Problem Solving Workshop #5 Tech Interviews and Competitive Programming Meetup

April 24, 2016 https://www.meetup.com/tech-interviews-and-competitive-programming/

Instructor: Eugene Yarovoi (can be contacted through the group Meetup page above under Organizers)

More practice questions: leetcode.com, glassdoor.com, geeksforgeeks.org
Books: Elements of Programming Interviews, Cracking the Coding Interview
Have questions you want answered? Contact the instructor, or ask on Quora. You can post questions and follow the instructor and other people who write about algorithms.

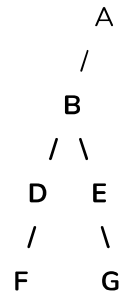
Try to find optimized solutions, and provide a time and space complexity analysis with every solution for the algorithms questions.

Solutions

Easy Problem

Given a binary tree, find the longest path in the tree (treating the edges as being undirected).

Example Input / Output:



The nodes in bold form the longest path: F -> D -> B -> E -> G

Solution: preprocess the tree and store at each node N the longest path to a leaf node from N (store length of path + pointer to child contributing to longest path, that way the path can always be traced). This can be done for all nodes by a simple recursive traversal. Then, traverse the tree again and for each node, consider forming a path combining the longest path to a leaf node from the left child + the longest path to a leaf node from the right child + the node itself. The longest path formed in this way will be the answer.

Edit: maybe this problem wasn't easy, maybe more like easy-medium or medium.

Medium-Hard Problem #1

You're given a string S, and a list L of pairs where each pair (i, j) is a rule that indicates that you may at any time swap the character at position i in S with the character at position j in S. Generate the lexicographically largest (last in dictionary order) string that you can by swapping characters. The same (i, j) pair may be reused any number of times to perform as many swaps as needed.

Solution: Build a graph where every position (every valid index in the string) is a vertex, and edges denote a rule that allows swapping between the two positions. We can show that if two indices are in the same connected component, they can eventually be swapped. As such, we should detect all the connected components, and sort the letters in descending order within each conected component.

Medium-Hard Problem #2

There is an MxN grid (matrix) where each cell contains a value (may be positive, zero, or negative). When you visit a cell, you collect the value in that cell. You start at (0,0), the upper left corner. The coordinate system is image coordinates: x increases going right, and y increases going down.

If you are located at position (a, b), you may jump next to any position (a+dx, b+dy) where dx and dy are integers greater than or equal to 1, and (a+dx, b+dy) is in-bounds of the grid. In other words, every jump is required to move you to the right and downwards by at least (but perhaps more than) one row and one column.

You can jump as many times as you want, and there is no specific cell where you have to end up. Determine the maximum sum of values you can obtain.

Solution: We can write the following recurrence for F(x, y), the maximum score achievable starting from x, y, that can then be solved by dynamic programming:

F(x, y) = max (0, F(x+1, y), F(x, y+1), F(x+1, y+1) + matrix[x][y])

In other words, your choices are to stop (take 0), take the value in the matrix and move to the next row and next column, or don't take the value and advance only one of these two positions. There would need to be a base case added and a correction for when the arguments are out of bounds.

Design Problem

Imagine you are Twitter. How would you design the "top 100 trending hashtags" feature, if it were not already designed? There is deliberate ambiguity in this problem -- you will need to make reasonable

assumptions. When coming up with your system design, you will need to modify / add to the design of one or more existing Twitter systems, so you will need to make reasonable assumptions for what the existing architecture looks like.

Solution: There are many possible directions in which you can take this problem. Here are some ideas we discussed:

- What is the meaning of “trending”? What are reasonable definitions? Ideas: count in last 12 hours, exponential weighting: a hashtag is counted as having occurred r^t times, where $r < 1$ and t is a time variable, so recent events count more and past events get multiplied by a progressively smaller factor to give them less significance.
- Full out solution where we collect all the data and then do analysis using map-reduce or other scalable technology vs. sampling, where we only run the analysis on a small random subset and get an approximate answer. Top-trending hashtags should appear even in heavily sampled data a lot, so sampling sounds like it could be a great idea.
- Do you run the analysis against the distributed databases or do you modify the front-ends or other services to forward the data for this analysis to a new cluster specifically set up for this task? Forwarding may make sense if you’re doing sampling, since many databases don’t have the ability to randomly sample without reading the data.