Day 0 Contest Analysis, Division AB

July 6, 2019

Filipp Rukhovich

Moscow Workshops ICPC Summer 2019

J. Jailbreak

Two prisoners are in cells t_1 and t_2 of $h \times w$-prison with walls, doors and free cells. We are to open as small number of walls as possible to give prisoners a way to go out. What is a small number?

Consider space outside a prison as one big empty cell s. Consider optimal pair of paths of prisoners. Let p be a first cell of second path which lie on first path. Obviously, it's optimal for second prisoner to go the same path after p as first one.

Iterate over all possible p-s (in fact, all non-wall cells). For each p, we should find shortest (in sense of number of doors) paths for p to t_1, t_2 and s and summarize it (an add 1 if p is a wall); the answer is minimal possible such a sum. To do it, find distances between t_1, t_2, s in $O(hw)$ time and all cells in using 0-1-BFS and then find an answer if $O(hw)$ time.

D. Destination Unknown

Given an undirected weighted graph with positive weights, and also vertices $s, g, h, v_1, v_2, \ldots, v_t$. We are to find for each v_i, $1 \leqslant i \leqslant v_t$ whether edge (g, h) lies on some shortest path between s and v_i or not.

Obviously, edge (g, h) of weight w lies on one of such shortest paths if and only if $dist(s, g) + w + dist(h, v_i) = dist(s, v_i)$ or $dist(s, h) + w + dist(g, v_i) = dist(s, v_i)$. Find shortest distances for s, g, h to all other vertices using Dijkstra's algorithm three times. After that, we can check each v_i in constant time.

The total complexity of the solution is $O(n^2)$.

G. Getting Through

Given a two-dimensional long straight corridor and n circles with centers inside it. We are to go through the corridor without intersecting with circles. And we are the circle too. What maximal radius could we have?

G. Getting Through

Use binary search by answer. So, suppose that we are a circle with radius R. Is it possible for us to negotiate the corridor?

To reveal that, understand where is the center of us cannot occur without intersections with circles and walls. To do that, move walls R units towards each other and increase radii of all circles by R.

After that, build a undirected graph with walls and circles as vertices; two vertices are connected by an edge if and only if correponding objects (circles and/or walls) intersect each other.

If two walls are connected by a path, then obviously, we cannot go through the corridor. Otherwise, we can do it from down to up starting from left wall and using “left hand rule”.

G. Getting Through

The complexity of the solution is $O(n^2 * log(MAX\_COORD/ABS\_ACCURACY))$; it can be done in $O(n^2 log\ n)$ if note that the answer is half of distance between some pair of objects or zero.

H. Hidden Camera

Given a convex polygon $A_0A_1A_2 \ldots A_{n-1}$ and an $\pi/2$-angle with vertex as a midpoint of on first side such that angles between sides of angle and A_0A_1 is $\pi/4$. We are to find an area of intersection of the polygon and the angle.

This intersection is a convex polygon vertices of which can be found by iterating over all sides and adding points of intersections of sides of polygon and rays to some vector v, so do vertices of polygon which lie in angle. After iterating, v contains a needed intersection, and we are just to find its area.

The complexity of the solution is $O(n)$.

B. Bribe

Given n henchmen of some criminal network, we can give bribes to some of them. If we give a bribe to some henchman, he will cooperate with us with some fixed probability or, in opposite case, take money and run away. We have a fixed amount of money and need to cooperate with c henchmen. If we work optimally, what will be a probability of success?

Use dynamic programming on subsets and submasks. Let $dp[mask][k]$, $0 \leqslant mask < 2^n$, $0 \leqslant k \leqslant n$ is a probability of success in assumption that we've already given bribes to subset of henchmen coded by n-bit number $mask$, and k of them are our cooperators.

B. Bribe

If $k \leqslant c$ then $dp[mask][k] = 1$; if $k < c$ and $mask = 2^n - 1$, then $dp[mask][k] = 0$.

Otherwise, we should try to bribe at least one more henchman. From $mask$, we can get what amount of money we have, and iterate over all henchmen we are able to bribe. In case of bribing such henchman j, we can calculate a probability of success as $p_j * dp[mask|(1 << j)][k+1] + (1 - p_j) * dp[mask|(1 << j)][k]$, for p_j as a probability of cooperating of j-th henchmen with us. Then, $dp[mask][k]$ can be found as maximum among all such henchmen.

The answer is $dp[0][0]$. The total complexity of the solution is $O(2^n * c * n)$.

C. Code

Given n encrypted messages M_1, M_2, ..., M_n, and decrypted message D which is a decrypted version of one of M_i, $1 \leqslant i \leqslant n$, we don't know what. The method of encrypting is replace each letter to another letter (may be the same) in such a way that same letters should be replaced by the same, and different letters should be replaced by another one. We have one more encrypted message X and are to decrypt as much letters as we can, directly or deductively.

C. Code

For each $i \in [1, n]$, we can easily check whether M_i can be an encrypted D or not. If so, then for some letters we can find its decrypted version. Important detail is that if message contains 25 different letters then we can decrypt all 26 letters by method of exclusion. Otherwise, each letter which doesn't occur in M_i cannot be decrypt uniquely and should be replaced by '?' in case of occurrence it in X.

The complexity of the solution is $O(|M_1| + |M_2| + \ldots + |M_n| + |D| + |X|)$.

E. Encoded Coordinates

Let P be a prime number; consider the following function modulo P. Then for tuple of integers (A, B, C, K, N, h), $0 \leqslant A, B, C, h, K < P$, $1 \leqslant N \leqslant 10^9$, we calculate integer sequences F, G, H in the following way:

- $F(1) = A$, $G(1) = B$, $H(1) = C$, $H(0) = h$;
- $\forall n \in \mathbb{Z}_{\geqslant 2}$, $F(n) = G(n-1) + H(n-1)$;
- $\forall n \in \mathbb{Z}_{\geqslant 2}$, $G(n) = K * F(n-1) + H(n-2)$;
- $\forall n \in \mathbb{Z}_{\geqslant 2}$, $H(n) = F(n-1) + K * G(n-1)$.

The result value of the function is $F(N)$ modulo P. Let it be function $D(P, A, B, C, K, N, H)$.

Given numbers $P, A_x, B_x, C_x, K_x, N_x, A_y, B_y, C_y, K_y, N_y$ and also $x = D(P, A_x, B_x, C_x, K_x, N_x, h)$ for some unknown parameter h. We are to find $y = D(P, A_y, B_y, C_y, K_y, N_y, h)$; if we cannot determine y uniquely then we should print "UNKNOWN".

E. Encoded Coordinates

Let $v[i] = \begin{pmatrix} F(i) \\ G(i) \\ H(i) \\ H(i-1) \end{pmatrix}$, $i \in \mathbb{Z}_+$. Then for any i, $v[i+1] = Mv[i]$, $M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ K & 0 & 0 & 1 \\ 1 & k & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$. Due to associativity of matrix multiplication, $v[n] = M^{n-1}v[1]$; $M^{n-1} =: M'$ can be calculated in $O(log\ n)$ time using binary exponentation.

E. Encoded Coordinates

Notice that $v[i] = v'[i] + h * v''[i]$, for $v'[i] = M' \begin{pmatrix} A \\ B \\ C \\ 0 \end{pmatrix}$,

$v''[i] = M' \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

Calculate $v'[N]$ and $v''[N]$ with A, B, C, K, N as A_x, B_x, C_x, K_x, N_x; denote it as $v'_x[N_x]$ and $v''_x[N_x]$. We know that first coordinate of $v[i]$ should be x; then we have an equation $x = x' + h * x''$; here x' and x'' are known first coordinates $v'_x[N_x]$ and $v''_x[N_y]$.

E. Encoded Coordinates

If $x'' \neq 0$ then h can be determined uniquely; we can do it in $O(P)$ (or $O(log\ P)$ using Fermat's little theorem) and then calculate y.

If $x'' = 0$ and $x' \neq x$ then the answer is "UNKNOWN" because of contradiction.

If $x'' = 0$ and $x' = x$ then h can be arbitrary remainder by modulo P; but it's possible that y does not depend on h. To check that, calculate $v''_y[N_y] = M^{N_y-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$. If first coordinate of $v''_y[N_y]$ is 0 then calculate $v_y[N_y]$ with $h = 0$; otherwise, the answer is "UNKNOWN".

The total complexity of the solution is $O(P + log\ N_x + log\ N_y)$.

A. Administrative Troubles

The spy-car rental company has a collection of cars. Each spy can pick any car up and then return it. During the driving, one of more accidents may occur. For each car, their catalog prices, costs of picking the car up and costs per kilometer driven are known, so do list of events of picking cars, returning it and accidents with their severities, in percentage of catalog price. For each spy, we are to check whether data about the spy is consistent or not and find total car cost he needs to pay.

A. Administrative Troubles

To solve the problem, create two maps, *carmap* for cars and *spmap* for spies. In *carmap*, we'll store all prices for each type of car in a class.

After reading first $N + 1$ lines of input and building the *carmap*, perform the events, one-by-one. For each spy, we'll store his current *state* which can be *FREE_OF_CAR*, *DRIVEN*, and *INCONSISTENT*, and also type of driven car (for *DRIVEN* state) and current cost he needs to pay.

Using these maps (or, better, unordered_maps), we are to perform each event easily; the only thing we need to do is to describe all changes of states and corresponding changes of costs accurately.

F. Flying Safety

Given undirected connected graph $G = (V, E)$ with n vertices and m edges, we are to find a minimal possible size of subset $E' \subseteq E$, such that $G' = (V, E')$ is still connected.

As it is known from graph theory, the answer is $n - 1$:))).

I.Incognito

Given n objects; each of them has a number and a category. We are to find a number of taking one or more objects in such a way that there are no two taken objects of one category.

For each category, find a number of objects of this category using map or unordered_map. Then, then answer is $(a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1)$; here $a_1, a_2, \ldots, a_k$ are numbers in map.