Princeton Competitive Programming Fall 22

Solutions for Week 4 problems

Please contribute solutions to problems you have solved. Avoid writing code here, a short paragraph or two describing the solution suffices. Try to include an analysis of your solution, preferably with big O notation. Feel free to add multiple solutions if you think your solution is sufficiently different from the ones already written up.

Div II A - Odd/Even Increments

Incrementing by one is the same as changing the parity (odd or even). Note that, since we can only increment all the odd indices simultaneously, if there are two terms at odd indices of different parity, we will never be able to make them both even. The same is true with even indices. However, if all the terms at odd indices are the same parity and all the terms at even indices are the same parity, we can always use the two operations to make the entire array even (i.e., if the even indices all contain odd terms, we can use the second operation to make all the terms even). Thus, this problem simplifies to checking whether all the integers at odd indices have the same parity and whether all the integers at even indices have the same parity, which can be checked with one pass of the array, or O(n) runtime.

-Solution by Devan Shah

Div II B - Bottle Arrangements

Consider the arrangement where we place all of our red bottles consecutively at the start, and then place all the white bottles afterwards (So for the first test case, the sequence would be RRRWW). Say the most red bottles any critic requests is *r* and the most white bottles any critic requests is *w*. Then I claim the sequence of *r* red bottles followed by *w* white bottles is the smallest such sequence that will satisfy all the critics. As a proof outline, consider the critic who wants *a* reds and *b* whites. We know *a* <= *r* and *b*<= *w*, so the sequence starting at *r*-*w*+1 and ending at *r* + *b* (sequence is 1-indexed) will satisfy that critic. With any smaller sequence, either the critic who wants *r* reds or the one who wants *w* whites will be disappointed. Thus, if *r* + *w* <= n, satisfying the critics is possible with a sequence like RR…RRWW...WW, and if r+w>n, it is impossible.

This algorithm is O(n) since we only need to traverse the array once.

-Solution by Devan Shah

Div II C - Maximal AND

Note that, to maximize the final possible value, we want to find the greatest binary digits place that we can convert from 0 to 1. Let's call the final value *A*. *A* will only have a 1 at the *m*th binary digits place if all a_i have a 1 at the *m*th binary digits place. To use our *k* operations wisely, we need to know how "close" we are to having all a_i have a 1 at the *m*th digits place. Thus, we can create an array accordingly that counts, for each digits place, how many a_i 's have a 1 at that place. If we can change a greater digits place to 1, it will obviously increase *A* more than changing a lesser digits place. Thus we consider the array from greatest *m* to least *m*, and keep track of how many operations we have left to use. Whenever we see a digit where less than or equal to *k* modifications could change that digit to a 1 in *A,* we do so and subtract the amount of operations we used from *k*. Note, we are using these operations to change that digit's place to a 1 in all the a_i's where it is not 1. Following this procedure, we can construct the largest such *A*.

Note, the main processing of this algorithm involves constructing the array which is O(log_2(digits of *A*) * n) since we effectively have a counter for each digit and there are log(*A)* digits. Since each a_i 2^31, this logarithm <= 31 so we can consider this algorithm O(n) as the log component remains small.

- Solution by Devan Shah

Div I A / Div II D - Middle Class

Effectively, we wish to redistribute the wealth so that as many people as possible are above the minimum wealth cutoff. To do so, it makes sense that we want to get as many individuals as possible exactly on the wealth cutoff line. Ideally, we want to apply the reform to as many people as possible such that they are all on or above the minimum wealth cutoff. Sorting the array lets us see who is close to the cutoff, because those are the people who it is easiest to boost to the wealth cutoff. Sorting the array from greatest wealth to least, we then sum the wealth of each individual (going from greatest to least) as long as total summed wealth >= (amount of people we summed over) * minimum wealth cutoff. The inequality is equivalent to saying redistribution over that group will bring everyone over the minimum wealth cutoff, and we want to find the largest group where this is possible.

Thus, we can return the amount of people we summed over and that is the solution the problem.

In this algorithm, we sort the array and then iterate through it once. Thus, our algorithm runs in O(n log n) since sorting is O(n log n).

- Solution by Devan Shah

The reform operation allows us, effectively, to transfer as much money from person *i* to person *j* as desired as long as the wealth of person *i* remains greater than or equal to the wealth of person *j*. This equivalence may not be obvious,

Div I B / Div II E - Radio Prize

Please add a solution!

Div I C - The Twin Tower

Please add a solution!

Div I D - Interstellar Travel

Please add a solution!

Div I E - Il Derby della Madonnina

Please add a solution!