Day 1 Contest

Bytedance Moscow Workshops Camp 2020

May 1, 2020

A. Allowed Swaps

Keywords: DFS.

Consider the graph G with n vertices where edges correspond to allowed swaps.

If G is a spanning tree, we can choose an arbitrary leaf v and move the label $v = p_u$ from its current position u along the unique path in the tree. After that we can remove v from the tree.

By fixing the leaves' labels one by one, we can then sort the permutation in at most $(n-1)+(n-2)+\ldots = n(n-1)/2$ swaps.

If G is arbitrary, sorting is not possible if v and p_v are in different component for any v.

Otherwise, we can sort labels in each of connected components of G by taking its spanning tree and applying the above method.

Total complexity: $O(n^2)$.

B. Banned Words

Keywords: multiple string matching, DP, matrix multiplication.

Consider the set $S = \{s_1, \ldots, s_k\}$ of all different prefixes of banned words.

For any index i and letter c, let $f(i, c)$ be the index j of the longest s_j that is a suffix of $s_i + c$.

Let $dp_{l,i}$ be the number of strings t of length l such that:

- t does not have banned substrings;
- the longest suffix of t that is present in S is s_i.

Initially, for $l = 0$ only the empty string needs to be considered. If s_1 is the empty string, then $dp_{0,1} = 1$, all other $dp_{0,i} = 0$.

In other cases, $dp_{l,i} = 0$ is s_i has a banned suffix. Otherwise, consider forward transitions for all letters c and perform $dp_{l+1,f(i,c)}$+=$dp_{l,i}$, unless $s_{f(i,c)}$ has a banned suffix.

If v_l is the vector with entries $(dp_{l,1}, \ldots, dp_{l,k})$, then $v_{l+1} = Av_l$ for the transition matrix A. Thus $v_l = A^l v_0$.

In given constraints, the set S and matrix A can be computed straightforwardly. A^l can computed in $O(k^3 \log l) = O(s^3 \log l)$ time, where s is the total length of banned words.

Total complexity: $O(s^3 \log l)$.

C. Cost of Subtree

Keywords: DSU.

Suppose that the smallest edge label in the answer is v. Construct a subgraph of the given tree by leaving only edges with labels at least v, then take the largest connected component.

To account for all different v, consider all edge by decreasing of their labels and maintain the subgraph described above.

If the current edge is (u, v), in the subgraph connect u and v. Try to update the answer with $v_i \times$(the largest component size).

The largest component can be maintained with DSU.

Total complexity: $O(n \log n)$.

D. Disk Troubles

Keywords: baby step-giant step.

The problem is to solve discrete logarithm problem in the field of order $N = 2^{32}$. Note that multiplication in this field can be computed fast (~ 32 operations) with bitwise manipulations.
Discrete logarithm can be solved with Shanks's algorithm (baby step-giant step) in $O(\sqrt{N}\log N)$ time.
Idea: if m is the solution to $x^m = y$, let $K = \sqrt{N}$ and represent $m = aK + b$ with $a, b < K$. We must then have $x^{aK+b} = y$, thus $(x^k)^a = yx^{-b}$.
For each query y compute all numbers $(x^K)^a$ and yx^{-b}, and find matches between them with a set data structure, or sorting and two pointers.
Note: to process multiple queries, we can decrease K and put $a = \lfloor m/K \rfloor$, $b = m \bmod K$. The numbers $(x^K)^a$ do not depend on the query, thus we only need to precompute them once. Then the complexity of processing Q queries becomes $O(N/K \log N + QK \log N)$, which is $O(\sqrt{QN}\log N)$ for $K \sim \sqrt{N/Q}$.
Note: this problem can be solved even faster by using decomposition of $\mathbb{F}_{2^{32}}$ with Pohlig-Hellman algorithm (basically independent BSGS for prime divisors of $2^{32} - 1$). For reference, see Codeforces problem 1310F.

E. Embedding

Keywords: DP, palindromic tree.

Suppose that t is a palindrome. Denote $p(t)$ to be the palindrome t without the first and the last letter, and $b(t)$ to be the largest suffix palindrome of t.

If $s_i = t$ is currently the last palindrome in the sequence, we can recursively obtain all continuations with the following process (starting from State 0).
State 0. Finish **OR** replace t with $p(t)$, go to State 1.
State 1. Replace t with $p(t)$, go to State 1 **OR** go to State 2.
State 2. Take $s_{i+1} = t$ and go to State 0 **OR** decide whether s_{i+1} is a prefix or suffix of t, replace t with suffix/prefix $b(t)$, go to State 3.
State 3. Replace t with suffix/prefix $b(t)$, go to State 3 **OR** take $s_{i+1} = t$, go to State 1.

If $dp_i(t)$ is the number of continuations with t from the state i, we have the recurrences:
$dp_0(t) = 1 + dp_1(p(t))$
$dp_1(t) = dp_1(p(t)) + dp_2(t)$
$dp_2(t) = dp_0(t) + 2dp_3(b(t))$
$dp_3(t) = dp_3(b(t)) + dp_0(t)$

String s of length n has at most n distinct palindromes. The palindromes t_i, as well as $p(t_i)$ and $b(t_i)$, can be found in linear time with palindromic tree (eertree), or potentially less efficiently with other string algorithms.
After that, use the recurrences to obtain the answer. The easiest way to produce the answer is to find the sum of $dp_3(r_i)$, where r_i is the largest palindrome ending at character i.
Total complexity: $O(n)$ or slightly slower.

F. Fox Labeling

Keywords: combinatorial probability, DP, integer partitions.

Suppose that indistinguishable groups have sizes $a_1, \ldots, a_s$, let A be the multiset of these numbers. Marking a random set of k foxes results in another set of groups with sizes in multiset B, where each of old groups either is unchanged, or breaks in two groups.
If $f(A)$ is the expected number of steps starting from the multiset A, and $p(A, B)$ is the probability to obtain multiset B from multiset A, then $f(A) = (1 + \sum_{B \neq A} p(A, B)f(B))/(1 - p(A, A))$.
Note that for all B either $A = B$, or $|B| > |A|$, thus the transition graph is acyclic, and $f(A)$ can be computed with simple DP.
$f(A)$ does not change is A is reordered, thus the number of states is bounded by the number of unordered partitions of n, which is less than 6000.

For a given multiset B, let us find all multisets A such that transition $A \to B$ is possible, along with $p(A, B)$. To do this, try all subsets of B with sum k. For any such subset $C = (c_1, \ldots, c_t)$, possible A's are produced by uniting each group in C with at most one group in $B - C$.
If the group c_i is united with a group d_i (possibly $d_i = 0$), then for this option we should add $\prod_i \binom{c_i+d_i}{c_i} / \binom{n}{k}$ to $p(A, B)$.
We can reduce the number of options by only considering unique multisets C. If c_x and b_x are multiplicities of x in C and B respectively, the number of ways to obtain a subset C from B is $\prod_x \binom{b_x}{c_x}$.
Further optimizations include special treatment of groups of size 1 and hashing the multisets.
With enough optimizations we can obtain a fast enough solution.

G. Game With Permutation

Keywords: constructive, randomized.

Ask random queries. Use local optimization to find a permutation maximizing the function "the number of matching numbers in all queries". Practically, asking 240 queries is enough to determine the unique answer with very high probability.
We think there should be a better constructive solution. We will update if we find one.

H. Histogram and Blue Rectangles

Keywords: convex hull trick.

Let h_i be the height of the column i, and $h_0 = 0$. For $i > 0$, let $left(i)$ be the largest $j < i$ such that $h_j < h_i$. Suppose that the rightmost column of a rectangle is the column i. Let $p_1, p_2, \ldots, p_k$ be the sequence defined by $p_1 = i$, $p_{j+1} = left(p_j)$, $p_k = 0$.
By considering maximal rectangles, we find that the largest area of a rectangle with rightmost column i is $\max_{j<k}(i - p_{j+1}) \times h_{p_j}$.

Construct a tree rooted at 0, where the parent of a node i is $left(i)$. Then for any node i the path to the root is the sequence $p_1, \ldots, p_k$. To maximize $(i - p_{j+1}) \times h_{p_j} = ih(p_j) - p_{j+1}h_{p_j}$ we will use dynamic convex hull trick. In the DFS we maintain a sequence of points (x_i, y_i). The sequence is always ordered by increasing of x_i, and can be changed by pushing or removing points from the right. We also maintain upper convex hull of the sequence.
• When moving from a parent node u to a child node v, push the point $(h_v, -u \cdot h_v)$ to the sequence and update convex hull.
• At a node v, find the largest area of a rectangle with rightmost column v as $\max x_i \cdot v + y_i$ with a convex hull trick query.
• When leaving from a child node v, rollback the convex hull to its previous state.

Note that using the usual stack updates to the convex hull can result in $\Omega(n^2)$ complexity. Instead, use BST and binary search to remove a suffix of the convex hull before pushing a new point.
This way, the **total complexity** is $O(n \log n)$.

I. Integer Equation Checker

Keywords: parsing.

Any expression can be checked for validity and correctness, e.g. with recursive descent or, in this case, just by finding all operators and equality signs.
Try all ways to replace at most two characters, and check if the resulting expressions are correct.
The given expression is very short, so complexity doesn't matter.

J. Joy With Cookies

Keywords: Grundy theory, scanline, linear equations.

Consider $2n$ possible cookies on top of a stack, with cookie $i + n$ being a rotated version of the initial cookie i. Let x_i and y_i be the dimensions of the cookie i.
The game is a sum of independent games for all initially placed cookies. Thus, we want to compute the Grundy

value G_i for each top cookie i.
We have that $G_i = \text{mex}(G_j \mid 1 \leqslant j \leqslant n, x_j < x_i, y_j < y_i)$.
Note that if cookie k can be placed on top of j, and j can be placed on top of i, then k can also be placed on top of i.
Thus, $G_i = \max(G_j \mid 1 \leqslant j \leqslant n, x_j < x_i, y_j < y_i) + 1$, since if we can obtain G_j, we can also obtain all values $0, \ldots, G_j - 1$ as possible moves from j (thus, as possible moves for i).
In case when no move from i is possible, $G_i = 0$.
Values of G_i can be found with a standard scanline and RMQ.

For the initial game with the cookie a_i, we can choose the starting Grundy value between G_{a_i} and G_{a_i+n}.
Instead, suppose that the starting value of the entire game is equal to $X = \oplus_i G_{a_i}$, and for any i we can independently choose to change X to $X \oplus Y_i$, where $Y_i = G_{a_i} \oplus G_{a_i+n}$. The goal is to change X to 0.
Thus, we need to find a solution for $X = \oplus_i Y_i z_i$ with variables $z_i \in \mathbb{Z}_2$. This can be done with Gaussian elimination over $\mathbb{Z}_2$.
X and all Y_i are $O(n)$, thus the number of their bits is $O(\log n)$, and Gaussian elimination can be done in $O(k \log n)$ time. **Total complexity**: $O((k+n)\log n)$.

K. Knights of Round Table

Keywords: constructive, 2-coloring.

If $p[x]$ is the type of potion for the knight x, we must have $p[a_i] \neq p[b_i]$ for all i.
If we also ensure that $p[1] \neq p[2], p[3] \neq p[4], \ldots, p[2n-1] \neq p[2n]$, then no three adjacent knights will have the same potion.
Construct a graph with $2n$ vertices, and edges corresponding to inequalities described above.
This graph is a union of two perfect matchings, thus it consists of disjoint cycles of even length.
Such a graph is easy to properly color with two colors.
Total complexity: $O(n)$.

L. Lands of Infinistan

Keywords: planar intersection graph.

Consider the planar intersection graph of all objects, with a common vertex at infinity. The number of regions can be found from Euler's formula $V - E + F = 1$.
If the conic is an ellipse, we need to introduce a vertex on it for the identity to work.
If there are no intersections, then $V = 1$ or 2 depending on the ellipse case, $E = n + b$, where n is the number of lines, b is the number of branches of the conic (1 for ellipse and parabola, 2 for hyperbola).
Each intersection point of multiplicity k increases V by 1, and E by k.
Find all intersections explicitly and count their multiplicities.
Line-line intersection are found by solving a 2×2 linear equation system. To find line-conic intersections, represent the line parametrically $p(t) = p_0 + vt$ with a point p_0 on the line and a direction vector v. Plug $p(t)$ into the conic equation and solve quadratic equation for t.

To find the type of a conic $ax^2 + bxy + cy^2 + O(x+y) = 0$, compute $D = b^2 - 4ac$. The conic is an ellipse when $D < 0$, a parabola when $D = 0$, and a hyperbola when $D > 0$ (provided these are the only possible cases).