LCC Takeup

By Max, Peter, Kenneth, and ChrisT

LCC ‘20 Contest 4 J1 - Tracy’s CPS

Count the number of competitors with a CPS less than or equal to 1.69. Then the answer is 0.5 * M/N. Remember to use floating point division to get the right answer.

To print the answer, make sure to only use 2 decimal places. This can be done using printf or cout << setprecision(2) << fixed in C++

LCC ‘20 Contest 4 J2 - Column Counting

Make a length N array freq representing the number of occurrences of K in the i^{th} column. Loop through the values a_{ij} in the input and increment freq[j] if a_{ij} == K.

LCC ‘20 Contest 4 J3 - Button Mashing

Implementation problem.

One approach would be to first process all the ‘ef’ and ‘fe’ moves because they take priority over the other moves. Keep track of which indices to skip. Iterate over the characters a second time to fulfill the other rules.

LCC '20 Contest 4 J4 - Tea

Recall that R(P) = P*Q. Multiplying the linear equation by P yields:

$AQP + BP^2 = CP$

$R(P) = (CP - BP^2)/A$

Note that R(P) is a quadratic function (for $A \neq 0$ and $B \neq 0$). We can find the price of the vertex with x = -b/2a or C/2B. Remember that integer division truncates, so you should use floating point math.

LCC ‘20 Contest 4 J4 - Tea, cont.

Consider the case $C = 0$. Then it follows that $R(0) = 0$ and $R(P) \leq 0$. Therefore, any price is elastic.

Consider the case $A = 0$. Then the linear equation becomes $P = C/B$. Notice that because B/C is a constant, there is only 1 price. Because $R(P)$ is undefined for all $P \neq B/C$, any price is elastic.

Consider the case $B = 0$. Then the linear equation becomes $Q = C/A$. Because MCPT can choose any price, there always exists an $R(P_2) > R(P_1)$ such that $P_1 > P_2$. Therefore, any price is inelastic.

LCC '20 Contest 4 J5 - CCO

Subtask 1: Loop through all possible combinations of mixtures, and find the one with the minimum cost.

Time complexity: $O(N * 2^N)$

Subtask 2: Let dp[i][j][k] be the minimum cost needed to make a solution with j mL of solution A, k mL of solution B with the first i mixtures.

dp[i][j][k] = min(dp[i - 1][j][k], *(don't include the current mixture)*

dp[i - 1][j - a[i]][k - b[i]] + c[i]) *(include the current mixture)*

Then loop through all the pairs (j, k) with ratio X : Y to find the answer.

Time complexity: $O(N^3)$

LCC ‘20 Contest 4 S1 - All Hail!

More implementation :D

Observe that the range of years can only be from -10^4 to 10^4, so we can fit all the years within a frequency array. Afterwards, iterate through each of the leaders and count up the amount of accomplishments during their respective reigns. Make sure to implement the rules for tiebreakers mentioned in the problem.

LCC ‘20 Contest 4 S2 - Pens and Pencils

The problem states that we want to get P disjoint continuous segment of pens. Note, that the most efficient way to get the P segments would be to convert every other pencil into a pen, starting with the 1st.

Now consider the case where Q is the number of operations we must perform and Q > P. We can simply attach pens to the end as separate operations, but they will remain as one set because they are continuous.

LCC ‘20 Contest 4 S2 - Pens and Pencils

The solution illustrated earlier will always require the minimum N, amount of pencils, to fulfill the requirements for Q and P. All that remains is figuring out if a given configuration is impossible to fulfill.

Case #1: N is too small to create P disjoint continuous segments

Case #2: N is too small to fulfill the Q operations

Case #3: Q is too small to get P disjoint segments

Implementing the conditions is left as an exercise to the reader.

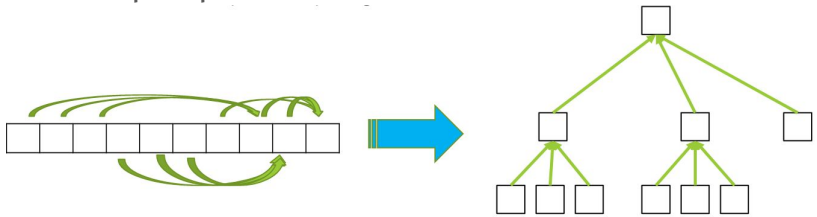
LCC '20 Contest 4 S3 - Deep Brackets

Notice that flipping the first M)s, and flipping the last M (s keeps our string balanced, and increases the depth as much as possible with 2M flips. By binary searching on the value of M, and looping through to find the depth, we can determine the minimum number of flips to achieve depth at least K.

Next, we need to find the lexicographically minimum set of 2M flips that still has a depth at least K. Of course, we still want to flip the first M)s (since they are the first M this is lexicographically least), then we loop through the string and maintain the depth until we find a point where the depth is K. After this, we flip the next M (s, which will be the first (s we can flip to maintain depth at least K.

LCC ‘20 Contest 4 S4 - Alan and Partner

The relationship can be converted into a tree structure as follows. From now on, we will talk about this tree structure. Then the purpose is to make the root of the tree (last node) as large as possible.



Consider some fixed X, let’s determine if we can make the root of the tree >= X. Denote a number >= X to be high, and a number < X to be low. Except for the leaf node, a node is high if two of the three nodes that affect that node are high. On the other hand, if there is less than one high, it will always be low. Therefore, if the required number of highs put in leaves for each of the three dependent subtrees is a, b, and c, then the new node requires a + b + c –max {a, b, c} highs put in leaves (two smaller ones).

LCC ‘20 Contest 4 S4 - Alan and Partner

Thus, we can do this tree dp to determine the minimum number of highs that Peter is required to put in the unknown slots. If this number is <= the number of highs we have in our array, then it is possible to get a root with value >= X.

Finally, we can binary search over X to find the maximum value of the root node.

LCC ‘20 Contest 4 S5 - Larry vs Theo

For subtask 1, the number of edges that can be added is predetermined regardless of what moves Larry and Theo make. Thus, we can just count the number of edges to be added to make our graph complete bipartite, and determine the winner based on its parity.

Similarly, if N is odd, our final complete bipartite graph is guaranteed to have an even number of edges, and thus since the total number of edges added is (final # - m)≡m (mod 2), the answer is once again predetermined.

We extend this idea to when N is even. We have some components, each of which can be represented by (0,0), (1,1), or (0,1). Where the two numbers indicate the parity of the parts modulo 2. Since N is even, we must have an even number of (0,1)s.

LCC '20 Contest 4 S5 - Larry vs Theo

The winner is determined by whether the final component is (0,0) or (1,1). If there are no (0,1)s, the parity of the final component is predetermined, since adding (0,0)s and (1,1)s is commutative.

If there are 2 (0,1)s left, whoever goes first wins, as they can either combine them into a (1,1) or a (0,0) depending on which parts they add an edge between.

If there are 4 (0,1)s left, no one wants to pair 2 (0,1)s, since the next player will win by matching the 2 remaining (0,1)s. Thus, we are bound to end up with 4 (0,1)s before someone is finally forced to make a move. The 4 (0,1)s each have an even number of edges, which means the answer is fixed. Similarly, if there are more than 4 (0,1)s, we still end up in a similar situation which has an even number of edges, so the answer is still fixed based on the parity of M.