PDS Tutorial Questions

Tutorial/Prep 8

Sep. 18, 2024

Information

- This document comprises tutorial questions for CS2700, which includes both conceptual/theory questions (relevant for CS2700) and programming questions (relevant for both CS2700 and CS2710; these questions can also be thought of as preparatory/practice programming questions for CS2710 Lab 8, and so also referred to as Prep 8).

Conceptual questions (for CS2700)

1. Prove that the maximum number of nodes in a binary tree is $2^{h+1} - 1$, where h is the height of (maximum depth of any node in) the tree. What is the maximum number of nodes in a k-ary tree?
2. Compare and contrast the preorder, inorder, and postorder traversals of a tree. How can each traversal be applied in practical applications?
3. Write pseudocode for a non-recursive inorder traversal of a binary tree using a stack. Discuss the advantages of using a non-recursive approach over a recursive approach.
4. How many distinct binary trees are possible for a binary tree whose pre-order sequence is 1, 2, 3?
5. Can you uniquely reconstruct a binary tree from its inorder and preorder traversals alone? For instance, is there an unique tree whose inorder traversal is 3,2,1 and preorder traversal is 1,2,3? If so, what is this unique tree?

Programming questions (for CS2710 Lab 8 preparation/practice, and for CS2700)

6. [QUEUE IMPLEMENTATION] Review the circular array/buffer/vector based implementation of queue seen in class. Is it harder or easier to implement the queue using a linked list data structure instead of a circular array? What are the pros/cons of doing so?
7. [DEMYSTIFY RECURSION & TREE TRAVERSAL] Write an algorithm to perform inorder, preorder, and postorder traversal of a binary tree. Employ recursion and draw a recursion call tree for simple examples to fully understand how the recursion works.
8. [LEVEL ORDER TRAVERSAL] Write an algorithm to perform level-order traversal of a binary tree (Breadth-First Traversal). Understand how a queue plays a key role in this traversal.
9. [BASIC FLOOD FILL] Given a 2D grid of '1's (filled cells) and '0's (empty cells), implement an algorithm to mark all connected '1's from a given starting position as '2'. Connection between two "1" cells can be via direct uni-hop (immediate up, down, left or right cell of a given grid cell) or indirect multi-hop (via intervening "1" cells) connection.