Day 4 Div. A Contest

Moscow Pre-finals Workshop 2020

April 22, 2020

A. Arcs

Keywords: greedy, RMQ.

Statement 1: the optimal answer consists of **at most three** arcs.

Proof. Indeed, consider the disjoint clockwise-ordered set of arcs $[l_1, r_1]$, ..., $[l_k, r_k]$ with $k \geqslant 4$. The set of arcs $[l_1, r_2], [l_2, r_3], \ldots, [l_k, r_1]$ covers each point at most twice, thus their total length is at most $2L$, where L is the circle length.

Thus we can find an arc $[l_i, r_{i+1}]$ of length at most $2L/k \leqslant L/2$ and replace $[l_i, r_i]$, $[l_{i+1}, r_{i+1}]$ with a valid arc $[l_i, r_{i+1}]$, improving the answer.

Statement 2: Let $p_0, \ldots, p_{n-1}$ be the ordered sequence of given points.

If the optimal answer contains at least two arcs, then the (cyclic) interval $(p_{(i-1) \bmod n}, p_{(i+1) \bmod n})$ intersects with at least one arc in the answer.

Proof. Assume the opposite. Let $[l_1, r_1]$ and $[l_2, r_2]$ be the (distinct) closest arcs to p_i from both sides. Arcs $[l_1, i]$, $[i, r_2]$ cover each point at most once, thus their total length is at most L.

If, say, length of $[l_1, i]$ is at most $L/2$, replace $[l_1, r_1]$ with $[l_1, i]$ to improve the answer.

Explicitly try all options to take a **single arc**.

By statement 2, any other answer can be obtained by excluding at most three intervals $I_i = (p_i, p_{(i+1) \bmod n})$ so that the remaining arcs are valid.

For each interval I_i find the largest $dr(i)$ such that the arc $[p_{(i+1) \bmod n}, p_{(i+dr(i)) \bmod n}]$ is valid.

Similarly, for each interval I_i find the largest $dl(i)$ such that the arc $[p_{(i-dl(i)) \bmod n}, p_i]$ is valid.

Try all options to *exclude* two intervals I_i, I_j with $(j - i) \bmod n \leqslant dr(i)$, $(i - j) \bmod n \leqslant dr(j)$ to obtain a valid answer of **two arcs**.

Now to find the answer for **three arcs**.

If the clockwise order of the three excluded intervals is I_i, I_j, I_k, then $dl(i)$ and $dr(j)$ provide a range of valid values of k.

Try all valid options of i, j ($j - i \bmod n \leqslant dr(i)$), and use static RMQ to find the shortest I_k in the range.

Total complexity: $O(n^2)$ with sparse-table or queue for RMQ.

$O(n^2 \log n)$ with anything else... on a good day.

B. Bytica's Algebra

Keywords: number theory, polynomial GCD.

If $A(x) \equiv 0 \bmod P$, the answer is P. Otherwise, the answer is the degree of the polynomial $\mathrm{GCD}(A(x), \prod_{i=0}^{P-1}(x - i))$ in $\mathbb{Z}_p$.

All integers satisfy $x^P - x \equiv 0 \mod P$ by little Fermat's theorem.

Both $\prod_{i=0}^{P-1}(x - i)$ and $x^P - x$ have degree P, and have P distinct roots in $\mathbb{Z}_p$, thus they must be the same polynomial.

Find $\mathrm{GCD}(A(x), x^P - x)$ with Euclid's algorithm.

For the first step, find $x^P \bmod A(x)$ with binary exponentiation of x modulo $A(x)$.

Since after that the Euclid's algorithm converges in at most n steps, the **total complexity** is $O(n^2 \log P + n^3)$.

C. Cutting Brackets

Keywords: brute-force, DP.

If a string s is not a balanced bracket sequence, then the cutting process can not finish. Indeed, any way to merge pairs of matched brackets produces a balanced sequence.

Find the answer with recursion, cutting all possible bracket pairs at each step.

If the current string t is not balanced, return 0.
If the current string t was processed before, return the memorized answer.
The number of distinct reachable balanced subsequences is small enough.

D. Drawing Clusters

Keywords: geometry, planar graphs, bipartite matching.

First, construct all clusters. One way of doing that is to:

- find pairwise intersections between all pairs of circles, get rid of repeated intersections;
- divide each circle into arcs with all intersection points;
- reconstruct faces of the planar graph with vertices being intersection points, and edges being arcs.

Take care of overlapping circles, precision when removing intersections, circles without intersections, infinite regions, etc.

If two clusters are adjacent, the number of circles containing each of them differs by 1.
We can properly color clusters in two colors depending on whether the number of circles containing each cluster is odd or even. That is, the graph G of adjacent clusters is bipartite.
It follows that when $k \geqslant 2$ the answer is equal to the number of clusters.
Otherwise, the answer is the maximum size of an independent set in G.
By Kőnig's theorem, the maximum size of an independent set is equal to (the number of vertices of G) - (the maximum matching size of G).
Since G is bipartite, find the maximum matching with Kuhn's algorithm.
Total complexity: roughly $O(n^6)$, but supposedly much faster.

Bonus tip: to avoid precision errors with detecting multiple intersection points, check for each triple of circles if they intersect at the common point: reduce to intersection of a circle and two lines (by subtracting the equations), obtain rational coordinates of lines' intersection, check if it lies on the circle.

E. Eyesight Development

Keywords: angle scanline, priority queue.

Consider a ray $r(\varphi)$ starting at the origin O and rotating counter-clockwise.
For each side of each rectangle let us locate the events when $r(\varphi)$ starts and stops intersecting this side.
Also, let us maintain the order of sides by the distance of their intersection with $r(\varphi)$.
Since rectangles are disjoint, the only changes to the order are insertions and removals.
A rectangle is visible from O if at any point one of its sides has the intersection with $r(\varphi)$ closest to O.

To maintain the order of sides during the scanline, use priority queue.
Use **dynamic comparator**: maintain the current φ, and compare pairs of sides with the current value of φ.
Instead of actual φ's it is better to use direction vectors to the points (which are always integer, and allow for precise solution).
Total complexity: $O(n \log n)$.

F. Ferry Express

Keywords: scanline.

If a coin falls to point x at time t, then to catch it with speed v we must have $vt \leqslant x \leqslant vt + L$.
Since v is integer, this can be transformed to $\lceil \frac{x-L}{t} \rceil \leqslant v \leqslant \lfloor \frac{x}{t} \rfloor$.
Find these segments for all coins, and use scanline to find the smallest value of v covered with most segments.
Total complexity: $O(n \log n)$.

G. Game Frotnite

Keywords: shortest path, DP.

Let $dp_{t,x,y}$ be the maximum length of a path we can take starting from the cell (x, y) at time t. If the cell (x, y) is already removed at time t, put $dp_{t,x,y} = 0$.
Calculate dp by decreasing of t. If (x, y) is not yet removed, then $dp_{t,x,y}$ is equal to 1 + maximum of $dp_{t+1,x',y'}$ for non-obstacle cells (x', y') reachable in zero or one steps.
The answer for each cell is $dp_{0,x,y}$.
Total complexity: $O(n^3)$ time with very small constant factor, and $O(n^2)$ memory if storing only two layers of $dp_{t,x,y}$.

H. Height Growing

Keywords: FFT.

The probability that the height is greater than, say, $H \sim 4 \log_2 n$ is negligible.

Let $p_{i,h}$ be the probability that the height of a random treap with i vertices is at most h. Put $p_{0,h} = 1$ for all $h \geqslant 0$ by definition.

Each vertex of a random treap is equiprobable to be the root (have the smallest priority), and, if the root key is known, the subtrees are random treaps of respective sizes.

This gives the recurrence $p_{n,h} = \frac{1}{n} \sum_{i=0}^{n-1} p_{i,h-1} p_{n-i-1,h-1}$ for $h > 0$.

Write $P_h(x) = \sum_{i=0}^{n} x^i p_{i,h}$. Obtain $P_h(x)$ by increasing of h as follows:

- take $Q(x) = x P_{h-1}^2 + 1$, and truncate it to $n+1$ terms;
- divide the i-th coefficient of Q by i for all $i > 0$.

Use FFT to find P_{h-1}^2, and stop at $h = H$.

The probability that the height is exactly h is $p_{n,h} - p_{n,h-1}$.

Total complexity: $O(Hn \log n) \sim O(n \log^2 n)$.

I. Improving Hackensburg

Keywords: DFS, atrocious input format.

Theorem. The network has at least one turn.

Starting from this turn, we can follow the network in a unique way and reconstruct the road directions.

To determine the largest number of intersection conversions, start by converting **all** intersections (say, there are I of them).

According to the road directions, the network decomposes into several, say, C cycles.

Statement The answer is $I - C + 1$.

Construct a graph G where each vertex corresponds to one of the C cycles.

If cycles i and j have a common intersection, make an edge between i and j.

If we choose any spanning tree in G and un-converse the intersections corresponding to the edges of the tree, the network will become connected again.

We can not un-converse any fewer intersections, since if we un-converse any intersection not in G, the number of cycles will only increase.

J. Just Interesting

Keywords: subtree DP.

Create a graph with $n+1$ vertices.

For each inequality $p_i > p_j$ create a directed edge (i, j). If the graph contains a cycle, the answer is 0.

Otherwise, if a node j does not have an incoming edge, create an edge $(n+1, j)$. Edges now form a tree rooted at $n+1$.

The problem now is to count permutations (topological orderings) π such that $\pi_s = t$, and $\pi_i > \pi_j$ for any edge (i, j). π_{n+1} is always $n+1$.

Let T_j denote the subtree of j, and w_j be the number of vertices in T_j. Suppose that for each child j of i we know x_j — the number of top. orderings of T_j. Consider the process of starting with i with no children, and add children one by one. Then x_i can be computed as follows:

- Initially put $x_i = 1$, $w_i = 1$.
- For each new child j, multiply x_i by $x_j \cdot \binom{w_i - 1 + w_j}{w_j}$ (choose any ordering of the current T_i, and any ordering of T_j, and merge them arbitrarily so that i is still the last). Also, add w_j to w_i.

This allows to compute x_i with subtree DP.

To account for $\pi_s = t$, if the subtree of i contains s, instead compute $dp_{i,p}$ — the number of top. orderings of T_i such that s is preceded by p other vertices. This allows for a similar recomputation:

- Initially, if $i = s$, put $x_{i,0} = 1$, $w_i = 1$. Otherwise, copy values of $x_{i,p}$ from the child j such that $s \in T_j$; also put $w_i = w_j + 1$.
- For each new child j (that can no longer contain s), for all p put $x_{i,p} = \sum_{k=0}^{p} x_{i,k} x_j \binom{p}{k} \binom{w_i + w_j - 1 - p}{w_i - 1 - k}$ (in the new ordering, choose how many out of p vertices preceding s come from the current T_i, and the number of ways to merge any pair of suitable orderings), and add w_j to w_i. The answer is $dp_{n+1,t-1}$.

If the hardest part ($x_{i,p} = \ldots$) has reasonable bounds of summation ($k \leqslant w_i$, $p - k \leqslant w_j$), then the **total complexity** is $O(n^2)$.

K. Kirchhoff

Keywords: linear equations, physics? in my contest???

Ohm's law: if two nodes i and j with electric potentials V_i and V_j are connected with resistance R, the current I_{ij} directly from i to j through R is equal to $\frac{V_i - V_j}{R}$.

Kirchhoff's circuit law (current conservation): for any node i we have $\sum_j I_{ij} = x_i$, where x_i is the current supply to i (positive if current enters the node i from outside, negative if the current leaves the node i, zero if not connected to the outside).

Let $C_{ij} = \sum \frac{1}{R}$, where the sum is over all resistances directly connecting nodes i and j. Combining the two laws above, for any node i we have

$$\sum_j (V_i - V_j) C_{ij} = V_i \sum_j C_{ij} - \sum_j V_j C_{ij} = x_i.$$

Construct a matrix A, where $A_{ii} = \sum_j C_{ij}$, $A_{ij} = -C_{ij}$ for $i \neq j$. If V and x are vectors of potentials and supplies respectively, we obtain a matrix equation $AV = x$.

If the node n is the *ground node* ($V_n = 0$), then by dropping the last row and column of A we obtain $A'V' = x'$. A' is non-singular for connected networks (*something-something, Kirchhoff's tree matrix theorem*), thus $V' = (A')^{-1}x'$.

If a unit of current enter through node i and leaves through node j, then $x_i = 1$, $x_j = 1$, all other $x_k = 0$. Find the values of V_i from $V' = (A')^{-1}x'$ and $V_n = 0$.

By Ohm's law, the effective resistance R_{ij} is then found by $\frac{V_i - V_j}{1}$.

If the matrix $(A')^{-1}$ is precomputed (e.g. with Gaussian elimination), each query can by answered in $O(1)$ since only two entries of x are non-zero, and only two entries of V are needed.

Total complexity: $O(n^3 + Q)$.