Problem Solving Workshop #27 Tech Interviews and Competitive Programming Meetup
August 19, 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, "Following Elements"

You're given an array A having N integers in the range[0...N-1].

Consider some index K in the range [0...N-1], and the set {A[K], A[A[K]], A[A[A[k]]], ... and so on}. Since all the values are in the range [0...N-1], the array accesses are always in-bounds, and eventually no new elements are added to the set (since they repeat). Essentially, this is the sequence formed by using K as an index into the array, then treating the result of that as another index, and so on.

Find the choice of K that generates the largest set (the one with the most distinct elements). If there are several equal choices, output any of them.

Example Input: A = [5, 4, 0, 3, 1, 6, 2]
Output: 2
Explanation: If we set K=2, then we generate the elements A[2] = 0, A[0] = 5, A[5] = 6, A[6] = 2. The elements after that will clearly just repeat. So, this set has 4 distinct elements. It also happens to be the largest you can generate with any choice of K (you can check that other choices result in smaller or equal sets). Here, it would have also been fine to output 0, 5, or 6, since these result in the same set of size 4.

Problem #2, "Multiplying Ranges"

You're given an array A of N floating-point numbers. These may be positive, negative, or zero. Don't worry about issues of floating-point precision for this problem.

You have to design a data structure that can answer queries of the form "What is A[i] * A[i+1] * ... * A[j-1] * A[j]" for any (i,j) that are supplied as parameters to the query (j >= i).

You're allowed to preprocess the array A before you start receiving and answering queries. Your preprocessing time should be O(N) or O(N log N) (since obviously the problem is trivial if you preprocess every possible query in O(N^2)).

Example: A = [1, 0, 1, 1.5, 0.3]
Query 1: i=2, j=4
Query 1 expected output: 0.45
Query 1 Explanation: 1 * 1.5 * 0.3 = 0.45
Query 2: i=0, j=3
Query 2 epexcted output: 0
Query 2 Explanation: 1 * 0 * 1 * 1.5 = 0

Guidance: if you're struggling with the problem, try solving the **sum**-query version of this first. In other words, what if you had to do queries of the form "What is A[i] **+** A[i+1] + ... + A[j-1] + A[j]"?

Problem #3, "Whistling Dwarves"

(Source: SPOJ, online judge available, see below)

Snow White and the N dwarfs live in the forest. Each morning the dwarfs form a long line and go whistling away to the mine. Snow White runs around them and snaps pictures to upload onto her favorite social network. Each picture will contain some contiguous subarray of the dwarves. That is, if the dwarves are in a line and can be assigned numbers 1, 2, ..., N, each picture will contain dwarves A, A+1, ..., B-1, B for some A, B.

Each dwarf has a colored cap, and there are C different colors. Snow White considers a picture to be pretty if strictly more than half of the caps on it are of the same color. That is, if there are K dwarves in the picture, a picture will be pretty if strictly more than K/2 caps are of the same color (that is, there exists a majority color).

Given the initial sequence of dwarves in terms of their cap colors, and a set of P pictures specified in terms of their A, B values (first and last dwarves in the picture), determine for each picture whether it is pretty, and what color is the majority color in each pretty picture (no need to determine majority color for non-pretty pictures).

Example Input: dwarfCaps = [red, blue, red, blue, red, blue, yellow, blue, yellow, yellow]
pictures = [(start=0, end=1), (start=0, end=2), (start=6, end=9)]
Output: [none, red, yellow]
Explanation: For the range [0, 1], there is no majority color (1/2 is not a majority). For the range [0, 2], red is the majority color (2/3 dwarves have a red cap). For the range [6, 9], yellow has 3/4 dwarves.

Guidance: If you're having trouble, first consider the case where N and P may be large, but C is small (say C = 3 for example). This is a lot easier. Then try to extend to cases where C may be large as well.

Online judge for problem. Don’t follow this link until you’ve solved the problem and want to code it, as there may be spoilers: http://www.spoj.com/problems/PATULJCI/