**BFS TRAVERSAL**

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

#define initial 1

#define waiting 2

#define visited 3

int n;

int adj[MAX][MAX];

int state[MAX];

void create\_graph();

void BF\_Traversal();

void BFS(int v);

int queue[MAX], front = -1,rear = -1;

void insert\_queue(int vertex);

int delete\_queue();

int isEmpty\_queue();

int main()

{

    create\_graph();

    BF\_Traversal();

    return 0;

}

void BF\_Traversal()

{

    int v;

    for(v=0; v<n; v++)

        state[v] = initial;

    printf("Enter Start Vertex for BFS: \n");

    scanf("%d", &v);

    BFS(v);

}

void BFS(int v)

{

    int i;

    insert\_queue(v);

    state[v] = waiting;

    while(!isEmpty\_queue())

    {

        v = delete\_queue( );

        printf("%d ",v);

        state[v] = visited;

        for(i=0; i<n; i++)

        {

            if(adj[v][i] == 1 && state[i] == initial)

            {

                insert\_queue(i);

                state[i] = waiting;

            }

        }

    }

    printf("\n");

}

void insert\_queue(int vertex)

{

    if(rear == MAX-1)

        printf("Queue Overflow\n");

    else

    {

        if(front == -1)

            front = 0;

        rear = rear+1;

        queue[rear] = vertex ;

    }

}

int isEmpty\_queue()

{

    if(front == -1 || front > rear)

        return 1;

    else

        return 0;

}

int delete\_queue()

{

    int delete\_item;

    if(front == -1 || front > rear)

    {

        printf("Queue Underflow\n");

        exit(1);

    }

    delete\_item = queue[front];

    front = front+1;

    return delete\_item;

}

void create\_graph()

{

|  |
| --- |
| int v; |
|  |

|  |
| --- |
| printf("\n Enter the number of vertices:"); |
|  |

|  |
| --- |
| scanf("%d", &n); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| printf("\n Enter graph data in matrix form:\n"); |
|  |

|  |
| --- |
| for(i=1; i<=n; i++) { |
|  |

|  |
| --- |
| for(j=1;j<=n;j++) { |
|  |

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

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |

}