Problem Solving Workshop #17 Tech Interviews and Competitive Programming Meetup

November 5, 2016 https://www.meetup.com/tech-interviews-and-competitive-programming/

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

Graph Problems Problem Solving Workshop

More practice questions: leetcode.com, glassdoor.com, careercup.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.

Problem #1, "Distributed Debugging"

You have a distributed system. When a sequence of events happens, different parts of the distributed system may see -- and log -- different subsets of those events. The events always appear in the same order in each component's log as the order in which they actually occurred, but each log may only have a subset of the events. Given the sequence recorded in each log, can you recover the original sequence (or say that there isn't enough information to do so)?

Example Input:
frontend server log: connected to server, successful login, logged out
authentication module log: failed login, successful login, logged out
security audit log: connected to server, failed login, initiated bank transfer
feature usage log: initiated bank transfer, logged out
Output: connected to server, failed login, successful login, initiated bank transfer, logged out
Explanation: the output shown is the only sequence of events consistent with the components seeing the events in the order shown in the input, given the problem's condition that each component always sees events in the order that they occur.

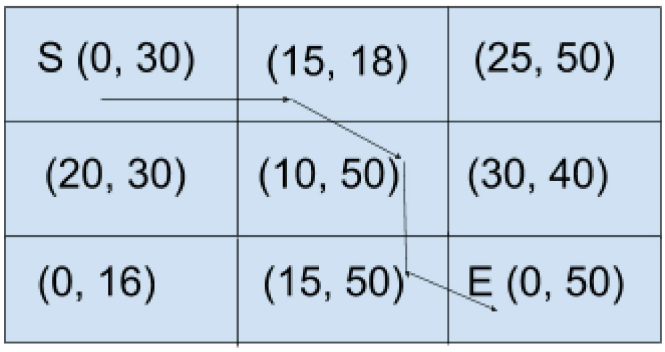
Problem #2, "Kayaking"

You are in a kayak inside a sea cave. The cave can be seen as rectangular grid where at any given time, you can move in the four cardinal directions, or stay still. The water is rising, and you need to escape the cave as quickly as possible. However, every position in the grid has a global time (possibly different for each position) after which you cannot enter or remain there, because it gets completely flooded and there's no room for air or your kayak. Some locations may also be too shallow to kayak into initially, and these can only be moved into ***after*** a certain global time. This means that sometimes, you might stay still

in your current location while you wait for the water to rise at a neighboring location. You start at time 0 and are given a parameter T, the amount of time it takes to move from one position to a valid adjacent position (if you start moving at time X, you remain at your current position until time X+T-1, and at time X+T you are in your new position). Given the allowed times for each cell of the grid, your starting position, and the position of the cave's exit, determine the smallest amount of time in which you can escape (or say it is impossible).

The input consists of a two-dimensional rectangular array, where each position in the array contains a 2-tuple of integers (earliest allowed time, latest allowed time). You start at time 0. Your output should be an integer representing the total time your escape will take.

Example Input: T = 3 and the following grid, where the starting position is labeled S and the exit is labeled E. The arrows indicate the path that would be taken to escape in the solution.



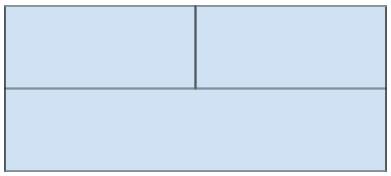
Output: 24
Explanation: We start in the upper left corner at time 0. We could wait until time 20 to move down, but instead it will be better to move right at time 15 (we actually start moving at time 12, so we will arrive at the upper middle cell at time 12 + T = 15, the earliest allowable time). Then, we cannot move right again because the move won't be available before the time limit of 18. So, we move down and get to the center cell at time 18. Then, we could wait for the move right to become ready at time 30, but it will be faster to move down (getting there at time 21), and then immediately move right, getting to the exit at time 24. Note that we couldn't move left because at time 21, the lower left cell was already flooded.

Follow-up variations:

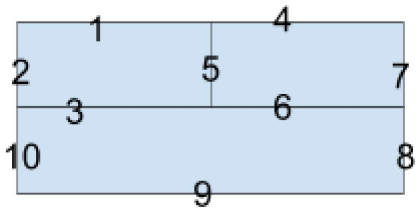
1. You can choose to start anywhere on the left wall, and need to get to any cell of your choice on the right wall.
2. Suppose instead of being given T, you are asked "Compute the value of T needed to be able to escape." In other words, how fast do you need to row?

Problem #3, "Bar Trick"

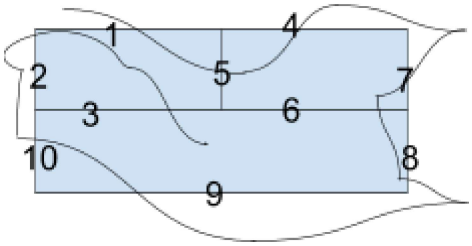
A stranger at a bar invites you to play the following game. They draw a diagram like this on their napkin, which consists of axis-aligned rectangles filling the space of a larger rectangular space:



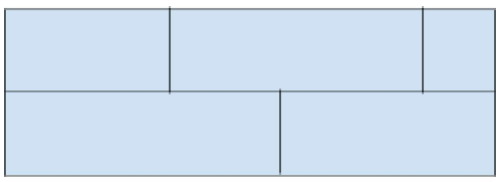
The requirement is to draw a connected curve (possibly self-intersecting, does not have to be a loop either) that crosses each line segment **exactly** once. The line segments are counted like this (the labels are just IDs, the ordering has no significance).



In other words, each line gets broken up into independent line segments by perpendicular lines. Here is the solution to the above:



Now, you are asked to solve this one. Either you have to show a solution, or claim it's impossible. If you find a solution, the stranger buys you a drink. If you claim it's impossible and you're correct, same thing. But if you claim it's impossible and the stranger shows a solution, you buy ***them*** a drink:



What do you do?

Problem #4, "Circus"

There's a connected, directed network of cities and roads connecting them. Some cities are safer than others, and each city has a unique number indicating its safety (higher is safer). A traveling circus leaves from a given city A, and each step of the journey, goes to the neighboring city that is safest (out of those to which there's a directed road from the circus's current city). As the circus doesn't consider whether it's been to a particular city before, it may be that by following this rule, it ends up repeating a loop that doesn't cover all the cities, or it ends up in a city from where there are no outbound roads (where it then stops).

A king wants the circus to come to the capital, but doesn't want to tell them to do so because it might not seem dignified for a king. So, instead, he plans to close off some roads (effectively deleting them from the road network) so that the circus will come to the capitol (another given city) naturally by following their safety rule. Since the king doesn't want to disrupt commerce, he wants to close as few roads as possible. Given a graph of cities and roads, where each city has a safety rating, as well the circus's starting city and the capitol, determine the smallest set of roads that, if closed, would result in the circus visiting the capital.