TAKE-OFF
PROGRAMMING CONTEST

DIU Take-Off Programming Contest
Fall 2022 [Main Round]

Organized By

Daffodil International University

DEPARTMENT OF CSE
DAFFODIL INTERNATIONAL UNIVERSITY

DIU INTRA CSE FEST 2022

Computer & Programming Club

Problem Analysis

Platform Support

Toph

Judging Panel

Judging Director

Abu Saleh

11th Semester

Department of CSE

Daffodil International University

Associate Judging Director

Masum Bin Hossain

10th Semester

Department of CSE

Daffodil International University

Judges

Rahat Islam Srijon	**Shimul Sutradhar**	**MD Dipo Sultan**	**Siam Ahmed**
12th Semester	12th Semester	11th Semester	11th Semester
S A H Nahid	**Saiful Islam**	**Nesar Ahmed**	**MD Fahim Istiak**
9th Semester	8th Semester	8th Semester	8th Semester

Index

Special Thanks

Rahat Islam Srijon

MD. Albin Hossain

Md Galib Hossain

Problem Set:

TOPC FALL 2022 Problem Set

Solution Code:

TOPC FALL 2022 Solution

A. 45th ICPC Unsolved Problem!

Category: Give away
Problem Setter: Masum Bin Hossain
Reviewer: Md Galib Hossain
Alternate Writer: Abu Saleh

Analysis:

~~**Setter is too busy to give the analysis.**~~

Time complexity: **O(1)**
Space Complexity: **O(1)**

B. Who Is the Champion?

Category: Simple if else
Problem Setter: Samia Dewan Ria
Reviewer: Siam Ahmed
Alternate Writer: Abu Saleh

Analysis:

To solve the problem we need **knowledge about if else**. As statement says that both number can not equal, so for the maximum number we should print champion and for minimum it should be runnerup.

Time complexity: **O(1)**
Space Complexity: **O(1)**

C. Opera & B2G1

Category: Simple math
Problem Setter: Siam Ahmed
Reviewer: Rahat Islam Srijon
Alternate Writer & Special Thanks : MD. Albin Hossain

Analysis:

Observation: This problem can be solved by simple iteration or by using mathematical observations. As the problem category is Simple Math, let's solve it with math. Forget about the term "Buy 2 Get 1". How many pizzas actually you can buy?If for buying 2 pizzas, you're getting 1 additional pizza for free, then for buying C pizzas, how many additional pizzas will you get?

Solution Idea: If the pizza price is X, and the dollar you have is M. Then the total number of pizzas you can buy is, C = M/X

For example, M = 20, X = 6, then C = (20/6) = 3.
Now you have the C. So, the additional pizzas you will get will be,
=C/2
= (M/X) / 2 [As, C = M/X]
= M/(2*X)
So, total number of pizzas will be = (M/X) + M/(2*X)

Time complexity: **O(T)**
Space Complexity: **O(1)**

D. ICPC to LCPC

Category: Nested if else
Problem Setter: Saiful Islam Ramim
Reviewer: Riadh Hasan
Alternate Writer: Abu Saleh

Analysis:

We will be given an input indicating the year we have traveled to . If the given year is less than 1582 then we should calculate the leap year according to "Julian Rule" which specifies , "If a year is divisible by 4 then it is a leap year". On the other hand if the given year is equal or greater than 1582 then we should calculate the leap year according to "Gregorian Rule" which specifies , "Every year that is a multiple of 4 is a leap year, unless it is a multiple of 100 that is not a multiple of 400" . And if the given year is a leap year then we have to print "**I can participate in LCPC**" otherwise print "**I have to travel back to the past**" without quotation marks .

Time complexity: **O(1)**
Space Complexity: **O(1)**

E. ICPC Seat-Plan

Category: Loop
Problem Setter: Md Galib Hossain
Reviewer: Anisur Rahman
Alternate Writer: Shimul Sutradhar

Analysis:

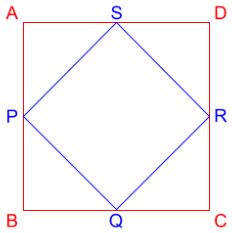
Here **N** row and every row has **M** PC's. that means the pc orientation looks like e N*M grid where every cell's value must be 1/0. The empty seats are denoted by **0.** So for the solution, you need to calculate the number of **0** in the N*M grid.
For that, we can use, a nested loop to take input and calculate the number of zeros.

Time complexity: **O(N*M)**
Space Complexity: **O(1)**

F. Squares

Category: Geometry
Problem Setter: MD. Albin Hossain
Reviewer: Saiful Islam Ramim
Alternate Writer: Siam Ahmed

Analysis:



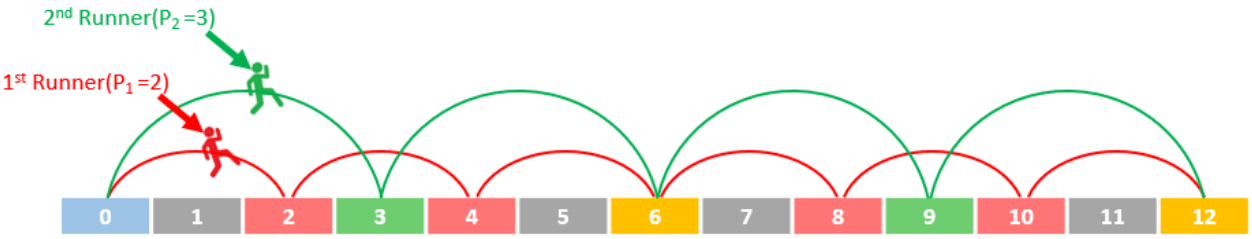
Given PQ. To find the area of the square ABCD we need to find a side AB. Then we can find the area of ABCD = AB^2.
Sides of square ABCD are equal, therefore AB = BC and P, Q are midpoints of AB, and BC respectively. So PB=BQ and AB= 2PB.
Consider the triangle △PBQ. angle ∠PBQ is 90° as ABCD is a square by definition.
We can get by **Pythagorean theorem**,
$PB^2 + BQ^2 = PQ^2$
$PB^2 + PB^2 = PQ^2$ [because PB = BQ]
$2PB^2 = PQ^2$
$4PB^2 = 2PQ^2$ [Multiply both side by 2]
$(2PB)^2 = 2PQ^2$
$AB^2 = 2PQ^2$ [because AB = 2PB]
AB^2=Area of ABCD square.

Time complexity: **O(1)**
Space Complexity: **O(1)**

G. Kotlin Heroes

Category: Number Theory
Problem Setter: Anisur Rahman
Reviewer: Abu Saleh
Alternate Writer: Abu Saleh

Analysis:



See the Figure Carefully.
Hint 1: We need a point where everyone can meet. What are the minimum point where everyone can jump?
Hint 2: Think for the number of two participants only. Say,
1st participate can jump 2 as = 2,4,**6**,8,10, ...,n*2.
2nd participate can jump 3 as= 3,**6**,9,12,15,. . . .n*3.
The minimum point will be multiplied by both as **6** this example.From this observation we can say that, this two participant can meet every multiple of **6.**
Hint 3: How do we find **6**. LCM of this two numbers.
So, find LCM of every participant jump step, and find how many **multiple** of **LCM** have upto **N.**

Total Number of meetup points of **K** participant's are : **N ÷ LCM of all participants jump step.**

Time complexity: **O(k*(log(min(lcm,p[i])))).**
Space Complexity: **O(1)**

H. So Long

Category: Data Structures
Problem Setter: Riadh Hasan
Reviewer: Shimul Shutradhar
Alternate Writer: MD. Albin Hossain
Special Thanks: Rahat Islam Srijon, Syed Ahsanul Huque Nahid

Analysis:

To solve this problem we can use a **stack data structure** as the list where we add the value at the end of the list and **pop value from the end for the list**. After that, we can simply implement the given game condition, and based on the condition we can find the winner of every round.

Time complexity: **O(T*Q)**
Space Complexity: **O(K)**

I. One Ball Many Chocolates

Category: Implementation
Problem Setter: Rahat Islam Srijon
Reviewer: Masum Bin Hossain
Alternate Writer: Saiful Islam Ramim

Analysis:

N and K are given.

N is the length and K is the length up to which the first drop can be made, so you have to make sure that the **first drop is on a cell containing a value less than or equal to K.**

You can use any power value less than or equal to N. Here the value of K is less than or equal to N and the power value is not related to neither N nor K.

So, you can simulate the process for **every possible value of K and power value** and maximize the answer. If you simulate you can see that the simulation would go at most sqrt(N). But the solution can be optimized to O(1) using a mathematical approach.

Time complexity: **O(N*K*(at most sqrt(K)))**
Space Complexity: **O(1)**

J. Freezed Standings

Category: Pre Stopper (String , Math)
Problem Setter: Abu Saleh
Reviewer: MD. Albin Hossain
Alternate Writer: Rahat Islam Srijon, Mahmud Sajjad Abeer (Alumni)

Analysis:

Simulating over the final string **T** and using some mathematics, this problem can be solved. Consider **N = 3** and **K = 3** . Take **a,b,c** as first 3 characters. So, string **S** becomes = "**abc**". To construct **T** concatenate **S** 3 times . Now, **T** = "**abcabcabc**"; After simulating over all substring we will get total of **24 distinct substrings** –

a , b, c — (3 substrings of length 1)
ab, bc, ca — (3 substrings of length 2)
abc , bca, cab — (3 substrings of length 3)
abca , bcab, cabc — (3 substrings of length 4)
abcab , bcabc, cabca — (3 substrings of length 5)
abcabc , bcabca, cabcab — (3 substrings of length 6)
abcabca , bcabcab, cabcabc — (3 substrings of length 7)
abcabcab , bcabcabc — (2 substrings of length 8)
abcabcabc — (1 substring of length 9)

1st observation: From all distinct substrings we can see that there are **3 substrings of 7 length**, **2 substrings of 8 length** and **1 substing of 9 length**. We can get a sequence from here that is the summation of first n natural numbers. As **N = 3** , so summation of first 3 natural numbers = **6** . We already find total of 6 substrings (orange color).

2nd observation: Now how many substrings are left there? There are 3 substring of all **1** to **6** lengths (blue and red color) . We can express it as **N * (NK - N)** .

3rd observation: Here some substring does not fill the requirements of repetition of at least one characters. How many substrings does not fill the requirements? From the above we see only **1** to **N (here, N=3)** lengths substrings do not fill the requirements (red color) , which is equal to **N*N** .

Finally, if we summarize all 3 observation - We will **add the 1st** and **2nd observation** answer to the final answer and **subtract the 3rd observation** answer.

Final answer equation:

= N*(N+1) / 2 + N*(NK - N) - N*N
= N*(N+1) /2 + N^2K - N^2 - N^2
= N*(N+1) /2 + N^2K - 2N^2

As we need to find the modulo of answer. We have to use Binary Exponentiation and Modular Inverse to calculate the answer.

K. A Giveaway

Category: Stopper
Problem Setter: Shimul Shutradhar
Reviewer: Abu Saleh, Anisur Rahman
Alternate Writer: Farhan Ahmed (IOI), Saiful Islam Ramim

Analysis:

For each line of product, we have to track which product has already been taken. For two lines we will take **two mask parameters**. The first one is to track if the first product is taken or not in a particular position. The second one is to track if the second product is taken or not in a particular position. Using this two-parameter run a backtrack solution will give the right answer.

The main problem with this backtrack solution is its time complexity O($n * n^n$).
Now we can do **DP in this backtrack solution**. For the DP table, **we need O(2^{n+n}) memory. Which will give a memory limit exit.**

If we observe that we don't need all bit for tracking. For 2nd product taking, we need to check if the first product is taken. So take the first product and set i'th position bit to one for the first product and for i'th position second product check if bit is one or not if the bit is one then take that product and set the bit to 2. Take the mask in 3 base number system for set bit 1 and 2.

Now do DP in this **3-base number system mask**.

Time complexity: **O($n * 3^n$)**
Space Complexity: **O (3^n)**

When You stop being afraid,

You feel good!

The End