Regional Contest Editorial

For Nigeria, Qatar, Sudan and Oman Regionals

A. Dictionary

Tags: Strings, Greedy

The input string ***S*** can be a palindrome, or not.

If it's not a palindrome, then you need to change ***0*** characters.

However, if it is a palindrome, then you need to change exactly ***1*** character to make the string a non-palindrome.

Therefore, the answer is either: ***0*** or ***1***.

B. Vowels

- If we consider the naive approach to use something like Bubble sort, whenever we find a vowel we keep swapping it with the previous non-vowels the answer will be the number of swaps we make.
- This is obviously **$O(N^2)$** complexity and will result in TLE, but if we observe this simple solution we can see that each vowel will be swapped with every non-vowels before it.
- So the answer will be to count the number of non-vowels before each vowel in our string.
- This can be done in **O(N)** complexity, by iterating over the string and keeping a counter to the number of vowels, and whenever we come across a vowel we simply add this counter to the our answer.

C. Teams

Tags: Math, Sortings

First of all, sort all the rating values, from smallest to greatest.

Iterate through all individuals, from $i = 0$ to $i = N - 3$.

If the difference between the rating of individual number $i + 2$ and the rating of individual i is smaller than S, then we will put the individuals number $i, i + 1, i + 2$ into a team, and continue the iteration from $i + 3$.

Else, then this individual cannot belong to any team.

The answer is the number of teams we formed.

D. Grid

Tags: Strings

Mike wants to make all strings equal, and can move any character to any place he wants. Therefore, we need to count for each English lowercase character: how many times it appears in all the words.

If the count of any character not divisible by ***N*** -the number of strings- that means there is going to be at least one string that will have more of this character than the other strings. This implies that the strings cannot be equal and the answer is **No**.

Otherwise, if for every character, it's count is divisible by ***N***, then all the strings can be equal and the answer is **Yes**.

E. Water and Borders

Tags: Bipartite Matching, Max flow

- Each time you choose a white cell it blocks its row and its column
- We can reduce the problem into bipartite matching between set of rows and set of columns, edges between row R and column C represents a white cell exist in grid[R][C]
- We want to know what maximum edges we can take s.t. Each row/column is selected at most once
- Maximum edges is equivalent to maximum white cells, So the answer is Maximum Cardinality Bipartite Matching(MCBM)

F. Magic Mirror

- It is clear that string T1 is only a permutation of string S1, and string T2 is a permutation of string S2, so strings T1 and T2 should have the same characters as strings S1 and S2 respectively.
- We can check those two conditions by comparing the frequency of characters between T1 and S1 and also between T2 and S2 they should both match.
- Then we can simulate the problem by using 2 nested loops where we can try to match string T1 to string S1 by replacing 2 positions and at the same time apply the same operation on string S1.
- After completing the previous step we could reach a state where strings S1 = ‘abba’, S2 = ‘abcd’, T1 = ‘abba’, T2 = ‘acbd’ we can see that we can swap the characters at positions 2 and 3 to make strings S2 and T2 equal.
- One way to solve this issue is to repeat step number 3 but this time on strings S2 and T2 where we can only swap 2 characters if there positions contain the same characters in string S1.
- If at the end string S1 is not equal to string T1 or string S2 is not equal to string T2 then there is no way to achieve the goal.
- Total complexity is **$O(N^2)$**.

G. NITD Problems Thieves

Tags: Dijkstra

- For each destination node ***u*** you want check whether there is a shortest path from ***S*** to ***u*** that goes through ***edge(A,B)***
- let ***dist(u,v)*** be the shortest path between ***u*** and ***v***
- If ***edge(A,B)*** lies on some shortest path then the sum of below values is equal to ***dist(S,u)***:
  - Shortest path from ***S*** to one of edge endpoints
  - Shortest path from the other endpoint to ***u***
  - Weight of ***edge(A,B)***
- Single source shortest path in weighted graph can be solved using Dijkstra

H. Pure Oxygen

- If we consider buying the stock in each day, we can calculate the maximum profit we can get by trying to sell it in any upcoming day and maximizing the profit.
- This can be done using 2 nested loops and whenever the profit from selling the stock is greater than the current profit we can simply update the profit and the pair ***(B, S)***.
- Total complexity $\mathbf{O(N^2)}$.

I. Happiness Pills

Tags: Dynamic Programming

- Observation: if we choose some problems to solve so the largest ***k*** **(k<=10)** problems' ***p_i*** are multiplied by largest ***k*** primes among the first **10 primes** (can be proved by contradiction), Sort elements ascending by their happinesses
- Let dp[i][t][k] be maximum possible happiness value using first ***i*** elements with remaining time ***t*** and first ***k*** primes
- Transitions will be like **0-1 knapsack** but you have to handle if there are no primes left you will multiply it by 1
- To avoid ***MLE*** you have to write it iteratively saving 2 rows only ***(i,i-1)*** [memory reduction trick]

J. Distributing Cookies

Tags: Math, Greedy

Let's denote the three values by: i, j and k, where $i < j < k$ and $i \leq \sqrt{x}$.

Now, let's iterate through all the possible values for i, from 1 to $\sqrt{x}$.

For each value of i, we want to calculate how many possible values of j do we have. Remember the constraints: $i < j$ and $j < k$. It's clear now that $i < j < \frac{x-i}{2}$ and $i+1 \leq j \leq \frac{x-i-1}{2}$. We can easily count the number of js than satisfy the equation in $O(1)$ and add them to the answer.

K. Median

Tags: Simulation, ad hoc

- You can simulate what problem needs as it will fit the time limit
- Each query you will calculate C = A + B using substr() function and sort it using sort() function
- So the total time complexity: O(Q * (|A|+|B|) * log(|A| + |B|))

L. Surprise Party!

Tags: Geometry, Math

Its guaranteed that the parents will always leave home before the planner arrives, we just need to check if the planner will leave before the parents arrive back home:

$$t_{departure} \leq t_{arrival}$$

It's obvious that:

$$t_{departure} = Start_{prep} + Duration_{prep}$$

$$t_{arrival} = (Start_{event} + Duration_{event}) + t_{car}$$

To calculate the time spent in the car: $t_{car} = \frac{D}{S}$, where:

$$D = \sqrt{(X_{home} - X_{event})^2 + (Y_{home} - Y_{event})^2}$$

M. Numbers

Tags: Math

If $Y - X = 3$, then we can add 3 to X (1 step), or add 2 then 1 (2 steps) or add three 1s (3 steps). It's obvious that adding the larger possible value is always optimal. Therefore:

While $Y - X \geq 3$, then it's optimal to add 3 to X.

If $Y - X = 2$, then it's optimal to add 2 to X.

If $Y - X = 1$, then it's optimal to add 1 to X.

If $Y - X = 0$, then we are add the number of 3s, 2s and 1s that we added to X, and that becomes the answer.