**PROGRAM 4 = DFS & BFS**

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

#include<stdio.h>

#include<time.h>

int arr[50][50],queue[50],visit[50],n,i,j,f=0,r=-1;

void bfs(int v)

{

for(i=1;i<=n;i++)

if(arr[v][i] && !visit[i])

queue[++r]=i;

if(f<=r)

{

visit[queue[f]]=1;

bfs(queue[f++]);

}

}

int main()

{

clock\_t start,end;

double time;

int v;

printf("\nEnter the number of nodes:");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

queue[i]=0;

visit[i]=0;

}

printf("\nEnter graph data in matrix form:\n");

for(i=1;i<=n;i++){

for(j=1;j<=n;j++){

scanf("%d",&arr[i][j]);

}

}

printf("\n Enter the starting vertex:");

scanf("%d",&v);

start = clock();

bfs(v);

end = clock();

time = ((double)(end - start))/CLOCKS\_PER\_SEC;

printf("\n The nodes which are reachable are:\n");

for(i=1;i<=n;i++)

if(visit[i])

printf("%d\t",i);

printf("\nTime taken : %lf\n",time);

}

OUTPUT :



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

#include<stdio.h>

#include<time.h>

int arr[50][50],reachable[50],n;

void dfs(int v)

{

int i;

reachable[v]=1;

for(i=1;i<=n;i++)

if(arr[v][i] && !reachable[i])

{

printf("\n %d->%d",v,i);

dfs(i);

}

}

int main()

{

clock\_t start,end;

double time;

int i,j,count=0;

printf("\nEnter number of nodes: ");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

reachable[i]=0;

for(j=1;j<=n;j++)

arr[i][j]=0;

}

printf("\nEnter the adjacency matrix: \n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&arr[i][j]);

start = clock();

dfs(1);

end = clock();

time = ((double)(end - start))/CLOCKS\_PER\_SEC;

printf("\n");

for(i=1;i<=n;i++)

{

if(reachable[i])

count++;

}

if(count==n)

printf("\nGraph is connected!");

else

printf("\nGraph is not connected!");

printf("\nTime taken : %lf\n",time);

}

OUTPUT:

