Assignment-2

Uninformed Search

BFS :

#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTICES 8 // Number of vertices in the graph

// Queue data structure for BFS

struct Queue {

int items[MAX\_VERTICES];

int front;

int rear;

};

// Graph representation using adjacency matrix

int graph[MAX\_VERTICES][MAX\_VERTICES] = {

{0, 1, 1, 1, 0, 0, 0, 0}, // A

{1, 0, 1, 0, 1, 0, 0, 0}, // B

{1, 1, 0, 1, 0, 1, 0, 0}, // C

{1, 0, 1, 0, 0, 0, 0, 0}, // D

{0, 1, 0, 0, 0, 0, 0, 0}, // E

{0, 0, 1, 0, 1, 0, 1, 1}, // F

{0, 0, 0, 0, 1, 1, 0, 1}, // G

{0, 0, 0, 0, 0, 1, 1, 0} // H

};

// Queue functions

struct Queue\* createQueue();

void enqueue(struct Queue\* q, int value);

int dequeue(struct Queue\* q);

int isEmpty(struct Queue\* q);

// BFS traversal function

void bfs(int startVertex) {

struct Queue\* q = createQueue();

int visited[MAX\_VERTICES] = {0}; // To keep track of visited vertices

visited[startVertex] = 1;

enqueue(q, startVertex);

printf("BFS traversal starting from vertex %c: ", 'A' + startVertex);

while (!isEmpty(q)) {

int currentVertex = dequeue(q);

printf("%c ", 'A' + currentVertex);

for (int i = 0; i < MAX\_VERTICES; ++i) {

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

enqueue(q, i);

visited[i] = 1;

}

}

}

printf("\n");

}

int main() {

bfs(0); // Start BFS traversal from vertex A (index 0)

return 0;

}

// Queue implementation

struct Queue\* createQueue() {

struct Queue\* q = (struct Queue\*)malloc(sizeof(struct Queue));

q->front = -1;

q->rear = -1;

return q;

}

void enqueue(struct Queue\* q, int value) {

if (q->rear == MAX\_VERTICES - 1) {

printf("Queue is full\n");

} else {

if (q->front == -1) {

q->front = 0;

}

q->rear++;

q->items[q->rear] = value;

}

}

int dequeue(struct Queue\* q) {

int item;

if (isEmpty(q)) {

printf("Queue is empty\n");

item = -1;

} else {

item = q->items[q->front];

q->front++;

if (q->front > q->rear) {

q->front = q->rear = -1;

}

}

return item;

}

int isEmpty(struct Queue\* q) {

return q->rear == -1;

}

DFS :

#include <stdio.h>

#include <stdbool.h>

#define VERTICES 8

// Function to perform DFS traversal using a stack

void dfs(int graph[VERTICES][VERTICES], int start) {

bool visited[VERTICES] = {false};

int stack[VERTICES];

int top = -1;

stack[++top] = start;

visited[start] = true;

printf("DFS traversal starting from vertex %c: ", start + 'A');

while (top != -1) {

int current = stack[top--];

// printf("%d\n",top);

printf("%c ", current + 'A');

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

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

stack[++top] = i;

visited[i] = true;

}

}

}

printf("\n");

}

int main() {

int adjacencyMatrix[VERTICES][VERTICES] = {

{0, 1, 1, 1, 0, 0, 0, 0}, // A

{1, 0, 1, 0, 1, 0, 0, 0}, // B

{1, 1, 0, 1, 0, 1, 0, 0}, // C

{1, 0, 1, 0, 0, 0, 0, 0}, // D

{0, 1, 0, 0, 0, 0, 0, 0}, // E

{0, 0, 1, 0, 1, 0, 1, 1}, // F

{0, 0, 0, 0, 1, 1, 0, 1}, // G

{0, 0, 0, 0, 0, 1, 1, 0} // H

};

dfs(adjacencyMatrix, 0); // Starting from vertex A (index 0)

return 0;

}