BFS CODE:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define size 7

void push(int a);

int pop();

void display();

void bfs(int graph[][7]);

int fpos = -1, rpos = -1;

int queue[size];

int main(){

int adj\_matrix[7][7] = {

{0, 1, 0, 1, 0, 0, 0},

{1, 0, 1, 1, 0, 1, 1},

{0, 1, 0, 1, 1, 1, 0},

{1, 1, 1, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 1},

{0, 1, 1, 0, 0, 0, 0},

{0, 1, 0, 0, 1, 0, 0},

};

for(int i = 0; i < 7; i++) queue[i] = NULL;

// display();

bfs(adj\_matrix);

return 0;

}

void bfs(int graph[][7]){

int visited[7];

for(int i = 0; i < 7; i++) visited[i] = 0;

push(0); visited[0]= 1;

while (fpos != size){

for(int i = 0; i < 7; i++){

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

push(i);

visited[i] = 1;

// break;

}

}

printf("%d ", pop());

// printf("%d\n", new\_node);

}

}

void push(int a){

if (fpos == -1 && rpos == -1){

queue[++rpos] = a;

fpos++;

return;

}

else if (rpos == size-1){

printf("Queue overflow condition\n");

return;

}

else{

queue[++rpos] = a;

return;

}

}

int pop(){

if (fpos == -1){

printf("Queue Underflow condition\n");

}

int n = queue[fpos];

queue[fpos] = (int) NULL;

fpos++;

return n;

}

void display(){

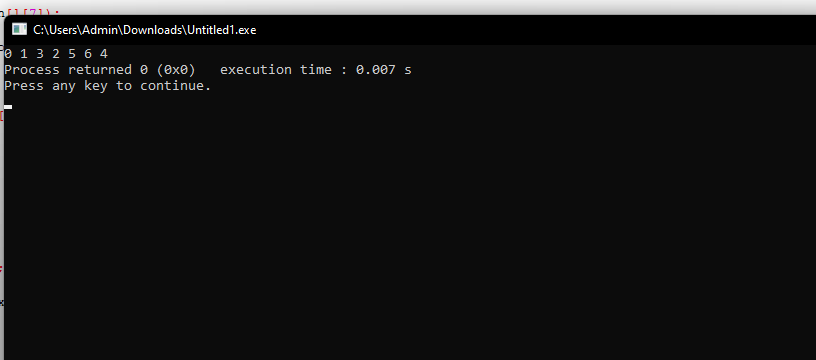
printf("Queue: ");

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

printf("%d ", queue[i]);

printf("\n");

}



DFS CODE:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define size 7

int pos = -1;

int stack[size];

void push(int a);

int pop();

void display();

void dfs(int graph[][7]);

int main(){

int adj\_matrix[7][7] = {

{0, 1, 0, 1, 0, 0, 0},

{1, 0, 1, 1, 0, 1, 1},

{0, 1, 0, 1, 1, 1, 0},

{1, 1, 1, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 1},

{0, 1, 1, 0, 0, 0, 0},

{0, 1, 0, 0, 1, 0, 0},

};

for(int i = 0; i < 7; i++) stack[i] = NULL;

// display();

dfs(adj\_matrix);

return 0;

}

void dfs(int graph[][7]){

int visited[7];

for (int i = 0; i < 7; i++) visited[i] = 0;

push(0); visited[0] = 1; printf("0 ");

// printf("%d ", pos);

// return;

// display();

while(pos != -1){

bool new\_node = false;

for(int i = 0; i < 7; i++){

// printf("%d ", graph[stack[pos]][i]);

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

new\_node = true;

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

push(i);

// display();

visited[i] = 1; printf("%d ", i);

break;

}

}

// printf("%d\n", new\_node);

if (!new\_node) pop();

}

}

void push(int a){

if (pos == size-1){

printf("Stack Overflow condition");

return;

}

stack[++pos] = a;

}

int pop(){

if (pos == -1){

printf("Stack Underflow condition");

return (int) NULL;

}

return stack[pos--];

}

void display(){

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

printf("%d ", stack[i]);

}

printf("\n");

}

