Problem Solving Workshop #23 Tech Interviews and Competitive Programming Meetup

May 6, 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, "All Doubles"

You're given a string S containing N lowercase English letters (it is important that the letters are limited to this range). Your goal is to count the number of contiguous substrings in S such that every distinct letter that appears in the substring appears an even number of times. Don't count the empty substring.

Example Input: S = "bddccb"
Output: 4
Explanation: "bddccb", "dd", "cc", "ddcc" are the 4 substrings in which every letter present in the substring occurs an even number of times. The substring "dc", for example, doesn't satisfy the condition.

The goal isn't to count distinct substrings. For example, for S = "aaaa" the answer should be 4, because each of the three "aa" substrings count, and then "aaaa" also counts.

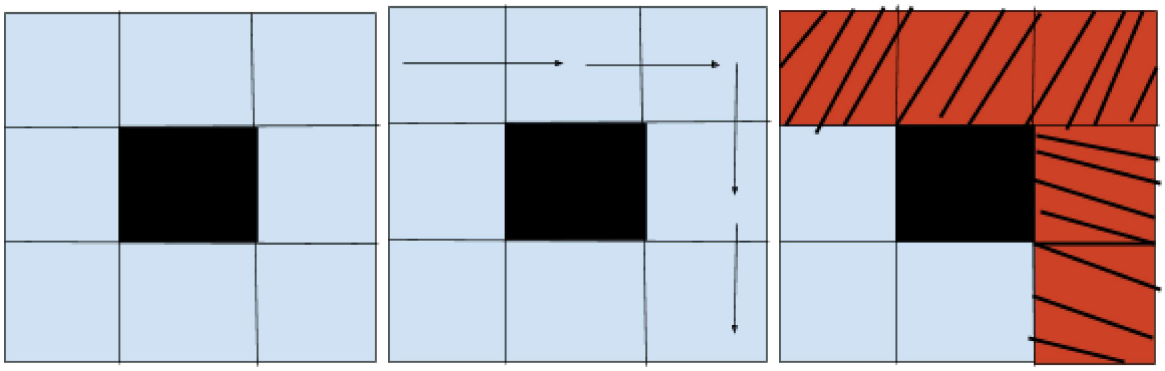
(a) **[Very Easy]** Give any correct solution. A naive solution may reasonably be $O(N^4)$ or $O(N^3)$.
(b) **[Easy-Medium]** Solve the problem in $O(N^2)$ time.
(c) **[Medium-Hard]** Solve the problem with a very good time complexity such as O(N log N) or O(N).
(d) **[Hard]** Consider parallelizing this solution now to share the work among K cores. The running time of your parallelized solution should be something like O(N log N / K) assuming all the cores can work in parallel. You can also assume that K is relatively small compared to N, so additive terms like "+ K" can be dropped from big-O analysis.
(e) **[Hard, recommend skipping until completing everything else]** What if you have unlimited cores, constrained only by the fact that it takes O(1) time for any thread to create and launch a new thread. How efficiently can you solve the problem then?

Problem #2, "Etchy-Sketch Patterns"

You are given an N x N grid, represented as a grid of booleans where a `True` value represents a wall at that location in the grid, and a `False` value represents an open space. You start at the upper-left corner of the grid (guaranteed to always be an open space), and you always have to end in the lower-right

corner (also guaranteed to be an open space). As you move, you can only pass through open spaces (not walls), and you paint every location you visit in the grid (like an etchy-sketch).

For example, the below diagram shows, if you have the grid in the first diagram, and you make the sequence of moves shown in the second diagram, you will paint the squares shaded in the third diagram:



Your goal is to determine in how many possible ways you can paint the grid. A way of painting is distinct from another if and only if the set of distinct squares you visit as you move from start to finish is different. Don't include in the count any runs where you don't get to the finish successfully.

(a) **[Easy-Medium]** Solve if you're only allowed to move right or down, by one cell at a time. An optimal solution would be $O(N^2)$.

(b) **[Medium]** You're allowed to move right, down, or up (but not left), one cell at a time, but you're not allowed to return to cells you've already been to (that would mean you're painting a location twice and that's sloppy!). An optimal solution would be $O(N^2)$.

(c) Same as part (c), but you're also allowed to revisit cells you've been to before. However, remember that a way of painting is distinct from another **only** if the set of **distinct** squares you visit as you move from start to finish is different. So, there are still finitely many ways to paint, since moving back and forth over and over again in the same column doesn't produce a new way of painting (the resulting picture is the same). **[Medium-Hard]** Solve this in $O(N^5)$ (or at least not exponential time). **[Hard]** Solve this in $O(N^3)$. It is possible even to solve it in the optimal $O(N^2)$.

Take-Home Problems (same as last time, since they haven't been discussed yet online)

1. **[Medium]** You're given an array of N integers, and an integer K >= 1. How many contiguous subarrays of the array sum up to a multiple of K?

2. **[Medium-Hard]** Given a string S, find the longest repeated substring. For example, if you have S = "abcdabc", **abc** is the longest repeating substring since it's the longest substring that occurs more than once. Consider both the version of the problem where we allow the repeating substrings to overlap and the case where we don't. Follow-up: how about longest repeating anagram (two substrings that are anagrams of each other)?