7A Contest

July 13, 2019

Ilya Zban

Discover Vladivostok 2019

A. Playing with an Array

In this problem we are given an array and we need to perform two operations:

- change value a_i to value v;
- count number of integers x that occur an odd number of times in segment $[l; r]$.

A. Playing with an Array

The key observation is that all values x are bounded by small number $C = 1000$.

We can store given array in segment tree, and maintain in each vertex a bitset with an oddity of number of occurences for all numbers $1, \ldots, 1000$. Bitset in vertex can be obtained as xor of bitsets of its children.

Building of a segment tree works in $\mathcal{O}(NC/w)$, and all queries can be performed in $\mathcal{O}(Q \log nC/w)$, where w is a size of word (64 bit).

B. Airlines 3

In this problem you need to construct an undirected complete graph with colored edges and without an odd cycles of single color. Also, for each two vertices that have two edges with an equal colors there should be a path of length 2 with this color between these two vertices.

You need to minimize a number of used colors in a graph.

B. Airlines 3

We can prove by induction, that for graph with n vertices we need to color the edges of graph in $n-1$ colors.

We should use the lemma (which can again be proved by induction) that each one-colored subgraph is a complete bipartite graph.

We prove that for each graph there is a one-colored cut, and as this cut is a complete bipartite graph, we reduce the problem to two independent problems, and this leads to $f(n) = 1 + f(k) + f(n-k-1) = n-1$.

To make the induction step that each graph has a one-colored cut we can see that after deletion of each vertex there would be one-colored cut on a graph with $n-1$ vertices, and two these cuts would have the same color. These cuts cover all vertices in a graph, so there is a cut in graph on n vertices.

B. Airlines 3

So, we have proved that we need to use at least $n - 1$ colors.

An example of appropriate graph is a graph with edge $i \leftrightarrow j$ $(i < j)$ colored in color i. We can see that this graph satisfies all required conditions and have the minimal possible number of edges.

C. Card Game

In this problem we have n players and the skill of each player has a uniform distribution on $[l_i; r_i]$, and we need to know all the probabilities of i-th player to be the strongest.

C. Card Game

Let's define the set of all given l_i, r_i as $x_1 < x_2 < \ldots < x_k$. For each player i we can take all intervals $[x_j; x_{j+1}] \in [l_i; r_i]$ and calculate the probability that i-th participant will be the strongest, and his skill will be in $[x_j; x_{j+1}]$.

C. Card Game

Let's define the set of all given l_i, r_i as $x_1 < x_2 < \ldots < x_k$. For each player i we can take all intervals $[x_j; x_{j+1}] \in [l_i; r_i]$ and calculate the probability that i-th participant will be the strongest, and his skill will be in $[x_j; x_{j+1}]$.

Obviously, there should be no participant with $l_t > x_{j+1}$. All the participants with $r_t <= x_j$ are also not interesting, they can't have stronger skill. For each other participant we should multiply the probabilities that they get skills less than x:

$$\frac{x_{j+1}-x_j}{r_i-l_i} \int_{x_j}^{x_{j+1}} dx \prod_{t=1, l_t \leqslant x_j, r_t \geqslant x_{j+1}}^{n} \frac{x-l_t}{r_t-l_t}.$$

C. Card Game

We can compute the answer with this method fast enough (in $\mathcal{O}(n^3)$ with divide and conquer), but it won't get accepted due to very big precision errors.

C. Card Game

We can compute the answer with this method fast enough (in $\mathcal{O}(n^3)$ with divide and conquer), but it won't get accepted due to very big precision errors.

The problem is in multiplication of $x - l_t$ — there are a lot of summands with different signs. We should change the integration order to multiply only positive polynomials. For example, we can reverse the integration order and get something like

$$\frac{x_{j+1}-x_j}{r_i-l_i} \int_0^{x_{j+1}-x_j} dz \prod_{t=1, l_t \leqslant x_j, r_t \geqslant x_{j+1}}^{n} \frac{x_j - l_t + z}{r_t - l_t}.$$

In this method there is much smaller precision error and it should get accepted.

D. Castle Site

In this problem we are given a triangle A, B, C and we need to find a point P that minimizes $f(P) = 6|A - P| + 3|B - P| + 2|C - P|$.

D. Castle Site

In this problem we are given a triangle A, B, C and we need to find a point P that minimizes $f(P) = 6|A - P| + 3|B - P| + 2|C - P|$.

The solution is just to print a point A, as $3 + 2 < 6$, and it is always optimal to move the resulting point to point A.

D. Castle Site

In this problem we are given a triangle A, B, C and we need to find a point P that minimizes $f(P) = 6|A - P| + 3|B - P| + 2|C - P|$.

The solution is just to print a point A, as $3 + 2 < 6$, and it is always optimal to move the resulting point to point A.

Also, you can use the two nested ternary searches by x and y coordinates to find a minimum of $f(x, y)$.

$f(x, y)$ is convex, as it is a sum of three convex functions.

E. Counting Rhyme

In this problem we were given a description of algorithm of choosing k elements from n by successive elimination and we need to determine cyclic shift x from a set of chosen elements.

E. Counting Rhyme

In this problem we were given a description of algorithm of choosing k elements from n by successive elimination and we need to determine cyclic shift x from a set of chosen elements.

We can see that the required cyclic shift x is less than $l = lcm(1, 2, 3, \ldots, n)$, because two cyclic shifts that have same remainder modulo l are effectively indistinguishable in our problem.

E. Counting Rhyme

In this problem we were given a description of algorithm of choosing k elements from n by successive elimination and we need to determine cyclic shift x from a set of chosen elements.

We can see that the required cyclic shift x is less than $l = lcm(1, 2, 3, \ldots, n)$, because two cyclic shifts that have same remainder modulo l are effectively indistinguishable in our problem.

So, the solution is to bruteforce all possible cyclic shifts $x \leqslant l$ and check if this cyclic shift gives the required set of chosen children. As $n \leqslant 16$, $l \leqslant 720\,720$ is small enough.

F. Divisibility by 11

In this problem we need to rearrange the digits of number x to get a largest possible number divisible by 11.

F. Divisibility by 11

Usually when we want to construct the lexicographically maximal object, we try to put the largest symbols at the beginning of the answer.

F. Divisibility by 11

Usually when we want to construct the lexicographically maximal object, we try to put the largest symbols at the beginning of the answer.

The remainder of a number $\overline{a_{n-1}a_{n-2}\dots a_0}$ modulo 11 is $\sum_{i=0}^{n-1}(-1)^i a_i$, so the digits at the even position contribute positive value to the remainder, and the digits on the odd position contribute the negative value.

F. Divisibility by 11

Let's sort the digits of x in non-increasing order and calculate the following dynamic programming:

$dp_{i,j,k}$ — is it possible to take digits from positions not less than i so that there are exactly k digits with coefficient $+1$, $n - i - k$ digits with coefficient -1, and the remainder of their sum modulo 11 is equal to j.

F. Divisibility by 11

Let's sort the digits of x in non-increasing order and calculate the following dynamic programming:

$dp_{i,j,k}$ — is it possible to take digits from positions not less than i so that there are exactly k digits with coefficient $+1$, $n - i - k$ digits with coefficient -1, and the remainder of their sum modulo 11 is equal to j.

$dp_{i,j,k} = dp_{i+1,j-a'_i,k-1}$ or $dp_{i+1,j+a'_i,k}$.

F. Divisibility by 11

Let's sort the digits of x in non-increasing order and calculate the following dynamic programming:

$dp_{i,j,k}$ — is it possible to take digits from positions not less than i so that there are exactly k digits with coefficient $+1$, $n-i-k$ digits with coefficient -1, and the remainder of their sum modulo 11 is equal to j.

$dp_{i,j,k} = dp_{i+1,j-a'_i,k-1}$ or $dp_{i+1,j+a'_i,k}$.

Now we can use a greedy algorithm to construct a number. If we have a partition of all digits into two sets — digits on the odd and on the even position, than we should place them in non-increasing order on each position.

F. Divisibility by 11

We can construct the number by placing the digits to odd and even positions simultaneously. We maintain the current number, the remainder r of current number modulo 11, take the maximal possible digit, and try to place it to the leftmost possible even or odd position (which one comes first).

F. Divisibility by 11

We can construct the number by placing the digits to odd and even positions simultaneously. We maintain the current number, the remainder r of current number modulo 11, take the maximal possible digit, and try to place it to the leftmost possible even or odd position (which one comes first).

How to check if we can construct the suffix of the number? We can use the precalculated values of dynamic programming: if we have k unused even positions, and the current remainder modulo 11 is r, we need to check if $dp_{i+1,11-r,k} = 1$.

The solution works in $\mathcal{O}(11n^2)$, where n is the length of a number.

G. Extremal Permutations 2

In this problem we need to find lexicographically next extremal permutation, where for each i either $p_i < p_{i-1}, p_i < p_{i+1}$, or $p_i > p_{i-1}, p_i > p_{i+1}$.

G. Extremal Permutations 2

The standard method to generate a lexicographically next permutation is to fix the rightmost first different element such that it is possible to construct the suffix with given conditions.

G. Extremal Permutations 2

The standard method to generate a lexicographically next permutation is to fix the rightmost first different element such that it is possible to construct the suffix with given conditions.

We can see that the lexicographically minimal extremal permutation is the $1, 3, 2, 5, 4, \ldots$.

G. Extremal Permutations 2

Suppose the first different element is at the position i. There are few cases.

Let the lowest value bigger than a_i between all a_j with $j > i$ is x. We will either put x at i-th position, or the first different element will at the left of i-th position.

- if $i > 1$ and $a_{i-1} > a_i$, than a_{i-1} should be bigger than x. Otherwise, we should move i.
- if $i > 0$ and $a_{i-1} > x$, than a_i should be a local minimum. There should be an integer $y > x$ among a_j with $j > i$, and we will put such minimal y at the position $i + 1$.

If these conditions are true, we can build a suffix of permutation by assigning $a_i := x$ and the suffix greedily in a way like $1, 3, 2, 5, 4, \ldots$. It is useful to prove such solution by bruteforce before submit.

H. Guests

In this problem we have an undirected graph and we need to find a largest subgraph such that for every vertex in this subgraph there are at least k other vertices that are incident to this vertex, and at least k other vertices that are not incident to this vertex.

H. Guests

We need to find a maximal subgraph. Why can't we take the whole graph? Either we can take the whole graph, or there is a vertex v such that there are less than k incident (or not incident) vertices to this vertex. But in this case this vertex can't be in the resulting subgraph anyway, ans we should delete this vertex from a subgraph.

H. Guests

We need to find a maximal subgraph. Why can't we take the whole graph? Either we can take the whole graph, or there is a vertex v such that there are less than k incident (or not incident) vertices to this vertex. But in this case this vertex can't be in the resulting subgraph anyway, ans we should delete this vertex from a subgraph.

We need to iterate the process of removing the definitely bad vertices from a graph, until the graph becomes empty or we get a valid graph that satisfies all conditions. As we dropped only useless vertices, the resulting subgraph would be maximal possible.

H. Guests

We need to find a maximal subgraph. Why can't we take the whole graph? Either we can take the whole graph, or there is a vertex v such that there are less than k incident (or not incident) vertices to this vertex. But in this case this vertex can't be in the resulting subgraph anyway, ans we should delete this vertex from a subgraph.

We need to iterate the process of removing the definitely bad vertices from a graph, until the graph becomes empty or we get a valid graph that satisfies all conditions. As we dropped only useless vertices, the resulting subgraph would be maximal possible.

To make this solution work in $\mathcal{O}(n^2)$ time we need to use a queue to maintain the useless vertices, and recalculate the degrees of neighbours after deletion of a vertice.

I. Lumberjacks

In this problem we are given an oriented forest, and we need to determine who is going to win in inverted green hackenbush, where player don't want to make the last turn.

I. Lumberjacks

It is known that green hackenbush is equivalent to nim by colon principle.

Nim, where players want to lose, is known as misere nim. It can be shown that the winning positions in a misere nim are almost the same as in usual nim, but if sizes of all piles are equal to one, the winning state is inverted.

I. Lumberjacks

The solution of the problem is to calculate a Sprague-Grundy values of each tree, determine a winner of the game and find a valid move.

- If all trees have the SG-values equal to one, then if there is an odd number of trees, second player is goind to win. Otherwise, the first player is going to win and we should find any way to cut the edge in any tree such that this tree will change its SG-value. It can be done by DFS with argument "we want to get SG-value x in this vertex".
- If there is a tree with SG-value bigger than one, we should do exactly the same algorithm, but we should avoid the case with going into the state with all ones.

J. Pilgrims

In this problem we are given a grid with $a_{i,j}$ pieces of food in a cell (i, j), and we need to choose a minimal possible number of right-down paths from $(1, 1)$ to (n, m), so there are at least $a_{i,j}$ paths that contain cell (i, j).

J. Pilgrims

In this problem we are given a grid with $a_{i,j}$ pieces of food in a cell (i, j), and we need to choose a minimal possible number of right-down paths from $(1, 1)$ to (n, m), so there are at least $a_{i,j}$ paths that contain cell (i, j).

We can look at all pieces of food as a partial ordered set, $x < y$ if there is a path from x to y. We need to find a partition of this set to a minimal number of chains.

J. Pilgrims

In this problem we are given a grid with $a_{i,j}$ pieces of food in a cell (i, j), and we need to choose a minimal possible number of right-down paths from $(1, 1)$ to (n, m), so there are at least $a_{i,j}$ paths that contain cell (i, j).

We can look at all pieces of food as a partial ordered set, $x < y$ if there is a path from x to y. We need to find a partition of this set to a minimal number of chains.

By Dilworth's theorem minimal number of covering chains in poset is equal to maximal antichain in this poset. In our problem we need to find a maximal possible cells (i, j) so there is no path between any of them. It can be done by following dynamic programming:

$$b_{i,j} = max(b_{i-1,j}, b_{i,j-1}, a_{i,j} + b_{i-1,j-1})$$

, and the answer is a $b_{1,m}$.

K. Skis and Skiers

In this problem we are given a two arrays a and b, and we need to find a permutation π to minimize $\sum_{i=1}^{n} |a_i - b_{\pi_i}|$.

K. Skis and Skiers

In this problem we are given a two arrays a and b, and we need to find a permutation π to minimize $\sum_{i=1}^{n} |a_i - b_{\pi_i}|$.

We should match the minimal number in a with the minimal number in b, the second minimal in a with the second minimal in b, and so on. Answer is $\sum_{i=1}^{n} |a'_i - b'_i|$, where a' and b' are the sorted copies of a and b.

Proof — if a'_1 is matched with b'_j, and b'_1 is matched with a'_i, we can make a matching between a'_1 and b'_1, a'_i and b'_j, and it will be not worse.

L. One More Round on the Spiral...

In this problem you are given a number a in cell $(0, 0)$, and you need to find a maximal possible number in cell (x, y) if an infinite grid contains a described spiral.

L. One More Round on the Spiral...

There are 8 possible spirals (for a fixed position of 1 there are 4 ways to choose position of 2, and 2 ways to choose position of 3). The main idea is that we should find a centers of all possible 8 spirals, so that the number a would be in cell $(0, 0)$, and take maximum value in cell (x, y) between these spirals.

For each of 8 possible spirals we can find a position of a if the center in $(0, 0)$, and that gives a real position of center (x_c, y_c).

After that we should compute the value in cell $(x - x_c, y - y_c)$ for a spiral with center $(0, 0)$.

To simplify the code we can use the fact that for standard spiral in cell (x, x) there is a number $(2x + 1)^2$, and we can rotate coordinates to avoid code copying.