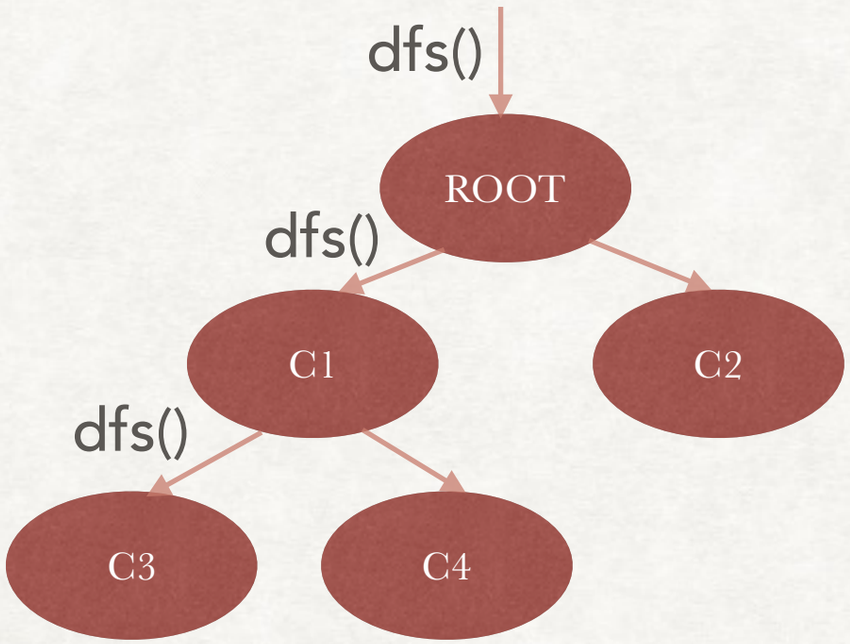
DEPTH FIRST SEARCH

CONCEPT

- Go deep first, usually use recursive approach



PROBLEM LIST

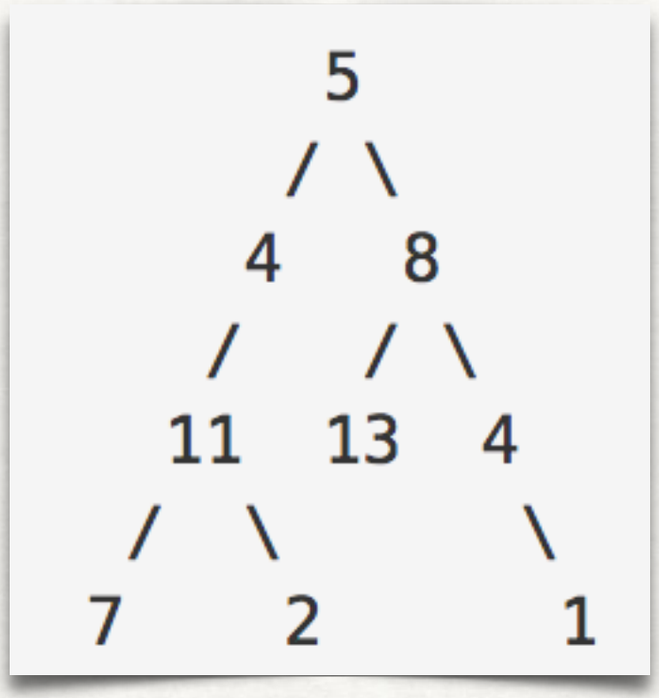
- 112. Path Sum
- 113. Path Sum II
- 199. Binary Tree Right Side View
- 200. Number of Islands
- 236. Lowest Common Ancestor of a Binary Tree
- 257. Binary Tree Paths
- 332. Reconstruct Itinerary
- 399. Evaluate Division
- 417. Pacific Atlantic Water Flow
- 464. Can I Win

NO. 112

PATH SUM

PROBLEM DESCRIPTION

- Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.



- Given the below binary tree and sum = 22, return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22.

IDEA

- dfs(root, sum)
  - dfs(root.left, sum - root.val) if root.left is not None
    - dfs(root.left, sum - root.val) if root.left is not None
    - dfs(root.right, sum - root.val) if root.right is not None
- dfs(root.left, sum - root.val) if root.right is not None

return True if sum == root.val else False

SOLUTION

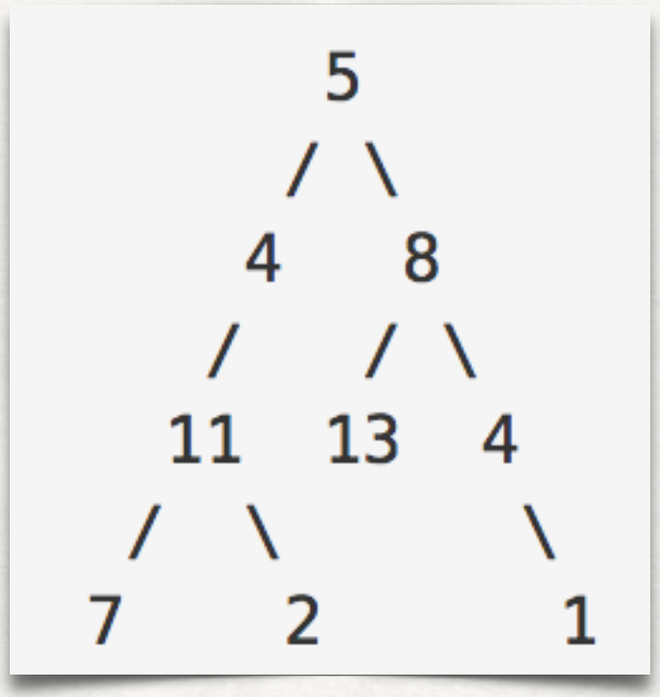
- https://github.com/Brady31027/leetcode/tree/master/112_Path_Sum

NO. 113

PATH SUM II

PROBLEM DESCRIPTION

- Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.



- For example, given the below binary tree and sum = 22
- return [[5,4,11,2], [5,8,4,5]]

IDEA

- Similar to Path Sum except
  - Keep tracking the nodes in local list
    - Since list is a local variable, system will put them in stack
  - stack contents will change when we go to another frame

SOLUTION

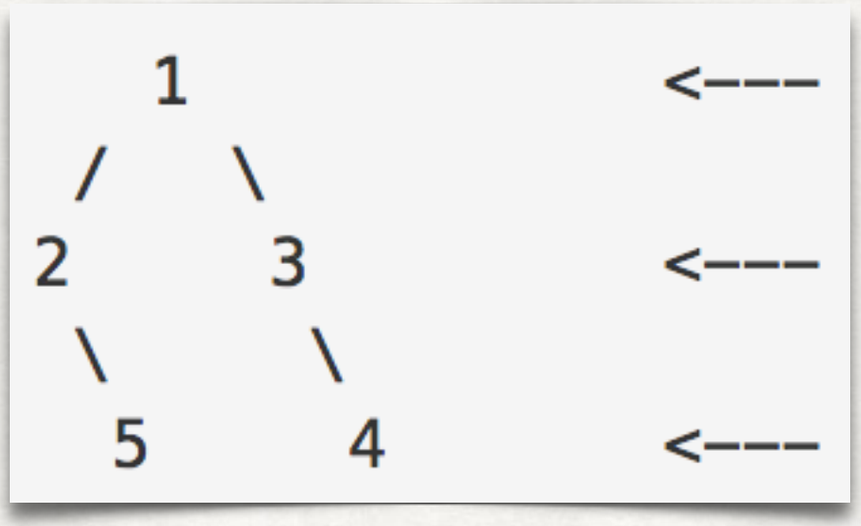
- https://github.com/Brady31027/leetcode/tree/master/113_Path_Sum_II

NO. 199

BINARY TREE RIGHT SIDE VIEW

PROBLEM DESCRIPTION

- Given a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.
- For example, given the following binary tree



- You should return [1, 3, 4].

IDEA

- Using BFS seems more straight-forward
  - level by level BFS
  - record the last node in every level
- How to solve this quiz applying DFS?

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/199_Binary_Tree_Right_Side_View

NO. 200

NUMBER OF ISLANDS

PROBLEM DESCRIPTION

- Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

```
11110
11010
11000
00000
```

Answer: 1

```
11000
11000
00100
00011
```

Answer: 3

IDEA

- Count connected component
  - DP is not easy
    - E.g.

1	1	1
	1	
1	1	1

- DFS
  - Build up a 2D visited map to track the history
  - Once we find a “land”, start to flood from it (recursive)
  - Every time we find another “land” which is never recorded in visited map, then it is a NEW ISLAND

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/200_Number_of_Islands

NO. 235

LOWEST COMMON ANCESTOR OF A BINARY SEARCH TREE

PROBLEM DESCRIPTION

- Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.
- According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow a node to be a descendant of itself).”
- For example, the lowest common ancestor (LCA) of nodes 2 and 8 is 6. Another example is LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself according to the LCA definition.

IDEA

- BST
  - Binary tree
  - left node <= root node <= right node

```
if (root.val - p.val) * ( root.val - q.val) <= 0:
        return root
```

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/235_Lowest_Common_Ancestor_of_a_Binary_Search_Tree

NO. 236

LOWEST COMMON ANCESTOR OF A BINARY TREE

PROBLEM DESCRIPTION

- Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.
- According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow a node to be a descendant of itself).”

IDEA

- Not a BST
  - Traverse to leaves
- Go left to see if there is target nodes (p or q)
- Go right to see if there is target nodes (p or q)
- If both left and right, current root is the LCA
- If left, the left child of current root is LCA
- If right, the right child of current root is LCA

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/236_Lowest_Common_Ancestor_of_a_Binary_Tree

NO. 257

BINARY TREE PATHS

PROBLEM DESCRIPTION

- Given a binary tree, return all root-to-leaf paths.

IDEA

- DFS from root, nothing special

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/257_Binary_Tree_Paths

NO. 332

RECONSTRUCT ITINERARY

PROBLEM DESCRIPTION

- Given a list of airline tickets represented by pairs of departure and arrival airports [from, to], reconstruct the itinerary in order. All of the tickets belong to a man who departs from JFK. Thus, the itinerary must begin with JFK.

IDEA

- DFS
- Maintain 2 type of subtrees
  - if src in the subtree, name it left tree (in order to fly back)
  - if src not in the subtree, name it right tree (no need to fly back)
  - ans = [start] + left + right
- Should have a better solution!

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/332_Reconstruct_Itinerary

NO. 399

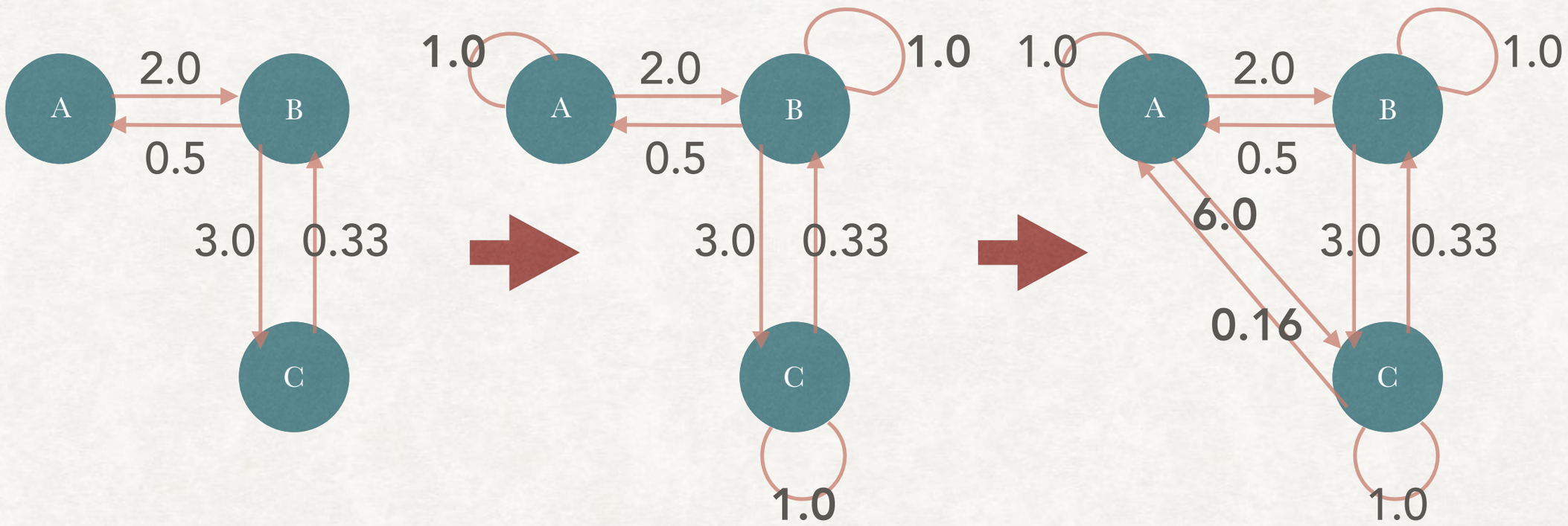
EVALUATE DIVISION

PROBLEM DESCRIPTION

- Equations are given in the format A / B = k, where A and B are variables represented as strings, and k is a real number (floating point number). Given some queries, return the answers. If the answer does not exist, return -1.0.
  - Given a / b = 2.0, b / c = 3.0.
  - queries are: a / c = ?, b / a = ?, a / e = ?, a / a = ?, x / x = ? .
  - return [6.0, 0.5, -1.0, 1.0, -1.0].

IDEA

- Floyd–Warshall algorithm
  - Find the shortest path in a directional graph
- Given a / b = 2.0, b / c = 3.0



SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/399_Evaluate_Division

NO. 417

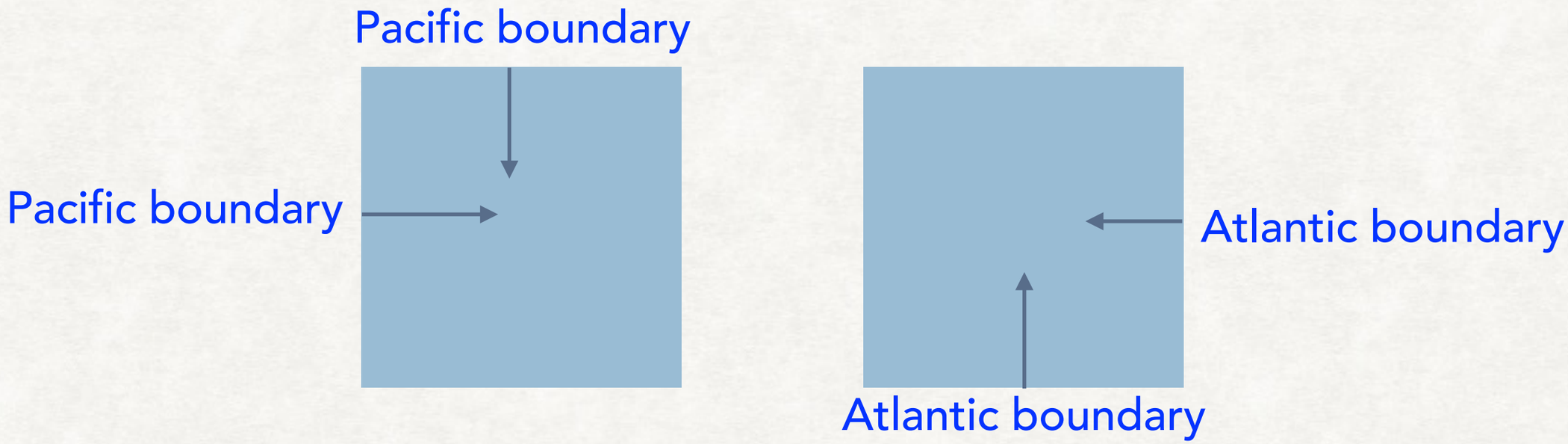
PACIFIC ATLANTIC WATER FLOW

PROBLEM DESCRIPTION

- Given an m x n matrix of non-negative integers representing the height of each unit cell in a continent, the "Pacific ocean" touches the left and top edges of the matrix and the "Atlantic ocean" touches the right and bottom edges.
- Water can only flow in four directions (up, down, left, or right) from a cell to another one with height equal or lower.
- Find the list of grid coordinates where water can flow to both the Pacific and Atlantic ocean

IDEA

- DFS from Pacific boundary and Atlantic boundary respectively



Answer is the intersection region that Pacific and Atlantic can reach by the following condition: height[y][x] >= height[prev_y][prev_x]

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/417_Pacific_Atlantic_Water_Flow

NO. 464

CAN I WIN

PROBLEM DESCRIPTION

- In the "100 game," two players take turns adding, to a running total, any integer from 1..10. The player who first causes the running total to reach or exceed 100 wins. What if we change the game so that players cannot re-use integers?

IDEA

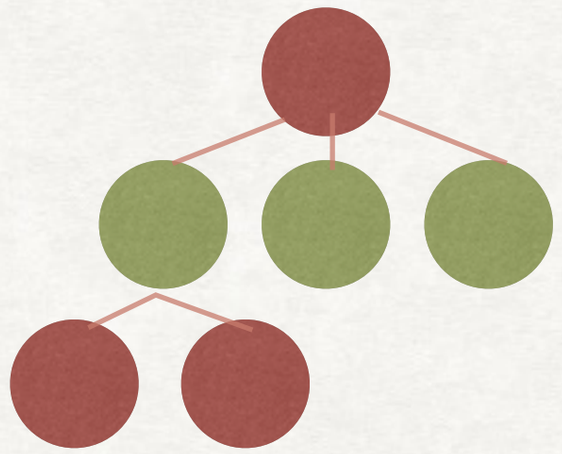
- Review Nim Game
  - https://github.com/Brady31027/leetcode/tree/master/292_Nim_Game
- We can not pick numbers being picked before…
  - The unchosen numbers
  - The remaining desiredTotal to reach

IDEA

- If player1 wants to win, player1 needs to put player2 into situations that player2 may lose

```
for i, choose in enumerate(choosable_mask):
    if not win(choosable_mask[:i] + choosable_mask[i+1:],
               desired_total, current_sum + choose,
               memo):
memo[mask_key] = True
break
```

player1 select choosable_mask[i]



recursively evaluate the results that player2 will get if player1 select choosable_mask[i]

use memo to remember the results for sub-problem

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/464_Can_I_Win