Problem Solving Workshop #28 Tech Interviews and Competitive Programming Meetup

September 2, 2017 https://www.meetup.com/tech-interviews-and-competitive-programming/

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

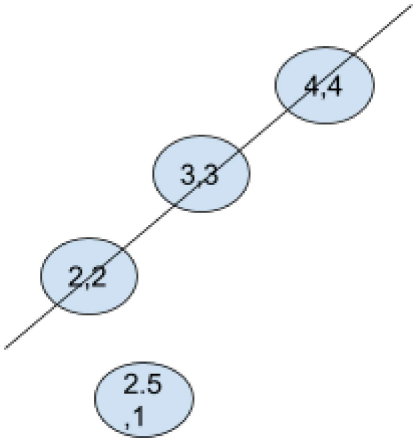
More practice questions: leetcode.com, glassdoor.com, geeksforgeeks.org
Books: Elements of Programming Interviews, Cracking the Coding Interview
Have questions you want answered? Contact the instructor, or ask on Quora. You can post questions and follow the instructor and other people who write about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

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

Problem #1, "Max Points on a Line" (Easy / Medium)

Given N points in 2D space (assume integer coordinates), find the largest set of points that are collinear (lie on the same line).

Example: [(2,2), (3, 3), (4, 4), (2.5, 1)]
Output: [(2,2), (3, 3), (4, 4)]
Explanation:

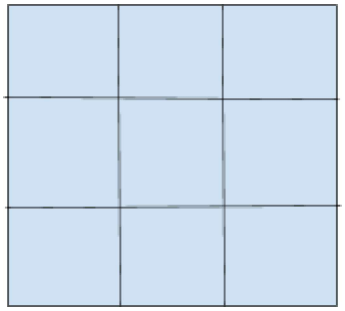


It is not possible to select a larger set of points so that they're all on the same line.

Complexity Guidance: **[Very Easy]** $O(N^3)$ time **[Easy]** $O(N^2)$ time / $O(N^2)$ space, **[Medium]** $O(N^2)$ time / $O(N)$ space.

Problem #2, "Square Counting" (Medium)

You're given an NxN grid, that starts out looking like this. For example;



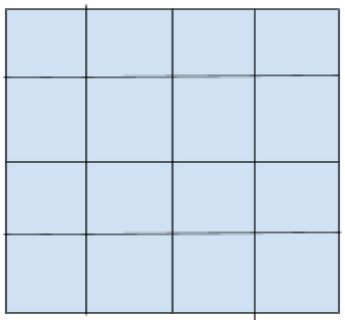
Your goal is to count how many total **squares** there are in the diagram. We count something as a square if it is an arrangement of 4 lines in the way you would normally expect square to be defined, without regard as to whether there are any extra lines or what is inside the square. A line segment may be part of multiple different squares. For example, the above diagram has 1 3x3 square, 4 2x2 squares, and 9 1x1 squares, for 14 total squares.

(a) **[Easy]** Provide an algorithm for counting the number of squares in an NxN grid.

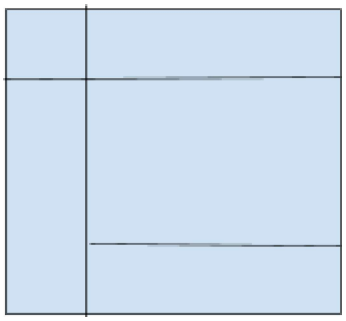
Example Input: 3 (meaning 3x3 grid)
Output: 14 (as explained above)

Example Input: 4
Output: 30

(b). Now, some grid lines may be erased. For example, a 4x4 grid starts out like this:



And then may have some lines erased to look like this:



Line segments can only be erased between intersection points of lines in the complete grid. In other words, the above drawing can be specified as a (N+1)xN matrix of booleans denoting which horizontal line segments (of length 1) to keep and a Nx(N+1) matrix denoting which vertical line segments to keep . The above drawing would be represented as:

Horizontal lines:
[T T T T
T T T T
F F F F
F T T T
T T T T]

Vertical lines:
[T T F F T
T T F F T
T T F F T
T T F F T]

(b) Count the number of **squares** in such an arrangment of lines.

The above example has 1 1x1 square (top left), 2 3x3 squares (overlapping, on the right side), and 1 4x4 square (the overall outline). So the answer would be 4.

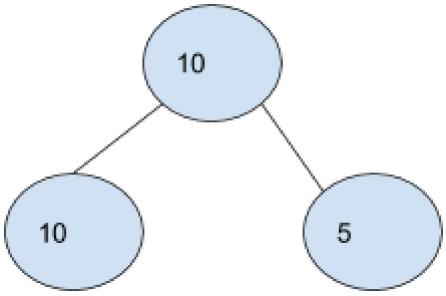
Complexity Guidance: **[Easy]** $O(N^4)$ naive solution, **[Medium]** $O(N^3)$ time, **[Hard, don't get too stuck on it]** $O(N^2 \log N)$ time.

Problem #3, "Graph Sums" (Hard)

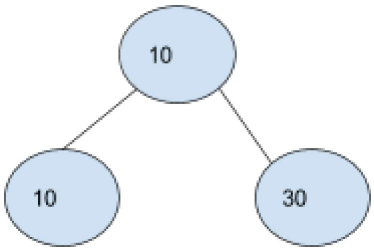
Consider a graph where every vertex has a value associated with it. We want to create a data structure on this graph that will support the following operations:

- Update(N, V): changes the value of node N to V.
- SumNeighbors(N): returns the sum of the values of all of N's neighbors.

Example:

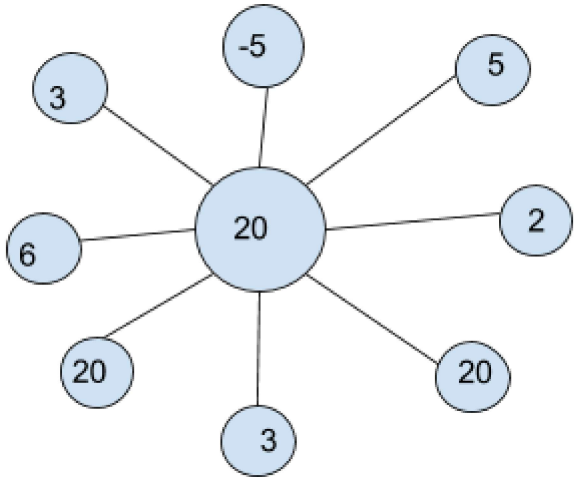


SumNeighbors(top node) -> 15 (Explanation: 5+10)
SumNeighbors(right node) -> 10 (Explanation: only connected to the top node)
Update(right node, 30)



SumNeighbors(top node) -> 40
SumNeighbors(right node) -> 10 (top node didn't change)

The obvious approach is to just store each node's value in the node itself so that Update can run in O(1), but then SumNeighbors runs in O(neighbor count of node N) time. The problem with this is, for example, with graphs that are structured like this:



Here the center node could take a long time to sum up (imagine an extension of this concept where the center node has a million neighbors).

Can you design an approach that better balances the running time of Update against the running time of SumNeighbors?