Problem Solving Workshop #21 Tech Interviews and Competitive Programming Meetup

March 4, 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, "Array Sampling" (Easy-Medium)

You're given K arrays of length N each. In how many distinct ways can you choose one number from each array such that:
number from array 0 < number from array 1 < number from array 2 < ... number from array K-1?

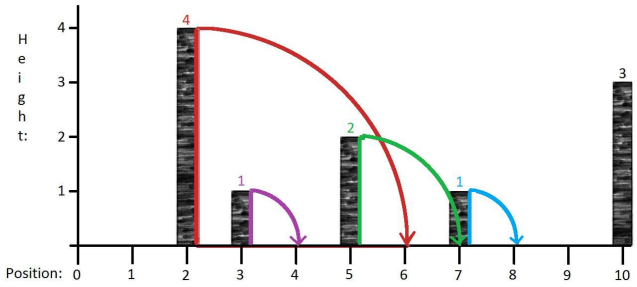
Example Input (K=2, N=3): [2, 3, 0], [4, 2, 1]
Output: 5
Explanation: we can choose (2, 4), (3, 4), (0, 4), (0, 2), (0, 1).

A "way" of choosing numbers is considered distinct from another "way" if it chooses different array indices. For example for the input [0, 0], [2, 2], the answer is 4, because there are 4 different ways to generate (0, 2) by taking different array indices.

Guidance: Think about the time complexity and try to find an efficient solution for two arrays first. Then try to extend it to K arrays.

Problem #2, "Dominoes" (Medium-Hard)

You have dominoes of different heights arranged in a line. When a domino is knocked over, it may knock over other dominoes, which may in turn knock over even more dominoes, and so on in a cascade. More precisely, the effect of knocking over to the right a domino at position *x* is that any and all dominoes with position greater than *x* and less than or equal to *x* + *dominoHeights[x]* are knocked over to the right as well. These dominoes may then knock over even more dominoes and so on until nothing else is in range to be knocked over.



In the above diagram, we have five dominoes of different heights. If we knock over to the right the red domino at position 2 (on the x-axis), which has height 4, it falls and knocks over all the dominoes up to and including position 2 + 4 = 6. The purple and the green dominoes are knocked over in the same direction in this process. The green domino then knocks over the blue domino. The rightmost domino is not affected by this cascade. We say that the cascade ends at position 8, since that is the farthest right that any domino participating in the cascade travels.

The goal of this problem is to compute the end position of the cascade for each possible starting domino. This is easy to accomplish in $O(n^2)$ by running an O(n) algorithm for each domino, but the goal is to come up with a more efficient algorithm.

The input for this problem will be given as an integer array, where the value at the i-th index will represent the height of the domino at position i. A value of 0 will denote the absence of a domino at that position. For the image above, the input is [0, 0, 4, 1, 0, 2, 0, 1, 0, 0, 3]. The output should be an array of the same size indicating the cascade end position for each domino, in this case [0, 1, 8, 4, 4, 8, 6, 8, 8, 9, 10, 13] (where there is no domino, the cascade ends at the position it begins).

Problem #3, "Zombie Apocalypse"

There has been a zombie apocalypse and to search for food you're splitting your (even-sized) group of N people into units of 2 people each. That way, each unit has a "leader" and a person to watch the leader's back.

People don't need any training to watch someone's back, but to be the leader they may need training, because the leader needs to be able to land headshots on zombies, use a chainsaw, etc. Understandably, some people would need more training than others to be able to be leaders. Some people were police officers prior to the apocalypse and some were...computer scientists. Let's say that for each person, you have a number that represents an estimate of how many hours it would take to train them to lead a unit.

(i) Given the training time for each person, split them into "leaders" and "watchers" so as to minimize the total training time required. **[Easy]** Any solution, **[Medium]** O(N) time.
(ii) Suppose now there's an additional constraint. In any given unit, the leader must be the person who's been in the group longer. In other words, even if someone wouldn't require a lot of training, they can't

lead someone who's been in the group for a longer time, to keep the circle of trust close. That doesn't mean that there can't be some relatively junior people in leader roles and some relatively senior watchers -- it just means that ***within*** any given unit, the more senior person must be the leader. You're now given a seniority for each person in addition to their training cost. Pair people up again, minimizing the total training time and respecting the constraints. **[Medium]** $O(n^2)$ time, **[Hard]** O(n log n) time.

Take-Home Problems

1. **[Medium-Hard]** Given a number M, generate the smallest positive decimal number consisting solely of 0s and 1s that is divisible by M.
For example, if M = 3, the answer is 111 because 111 is divisible by 3 and is the smallest positive number consisting of only 0s and 1s in its decimal representation to be divisible by 3. (0 is not a positive number.)
Target Complexity: $O(M^2)$ time.

2. **[Medium-Hard]** You have a gigantic set of records (think rows in a DB table), and you've transmitted this dataset to someone else. They've made some changes, but haven't kept track of what changes they made. If they've only made a small number of changes, how would you sync your set of records to theirs, without retransmitting the entire file?
If they've modified K records and the full dataset contains N records, analyze your solution in terms of how many records must be transmitted. O(N) records would be retrasmitting the file, so you're looking for something sublinear in N. K might appear in your analysis becuase you might have to transmit more data if they changed more data.

3. **[Medium]** Given a string S and a subsequence w, in how many distinct ways can you find w as a subsequence of S? A string w is a subsquence of S if some characters can be deleted from S to form w, without reshuffling the remaining characters.. Two ways are distinct if they keep different indices to form w.
Example: S = "abcab", w= "ab", output = 3. You can take the first a and then either of the two b's, or the second a and the second b.