11/01/2021

**Experiment No:21**

**GRAPH TRAVERSAL**

**AIM:**

Write a program to create a graph using arrays and perform the following operations:

1. DFS Traversal

2. BFS Traversal

**DATA STRUCTURES USED:**

Graph using Arrays,Stack,Queue.

**ALGORITHM:**

Algorithm DFS

START

1. Push the starting vertex into the stack

2. While stack not empty

3. Pop a vertex v

4. If v is not in VISIT

5. Visit the vertex x

6. Store v in VISIT

7. Push all the adjacent vertices of v into stack

8. EndIf

9. EndWhile

STOP

Algorithm BFS

START

1. Enqueue starting vertex

2. Visit the vertex

3. Store the vertex in VISIT

4. While queue not empty

5. Dequeue a vertex v

6. For all the adjacent vertices w of v

7. If w is not in VISIT

8. Enqueue w

9. Visit w

10. Store w in VISIT

11. EndIf

12. EndFor

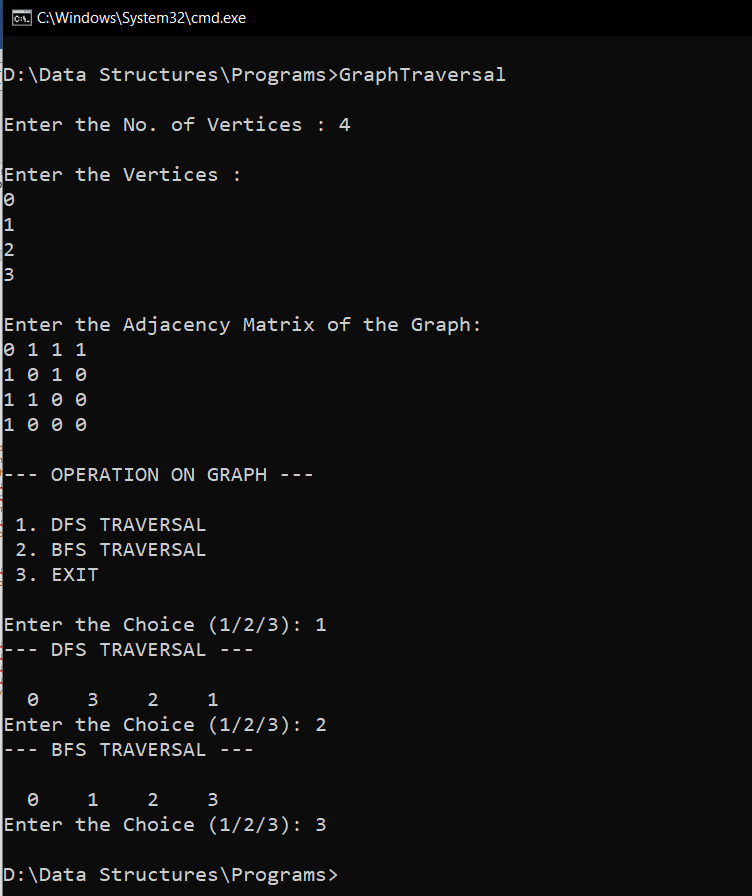
13. EndWhile

STOP

**PROGRAM:**

#include<stdio.h>  
#include<stdlib.h>  
int stack[100];  
int queue[100];  
int top = -1;  
int front = -1,rear=-1;  
void push(int x){  
 stack[++top] = x;  
}  
  
int pop(){  
 if(top!=-1){  
 int x = stack[top];  
 top--;  
 return x;  
 }  
}  
void enqueue(int x){  
 if(front == -1){  
 front = 0;  
 rear = 0;  
 queue[rear]=x;  
 }else{  
 queue[++rear]=x;  
 }  
}  
int dequeue(){  
 if(front != -1){  
 int x = queue[front];  
 if(front == rear){  
 front = -1;  
 rear = -1;  
 }else{  
 front = front + 1;  
 }  
 return x;  
 }  
}  
void dfs\_traversal(int n ,int value[], int adj[][n]){  
 int flag = 0;  
 int index=0,j,k;  
 int vertex;  
 int visit[n];  
 push(value[0]);  
 while(top!=-1){  
 vertex = pop();  
 for(j=0; j<n; j++){  
 if(visit[j] == vertex){  
 flag =1;  
 }  
 }  
 if(flag == 0){  
 visit[index] = vertex;  
 printf(" %d ",vertex);  
 for(j=0; j<n; j++){  
 if(value[j] == vertex){  
 for(k=0; k<n; k++){  
 if(adj[j][k] == 1 ){  
 push(value[k]);  
 }  
 }  
 break;  
 }  
 }  
 index++;  
 }  
 flag = 0;  
 }  
}  
void bfs\_traversal(int n ,int value[], int adj[][n]){  
 int index = 0;  
 int flag = 0,vertex,j,k,i;  
 int visit[n];  
 enqueue(value[0]);  
 printf(" %d ", value[0]);  
 visit[index++] = value[0];  
 while(front!= -1){  
 vertex = dequeue();  
 for( j=0; j<n; j++){  
 if(value[j] == vertex){  
 for(k=0; k<n; k++){  
 if(adj[j][k] == 1){  
 for( i=0; i<n; i++){  
 if(visit[i] == value[k]){  
 flag = 1;  
 }  
 }  
 if(flag == 0){  
 enqueue(value[k]);  
 printf(" %d ", value[k]);  
 visit[index] = value[k];  
 index++;  
 }  
 flag = 0;  
 }  
 }  
 break;  
 }  
 }  
 }  
}  
void main(){  
 int n,i,j,op;  
 char ans='y';  
 printf("\nEnter the No. of Vertices : ");  
 scanf("%d", &n);  
 int adj[n][n],value[n];  
 printf("\nEnter the Vertices : \n");  
 for(i=0; i<n; i++){  
 scanf("%d", &value[i]);  
 }  
 printf("\nEnter the Adjacency Matrix of the Graph:\n");  
 for(i=0; i<n; i++){  
 for(j=0; j<n; j++){  
 scanf("%d", &adj[i][j]);  
 }  
 }  
 printf("\n--- OPERATION ON GRAPH --- \n\n");  
 printf(" 1. DFS TRAVERSAL\n");  
 printf(" 2. BFS TRAVERSAL\n");  
 printf(" 3. EXIT \n");  
 while(ans=='y'){  
 printf("\nEnter the Choice (1/2/3): ");  
 scanf("%d",&op);  
 switch(op){  
 case 1:printf("--- DFS TRAVERSAL ---\n\n");  
 dfs\_traversal(n,value,adj);  
 break;  
 case 2:printf("--- BFS TRAVERSAL ---\n\n");  
 bfs\_traversal(n,value,adj);  
 break;  
 case 3:ans='n';  
 break;  
 default:printf("Enter a Valid Input\n");  
 }  
 }  
}

**OUTPUT:**



**RESULT:**

The given operations are performed on a graph using arrays.