**Vakamalla Keerthi priya**
**(1BM19CS176)**

**ADA LAB TEST-1**

4) Write
program to do the following:

a) Print all the nodes reachable
from a given starting node in a digraph using BFS method.

b) Check whether a given graph is
connected or not using DFS method.

Modification:

BFS, given an undirected graph, print all connected components line by
line. For Eg: consider the following
graph.

**A)**

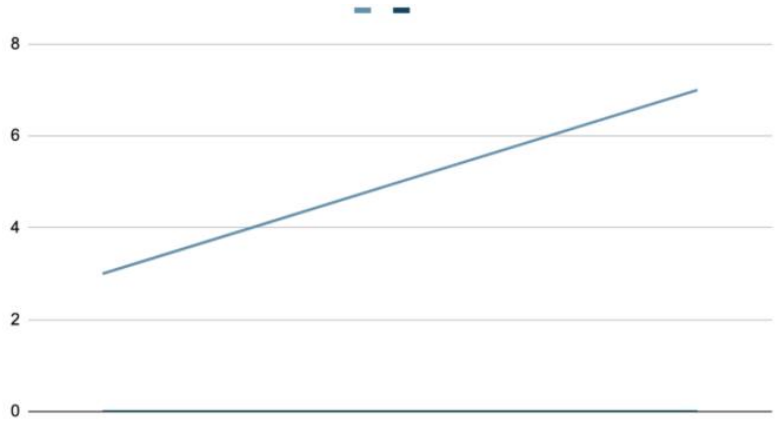
```
#include<stdio.h>
#include <time.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v)
{
   for(i = 1; i <= n; i++)
   if(a[v][i] && !visited[i])
   q[++r] = i;
   if(f <= r)
   {

      visited[q[f]] = 1;
      bfs(q[f++]);

   }

}
```

```
int main()
{
    int v;
    clock_t start, end;
    double t;
    printf("Enter the number of vertices: ");
    scanf("%d",&n);
    for(i=1; i <= n; i++)
    {

        q[i] = 0;

        visited[i] = 0;

    }

    printf("\nEnter graph data in matrix form:\n");
    for(i=1; i<=n; i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

    printf("Enter the starting vertex: ");
    scanf("%d", &v);
    bfs(v);
    printf("\nThe node which are reachable are:");
    for(i=1; i <= n; i++)
    {
        if(visited[i])
        printf(" %d", i);
        else
        {
            printf("\nBFS is not possible. All nodes are not reachable!");
            break;
        }

    }
    start = clock();
    bfs(v);
    end = clock();
```

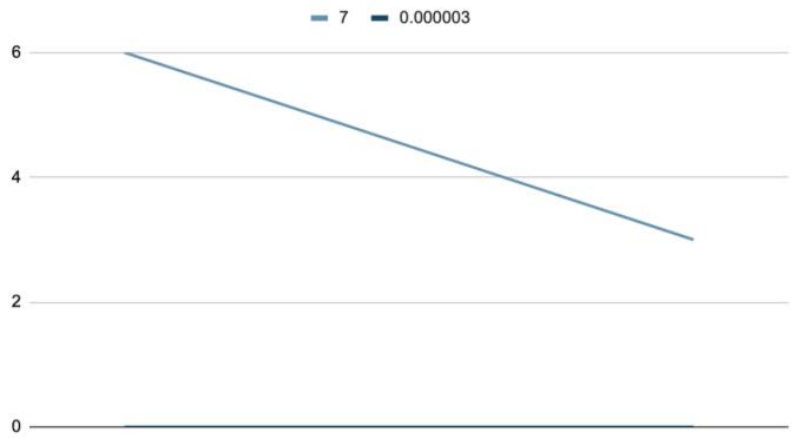
```
    t = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("\n");
    printf("\nTime taken by BFS : %lf\n", t);
    printf("\n");
    return 0;
}
```

```
Enter the number of vertices: 3

Enter graph data in matrix form:
1 1 1
0 1 0
0 1 1
Enter the starting vertex: 1

The node which are reachable are: 1 2 3

Time taken by BFS : 0.000002
```

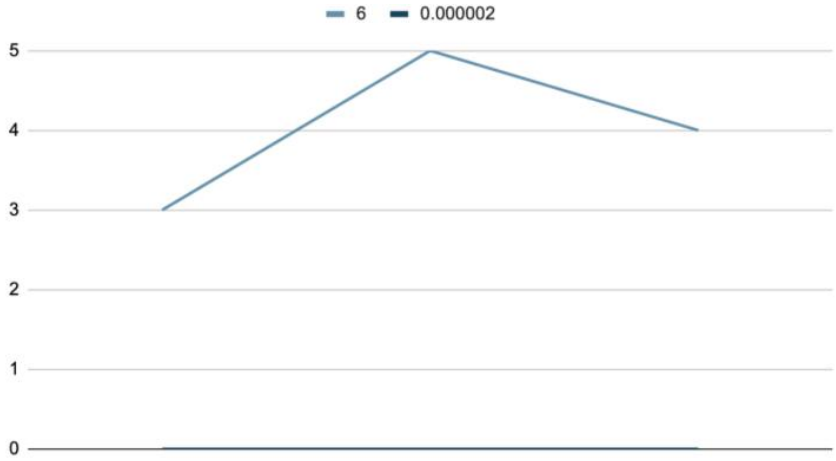
**BFS(ascending order):**




**BFS(descending order):**




**BFS(random order):**




**B)**

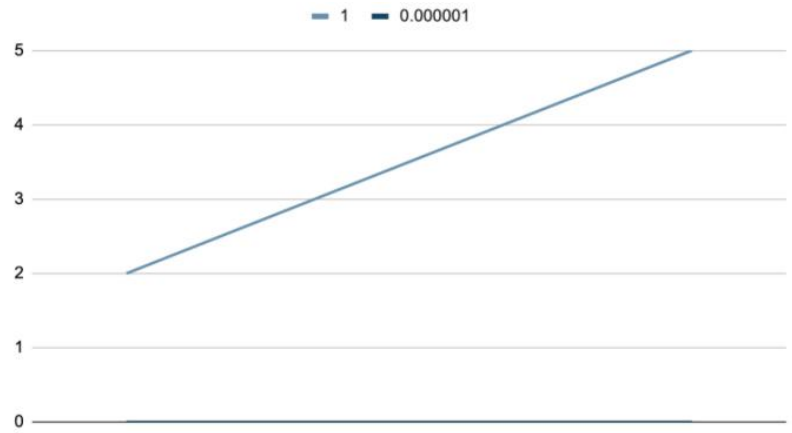
```
#include <stdio.h>
#include <time.h>
void dfs(int n, int cost[10][10], int u, int s[])
{
int v;
s[u]=1;
for(v=0;v<n;v++)
{
if((cost[u][v]==1) && (s[v]==0))
dfs(n,cost,v,s);
}
}
int main()
{
int n,i,j,cost[10][10],s[10],con,flag;
clock_t start, end;
```

```
double t;
printf("Enter the number of nodes\n");
scanf("%d",&n);
printf("Enter the adjacency matrix\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
scanf("%d",&cost[i][j]);
}
con=0;
for(j=0;j<n;j++)
{
for(i=0;i<n;i++)
s[i]=0;
dfs(n,cost,j,s);
flag=0;
for(i=0;i<n;i++)
{
if(s[i]==0)
flag=1;

}
if(flag==0)
con=1;

}
if(con==1)
printf("\nGraph is connected\n");
else
printf("\nGraph is not connected\n");
start = clock();
 dfs(n,cost,j,s);
 end = clock();
 t = ((double) (end - start)) / CLOCKS_PER_SEC;
 printf("\n");
 printf("\nTime taken by DFS : %lf\n", t);
 printf("\n");
return 0;
}
```

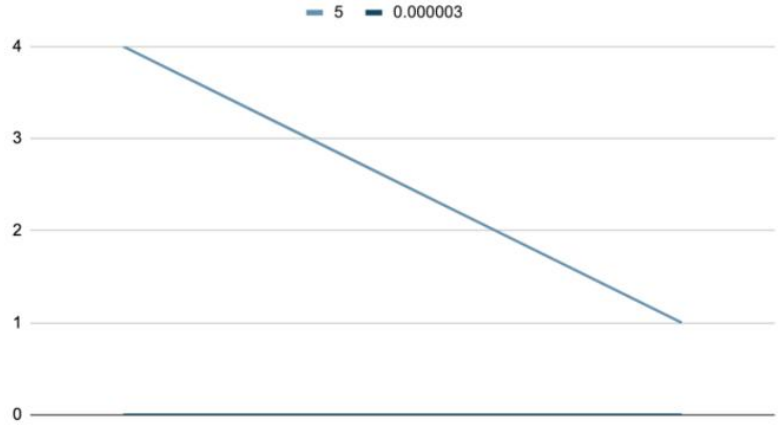
```
Enter the number of nodes
4
Enter the adjacency matrix
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0

Graph is connected


Time taken by DFS : 0.000002
```

```
Enter the number of nodes
2
Enter the adjacency matrix
1 2
2 3

Graph is not connected


Time taken by DFS : 0.000002
```

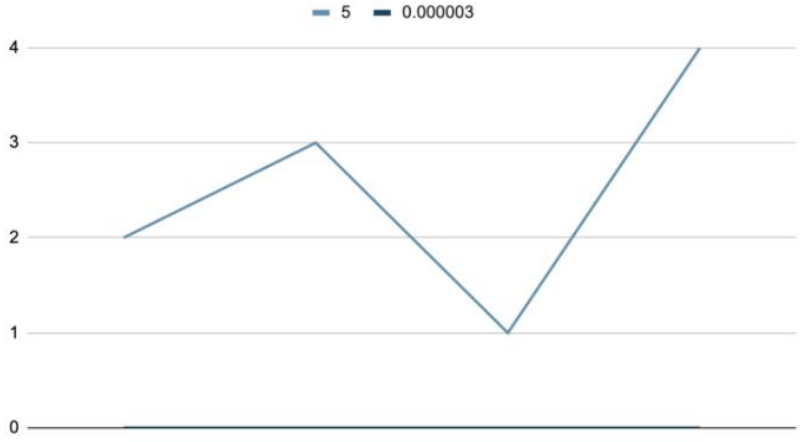
**DFS(ascending order):**




**DFS(descending order):**




**DFS(random order):**




**MODIFICATION:**

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
```

```
#include<string.h>
#include<time.h>
int q[100];
int visited[100];
int adj[20][20];
int n;

int front=-1, rear=-1;
void enqueue(int v)
{
   if(front==-1 && rear==-1)
   {
      front=rear=0;
   }
   if(rear==n-1)
   {
      printf("Queue Full\n");
      return;
   }
   q[rear]=v;
   rear++;
}
int dequeue()
{
   int val;
   if(front==-1 || front>rear)
   {
      //printf("Queue Underflow\n");
      return -1;
   }
   val=q[front];
   if(front==rear || front>rear)
   {
      front=-1;
      rear=-1;
   }
   front++;
   return val;
}
void bfs(int v)
{
   for(int i=0;i<n;i++)
   {
      if(adj[v][i]==1 && visited[i]==0)
```

```
        {
            enqueue(i);
            printf("%d\t",i);
            visited[i]=1;
        }
    }
    int val=dequeue();

    if(val!=-1)
    {
        bfs(val);
    }
    else
    {
        //printf("\n");
        return;
    }
}

int main()
{

    int flag=0;
    int ci=2;
    int v,count = 1;
    printf("Enter the Number of the vertex\n");
    scanf("%d",&n);
    printf("Enter the Entries Of The Adjacent Matrix\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&adj[i][j]);
        }
    }
    printf("Enter the Starting Vertex\n");
    scanf("%d",&v);
    printf("BREADTH ORDER TRAVERSAL FOR FOREST 1 IS\n");
    printf("%d\t",v);
    visited[v]=1;
    bfs(v);
```

```
   for(int i=0;i<n;i++)
   {
      if(visited[i]!= 1)
      {
         printf("\nTRAVERSAL \n");
         printf("\n%d\t",i);

         visited[i]=1;
         bfs(i);
         count++;
         flag = 1;
      }
   }
   if(flag==0)
   {
      printf("\nGRAPH IS CONNECTED\n");
   }
    if(flag==1)
   {
      printf("\nGRAPH IS NOT CONNECTED AND HAS %d PARTS\n",count);
   }
}
```

```
Enter the Number of the vertex
3
Enter the Entries Of The Adjacent Matrix
1 1 1
0 1 0
0 0 1
Enter the Starting Vertex
1
BREADTH ORDER TRAVERSAL FOR FOREST 1 IS
1
TRAVERSAL

0       2
GRAPH IS NOT CONNECTED AND HAS 2 PARTS
```