1. **Graph DFS implementation in C**

#include <stdio.h>

int A[10][10],visited[10],N;

void DFS(int i)

{

int j;

printf("%d ",i);

visited[i]=1;

for (j=0; j<N; j++)

{

if (A[i][j]==1 && !visited[j])

{

DFS(j);

}

}

}

int main()

{

int i,j,start;

printf("Input the number of vertices: ");

scanf("%d", &N);

printf("Enter Adjacency Matrix..");

for (i=0; i<N; i++)

for (j=0; j<N; j++)

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

for (i=0; i<N; i++)

visited[i]=0;

printf("Input the starting vertex for BFS traversal: ");

scanf("%d", &start);

DFS(start);

return 0;

}

Output:

Input the number of vertices: 4

Enter Adjacency Matrix..

0 1 0 1

1 0 1 0

0 1 0 0

1 0 0 0

Input the starting vertex for BFS traversal: 3

1. 0 1 2
2. **Graph BFS implementation**

#include <stdio.h>

#include <stdlib.h>

int A[10][10], visited[10], N;

// Function to perform BFS traversal

void BFS(int v)

{

int visited[10] = {0};

int queue[10];

int front = -1, rear = -1;

// Mark the startVertex as visited and enqueue it

visited[v] = 1;

queue[++rear] = v;

printf("BFS Traversal Order: ");

while (front != rear)

{

int currentVertex = queue[++front];

printf("%d ", currentVertex);

for (int i = 0; i < N; i++)

{

if (A[currentVertex][i] == 1 && !visited[i])

{

visited[i] = 1;

queue[++rear] = i;

}

}

}

printf("\n");

}

int main() {

int i,j,start;

// Input the number of vertices

printf("Input the number of vertices: ");

scanf("%d", &N);

printf("Enter Adjacency Matrix..");

for (i=0; i<N; i++)

{

for (j=0; j<N; j++)

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

}

// Input the starting vertex for BFS traversal

printf("Input the starting vertex for BFS traversal: ");

scanf("%d", &start);

// Perform BFS traversal

BFS(start);

return 0;

}

Output:

Input the number of vertices: 4

0 1 1 0

1 0 0 1

1 0 0 1

0 1 1 0

Input the starting vertex for BFS traversal: 3

BFS Traversal Order: 3 1 2 0