

program 12: BFS & DFS

Date _____

Page _____

a) Binary Search Breadth First Search ..

b) Depth First Search

pseudocode void BFS (int start) {

 while (front < rear) {

 int node = queue[front++];

 for (int i=0; i<n; i++) {

 if (graph[node][i] == 1 && !visited[i]) {

 ("enqueue visited[i]=1, index "i") string
 queue[rear++] = i; // (i, 0, 33) front

 ("add vertex node(i) value "i") string

void DFS (int v) {

 visited[v] = 1; // (v, 0, 1) init.

 for (int i=0; i<n; i++) {

 if (adj[v][i] == 1 && !visited[i]) {

 DFS(i); // (i, 0, 0, 1) init.

 }

 ("exit from visited node "v") string

code: #include <stdio.h> int h, l, n;

int graph[20][20], visited[20], n;

 ("enter matrix 2x20 ")); // input

void BFS(int start) {

 int queue[20], front = 0, rear = 0, n;

 visited[start] = 1;

 queue[rear++] = start;

 ("initialise to read more reading begin")

 while (front < rear) {

 int node = queue[front++];

 printf ("%d ", node);

 for (int i=0; i<n; i++) {

 if (graph[node][i] == 1 && !visited[i]) {

visited[i] = 1; // initializes all vertices as unvisited
 queue[rear++] = i; // adds start vertex to queue
 ?
 } } // (front >= rear) == false
 : [++ front] = start tri
 int main() { int n; // 0 <= n <= 100;
 if (scanf("%d", &n) != 1) {
 printf("enter number of vertices: ");
 scanf("%d", &n);
 printf("enter adjacency matrix: \n");
 for (int i=0; i<n; i++) {
 for (int j=0; j<n; j++) {
 scanf("%d", &graph[i][j]);
 } } // for i = 0 to n-1 for j = 0 to n-1
 for (int i=0; i<n; i++) {
 visited[i] = 0; // initializes all vertices as unvisited
 } } // for i = 0 to n-1

Output: Enter number of vertices: 4

Enter adjacency matrix: > (front = start) = 0

0 1 1 0 // (front) = start = 0 <= i <= n-1

0 0 1 0 // (start <= i <= n-1) = 1 <= j <= n-1

1 0 0 1 // (0 <= i <= n-1) = 1 <= j <= n-1

0 0 0 0 // (0 <= i <= n-1) = 1 <= j <= n-1

Enter starting vertex: 2

BFS traversal : 2031

code: #include <stdio.h>

#define MAX 10

int visited[MAX], adj[MAX][MAX];

int n;

void DFSC(int v) {

visited[v] = 1;

printf("%d", v);

for (int i=0; i<n; i++) {

if (adj[v][i] == 1 && !visited[i]) {

DFSC(i);

}

}

int main() {

printf("Enter number of vertices: ");

scanf("%d", &n);

printf("Enter adjacency matrix: \n");

for (int i=0; i<n; i++) {

for (int j=0; j<n; j++) {

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

}

}

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

visited[i] = 0;

```
printf("DFS Traversal starting from vertex  
DFS(0);  
return 0;
```

۳

Enter number of vertices: 6

Enter adjacency matrix in tabular form.

0 1 1 0 0 0

0 0 0 1 1 0

6 0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

3 6 9

0 0 0 0 0 0

~~DFS Traversal starting from vertex 0:~~

~~0 1 3 4 5 2 1, 001:0=1 101, 001~~

~~3 (G3) belief 1~~ ~~1 = C7EV7~~ the 2 11

~~OK~~
See you
15/12/15

if we try to reduce redshift differences

(e.g., "bogus" passes)

3. *Ammonia* with carb.

3 (4+6) as if (or if true) soft

1. (1980) 108, "b," 1982

(+signi) (as i tai) sal-

$$i_0 = [1] \text{ kg} / \text{m}^2$$