T192

Author: Alex Tung
Setter: Ian

tl;dr

Given an array A containing N integers, where 0 <= A[i] < 3.

There are 3 operations:

1. L R x, change A[L.. R] = x
2. L R, A[i] = (A[i] + 1) % 3, L <= i <= R
3. L R, find number of pairs(p, q) such that A[p..q] is colorful.

Subtask 1

Just do what the problem states:

For operation 1: for loop change all elements.

For operation 2: for loop change all elements.

For operation 3: a nested for loop to calculate (p, q) that is colourful.

O(Q * N^2)

O(Q * N^3) may pass too as there are only simple iteration.

Subtask 2

Similar for subtask 1, we could see the bottleneck is calculating the no. of colourful pairs.

We could apply two-pointer technique to calculate.

O(Q * N)

Subtask 3

Here there is only point update and range query.

We could come up with a segment tree solution.

Each node maintaining the no. of colourful strips in its segment, and the first and last occurence of the three colours.

When we merge two nodes V, U (V is on the left side and U on the right side), We try to calculate colourful strips that starts in V and ends at U. We could use the last occurrences of colours in V and first occurrences of colours in U to calculate.

O(Q log N)

Subtask 4

Here we have range update and range query.

So, we have to apply lazy propagation.

When we try to propagate from V (parent) to U (child), we need to handle some cases.

We first change U using the lazy value of V. Then,

If the lazy value of V is related to operation 1, then we know we could just simply change the lazy value of U to V and initialize node U information (like setting the ans = 0, first and last occurence of that colour = (x, y), other colour = -1).

Subtask 4

If the lazy value of V is related to operation 2:

if the lazy value of U is related to operation 2/ no lazy value:

change the lazy value (+ lazy value of V) % 3.

if the lazy value of U is related to operation 1:

we have to shift the colour of the lazy value (+ lazy value of V) % 3

Notice that the related operation of lazy value of U is unchanged.

O(Q log N)

Subtask 5

The difference of subtask (3) 4 and 5 is the constant of your merging function of the segment tree node.

To pass subtask 5, your merging function should have constant <= 3 or applied some constant optimization in it.

You should carefully implement the segment tree, avoid using vector in the node.