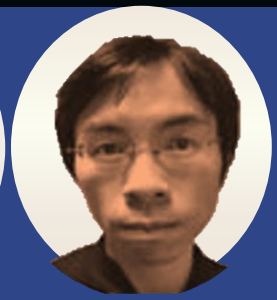
Dauphine | PSL
UNIVERSITÉ PARIS

... → COMPUTER SCIENCE, DECISION-MAKING, AND DATA → FURTHER EDUCATION

Algorithmic and advanced Programming in Python

Eric Benhamou eric.benhamou@dauphine.eu
Chien-Chung.Huang chien-chung.huang@ens.fr
Sofía Vázquez sofia.Vazquez@dauphine.psl.eu


Lab 8

Problem 1: maximum element in a binary tree

- Give an algorithm for finding maximum element in binary tree. Give Time Complexity and Space Complexity. Give at least two solutions like what we saw in master class for order traversal

Problem 1: maximum element in a binary tree

- Give an algorithm for finding maximum element in binary tree.

- **Solution 1**: One simple way of solving this problem is: find the maximum element in left subtree, find the maximum element in right sub tree, compare them with root data and select the one which is giving the maximum value. This approach can be easily implemented with recursion.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 1: maximum element in a binary tree

- Give an algorithm for finding maximum element in binary tree.

- **Solution 2**: do it without recursion. Using level order traversal: just observe the element's data while deleting.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 2: Searching an element in binary tree

- Give an algorithm for searching an element in binary tree. Give Time Complexity and Space Complexity. Give at least two solutions like what we saw in master class for order traversal

Problem 2: Searching an element in binary tree

- Give an algorithm for searching an element in binary tree.

- **Solution 1**: Given a binary tree, return true if a node with data is found in the tree. Recurse down the tree, choose the left or right branch by comparing data with each node's data.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 2: Searching an element in binary tree

- Give an algorithm for searching an element in binary tree.

- **Solution 2**: do it without recursion. We can use level order traversal for solving this problem. The only change required in level order traversal is, instead of printing the data, we just need to check whether the root data is equal to the element we want to search.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 3: Inserting an element in binary tree

- Give an algorithm for inserting an element into binary tree. Give Time Complexity and Space Complexity.

Problem 3: Inserting an element in binary tree

- Give an algorithm for inserting an element into binary tree.

- **Solution**: Since the given tree is a binary tree, we can insert the element wherever we want. To insert an element, we can use the level order traversal and insert the element wherever we find the node whose left or right child is nil.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 4: Size of a binary tree

- Give an algorithm for finding the size of binary tree. Give Time Complexity, Space Complexity and two solutions

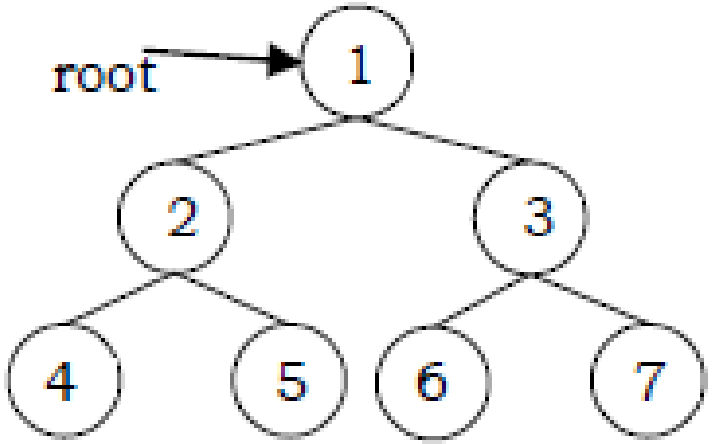
Problem 4: Size of a binary tree

- Give an algorithm for finding the size of binary tree.

- **Solution 1**: Calculate the size of left and right subtrees recursively, add 1 (current node) and return to its parent.
- Time Complexity: O(n). Space Complexity: O(n).
- **Solution 2**: Solve without recursion using level order traversal.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 5: reverse order

- Give an algorithm for printing the level order data in reverse order. For example, the output for the below tree should be: [[4 5 6 7] [2 3] [1]]



Problem 6: delete a tree

- Give an algorithm for deleting the tree.

Problem 6: delete a tree

- Give an algorithm for deleting the tree.

- **Solution:** To delete a tree, we must traverse all the nodes of the tree and delete them one by one. So which traversal should we use: InOrder, PreOrder, PostOrder or Level order Traversal?
- Before deleting the parent node, we should delete its children nodes first. We can use post order traversal as it does the work without storing anything. We can delete tree with other traversals also with extra space complexity. For the following, tree nodes are deleted in order – 4, 5, 2, 3, 1.

Problem 7: depth of a binary tree

- Give an algorithm for finding the height (or depth) of the binary tree.

Problem 7: depth of a binary tree

- Give an algorithm for finding the height (or depth) of the binary tree.

- **Solution 1**: Recursively calculate height of left and right subtrees of a node and assign height to the node as max of the heights of two children plus 1. This is similar to $PreOrder$ tree traversal (and DFS of Graph algorithms).
- Time Complexity: O(n). Space Complexity: O(n).
- **Solution 2**: using level order traversal. This is similar to BFS of Graph algorithms. End of level is identified with nil.
- Time Complexity: O(n). Space Complexity: O(n).

Problem 8 (bonus): combination of words

- **This is a typical interview question when applying to Google or Facebook**
- Given a set of positive numbers, find all possible combinations of words formed by replacing the continuous digits with the English alphabet's corresponding character, i.e., subset {1} can be replaced by A , {2} can be replaced by B, {1, 2} can be replaced by L {2, 1} can be replaced by U, etc.

- **Input:** digits[] = { 1, 2, 2, 1 }
  Output: ABBA, ABU, AVA, LBA, LU

  {1, 2, 2, 1} = ABBA
  {1, 2, 21} = ABU
  {1, 22, 1} = AVA
  {12, 2, 1} = LBA
  {12, 21} = LU

Problem 9 (bonus): binary trees with same inorder traversal

- **This is a typical interview question when applying to Google or Facebook**
- Given an inorder sequence of a binary tree, find all possible binary trees having that same inorder traversal
- For example, there are 5 binary trees with inorder traversal [1, 2, 3] as shown below