**BREADTH FIRST SEARCH-BFS**

**Aim:**

The aim of the program is to implement Breadth First Search using C programming Language.

**Algorithm:**

1. Start.
2. Create an empty queue Q.
3. Mark all vertices as unvisited.
4. Mark S as visited.
5. Enqueue s into Q.
6. Dequeue the front vertex from Q.
7. Traverse the graph.
8. Mark v as visited.
9. Enqueue v into Q.
10. End.

**Coding:**

#include<stdio.h>

#include<stdlib.h>

struct queue

{

int size;

int f;

int r;

int\* arr;

};

int isEmpty(struct queue \*q){

if(q->r==q->f){

return 1;

}

return 0;

}

int isFull(struct queue \*q){

if(q->r==q->size-1){

return 1;

}

return 0;

}

void enqueue(struct queue \*q, int val){

if(isFull(q)){

printf("This Queue is full\n");

}

else{

q->r++;

q->arr[q->r] = val;

// printf("Enqued element: %d\n", val);

}

}

int dequeue(struct queue \*q){

int a = -1;

if(isEmpty(q)){

printf("This Queue is empty\n");

}

else{

q->f++;

a = q->arr[q->f];

}

return a;

}

int main(){

// Initializing Queue (Array Implementation)

struct queue q;

q.size = 400;

q.f = q.r = 0;

q.arr = (int\*) malloc(q.size\*sizeof(int));

// BFS Implementation

int node;

int i = 1;

int visited[7] = {0,0,0,0,0,0,0};

int a [7][7] = {

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

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

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

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

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

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

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

};

printf("%d", i);

visited[i] = 1;

enqueue(&q, i); // Enqueue i for exploration

while (!isEmpty(&q))

{

int node = dequeue(&q);

for (int j = 0; j < 7; j++)

{

if(a[node][j] ==1 && visited[j] == 0){

printf("%d", j);

visited[j] = 1;

enqueue(&q, j);

}

}

}

return 0;

}

**Output:**

1 0 2 3 4 5 6

**Result:**

The program has been successfully implemented.

**DEPTH FIRST SEARCH-DFS**

**Aim:**

The aim of the program is to implement Depth First Search using C programming Language.

**Algorithm:**

1. Start.
2. Create a stack and push the starting vertex.
3. Mark the starting vertex as visited.
4. Pop a vertex from the stack.
5. If the neighbour has not been visited, mark it as visited
6. Push the neighbour onto the stack.
7. End.

**Program:**

#include<stdio.h>

#include<stdlib.h>

int visited[7] = {0,0,0,0,0,0,0};

int A [7][7] = {

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

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

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

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

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

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

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

};

void DFS(int i){

printf("%d ", i);

visited[i] = 1;

for (int j = 0; j < 7; j++)

{

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

DFS(j);

}

}

}

int main(){

DFS(0);

return 0;

}

**Output:**

0 1 2 3 4 5 6

**Result:**

The program has been successfully implemented.