**[Breadth/Depth First Search](https://leetcode.com/explore/learn/card/graph/) for Graph/Tree Problems**

1. [Delete Node in a BST](https://leetcode.com/problems/delete-node-in-a-bst/)

Diagram

Description automatically generated

Text

Description automatically generated

1. [Populating Next Right Pointers in Each Node II](https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii/)

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. [Network Delay Time](https://leetcode.com/problems/network-delay-time/)

Text, letter

Description automatically generated

Dijkstra's Algorithm

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. [Deepest Leaves Sum](https://leetcode.com/problems/deepest-leaves-sum/)

A picture containing text

Description automatically generated

1. [Shortest Path in a Binary Matrix](https://leetcode.com/problems/shortest-path-in-binary-matrix/)

Finding the shortest path between two nodes in a graph is almost always done in BFS. How to get all the neighbors of a cell? In traditional graph representations, this would be the equivalent of examining all the edges of a given node.

For Grids, we identify each neighbor by its row and column offset from the give cell.

Table

Description automatically generated

The most common pattern is to put these offsets into a list.

directions = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]

Note: Most cells have 8 neighbors, corner cells only have 3 neighbors, and edges cell have 5 neighbors, to handle this, we can start checking that the neighbor’s row and column actually are within the dimensions of the grid.

You should **always** discuss the possibility of overwriting the input with your interviewer and clarify what kind of environment your algorithm is expected to run in. Sometimes they won't care, sometimes they'll state it has to run in a multithreaded environment, or sometimes they'll have a particular preference as it impacts what they're trying to see from you.

**Approach 1: BFS, Overwriting Input**

We get the following

**Bubble chart

Description automatically generated with low confidence**

A picture containing text

Description automatically generated

Approach 2: BFS (Without overwriting the input)

*Keeping track of how many cells at each distance are on the queue*

A picture containing timeline

Description automatically generated

1. [Find a Corresponding Node of a Binary Tree in a Clone of that Tree](https://leetcode.com/problems/find-a-corresponding-node-of-a-binary-tree-in-a-clone-of-that-tree/)

Graphical user interface, text, application

Description automatically generated

1. [Critical Connections in a Network](https://leetcode.com/problems/critical-connections-in-a-network/)

Graphical user interface

Description automatically generated with medium confidence

1. [Find Largest Value in Each Tree Row](https://leetcode.com/problems/find-largest-value-in-each-tree-row/)

Text

Description automatically generated

1. Interview Question-Zillow

Given an ascending integer array e.g. [0,0,0,2,2,3,3]

Return an array of the ascending indices (an index where the number is number is larger than its previous element).

For the given input, the answer is [3,5],

The new condition:

The new condition: The number of unique values in the ascending array is less than 5, However the array's length could still be very large. [0,0,0,0,0....0,0, 1,1,1,1,1,1....,1, 4,4,4,4]

Text

Description automatically generated

1. [Two Sum IV – Input is a BST](https://leetcode.com/problems/two-sum-iv-input-is-a-bst/)

Graphical user interface, text

Description automatically generated

**Approach 2:** In this approach, we make use of the fact that the given tree is a binary search tree. Now, we know that the inorder traversal of a BST gives the nodes in ascending order. Thus, we do the inorder traversal of the given tree and put the result in a list which contains the node in sorted in ascending order.

Once this is done, we make use of two pointers left and right pointing to the beginning and end of the sorted list.

* Check if the sum of the elements pointed by left and right is equal to required sum k. if so return true,
* Otherwise, if the sum of current two elements is lesser than the required sum k, update left to the point to the next element. This is done , because, we need to increase the sum of the current elements, which can only be done by increasing the smaller number.
* Otherwise, if the sum of the current two elements is larger than the required sum k, update right to the previous element. This is done because, we need to decrease the sum of the current elements, which can only be done by reducing the larger number.
* Continue till left crosses the right pointer.
* If two pointers cross each other, return false.

Text

Description automatically generated