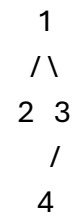
MSML606 HW3: Binary Trees, Binary Search Trees

[Programming Part]

1. **Construct a binary tree**
   - Write a function/method to construct a **binary tree** given a level-by-level representation (i.e., BFS output)
   - Assume that there are no duplicate values in the tree.
   - Example:
     - Input: "1,2,3,None,None,4,None" (assume the input contains only numbers or 'None')
     - Output: Binary tree (The root node of the reconstructed tree)



2. **In-Order traversal**
   - Write two functions/methods to do in-order traversal of the tree.
     - InOrderRecursive: A recursive implementation
     - InOrderIterative: An iterative implementation

3. **Time and space complexity analysis**
   - Write the following utility functions to generate test cases:
     (a) Write a function to generate a random permutation of N unique numbers. You may use Python's random package for this.
     (b) Write a function to generate a complete binary tree given N unique numbers
     (c) Write a function to generate a skewed binary tree (e.g., all nodes have only left children or only right children) given N unique numbers

   - Using these functions, compare the time and space complexity of 'InOrderRecursive' and 'InOrederIterative' for both complete binary tree and skewed tree.
     - Vary the input size N (e.g., N=10, 100, 1000, ...). Adjust the range until you observe meaningful differences between the methods.
     - Measure the exact running time using timers (e.g., time.perf_counter())
     - Measure memory usage using tools like 'tracemalloc'

    - For each input size, run both traversal methods multiple times (e.g., 100 iterations) to get reliable measurements
    - Handle exceptions during the computation (e.g., 'RecursionError') and document when they occur

4. **Validate Binary Search Tree (BST)**
    - Write a function to check if a given binary tree satisfies BST properties
    - Input:
        - The function will take the root node of the binary tree as input
        - This root node should be generated from a level-by-level representation (i.e., BFS output) of a binary tree. You may use the function you implemented for Problem 1 to construct this tree.
    - Output: return True if the tree satisfies BST properties; otherwise return False.

[Report] Your report should include following items

1. Briefly discuss your approach for each problem above, including any assumptions made.
2. Analyze the time complexity of your methods for each problem.
3. For problem 3, include a plot or table comparing running times and memory usage for recursive and iterative methods on both complete binary trees and skewed trees. Discuss any observations or trends in running times and memory usage based on your measurements. Provide an asymptotic analysis to explain your results.