Programming League National 2022
Preliminary Round Editorial

Contest Team

March 2022

Contents

1 Introduction

The problems are sorted by categories, Closed Category then Open Category.

Programming League Contest 2022 is hosted using Codeforces.

Here are the contest links:

- Preliminary Round [Closed]
- Preliminary Round [Open]

We would like to appreciate Mike Mirzayanov for Codeforces and Polygon platform. Huge thanks to our problem authors and testers.

- Tay Qi Xiang, UM' 24
- Chin Shan Hong, UM' 24
- Chong Yi Fong, UM' 24
- Lim Hong Zhi, UM' 24
- Chan Chee Sun, UM' 24
- Lei Wing Yee, UM' 24
- Lim Yun Kai, Senior Software Developer at Grab

2 Closed Category

2.1 Sentence Reconstructing

Author: Lei Wing Yee

Tags: Implementation, Strings

Abridged Statement

Given two strings for example, $str1$ and $str2$ which can contain any characters, the task is to check whether $str2$ can be formed from the characters of $str1$.

Solution

It can be seen from the problem that the order of the characters does not matter. Only the count of each character matters. All the number of characters in $str2$ should be present in $str1$. You need to maintain a count array/frequency array of size 256, which would store the frequency of each character in the given strings.

To do this, first traverse through the first string, $str1$, and keep a count of the number of characters. Now traverse through the second string, $str2$, and keep subtracting the count of each character. If the count becomes negative at any point, that means the second string cannot be formed from characters of the first string, so return false, else return true.

Time Complexity : $O(n)$

2.2 Pawsome Cards

Author: Chan Chee Sun

Tags: Implementation

Abridged Statement

Check if all items with x and y values h and w, if $x \leq h$ and $y \leq w$ or $y \leq h$ and$x \leq w$.

Solution

Based on each type of query

If query is "+"

- Ensure value x is always bigger than y (switch them if they're not)
- If new value x is bigger than the older value x, update the value x to the new value (same goes to y)

If query is "?"

- Compare value x to h and y to w OR y to h and x to w
- Print 1 if either statements are correct, and 0 if both are wrong

Time Complexity : $O(n)$

2.3 Purrhaps a Jog?

Author: Chan Chee Sun

Tags: Implementation, Math

Abridged Statement

Find the sum of last digit of all integers from 1 to a divisible by b

Solution

The problem is based on simple observation.

Let $k = \lfloor \frac{a}{b} \rfloor$ be the number of integers from 1 to a divisible by b.
Since the last digits are needed to be added in the original sum. So, the last digit can be in the range of 0 to 9 and the cycle can be formed by observing the array pattern.

So for each cycle, sum = sum of first 10 last digits, after this, we can divide k by 10 and add the last digit of remaining numbers from starting.

Hence, Sum = (no. of the cycle $\times$ sum of the last digit of first 10 integers divisible by b) + (sum of the last digit of k%10 integers divisible by b).

Time Complexity : $O(log(\frac{a}{b}))$

2.4 Money Bag

Author: Tay Qi Xiang

Tags: Dynamic Programming

Abridged Statement

Given a bag with volume V, and n items each with their respective volume, find the minimum possible volume of the bag after filling in the items.

Solution

This is a variant of the 0/1 knapsack, known as the maximum subset-sum problem.

The problem requires finding the minimum remaining space, that is, asking for the maximum loadable weight. For each object, there are two states: loaded and unloaded.

Then, for any weight m the maximum value $f(m)$,

$$f(m) = max(f(m - w[i]) + w[i], f(m))$$

Among them,
w is the weight (i.e. value)
$f(m - w[i])$ refers to the maximum weight that the remaining capacity of the bag can hold after item i is loaded
$f(m - w[i]) + w[i]$ refers to the maximum weight that the bag can hold after item i is loaded

Time Complexity : $O(nV)$

2.5 Ice Cream

Author: Lim Hong Zhi

Tags: Math, Combinations, BigInteger

Abridged Statement

Given n and r. Find the number of combination without repetitions.

Solution

We can apply the formula for combination without repetition : $\binom{n+r-1}{r}$ to get our answer.

Whereas the formula for combination is $\binom{n}{r} = \frac{n!}{(n-r)!r!}$.

$$\binom{n+r-1}{r} = \frac{(n+r-1)!}{(n+r-1-r)!r!} \tag{1}$$

$$= \frac{(n+r-1)!}{(n-1)!r!} \tag{2}$$

Since the answer will exceed 2^{64}, we will need to use BigInteger for Java or implementing our own BigInteger class for C++.

Time Complexity : $O(n+r)$

3 Open Category

3.1 Number Reconstruction

Author: Tay Qi Xiang

Tags: Math

Abridged Statement

Given an integer S. Find the possible values for integer n such that $n - \lfloor \frac{n}{10} \rfloor = S$.

Solution

We cannot simply go through every single possible values of n because the number of operations is too large. (Time Complexity : $O(n)$)

Instead, we can do much better.

Idea 1:

The Euclidean Division states that there exists some integer t such that

$$\lfloor \frac{n}{10} \rfloor = \frac{n-t}{10} \text{ , where } t = \{ t \mid 0 \leq t \leq 9, t \in \mathbb{Z} \} \tag{1}$$

Thus,

$$S = n - \lfloor \frac{n}{10} \rfloor = \frac{9n-t}{10} \tag{2}$$

To get back our original number n, we multiply the number, S by $10/9$

$$\frac{10}{9} \times \frac{9n-t}{10} = n - \frac{t}{9} \tag{3}$$

We can see that there only exists 2 possible integer solutions, one of it is n and the other one is $n-1$ when $t = 9$. Hence, to get the answer, we multiply S by $10/9$, and when S is divisible by 9, the other solution is $n-1$.

Time Complexity : $O(1)$

Idea 2: (Sub-optimal but still accepted)

Since the question required 3 digits or above, we can represent N as $100a+10b+c$; $M = 10a + b$.

$$S = N - M \tag{1}$$
$$S = (100a + 10b + c) - (10a + b) \tag{2}$$
$$S = 90a + 9b + c \tag{3}$$
$$a = \frac{S - 9b - c}{90} \tag{4}$$

There only exists 10 possible values for b $(0 - 9)$ and 10 possible values for c $(0 - 9)$. Thus, we only need to loop through b and c a maximum of 100 times to find the solution.

The answer will be in the form of $(100 \times \frac{S-9b-c}{90} + 10 \times b + c)$.

Time Complexity : $O(1)$

3.2 Covid

Author: Chin Shan Hong

Tags: Disjoint Set Union(DSU), DFS, Flood Fill

Abridged Statement

Given a group of size m × m people, where 0 represent healthy people, 1 represent people infected by Covid-19 virus, and 2 represent people infected by Omicron. Find the largest cluster of Covid-19 and Omicron. If the largest cluster size of Omicron is larger or equal to largest Covid -19 cluster, output "MOH should focus on Omicron". Or else the program should output "MOH should focus on Covid-19".

Solution

This problem can be solved using Union Find data structure.

We need to create a one-dimensional array of size m × m called parent to keep track of the clusters where each people belong to. We also need to create another one-dimensional array of size m × m called clusterSize to store the size of each cluster. Initially, all the people have aparent (cluster) which is itself, where parent[clusterIndex] = clusterIndex. And the initial size of all the clusters is 1, where clusterSize[clusterIndex] = 1. After that, we loop through the group of people given in the input, if the people are healthy then we can ignore it. Otherwise, if the people are labelled as 1 or 2, then we need to group them together into clusters.

When we identified the people are labelled as 1 or 2, we need to check the whether people adjacent to them horizontally and vertically have the same label with them or not. If yes, we will check whether they belong to the same cluster with the parent array using recursion. If we found both of the people have the same clusterIndex, then they belong to the same cluster and we do nothing. Or else we need to join them into one largest cluster. When merging two clusters into one cluster, we need to merge the smaller cluster into the larger cluster and update the clusterSize array where clusterSize[clusterIndex] += clusterSize[anotherClusterIndex]. Also, we need to update the parent array where parent[yClusterIndex] = parent[xClusterIndex] if y cluster is smaller than x cluster and vice versa. That means the y cluster is already part of x cluster and vice versa.

When we finish merging all the clusters, we loop through the clusterSize array to find the largest Covid-19 cluster and largest Omicron cluster and compare them. If the size of Omicron cluster is greater than or equal to Covid-19 cluster, we output "MOH should focus on Omicron". Else we output "MOH

should focus on Covid-19".

Note that this question can be solved with dfs with flood fill.

Time Complexity : $O(n^2)$

3.3 Meow's Pizzeria

Author: Lei Wing Yee

Tags: Greedy, Priority Queue

Abridged Statement

Meow and his friend, Miau are pizza delivery guys. They take tips of A[i] and B[i] corresponding to the i^{th} order. Note that each order will be taken by only one person. Meow can take at most X orders and Meow's friend can take Y. You must divide the orders between Meow and his friend such that total tips are maximized. You are also given a constraint that $X + Y \geq n$, which will guarantee that none of the orders will go unprocessed.

Solution

We can use following greedy strategy. Sort all the orders by $D[i] = |A[i] - B[i]|$. Process the order one by one in the above sorted order and assign each order to either Meow or his friend greedily depending on relation between A[i] and B[i].

Let us denote $D[i] = |A[i] - B[i]|$. Now we will sort all the orders in decreasing order of D and then we will process the orders one by one. We can have following two cases.

1. If $A[i] > B[i]$, then we will try to assign it to Meow if possible (If after the assignment, limit of orders is not crossed). Otherwise, we will assign it to Miau.
2. If $B[i] > A[i]$, then we will try to assign it to Miau if possible. Otherwise, we will assign it to Meow.

Note that the condition $X + Y \geq n$ guarantees that we will be able to assign the order to one of the persons.

Time Complexity : $O(nlog(n))$

3.4 Line Intersection

Author: Chong Yi Fong

Tags: Coordinate Geometry

Abridged Statement

Given n lines represented by 2 end-points, find whether all lines intercept with each other.

Solution

We use a counterclockwise line intersection algorithm to determine whether any 2 lines are intercepted. We also check whether the 2 lines have the same point, and they are deemed as intercepted. By iterating through all combinations of any 2 lines in the given list, we can find whether all lines are intercepted with each other.

Time complexity: $O(n^2)$

3.5 Exploit Resources

Author: Tay Qi Xiang

Tags: Dynamic Programming

Abridged Statement

Given n planets in sequence order, and power level of p. Each planet can be either resource-based or maintenance-based. For resource-based, if choose to mine, you will earn $a_i \times p$ and $p = p \times (1 - 0.01k)$. For maintenance-based, if choose to repair, you will pay $a_i \times p$ and $p = p \times (1 + 0.01c)$. Find the maximum earning.

Solution

If we select from 1 to n, there will be an aftereffect. Thus, we will select the planets from n to 1. Why?

The initial value is L, and after mining once (same logic applies to repair), it becomes $L \times (1 - k\%)$, if we mine again, we will get $money + L \times (1 - k\%) \times a_i$. This means for every time we mine, every subsequent a_i will reduce by $k\%$, i.e. the latter decision will be affected by the previous one.

Hence, let $f[i]$ be the maximum money earned from i to n (1-indexed). We will get the following DP relationship:

$$f[i] = \begin{cases} max(f[i+1], f[i+1] \times (1 - k\%) + a_i : \text{type} = 1 \\ max(f[i+1], f[i+1] \times (1 + c\%) - a_i : \text{type} = 2 \end{cases}$$

Our final answer will be $f[1] \times L$. (1-indexed)

Time complexity : $O(n)$