Q1:

1. DFS-iterative (G, s): //Where G is graph and s is source vertex

Let S be stack

S.push(s) //Inserting s in stack

Mark s as visited.

While (S is not empty):

//Pop a vertex from stack to visit next

V = S.top()

S.pop()

//Push all the neighbors of v in stack that are not visited

For all neighbours w of v in Graph G:

If w is not visited:

S.push(w)

Mark w as visited

DFS-recursive(G, s):

Mark s as visited

For all neighbors w of s in Graph G:

If w is not visited:

DFS-recursive(G, w)

1. BFS(G, s) //Where G is the graph and s is the source node

Let Q be queue.

Q.enqueue(s) //Inserting s in queue until all its neighbor vertices are makred

Mark s as visited

While (Q is not empty)

//Removing that vertex from queue, whose neighbor will be visited now

V = Q.dequeue()

//Processing all the neighbors of v

For all neighbors w of v in Graph G

If w is not visited

Q.enqueue(w) //Stores w in Q to further visit its neighbor

Mark w as visited

Q2:

1. Figure(a)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0 |  | 1 | 1 |  |  |  |  |  |  |
| 1 |  |  |  |  | 1 |  |  |  |  |
| 2 |  | 1 |  |  |  | 1 |  |  |  |
| 3 |  | 1 | 1 |  | 1 | 1 |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  | 1 |  |  |  |  |
| 6 |  |  |  |  |  |  |  | 1 | 1 |
| 7 |  |  |  |  |  |  |  |  | 1 |
| 8 |  |  |  |  |  |  |  |  |  |

Figure(b)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 0 |  | 1 | 1 | 1 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  | 1 |  |  |  |  |
| 2 |  |  |  | 1 |  |  | 1 |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  | 1 |  |  |  |  |  | 1 | 1 |  |  |
| 5 |  |  |  | 1 |  |  |  |  |  |  | 1 |  |
| 6 |  |  |  |  |  |  |  | 1 |  |  |  | 1 |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  | 1 |  |  | 1 |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  | 1 |  | 1 |  | 1 |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  |

b)

Figure (a)

0: [1, 2]

1: [4]

2: [1, 5]

3: [1, 2, 4, 5]

4: []

5: [4]

6: [7, 8]

7: [8]

8: []

Figure (b)

0: [1, 2, 5]

1: [7]

2: [3, 6]

3: []

4: [2, 8, 9]

5: [3, 10]

6: [7, 11]

7: []

8: [6, 9]

9: []

10: [7, 9, 11]

11: []

c) The Breadth first traversal order for graph (a) is: 0->1->2->4->5->6->7->8

The Breadth first traversal order for graph (a) is: 0->1->2->5->7->3->6->10->11->9->4->8

d) The Depth first traversal order for graph (a) is: 0->1->4->2->5->3->6->8->7

The Depth first traversal order for graph (b) is: 0->1->7->2->3->6->11->5->10->9->4->8