=0;i<v;i++)
    for(j=0;j<v;j++)
        am[i][j]=0;
    cout<<"\n Vertex = { ";
    for(i=1;i<=e;i++)
    cout<<i<<",";
    cout<<"\b }";
    cout<<"\n\n Enter "<<e<<" edges : \n\n";
    for(i=0;i<e;i++)
    {
        cout<<"\n EDGE "<<i+1<<"\n";
        cout<<"\n Enter source vertex : ";
        cin>>s;
        cout<<"\n Enter destination vertex : ";
        cin>>d;
        if(s<=e&&s>=1&&d<=e&&s>=1)
        {
        am[s-1][d-1]=1;
        am[d-1][s-1]=1;
        }
        else
        {
```

```
            cout<<"\n Invalid Vertex ! Enter again...";
            i--;
        }
    }

    cout<<"\n The adjacency Matrix is : \n\n\n";
    for(i=-1;i<v;i++)
    {for(j=-1;j<v;j++)
    {
        if(i==-1)
        {
            if(j==-1)
                cout<<"    ";
            else
                cout<<j+1<<" ";
        }
        else if(j==-1)
        {
            cout<<i+1<<" ";
        }
        else
            cout<<" "<<am[i][j];
    }
        cout<<"\n\n ";
    }
    return 0;
}
```

Output :

```
Enter number of vertex : 4
Enter number of edges : 4
Vertex = { 1,2,3,4 }
Enter 4 edges :

EDGE 1
Enter source vertex : 1
Enter destination vertex : 2
EDGE 2
Enter source vertex : 2
Enter destination vertex : 3
EDGE 3
Enter source vertex : 3
Enter destination vertex : 4
EDGE 4
Enter source vertex : 4
Enter destination vertex : 1
The adjacency Matrix is :

   1 2 3 4
1  0 1 0 1
2  1 0 1 0
3  0 1 0 1
4  1 0 1 0
```

Problem Statement-2 :

Implement the adjacency list representation of a graph.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

class graph
{
    int vertices;
    int edges;
    list<int> *adjacency_list;
public :
    graph(int a,int b)
    {
        vertices=a;
        edges=b;
        adjacency_list=new list<int>[a];
    }
    void add_edge()
    {
        int s,d;
        cout<<"\n Enter the source vertex : ";
        cin>>s;
        cout<<"\n Enter the destination vertex : ";
        cin>>d;
        adjacency_list[s].push_back(d);
    }
    void dispList()
    {
        cout << "\n Adjacency List : \n";
        list<int>::iterator i;
        for (int j=0;j<vertices;j++)
        {
            cout<<"\n "<<j<<" -->";
            for (i = adjacency_list[j].begin();i !=
adjacency_list[j].end();i++)
                cout << " " << *i << " ";
            cout << "\n";
        }
    }
```

```
};

int main()
{
    int v,e,i,s;

    cout<<"\n Enter the number of vertexes : ";
    cin>>v;
    cout<<"\n Enter the number of edges : ";
    cin>>e;
    cout<<"\n Enter the "<<e<<" edges : \n\n";

    graph g(v,e);
    for(i=0;i<e;i++)
    g.add_edge();
    g.dispList();
    cout<<"\n\n";
    return 0;
}
```

Output :

```
Enter the number of vertexes : 4
Enter the number of edges : 6
Enter the 6 edges :

Enter the source vertex : 0
Enter the destination vertex : 1
Enter the source vertex : 0
Enter the destination vertex : 2
Enter the source vertex : 0
Enter the destination vertex : 3
Enter the source vertex : 2
Enter the destination vertex : 1
Enter the source vertex : 2
Enter the destination vertex : 3
Enter the source vertex : 3
Enter the destination vertex : 1
Adjacency List :
0 --> 1  2  3
1 -->
2 --> 1  3
3 --> 1
```

PRACTICAL - 7

Problem Statement-1 :

Implement Depth First Search in a graph. Use the adjacency list representation.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

class graph
{
    int vertices;
    int edges;
    list<int> *adjacency_list;
public :
    graph(int a,int b)
    {
        vertices=a;
        edges=b;
        adjacency_list=new list<int>[a];
    }
    void add_edge()
    {
        int s,d;
        cout<<"\n Enter the source vertex : ";
        cin>>s;
        cout<<"\n Enter the destination vertex : ";
        cin>>d;
        adjacency_list[s].push_back(d);
    }
    void dfs(int a,int checked[])
    {
        cout<<" "<<a<<" ";
        checked[a]=1;
        list<int>::iterator i;
        for(i=adjacency_list[a].begin();i!=adjacency_list[a].end();i++)
          if(checked[*i]==0)
                dfs(*i,checked);
     }
```

```
    void initialize_dfs(int start)
    {
            int *checked= new int[vertices];
            for(int i=0;i<vertices;i++)
                checked[i]=0;
            dfs(start,checked);
    }
};

int main()
{
    int v,e,i,s;

    cout<<"\n Enter the number of vertexes : ";
    cin>>v;
    cout<<"\n Enter the number of edges : ";
    cin>>e;
    cout<<"\n Enter the "<<e<<" edges : \n\n";

    graph g(v,e);
    for(i=0;i<e;i++)
    g.add_edge();

    system("cls");
    cout<<"\n Enter the start node : ";
    cin>>s;
    cout<<"\n DFS Sequence :  ";
    g.initialize_dfs(s);
    cout<<"\n\n";
    return 0;
}
```

Output :

```
Enter the number of vertexes : 4
Enter the number of edges : 6
Enter the 6 edges :

Enter the source vertex : 0
Enter the destination vertex : 1
Enter the source vertex : 0
Enter the destination vertex : 2
Enter the source vertex : 0
Enter the destination vertex : 3
Enter the source vertex : 1
Enter the destination vertex : 3
Enter the source vertex : 2
Enter the destination vertex : 1
Enter the source vertex : 2
Enter the destination vertex : 3
```

```
Enter the start node : 0
DFS Sequence :   0  1  3  2
```

Problem Statement-2 :

Find the number of disconnected components in an undirected graph. Use the adjacency list representation.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;
class graph
{
    int vertices;
    int edges;
    list<int> *adjacency_list;

public :
    graph(int a,int b)
    {
        vertices=a;
        edges=b;
        adjacency_list=new list<int>[a];
    }
    void add_edge()
    {
        int s,d;
        cout<<"\n Enter the source vertex : ";
        cin>>s;
        cout<<"\n Enter the destination vertex : ";
        cin>>d;
        adjacency_list[s].push_back(d);
        adjacency_list[d].push_back(s);
    }
    void dfs(int a,int checked[])
    {
        checked[a]=1;
        list<int>::iterator i;
        for(i=adjacency_list[a].begin();i!=adjacency_list[a].end();i++)
        if(checked[*i]==0)
              dfs(*i,checked);
    }

    int get_components()
    {
```

```
        int comp=0;
        int *checked= new int[vertices];
            for(int i=0;i<vertices;i++)
                checked[i]=0;
            for(int i=0;i<vertices;i++)
                if(checked[i]==0)
                   comp++;
                    dfs(i,checked);
        return comp;
    }
};

int main()
{
    int v,e,i,s;
    cout<<"\n Enter the number of vertexes : ";
    cin>>v;
    cout<<"\n Enter the number of edges : ";
    cin>>e;
    cout<<"\n Enter the "<<e<<" edges : \n\n";
    graph g(v,e);
    for(i=0;i<e;i++)
    g.add_edge();
    cout<<"\n Number of Components : "<<g.get_components();
    cout<<"\n\n";
    return 0;
}
```

Output :

```
Enter the number of vertexes : 9
Enter the number of edges : 7
Enter the 7 edges :

Enter the source vertex : 0
Enter the destination vertex : 2
Enter the source vertex : 1
Enter the destination vertex : 5
Enter the source vertex : 1
Enter the destination vertex : 7
Enter the source vertex : 3
Enter the destination vertex : 4
Enter the source vertex : 4
Enter the destination vertex : 6
Enter the source vertex : 6
Enter the destination vertex : 8
Enter the source vertex : 8
Enter the destination vertex : 3
Number of Components : 3
```