Programming League 2021
Finale Round Editorial

Contest Team

May 2021

Contents

1 Introduction

The problems are sorted by categories, Closed Category then Open Category.
Contests link:
Programming League Contest 2021 - Finale Round [Closed]
Programming League Contest 2021 - Finale Round [Open]
We would like to appreciate Mike Mirzayanov for Codeforces and Polygon platforms. Huge thanks to our problems authors and testers.

- Amos Tan Li Sheng, UM '24
- Chooi He Lin, UM '23
- Kenneth Yong Hau Woon
- Lee Zong Yu, NTU '24
- Ong Jack Min, UM '23
- Tan Jun Wei, UM '23
- Yeoh Zi Song, MIT '24

2 Cat Food

Author: Chooi He Lin

Abridged Statement

Consider an array where the elements are either 0 or 1. We are given the starting and ending indexes of each contiguous segment of 1's. We are also given an integer x, and are allowed to either jump ahead x steps (without collecting any 1's) or walk 1 step (and collect a 1 in that position, if it exists). Find the minimum number of steps needed to collect all 1 values, if we start right before the array.

Solution

One can solve the problem using greedy algorithm. First, sort the sequences provided according to the start index of the sequences. Next, loop through the sequences. If we can jump to skip x meters without skipping the first meter of next sequence, we jump from i^{th} meter (the meter we currently at) to $(i+x)^{th}$ meter, else we walk to the end meter of the next sequence.

Time Complexity: $O(NlogN)$

3 The Juggernaut

Author: Ong Jack Min

Abridged Statement

Given jugs of capacity A and B. Count the amount of integers in an array of integers that are reachable using the 4 operations:

1. Fill a jug to full.
2. Pour the contents of one jug completely into another.
3. Pour the contents of one jug into another jug until the other jug is full.
4. Empty a jug.

Solution

There are 2 conditions that are necessary for Juggernaut Meow to be able to serve the request r.

1. $A + B \geq r$
2. $r \mod GCD(A, B) = 0$

As Juggernaut Meow needs to serve the Milo in one serving, the maximum request he can serve is the maximum capacity of the 2 jugs A + B.

The operations given can be reduced to adding or subtracting the integers A or B. Thus, in order for an integer q to be reachable, it must have integer solutions to the equation $xA + yB = q$. This means it must satisfy $xA + yB = q$ where $x, y \in \mathbb{Z}$. This is only true if q is a multiple of the greatest common divisor of A and B, GCD(A,B).

A proof is given here: https://math.stackexchange.com/questions/20717/how-to-find-solutions-of-linear-diophantine-ax-by-c

Time Complexity: $O(N + log(min(A, B)))$

4 Meow Logic

Author: Lee Zong Yu

Abridged Statement

Determine the validity of an argument with N premises and a conclusion and all the statements are in prefix expression that contains p, q, r, s, t — the logical variables (operand) which can take value 1 (True) or 0 (False) and &, |, >, =, ! — logical operators meaning AND, OR, IMPLY, EQUAL, NOT respectively.

Theorem

"Argument is valid if and only if conjunction (AND) of all the premises is true implies the conclusion is a tautology."

Solution

There are several ideas we can get from the Theorem,

1. For some values combination of the logical variables, if the combination returns a 0 (False) for any of the premises, we can ignore the combination as it won't affect the validity of the argument.

2. If any of the values combination of the the logical variables return a 1 (True) for all the premises but return a 0 (False) for the conclusion, the argument is INVALID.

3. Conversely, If ALL the values combination of the logical variables return either a 0 (False) for any of the premises (1^{st} point) or 1 (True) for all the premises and the conclusion, then the statement is VALID.

We can solve these arguments by using Truth Table. Consider the following example,
Argument with premises: $p \rightarrow q$ and q with a conclusion: p. By drawing out the truth tables

Row	p	q	$p \rightarrow q$	q		p	
1	0	0	1	0			
2	0	1	1	1	Critical Row	0	(Counter Examples)
3	1	0	0	0			
4	1	1	1	1	Critical Row	1	

We can see the row 2 and row 4 are critical row, a critical row is a row where all the premises are True (1). After determining the critical rows, we can check whether ALL the conclusions of those critical rows return 1, if any of it return a 0, the arguments are said to be invalid. As in this case, row 2 is one of the

counterexamples, therefore, it is invalid.

To implement it with coding, a complete search, a.k.a brute force solution should works (reason and complexity analysis will be explained below). By looping through all the possible combinations of the five variables (by implementing five for loop or using a data structure known as *Bitmask*), we may determine the critical rows and also the validity of the argument by method mentioned above.

The main troublesome part of the problem is to design a **bugfree** code in evaluating the prefix expression. First, the reason of using prefix expression instead of infix expression (the type that most commonly seen) is for ease and convenience as evaluating a infix expression are complicated in the sense that we need to know its left-to-right association, operator precedence rules and parentheses. Before going into prefix expression, Let have understanding in solving another form of expression which is *Postfix Expression* in which operator comes after operand ($operand_1\ operand_2\ operator$). We can utilize a data structure, *stack* to evaluate this.

One key observation is: after evaluating an expression, it will become an operand. For example, $operand_1\ operand_2\ operator$, $operand_1$ may be equal to $operand_3\ operand_4\ operator_2$. Let say $1 \times 2 + 3$ in postfix form is $12 \times 3+$. You may notice that $12\times$ is $operand_1$, while 3 is $operand_2$. And eventually, it will reach a last operand which is the final answer for the expression. Therefore using a stack where it only stores operands, looping through the string, push operands into a stack whenever you encounter an operand. When encountering a logical operator, pop out the top two operand and evaluate them based on the logic of operator. After that, push the answer back to the stack. This process is continued until eventually there is single logical variable there. It is obvious that this runs in $O(len(p))$, where p is the premise. Knowing this, you can actually reverse the prefix expression (the reason of using prefix is because prefix is more readable than postfix) and it will become similar to a postfix expression (although not exactly) and evaluate based on the algorithm above.

Time Complexity: $O(2^5 \times \sum_{i=1}^{n} len(p_i))$

5 Meow SQL

Author: Ong Jack Min

Abridged Statement

Count the number of characters in all the unique lines.

Solution

We can easily count the number of characters by knowing the length of the strings. In order to only count unique values, we can ingest all the input into a set and count using the set. This is because a set will only have unique values. Unordered set insertions can be expected to run in $O(1)$ while ordered set insertions are $O(logN)$. Either one would pass the question constraints.

Time Complexity: $O(N)$

6 Meow Maze

Author: Amos Tan Li Sheng

Abridged Statement

Given a $N \times M$ grid, Nyan Cat starts at 'S' and wants to reach the exit of the maze 'E'. There are blocked cells represented by '#' and empty cells represented by '.'. Nyan Cat can move from one cell to a neighbouring cell (any cell that shares a side with the current cell) in one second. The maze also contains several (possibly none) teleportation portals which are represented by 'O'. Nyan Cat can teleport from one 'O' cell to another 'O' cell instantly (in 0 seconds). Find the minimum number of seconds required for Nyan Cat to reach the exit of the maze. Output -1 if it is impossible.

Solution

Notice when going from 'S' to 'E', we either teleport only once, or never teleport at all. Consider a sequence of steps where we teleport more than once, for example (Teleport from A to B, walk from B to C, teleport from C to D). If we teleport from A, A must be a 'O' cell. Similarly, if we teleport to D, D must be a 'O' cell, and therefore we could have saved time by teleporting from A to D directly.

Hence, we will use BFS to find SE: the minimum time needed to walk directly from 'S' to 'E', SO: the minimum time needed to walk from 'S' to a 'O' cell, and EO: the minimum time needed to walk from a 'O' cell to 'E'. Note that SE represents the minimum time needed to walk from 'S' to 'E' without teleporting, and $SO + EO$ represents the time needed to walk from 'S' to 'E' by teleporting exactly once.

Let us initialize SE, SO and EO with a large integer to represent infinite time (e.g. 10^9). We will use BFS from 'S' to determine SE (if 'E' is reachable from 'S') and SO (if any 'O' cell is reachable from 'S'). We keep track of the distances of each cell from 'S' using an array, and update SO when we reach an 'O' cell, and SE when we reach 'E'. After that, we perform BFS from 'E' another time to determine EO in a similar manner. If $min(SE, SO + EO) = 10^9$, we cannot reach 'E' from 'S' no matter what path we take, otherwise we have found the answer.

Time Complexity: $O(NM)$

7 Meow and Miau

Author: Amos Tan Li Sheng

Abridged Statement

Meow and Miau are playing a game, Meow has A cards and Miau has B cards. The cards are either white or grey. They shared an array with a total of N integers, $c_1, c_2, c_3, \ldots, c_N$. During the i^{th} turn of the game, Meow will pass c_i cards to pass Miau, and Miau will pass c_i cards (may include what Miau just received from Meow) to pass Meow. The score of the game is $|a_g - b_w|$. Meow wants to maximize the score while Miau wants to minimize the score.

Solution

Notice that the score, $|a_g - b_w|$, is fixed.

$$|a_g - b_w| = |(A - a_w) - b_w| = |A - a_w - b_w|$$

Neither of them will affect the scoring of the game.

Time Complexity: $O(1)$

8 The Prime Meow

Author: Amos Tan Li Sheng

Abridged Statement

Given two arrays, a and b, each array contains N integers. Meow writes from a_i to b_i inclusively for each i and always keeps prime numbers. Find the sum of the numbers Meow writes.

Solution

This is a harder version of 'The Biased Meow' at 15 and this question is similar to the problem, 'Meow String', in the Preliminary Round [Closed]. Meow always skips prime numbers, hence Meow always writes non-prime numbers and they are continuous non-prime numbers. We handle the sum one by one for each a_i to b_i. We use dynamic programming by having prefix sums and the well-known Sieve of Eratosthenes to determine the prime numbers.

Firstly, denote $prime[i]$ as whether i is a prime. (Use Sieve of Eratosthenes)
Secondly, denote $dp[i]$ as the sum of non-prime numbers from 1 to i.

$$dp[i] = dp[i-1] + (prime[i]?0 : i)$$

Thirdly, calculate the sum of non-primes numbers from a_i to b_i by using

$$sum = dp[b_i] - dp[a_i - 1]$$

Lastly, sum up all the sums.

Time Complexity: $O(MloglogM)$ where $M = max\{b_i : i = 1..n\}$

9 Hash Cat

Author: Ong Jack Min

Abridged Statement

Given the hashing algorithm:

1. Split the message into packets of 4 ASCII characters. Concatenate their bytes to form a 4-byte integer, m.
2. Generate N pairs of integers (a_i, b_i) and multiply them with m to generate N integers $a_i b_i m$.
3. Sum all N integers and modulo the result by an 8 decimal digit prime number P to form E.

$$E = (\sum_{i=1}^{N} a_i b_i m) mod P$$

Solution

We can first use the commutative and distributive property of modular arithmetic to factor the equation:

$$E = (\sum_{i=1}^{N} a_i b_i m) \mod P$$

$$E = m(\sum_{i=1}^{N} a_i b_i) \mod P$$

This means we only need to calculate the term after the m once for all the packets. Let this number be k for key.

$$k = (\sum_{i=1}^{N} a_i b_i) \mod P$$

$$E = mk \mod P$$

Now if we were to hash all the possible m to see if there is a matching E, we would need to iterate through $(26 + 26 + 10)^4$ Q possibilities. As this number is in the 12 digit range, this would yield a TLE solution.

However, we can actually get an equation that will yield all numbers that hash to E. We then would only need to make sure that the generated m is valid. We can do this by multiplying both sides by the modular multiplicative inverse of k, k^{-1}:

$$Ek^{-1} = mkk^{-1} \mod P$$

$$Ek^{-1} = m \mod P$$

The modular multiplicative inverse can be obtained using the Extended Euclidian Algorithm in $O(log(min(A, B)))$ time. We can then infer that adding a multiple of P will not change its value under $\mod P$.

$$q \in \mathbb{Z}, Ek^{-1} = Ek^{-1} + qP = m \mod P$$

If we test for the validity of the numbers generated by our formula, the worst case number of integers we would need to test for each packet would be:

$$\lfloor \frac{highest\, m - lowest\, m}{P} \rfloor$$

$$\lfloor \frac{2054847098(\text{“}zzzz\text{”}) - 808464432(\text{“}0000\text{”})}{10000019} \rfloor = 124$$

Time Complexity: $O(Q + log(min(A, B)))$

10 Meow Flower

Author: Amos Tan Li Sheng

Abridged Statement

Given a $N \times M$ grid, each cell is either '1' as being occupied or '0' as being emptied. Meow can swap at most K times between an empty cell and a occupied cell. After at most K swaps, find the maximum square are of a grid which contains only occupied cell.

Solution

This problem can be solved by using 2D prefix sums. Denote $s[i][j]$ as whether the cell at i^{th} row and j^{th} column is occupied and $dp[i][j]$ as the number of occupied cells for a rectangular grid from $(1, 1)$ to (i, j) or (1^{st} row, 1^{st} column) to (i^{th} row, j^{th} column).

$$dp[i][j] = dp[i][j-1] + dp[i-1][j] - dp[i-1][j-1] + (s[i][j] == 1)$$

After doing it, we need a variable, $total$, as the total number of occupied cells in the $N \times M$ grid for later use. In order to solve this problem, we use binary search to search the maximum size of the length of the square grid. Denote len being the current length of the square grid we are searching. We will iterate the bottom right point of the available square grids from (len, len) to the (N, M), which is a 2D Sliding Window.

For each $len \times len$ square grid which has the bottom right point (i, j), the number of occupied cells are

$$dp[i][j] - dp[i-len][j] - dp[i][j-len] + dp[i-len][j-len]$$

Next, we need to check whether if the square grid contains $len \times len$ occupied cells after at most K swaps, since we know the total number of occupied cells, denote cnt as the number of occupied cells in the square grid before any swap.

$$cnt = cnt + min(k, total - cnt)$$

The new cnt is the maximum number of occupied cells of the square grid after at most K swaps. If the new $cnt \geq len \times len$, the square grid can contain all occupied cells after at most K swaps.

Time Complexity: $O(NMlog(min(N, M)))$

11 Meow does not dream of killer bunny senpai

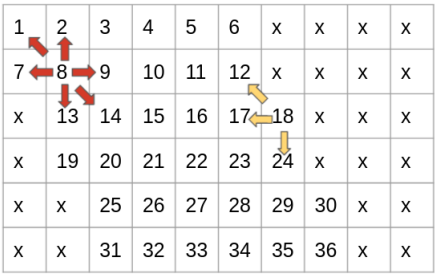
Author: Ong Jack Min

Abridged Statement

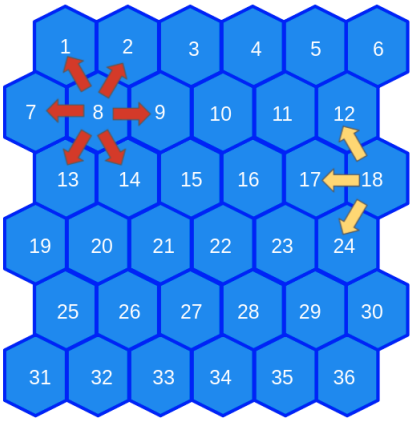
Given a hexagonal tessellation of with R rows and C columns, with 'm' denoting the presence of a musical cat, 'b' denoting the presence of a killer bunny, and '.' denoting an empty tile, find the number of musical cats that are reachable by the killer bunnies.

Solution

There are a few ways one can model a hexagonal tessellation. One of the ways is to model it as a grid by using an offset every 2 rows. Any tile can then traverse to its North, South, East, West, North-West and South-East.



Grid representation of sample case 1.



Hexagonal representation of sample case 1.

We can then use any sort of graph traversal algorithm starting from the killer bunnies such as DFS or BFS to traverse the graph and count the number of

bongo cats encountered. We should make sure we store a global visited for all the killer bunnies to prevent a TLE solution from running an O(RC) search RC times on the case where the whole grid is killer bunnies.

If we use Python, an explicit DFS using stack or explicit BFS using queue must be used as the recursive approach will exceed the stack frame limit of the Python interpreter.

Time Complexity: $O(RC)$

12 Meow Writing Numbers

Author: Yeoh Zi Song

Abridged Statement

Suppose we have a positive integer N. Consider the concatenation of all integers between 1 and N, 1234567891011...N. We erase some of its digits (possibly none) from the concatenation. Given the result, determine the minimal N that could have produced the string given.

Solution

This problem can be solved by greedy. We list down the positive integers one by one. We keep a pointer that initially points to the first letter of s. Whenever the pointed character in the string s matches the corresponding digit of the integer, we move the pointer one step to the right and continue. Repeat this process until the pointer reaches the end.

However, we still need to know whether the answer can be large. The key is to note that the answer will never exceed 10^6, because after writing down 10 consecutive numbers, at least one of them has last digit equals to the current digit, so the pointer will move to the right at least once when we write down 10 consecutive numbers. Thus, in the worse case, we'll only list down the numbers from 1 to 10^6, which is definitely fast enough.

Time Complexity: $O(|S|)$, where $|S|$ is the length of the string given.

13 Meow Christmas Tree

Author: Yeoh Zi Song

Abridged Statement

Given a Christmas Tree (consisting of smaller Christmas Trees) of size N, its cells are coloured either black or white. Count the number of Christmas Trees of size 1, 2, ..., N that are fully coloured white and black respectively.

Solution

This is a dynamic programming problem. We let $dp[i][j]$ denote the maximum size a Christmas Tree with the same color in its interior that can be formed with the top triangle being the j-th triangle on the i-th row. (note that in this case j must be odd, or else the triangle is inverted and does not count as a Christmas Tree) Now, the dp transitions are obvious. The triangles at the bottom have $dp[i][j] = 1$. When we consider the j-th triangle on the i-th row, we look at the triangles below it, i.e. the $j, j+1$ and $j+2$-th triangle on the $(i+1)$-th row. Note that the $j+1$-th triangle will be inverted in this case, since $j+1$ is even. Now, if they're not all of the same color as the current triangle, then we can safely say that $dp[i][j] = 1$, since the triangle cannot be extended anymore. Otherwise, we take $dp[i][j] = min(dp[i+1][j], dp[i+1][j+2]) + 1$.

After calculating all the dp values in $O(n^2)$ time, we can finish the solution by counting for each i, how many dp values are equal to i, and then we can count the number of Christmas Tree of size i for all i easily.

Time Complexity: $O(n^2)$

14 Meow Swap

Author: Amos Tan Li Sheng

Abridged Statement

Given a permutation of N integer, Meow is required to swap the integers to make the sequence in decreasing order. During each swap, Meow could only swap the i^{th} integer with either $(i-2)^{th}$ integer (if there is integer at index $i-2$) or $(i+2)^{th}$ integer (if there is integer at index $i+2$). Find the minimum number of swaps that Meow needed to make the sequence in decreasing order. Output -1 if it is impossible.

Solution

Since the desired outcome would be a decreasing permutation and we can only swap indices with the same parity, we conclude that if N is odd, the i^{th} integer in the permutation must be equal the its parity, $i\%2$; if N is even, the i^{th} integer in the permutation must not be equal the its parity, $(i+1)\%2$. If there is integer which does not follow the above rule, the task is impossible.

Next, we could break the permutation into two arrays of integers — one having all the odd indices and one having all even indices. This modification simplifies our situation to just count the number of 'bubble sort' swaps (the number of swaps between adjacent indices) of the new arrays to make them decreasing order. (Note: this is a well-known problem called "counting the number of inversions")

Let us iterate the new arrays one by one. Denote a_i as the i^{th} integer of the current array. We could start swapping from the smallest integer to the largest integer of the array, as long as the number on the right is greater than current integer, we swap them. For each a_i, the number of swaps required is the number of $a_j > a_j$, where $j > i$. This is doable with a well known Data Structure — Segment Tree. The minimum number of swaps that Meow needed to make the sequence in decreasing order is the sum of all the number of swaps of each integer in two arrays. This problem can also be solved with Order Statistics Tree and Merge Sort.

Time Complexity: $O(NlogN)$

15 The Biased Meow

Author: Amos Tan Li Sheng

Abridged Statement

Given two arrays, a and b, each array contains N integers. Meow writes from a_i to b_i inclusively for each i and always keeps even numbers. Find the sum of the numbers Meow writes.

Solution

Meow always skips even numbers, hence Meow always writes odd numbers and they are continuous odd numbers. We handle the sum one by one from each a_i to b_i. Let us deduce the formula of a continuous of odd numbers from a_i to b_i. Denote

- x as the first odd number Meow writes and
- m as the number of odd numbers Meow writes.

$$\begin{aligned}
&x + (x+2) + (x+4) + \cdots + (x+2(m-1)) \\
=&(x + x + 2(m-1)) \times \frac{m}{2} \\
=&2 \times (x + m - 1) \times \frac{m}{2} \\
=&(x + m - 1) \times m
\end{aligned}$$

Next, we need to determine the value of m, we can do some offset for a_i and b_i to make it odd. When a_i is even, we increase a_i by 1. When b_i is even, we decrease b_i by one. Note that if original a_i and b_i are both equal and even, this will end up $a_i > b_i$, which we will need to ignore.

By using formula, $b_i = a_i + 2(m-1)$, we can determine m. Hence, the answer for this problem is $\sum_{i=1}^{n}(a_i + m - 1) \times m, m = \frac{b_i - a_i}{2} + 1$.

Time Complexity: $O(N)$

16 Meow Planet

Author: Lee Zong Yu

Abridged Statement

Given an array of integers, Meow and Miau each choose an integer (cannot be the same index) such that Meow want to maximize the absolute differences between the integers they choose and Miau want to minimize it. Output the absolute difference if Meow choose first and if Miau choose first assume that they plays optimally.

Solution

Let us consider the case where Meow chooses first. Since Meow knows that for any integer he chooses, Miau will choose the integer that have the minimum absolute difference. By this, Meow would have choose the maximum out of all the minimum absolute difference, a.k.a. Maximin. If Miau choose first, Miau would want to minimize the maximum absolute difference, a.k.a. Minimax. In other words, the first answer is $\max_{i=1}^{i=n} \min_{j=1}^{j=n} |a_i - a_j|$, such that $i \neq j$, second answer is $\min_{i=1}^{i=n} \max_{j=1}^{j=n} |a_i - a_j|$, such that $i \neq j$. However, we can notice that this solution has a time complexity of $O(N^2)$ and it will give us the verdict — Time Limit Exceeded.

How do we optimise it? Note that the when the array is sorted, there are only two possible values for choosing j for the inner loop. To minimize the inner loop of first answer, we would just choose one of the neighbours of a_i. To maximize the inner loop for second answer, we would just choose one of the integers from both ends (a_1 and a_n) (proof is attached below). The first answer is $\max_{i=1}^{i=n} \min(|a_i - a_{i-1}|, |a_i - a_{i+1}|)$ and second answer is $\min_{i=1}^{i=n} \max(|a_i - a_1|, |a_i - a_n|)$. Sorting the array requires $O(N \log N)$ time, and after that we can find the answer in $O(N)$ time, therefore, the time complexity is $O(N \log N)$

Proof

Given a sorted array a with N elements $a_1 \leq a_2 \leq \ldots \leq a_n$.
Denote a_i as the chosen number of the first player and a_j as the chosen number of the second player.

$$\begin{aligned} a_1 \leq a_2 \leq \ldots \leq a_i \\ -a_i \leq -a_{i-1} \leq \ldots \leq -a_2 \leq -a_1 \\ a_i - a_i \leq a_i - a_{i-1} \leq \ldots \leq a_i - a_2 \leq a_i - a_1 \\ 0 \leq a_i - a_{i-1} \leq \ldots \leq a_i - a_2 \leq a_i - a_1 \end{aligned}$$

$\therefore a_i - a_{i-1}$ is the minimum and $a_i - a_1$ is the maximum if $j < i$.

$$\begin{aligned} a_i &\leq a_{i+1} \leq \ldots \leq a_n \\ a_i - a_i &\leq a_{i+1} - a_i \leq \ldots \leq a_n - a_i \\ 0 &\leq a_{i+1} - a_i \leq \ldots \leq a_n - a_i \end{aligned}$$

$\therefore a_{i+1} - a_i$ is the minimum and $a_n - a_i$ is the maximum if $j > i$.

Time Complexity: $O(N \log N)$