Day 4 Contest Analysis, Division AB

July 11, 2019

Filipp Rukhovich

Discover Vladivostok 2019

A. Survival Route

Given a circle *circCr* of radius r with center C and two points A to B on the sphere with center O and radius R. We are to find a length of shortest path between A and B in assumption that we cannot move inside the circle.

Suppose without loss of generality that $R = 1$, and O is the origin. Let *circAB* be circle with center O which is containing A and B. If $A = -B$ then the answer is π; otherwise, this circle is defined uniquely.

A. Survival Route

As it is known, the shortest path between two points on the sphere is a length l of a shortest arc α_{AB} of $circAB$ between A and B. If the distance between this arc and C is not less then R then the arc is the answer and can be found as $arccos((A, B))$ with (v_1, v_2) as scalar product of vectors v_1 and v_2.

To find this distance, find perpendiculars from C to $circAB$. If C is a pole with respect to $circAB$ (i.e. $[A, B] \in \{-C, C\}$) then answer is l; otherwise, the bases of a perpendicular are points of intersection of $circAB$ and a plane containing C and perpendicular to a plane OAB. If one of these bases is on α_{AB} then the needed distance is a distance between this base and C; otherwise, the answer to the problem is l.

A. Survival Route

Suppose that I is not the answer. Then, to achieve the goal by shortest path we should:

- build one of tangent lines (on the sphere, lines are circles with center in O) from A to *circCr* with A_t as a point of tangency;
- build one of tangent lines from B to *circCr* with B_t as point of tangency;
- say that the shortest path is to achieve A_t from A "straightforwardly" (in a sense of a sphere), then achieve B_t moving on the flat circle which is the bound of forbidden zone and the achieve B straightforwardly.

A. Survival Route

To find a tangent line from A, build a plane AOC and rotate it on angle r around line AO. Then, line is an intersection of this plane and the sphere, and A_t is one of the bases of perpendicular from C to this circle. Not that there are two possible A_t-s, so do B_t-s; then, it is four possible variants of path, and one should choose the optimal one.

The complexity of the solution is $O(1)$.

E. Pea-city

Given n points on Euclidean plane, we are to cover them by a rectangle of a minimal possible area.

First of all, find a convex hull of a given set; obviously, it's necessary and sufficiently for us to cover this convex polygon.

Secondly, we can get from "method of compression" that in the optimal solution, four sides of covering rectangle lie on four lines tangent to some four points; two of these tangent lines are parallel, and two other are perpendicular to first two.

E. Pea-city

All such fours of vertices can be found in $O(n)$ using "method of four pointers". For each such four, the polar angle of direction of one of tangent lines lie in some segment; on can find this segment and then calculate the answer checking the ends of the segment and also divide the segment to two parts and use a ternary search on each of them (for $\alpha \in [\alpha_1, \alpha_2]$, the area will be $ABcos(\alpha + \phi_A)cos(\alpha + \phi_B)$, so on each segment of length less than $\pi/2$ area will be unimodular).

The complexity of the solution is $O(n * (log\ n + log(MAX\_COORD/PRECISION)))$.

K. Dividing an orange

For convenience of notation, we'll use a bit alternative version of the statement.

n people divide k oranges in the following way. Each person has a number from 1 to n. Person n supposes a variant who and how many oranges get. Afterwards all people vote "for" or "against". If at least half of the people vote "for" then the decision is accepted; otherwise n-th person is ostracised, and the procedure is repeated for $n - 1$ people.

The question is: given n, k and m numbers a_1, a_2, ..., a_m, find a minimum and maximum number or oranges a_i-th, $1 \leqslant i \leqslant m$, person can get (or -1 -1 if the person will be definitely ostracised) if the all "play optimally"; each person wants to maximize its own number of oranges and thinks that no oranges is better than ostracizion. Between the cases with equal numbers, each person wants to ostracized a maximum number of persons.

K. Dividing an orange

Fix some k and try to understand the disposition for any n from 1 to ∞. For each n, the result will be written in format $res_1, res_2, \ldots, res_n$; here res_i is a number of segment of numbers of oranges i-th person can get if there are n persons or _ if i-th person will be ostracized.

For $n = 1$ it will be "k", for $n = 2$ — "0 k". For $n = 3$, 3rd needs to have at least one more voice from first. 2nd will vote "for" only if 3rd gives him $k + 1$ orange or more which is impossible; but for 1st only one orange is enough to vote "for". Then, for $n = 3$ the disposition is "1 0 k-1".

For $n = 4$, 4th needs in one more voice and he can "buy" it from 2nd for 1 orange, so it will be "0 1 0 k-1".

K. Dividing an orange

For $n = 5$, 5th need two more voices, and he has exactly one way to get them for 2 oranges; then, the disposition will be "1 0 1 0 k-2". Repeating the reasoning, we will get the following dispositions:

$n = 6$: 0 1 0 1 0 k-2

$n = 7$: 1 0 1 0 1 0 k-3

$n = 8$: 0 1 0 1 0 1 0 k-3

...

$n = 2k$: 0 1 0 1 0 1 ... 0 1

$n = 2k + 1$: 1 0 1 0 1 0 ... 1 0 0 .

K. Dividing an orange

For $n = 2k + 2$, $2k + 2$-th needs to get $k + 1$ "for" including himself; he can "buy" k votes but in this case, he can choose k "for"-ers for 1 orange from $k + 1$ persons. Som the disposition for $n = 2k + 2$ will be:

0 [0, 1] 0 [0, 1] ... 0 [0, 1] [0, 1] 0.

As we can see, first $2k + 2$ persons will never be ostracized for $n \geqslant 2k + 2$; it means that each of these persons will for "against" suggestion with 0 oranges for himself. Then, for $n = 2k + 3$, any suggestion of $2k + 3$-th persion will get at least $k + 2$ votes "against", and it will be ostracized:

$n = 2k + 3$: 0 [0, 1] 0 [0, 1] ... 0 [0, 1] [0, 1] 0 -1.

K. Dividing an orange

It means that if $n = 2k + 4$ then $2k + 3$-th will vote "for" any suggestion of $2k + 4$; in other words, price of his voice will be 0. It gives $2k + 4$-th 2 free-of-charge votes "for", and k more votes he can buy from any of the remaining persons (because 1 is better than [0, 1] or 0). Then, the disposition will be

$n = 2k + 4$: [0, 1] [0, 1] [0, 1] . . . [0, 1] 0 0.

For $n > 2k + 4$, first $2k + 4$ persons will be definitely not ostracized; then, they will give at least $k + 4$ votes "against". So, $2k + 5$-th, $2k + 6$-th, $2k + 7$-th will be ostracized:

$n = 2k + 5$: [0, 1] [0, 1] [0, 1] . . . [0, 1] 0 0 -1

$n = 2k + 6$: [0, 1] [0, 1] [0, 1] . . . [0, 1] 0 0 -1 -1

$n = 2k + 7$: [0, 1] [0, 1] [0, 1] . . . [0, 1] 0 0 -1 -1 -1

K. Dividing an orange

But they will be ostracied only if $n \leqslant 2k + 7$; if $n = 2k + 8$, then these three will vote "for" any suggestion, and the disposition will be:

$n = 2k + 8$: [0, 1] [0, 1] . . . (2k+4 times) . . . [0, 1] 0 0 0 0

Analogically, if $2k + 8 < n < 2k + 16$, people from $2k + 9$th to $2k + 15$-th will be ostracized but for $n = 2k + 16$, last eight persons gets 0, and others — [0, 1]. Repeating this reasoning, we can can an answer for any n, k, a_i in $O(logn)$, so the total complexity of the solution is $O(Q\ log\ n)$.

C. Chocolate Frogs

Given convex polygon with n vertices, we are to find a number of ways to split it into some number of pieces by non-intersecting diagonals in such a way that exactly k of them are triangles.

Let $dp[i][j][flag]$, $3 \leqslant i \leqslant n$, $0 \leqslant j \leqslant i-2$, $0 \leqslant flag \leqslant 1$, be a number of way to split convex i-gon into exactly j triangular faces, s.t. edge connecting vertices 1 and n lies in face which is triangular if $flag = 1$ and non-triangular if $flag = 0$.

C. Chocolate Frogs

Suppose edge $(1, i)$ lies in triangular face. Iterate over third vertex v of the face. Then, we have v-gon and $i - v + 1$-gon (cases $v = 2, n - 1$ are left to the reader) and should get $j - 1$ triangular faces from them; $j' \in [0, j - 1]$ should be got from first one, and $j' - 1 - k$ to second one, so we should add $dp[i'][j'] * dp[i - i' + 1][j - 1 - j']$ for each such j' (and for each v).

Suppose now that edge $(1, i)$ lies in non-triangular face. Iterate over vertex v which is a second vertex connected with 1 in the face. Suppose for simplicity that $v \in [4, i - 3]$ (other cases can be perform by the same way). Then, $j' \in [0, j]$ triangles from v-gonal part can be got by $dp[v][j'][0] + dp[v][j'][1]$ ways, and remaining $j - j'$ from the $i - v + 1$-gonal part - by $dp[i - v + 1][j - j'][0]$ parts.

C. Chocolate Frogs

The answer to the problem can be found as $dp[n][k][0] + dp[n][k][1]$.

The complexity of the solution is $O(n^4)$; but in fact it's "$O(n^4/24)$", so it has a change to be accepted in case of being accurate in details of implementation (less number of taking by modulo, using array instaed of vectors etc.).

Also it's possible to precalculate all possible answer, store them into some compressed way and solve a problem into $O(1)$:)

F. Beautiful sums

The beauty index of some integer x is a number of ways to represent x as a sum of consecutive positive integer,

Given $n \leqslant 10^5$, we are to find the smallest integer for given beauty index n.

First of all one can easily see that if n is represented as sum of k consecutive numbers then:

- if k is odd then k is a divisor of n, and n/k is a middle of the sequence;
- if k is even then $k/2$ is a divisor of n, and $2n/k$ is odd sum of two elements in the middle of the sequence.

Based on these facts, one can calculate beauty indexes for first $10^6 - 10^7$ elements and reveal a "magic" fact: for any x, its beauty index is equal to number of odd division of x.

F. Beautiful sums

Using this fact, we are to find a minimal odd number which has exactly n divisors. This number can be represented as $p_1^{a_1-1} p_2^{a_2-1} \dots p_z^{a_z-1}$; here, $p_1 < p_2 < \dots p_z$ form a sequence of first z odd prime numbers (i.e., 3,5,7,11,13,17 etc), $a_1 \geqslant a_2 \geqslant a_3 \geqslant \dots \geqslant a_z \geqslant 2$ are integers, s.t. $a_1 a_2 \dots a_z = n$. Generate all such a sequences (a_i) using "brute-force" recursive algorithm; to compare two numbers, we can just compare their logarithms.

D. LWDB

Given a weighted tree with n vertices and positive weights. We are to perform the following queries:

1. given vertex v, distance d and color c, paint all tree vertices at the distance not exceeding d from the vertices v in color c;
2. given vertex v, return its color.

D. LWDB

Build centroid decomposition (CD) for given tree; also, in each subtree of CD, calculate distances from centroid to all other vertices. The idea is that we need to do the following actions to perform the query of first type with parameters v, d, c:

- consider all $\log n$ subtrees containing v;
- for each such a subtree with centroid u, store in this subtree, we want to paint all vertices with distances no more than $d - dist(u, v)$ should be colored in color c.

D. LWDB

For each subtree, all queries of such a type we will store in a stack in such a way that distances should be strictly decreasing from bottom to top. Then, if new query (of new type) with distance d' and color c' appears, then we remove all queries with distances no more than d' from the stack (all vertices meaningful for these queries will be repainted) and then add current one.

To perform second query for vertex v, just go over all subtrees containing v; for each of them, take $d' = dist(v, u)$ for u as centroid of the subtree and find the color of v in sorted stack using *lower_bound* algorithm (let's store stack in the vector or deque).

The complexity of precalculation is $O(n\ log\ n)$, answering the query of the first type — amortisized $O(log\ n)$, of the second type — $O(log^2 n)$.

I. Accounting Numeral System

Given integers n, m, $1 \leqslant n \leqslant 10^{16}$, $2 \leqslant m \leqslant 1000$, find a representation

$$n = C_{x_m}^m + C_{x_{m-1}}^{m-1} + \ldots + C_{x_1}^1,$$

for some integer $x_1, x_2, \ldots, x_m$, such that $0 \leqslant x_1 < x_2 < \ldots < x_m$.

Fact 1: for any nonnegative integer x:
$C_x^m \leqslant C_{x-1}^m + C_{x-2}^{m-1} + \ldots + C_{x-m}^1 + 1$ (can be proved by induction);

Fact 2: taking different sequences of x_i, $1 \leqslant i \leqslant m$ such that $x_m \leqslant x$ for some x, one can get all numbers from 0 to $C_x^m + C_{x-1}^{m-1} + \ldots + C_{x-m+1}^1$ and only those numbers (can be proved by induction using Fact 1).

I. Accounting Numeral System

Facts 1, 2 give us a possibility to solve the problem by greedy way. As x_m we take maximum possible x such that $C_x^m \leqslant n$; then, subtract $C_{x_m}^m$ from n and repeat the procedure for x_{m-1}, x_{m-2}, $\ldots$, x_1.

The only technical thing is how to calculate C_x^j for some integer x, j, accurately. We suggest the following scheme:

- if $m \geqslant 10$ then $x_m \leqslant 1000$; so we can precalculate C_x^y for and $x, y \leqslant 1000$ using the fact that $C_x^y = C_{x-1}^y + C_{x-1}^{y-1}$. If some C_x^y becomes more than n we replace it by $n+1$ to avoid an overflow;

I. Accounting Numeral System

- otherwise, any C_x^y, $y \leqslant m$, can be calculated as $\frac{x*(x-1)*...*(x-m+1)}{m*(m-1)*(m-2)*...*1}$. Before any multiplications, we reduce the fraction in $O(m^2 * \log\ x)$; after that, we calculate a product of the remaining numbers. When we want to multiplicate current C on some z, we should check whether $C \leqslant n/z$; if not then this C_x^y is more than n, an it doesn't have a sense to finish the calculation.

J. Ceizenpok's formula

Given $n, k, m, 1 \leqslant n \leqslant 10^{18}, 0 \leqslant k \leqslant n, 2 \leqslant m \leqslant 100000$, find $\binom{n}{k} \bmod m$.

First of all, factorize m, and let it be $m = p_1^{a_1} p_2^{a_2} \dots p_z^{a_z}$. If we find $\binom{n}{k} \bmod p_i^{a_i}$ for each $i = 1, 2, \dots, z$, then we can restore answer in $O(mz) = O(m \log m)$ using Chinese remainder theorem (or faster, but it's not necessary).

So, suppose that $m = p^a$, p is prime, a is positive integer.

Note that $\binom{n}{k} \bmod m = \frac{n!}{k!(n-k)!} \bmod m$. Represent $n!$ as $p^{c_n} q_n$, c_n is nonnegative integer, q_n is not divided by p; find c_n and $q_n \bmod m$. Do the same with $k!$ and $(n-k)!$; after that, we can find such a representation of $\binom{n}{k}$ using division by modulo in $O(m)$.

But how to find c_n and $q_n \bmod m$?

J. Ceizenpok's formula

Obviously, $c_n = \lfloor n/p \rfloor + \lfloor n/(p^2) \rfloor + \ldots + \lfloor n/(p^{\lfloor \log_p n \rfloor}) \rfloor$; because of cyclicity, $q_n \mod m = (q_m \mod m)^{\lfloor n/m \rfloor} * (q_{n \mod m} \mod m) \mod m$ (let q_0 be 1).

So, both c_n and q_n can be calculated in $O(m + \log n)$; then, the whole solution works in $O((m + \log n) * \log m)$.

B. Dispersed parentheses

Given n, k, we are to find a number of disperse parentheses sequences (parentheses sequences with some number of zeroes being inserted at arbitrary places) of length n with depth k.

Let $dp[i][j][l]$, $0 \leqslant l \leqslant j \leqslant i \leqslant n$, is a number of such a sequences of length i consisting of '(', '0' and ')' that:

- for any prefix of the sequence, difference between numbers of '('-s and ')'-s is nonnegative;
- maximal possible such a difference is j;
- such a difference for the whole sequence is l.

Use "forward dynamics". From any state $dp[i][j][l]$, one can move to states $dp[i+1][max(j, l+1)][l+1]$ (if $l < k$), $dp[i+1][j][l]$ and $dp[i+1][j][l-1]$ (if $l > 0$). It means that we can calculate dp in $O(nk^2)$ and find the answer as $dp[n][k][0]$.

G. Nano alarm-clocks

Given n stopped nano alarm-clocks which works in assumption that there are 12 hours, 10^6 minutes in each hour and 10^6 seconds in each minute. We are to add some times to each of watch in such a way that all clocks show the same time, and total added time should be as minimal as possible.

G. Nano alarm-clocks

Iterate over all possible equal times t from 0:0:0 to 11:999999:999999. For each t, we'll find a needed total time *tans* we need to add to get t. For 0:0:0, *cans* can be found straightforwardly. After that, after each adding a second to t. we should do the following:

- add a second to all watches, i.e. add n to *tans*;
- for each watch which shows time t, subtract $12 * 10^6 * 10^6$ from *tans*.

Note that if there is a sequence of second without any watches showing that time, then we can perform the seconds in $O(1)$; it reduces the complexity to $O(n\ log\ n)$ - time of sorting of all the times on the watches.

H. Lunch

Given n points in a row numbered from 1 to n. We want to start from point s, visit all vertices exactly once and finish in point f. On each move, we can go to the adjacent cell or jump over it; we want to minimize the number of moves of first type, or "non-jumps".

Suppose for simplicity that $s < f$.

H. Lunch

Consider the case $s = 1$, $f = n$. Let $dp[i]$, $1 \leqslant i \leqslant n$, be a minimal number of jumps we need to start from 1, visit all points from 1 to i and only those points and finish in point i. For basis, we can see that $dp[2] = 1$, $dp[3] = 2$. For each $i \leqslant 4$, we have two variants:

- first we do "non-jump"; it will give a path with $1 + dp[i - 1]$ non-jump;
- first we do jump; then, we are forced to make a "non-jump" back (it's the only variant not to finish in point 2); after that, we do jump to 4th point and reduce the problem into $dp[i - 3]$. So, $dp[i] = min(dp[i - 1], dp[i - 3]) + 1$.

H. Lunch

Suppose then that $s = 1$, $f < n$. Then, only one possible optimal way exist:

- visit all points from 1 to $f - 1$ and finish in $f - 1$-th; it can be done in $dp[f - 1]$ "non-jumps";
- jump to $j + 1$, $j + 3$, $j + 5$, ..., as many times as we can;
- "non-jump" to n if we are in $n - 1$ or to $n - 1$ otherwise;
- make a sequence of jumps until f-th point is achieved.

H. Lunch

Suppose then that $1 < s < f < n$. Then, if $s + 1 == f$ then the answer does not exist (we should go from s to the smaller points so first point more than s which should be visited will be f). Otherwise, the answer is $2 + dp[s - f - 1]$ (jump from 2 or 1, make one "non-jump" , jump to $s + 1$, visit all points from $s + 1$ to $f - 1$, jump to $n - 1$ or n, make one "non-jump" and jump to f).

The complexity of the solution is $O(n)$; it can be reduced to $O(1)$ if note that $dp[i] = \lfloor (i - 1)/3 \rfloor + (i - 1) \mod 3$.

L. The Pool for Lucky Ones

Given a row of n lanes; a_i people swim in the i-th of them $i = 1, 2, \ldots, n$. The person feels himself unhappy iff he swims in one of the lane with maximum possible a_i. We can ask one person to go to the neighbouring lane. What minimal number of unhappy people we can achieve?

Let A is a maximal among all a_i-s. After moving of a single person, maximum may become $A, A + 1$ or $A - 1$; all we need is to maintain numbers of lanes with A, $A + 1$ and $A - 1$; having these numbers, we can simply restore the answer. The complexity of the solution is $O(n)$.