Day 1: Kyoto Contest 2 Editorial

Um_nik

February 3, 2022

1 Announcements

Calculate the number of different values of $\lceil \frac{S_i}{T} \rceil$.

2 Build The Grid

```
.######
.#.....
.#.###.
.#.#.#.
.#...#.
.#####.
.......
```

3 Coins and Boxes

We start on the left and we have to visit all the objects, including the rightmost one. Let's look at the segments between consecutive objects (including the starting point). We have to cross each segment at least once; we have to cross each segment to the right of where we will finish at least twice, and all the segments to the left of the finish must be crossed odd number of times. We can't cross some segment only once if there are more boxes than coins to the left of it, because we can't open all the boxes using only the coins on the left, and we won't be able to return after we cross the segment. For each segment determine if it is ok to cross it once. Now fix the finish point. If we twice cross each segment on the right; once cross all the segments on the left that is ok to cross once and cross all other segment three times, it is actually possible to construct a valid path. By doing scanline on finish point it is easy to maintain the total distance.

4 Destructive Game

The game is a direct sum of independent games, for each game we can calculate its Grundy value and take their xor. For odd b Grundy value is just the parity of a. For even b Grundy value is determined by $(a \bmod (b+1))$: it is 2 if $(a \bmod (b+1)) = b$ and parity of $(a \bmod (b+1))$ otherwise. To prove that notice that each move changes $(a \bmod (b+1))$ by 1.

5 Edges, Colors and MST

Edge is not a part of MST if it is bigger than all the edges of MST on a path connecting endpoints of the edge in MST. Lexmin sequences are constructed greedily. Let's iterate over edges. If we encounter an edge from MST without assigned value, assign the smallest value we have left. If we encounter an edge not from MST without assigned value, we have to make sure that all the edges of MST on the path between endpoints have smaller values. Let's say there are k of them, then the value of current edge cannot be smaller than $(k+1)$-th minimum of remaining values. It turns out it is always possible to assign the $(k+1)$-th minimum. First k minimums should be distributed between edges of MST on the path between endpoints, and it is always better to assign them in order they appear in the list of edges. Notice that this algorithm always uses the smallest possible numbers, so all the assigned numbers are smaller than all the not assigned.

To implement it you need to find all the edges without assigned numbers on a path in a tree. It is OK to do it in time proportional to their number, as the total number of edges in the tree is $O(n)$. Use path compression (DSU) to skip segments of edges with assigned numbers.

6 Flatland Currency

Notice that coin denominations form a sequence where each number is disible by the next one. We only care about number of coins with denomination 1 in the end, so it doesn't make sense to use them, and it is easy to see that we only care about total value of all other coins. If we pay for something of cost A, we will get $(5 - A \bmod 5) \bmod 5$ coins of denomination 1, and will spend $\lceil \frac{A}{5} \rceil$ money. It doesn't make sense to buy items in bunches, because it can save money only if it produces less coins of denomination 1 than any of the items by themselves.

This gives us a reduction to a knapsack problem: each item costs $\lceil \frac{A}{5} \rceil$ and produces $(5-A \bmod 5) \bmod 5$ coins of denomination 1. Let's say that we have n items, i-th of them has cost c_i and value w_i. The important thing about this knapsack instance which allows us to solve it is that $w_i \le 4$. Let's group items by value, if we fix the number of items of given value we want to take, it is obvious that we should take the cheapest ones. Let's sort them in advance. If we knew that from each value group we will take items with total value divisible by 12, we would be able to glue them together in groups with value 12, which will allow greedy solution, since all the groups have the same value. It is not true, but we can make it true by bruteforcing the remainder of total value modulo 12 in each value group. It will work in $12 \cdot 6 \cdot 4 \cdot 3 \cdot n$.

7 Game with Balls and Boxes

Let's assume we have somehow done the first round. In the second round we will have to open all the boxes that still don't have the right ball. Thus in the first round we want to minimize cost we pay now for boxes opened in the first round + cost for boxes with wrong ball in the second round. The permutation is a collection of independent cycles. For each cycle we can either open all the boxes in the first round and fix all of them, or open some (cyclic) segments which will allow to fix all the boxes except one of the borders. Choosing the best option can be done with linear DP where we store whether we decided to open the first and the last boxes.

8 High Powers

$s = a+b+c$, $t = ab+bc+ca$, $u = abc$ means that a, b and c are the different roots of polynomial $x^3 - sx^2 + tx - u$. That means that for any $n \ge 3$ $a^n = sa^{n-1} - ta^{n-2} + ua^{n-3}$ (and the same holds for b and c).

$$F(n,m) = \frac{a^n(b^m - c^m) + b^n(c^m - a^m) + c^n(a^m - b^m)}{(a-b)(b-c)(c-a)}$$

For $n \ge 3$

$$\begin{aligned}
F(n,m) = \frac{a^n(b^m - c^m) + b^n(c^m - a^m) + c^n(a^m - b^m)}{(a-b)(b-c)(c-a)} &= \frac{(sa^{n-1} - ta^{n-2} + ua^{n-3})(b^m - c^m)}{(a-b)(b-c)(c-a)} + \\
+ \frac{(sb^{n-1} - tb^{n-2} + ub^{n-3})(c^m - a^m)}{(a-b)(b-c)(c-a)} &+ \frac{(sc^{n-1} - tc^{n-2} + uc^{n-3})(a^m - b^m)}{(a-b)(b-c)(c-a)} = \\
= s \cdot \frac{a^{n-1}(b^m - c^m) + b^{n-1}(c^m - a^m) + c^{n-1}(a^m - b^m)}{(a-b)(b-c)(c-a)} &- t \cdot \frac{a^{n-2}(b^m - c^m) + b^{n-2}(c^m - a^m) + c^{n-2}(a^m - b^m)}{(a-b)(b-c)(c-a)} + \\
+ u \cdot \frac{a^{n-3}(b^m - c^m) + b^{n-3}(c^m - a^m) + c^{n-3}(a^m - b^m)}{(a-b)(b-c)(c-a)} &= sF(n-1,m) - tF(n-2,m) + uF(n-3,m)
\end{aligned}$$

Similarly, for $m \ge 3$ $F(n,m) = sF(n,m-1) - tF(n,m-2) + uF(n,m-3)$.

It's then straightforward to check that

$$\begin{pmatrix} F(n,m) & F(n+1,m) & F(n+2,m) \\ F(n,m+1) & F(n+1,m+1) & F(n+2,m+1) \\ F(n,m+2) & F(n+1,m+2) & F(n+2,m+2) \end{pmatrix} \begin{pmatrix} 0 & 0 & u \\ 1 & 0 & -t \\ 0 & 1 & s \end{pmatrix} = \begin{pmatrix} F(n+1,m) & F(n+2,m) & F(n+3,m) \\ F(n+1,m+1) & F(n+2,m+1) & F(n+3,m+1) \\ F(n+1,m+2) & F(n+2,m+2) & F(n+3,m+2) \end{pmatrix}$$

and

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ u & -t & s \end{pmatrix} \begin{pmatrix} F(n,m) & F(n+1,m) & F(n+2,m) \\ F(n,m+1) & F(n+1,m+1) & F(n+2,m+1) \\ F(n,m+2) & F(n+1,m+2) & F(n+2,m+2) \end{pmatrix} = \begin{pmatrix} F(n,m+1) & F(n+1,m+1) & F(n+2,m+1) \\ F(n,m+2) & F(n+1,m+2) & F(n+2,m+2) \\ F(n,m+3) & F(n+1,m+3) & F(n+2,m+3) \end{pmatrix}$$

Since

$$\begin{pmatrix} F(0,0) & F(1,0) & F(2,0) \\ F(0,1) & F(1,1) & F(2,1) \\ F(0,2) & F(1,2) & F(2,2) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} F(n,m) & F(n+1,m) & F(n+2,m) \\ F(n,m+1) & F(n+1,m+1) & F(n+2,m+1) \\ F(n,m+2) & F(n+1,m+2) & F(n+2,m+2) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ u & -t & s \end{pmatrix}^m \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & u \\ 1 & 0 & -t \\ 0 & 1 & s \end{pmatrix}^n$$

9 Items and Heroes

Answer is Yes if and only if $\sum_{v \in S_u} A_v \geq \sum_{v \in S_u} C_v$ for every u, where S_u is the subtree of u (Hall's theorem). Let's maintain $\sum_{v \in S_u}(A_v - C_v)$ for each u. Update is $+=$ on path to root, to check for answer we should check that minimum is non-negative. All of that can be done with Heavy-Light Decomposition.

10 Juggler's Trick

At first let's assume that there are no W balls. After all the operations we will remove some segments of the initial array. Segments are independent, let's learn how to check if we can remove some segment. Obviously, the length of the segment should be divisible by $(r+b)$. If $k = \frac{length}{r+b}$ then we will spend exactly k operations and will remove kr red balls and kb blue balls, so our segment should have exactly that number of them. It turns out that these conditions are sufficient. Let's prove that by induction on k. Base $k = 0$ is obvious, to prove the step we need to show that there is a segment with which we can start (so it has exactly r red and b blue balls). Let's divide the segment into k segments of length $(r+b)$. Either there will be a good segment already, or there will be segments with strictly less and strictly more red balls. Let's move one into the other, the number of red balls is changing by at most 1 each step, so at some point we will encounter the segment with exactly r red balls.

White balls don't change much, now we want to have not more than kr red balls and not more than kb blue balls. That allows us to do $O(n^2)$ DP: maximum number of operations on prefix. The transitions are either to skip the next ball, or start a new segment here, for which we will need to iterate over k. To check that segment is good in $O(1)$ we will need to precalculate prefix sums for number of red and blue balls: R_p and B_p respectively.

To speed it up, we will need to make the transitions of the second type faster. Let's calculate the DP backwards. When can we do the transition from q to p?

1. $q < p$
2. $(p-q) \bmod (r+b) = 0 \Leftrightarrow p \bmod (r+b) = q \bmod (r+b)$
3. $k = \frac{p-q}{r+b}$
4. $R[p] - R[q] \leq kr = \frac{rp}{r+b} - \frac{rq}{r+b} \Leftrightarrow R[p] - \frac{rp}{r+b} \leq R[q] - \frac{rq}{r+b}$
5. $B[p] - B[q] \leq kb = \frac{bp}{r+b} - \frac{bq}{r+b} \Leftrightarrow B[p] - \frac{bp}{r+b} \leq B[q] - \frac{bq}{r+b}$

Thus to choose the best transition for given p, we should look at all the $q < p$ with the same remainder modulo $(r+b)$, and choose among the ones that satisfy $R[p] - \frac{rp}{r+b} \leq R[q] - \frac{rq}{r+b}$ and $B[p] - \frac{bp}{r+b} \leq B[q] - \frac{bq}{r+b}$. That's a rectangle on a plane, so we can use 2D Segment Tree (a separate one for each remainder modulo $(r+b)$) to choose the best option.

11 King's Palace

Let's do meet-in-the-middle: split the walls into groups of sizes B and $N-B$, bruteforce all the options in each group. Now we have to somehow calculate the number of good options from the second group for each option in the first group. To desribe the constraints that the first group imposes on the second, we will use a bitmask of size $3(N-B)$: for each wall and each color store if it is forbidden. Option for the second group is good if it doesn't use any of the forbidden colors, which means that one should use sum-over-subset DP to calculate the number of good options. The solution works in $O(3^B + 3^{N-B} + (N-B)2^{3(N-B)})$ time, which is fine for $B = 6$ or $B = 7$.

12 Lion and Zebra

TODO

13 Math String

Let's first come up with a linear time DP which stores some constant number of numbers for each prefix and recalculates them when you append a new symbol. The things one need to store are:

- sum of everything before the last + sign
- product of everything after the last + sign
- product of everything after the last + sign except for the last (incomplete) number

It would be enough if one would just say what symbols should we append, but since we are interested in the sum over all options, we need to store sums of all that over all options, and we need to be careful to understand that sometimes 1 is actually the number of possible options, so we need to store that also. And we need to distinguish between the cases when the last symbol is digit or not, since the transitions are a bit different (and we cannot put two signs next to each other).

Once you have this DP, it should be straightforward to see that transition is linear, which means that we can wrap it into multiplication by a matrix, and then change multiplication by the same matrix n times into matrix exponentiation.