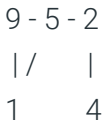
Graph-1 Questions

Question 1 :

Detect cycle in an undirected graph using BFS

We have an an undirected graph, how to check if there is a cycle in the graph? For example, the following graph has a cycle 1-0-2-1..

Input :



Output : Yes

Question 2 :

Find Minimum Depth of a Binary Tree

We have a binary tree, find its minimum depth. The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node. For example, minimum height of below Binary Tree is 2.

Sample Input 1 :

Sample Output 1 : 2

Question 3 :

Minimum time required to rot all oranges

We have a matrix of dimension m*n where each cell in the matrix can have values 0, 1 or 2 which has the following meaning:

0: Empty cell 1: Cells have fresh oranges 2: Cells have rotten oranges

We have to determine what is the minimum time required so that all the oranges become rotten. A rotten orange at index [i,j] can rot other fresh orange at indexes [i-1,j], [i+1,j], [i,j-1], [i,j+1] (up, down, left and right). If it is impossible to rot every orange then simply return -1.

Input: arr[][C] = { {2, 1, 0, 2, 1},
{0, 0, 1, 2, 1},
{1, 0, 0, 2, 1}};

Output:

All oranges cannot be rotten.

Explanation:

At 0th time frame:

{2, 1, 0, 2, 1}

{0, 0, 1, 2, 1}

{1, 0, 0, 2, 1}

At 1st time frame:

{2, 2, 0, 2, 2}

{0, 0, 2, 2, 2}

{1, 0, 0, 2, 2}

At 2nd time frame:

{2, 2, 0, 2, 2}

{0, 0, 2, 2, 2}

{1, 0, 0, 2, 2}

...

The 1 at the bottom left corner of the matrix is never rotten.

Question 4 :

Find the size of the largest region in the Boolean Matrix

We have a matrix where each cell contains either a '0' or a '1', and any cell containing a 1 is called a filled cell. Two cells are said to be connected if they are adjacent to each other horizontally, vertically, or diagonally. If one or more filled cells are also connected, they form a region. find the size of the largest region.

Input: M[][5] = { {0, 0, 1, 1, 0}, {0, 0, 1, 1, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0 1} }

Output: 4

Question 5 :

Word Ladder

We have a dictionary, and two words 'start' and 'target' (both of same length). Find length of the smallest chain from 'start' to 'target' if it exists, such that adjacent words in the chain only differ by one character and each word in the chain is a valid word i.e., it exists in the dictionary. It may be assumed that the 'target' word exists in dictionary and length of all dictionary words is same.

Input:
Dictionary = {POON, PLEE, SAME, POIE, PLEA, PLIE, POIN},
start = TOON, target = PLEA
Output: 7

Explanation: TOON – POON – POIN – POIE – PLIE – PLEE – PLEA