J194 Graffiti

Author: Alex Tung
Setter: Tony, Percy, Ian

tl;dr

Given a lower triangular 0-1 matrix with length N. There is K 1-element in the matrix. A good matrix is defined as :

> Element in (i, j) is 1 if and only if elements in (i + 1, j) and (i + 1, j) are also 1.

Find out how many 0-element in the matrix you need to change to 1 to make the matrix good.

Subtask 1 (K = 1)

output ((1 + (n - i + 1)) * (n - i + 1)) / 2

as it is equal 1 + 2 + … + (n - i + 1) (sum of arithmetic sequence)

Subtask 2

Like subtask 1, calculate the area of the two triangle, then subtract the area that is counted twice. There are three case,

1. the repeated area is triangle 1
2. the repeated area is triangle 2
3. a sub triangle of triangle 1, 2
4. no repeated area

We could use some “ifs” to determine which case it is, 1,2,4 is easy. Case 3 need some calculation.

Subtask 3 (N = 2)

um…. there is only 3 element… only two cases… as K >= 1, it implies answer equals 3 - K.

Subtask 4 (N <= 20)

Almost every brute force solution could pass this.

For example, we could write a recursion

```
need_to_be_one(i, j)

    if (i == N)
```

Subtask 5 (N <= 3000)

We could come up with a O(N^2) solution:

1. We iterate from the triangle top to bottom, if element(i, j) is 1, then, of course, element(i + 1, j) and element(i + 1, j + 1) must be 1.
2. We iterate from bottom to top, if element(i + 1, j) and element(i + 1, j + 1) then element(i, j) must be 1.

Why this work? Our first step would determine the final state of the bottomest row. Then we go from bottom to top and finalize every row.

Subtask 6 (N <= 10^6)

Subtask 5 could give us some inspiration. For every combination of the state of the bottomest row, there is only one unique good matrix with that specific last row. (Don’t fully believe it first, think about it yourself)

For every (i, j) that the element is 1, it would turn every element in [j, j + (n - i)] of the last row to 1. (validate it with subtask 5).

We could divide the last row into some segments (without intersection). Notice every segment would then build up a sub-triangle in the final state, and every segment’s sub-triangle would be mutually exclusive. We could use some formula similar to subtask 1 to calculate the triangle area.

Subtask 6 (N <= 10^6)

Now the problem become how to find the segments. As N <= 10^6, we need to have some O(N) or O(N log N) method to find them. One way is to use delta array (partial sum).

For every (i, j) that element(i, j) is 1, we add 1 to A[j, j + (N - i)]. At last, the segments would be the consecutive element in A that is >= 1. Now problem reduced to how to add value to an interval efficiently. Delta array is a good trick to implement this. For every (i, j) that element(i, j) is 1, add 1 to delta[j] and -1 to delta[j + (N - i)]. Then iterate p from 2 to N and add delta[p - 1] to delta[p] for 1 < p <= N.

More about delta array (partial sum)
https://assets.hkoi.org/training2018/optimization.pdf

Full solution

Full solution is based on subtask 6, we have to find the segments more efficiently.

For every (i, j) that element(i, j) is 1, we transform them to segment [j, j + (N - i)]. Then we union segments if they have intersection.

How to do?

First we could sort the segment by their left boundary, then iterate all segment. We maintain a union segment [L, R]. Let the current iterated segment be [X, Y].

If X > R, this segment has no intersection with previous segments, we add segment [L, R] to our final segment set and change L to X and R to Y.

Full Solution

If X < R, we update R to max(R, Y), union this segment with previous segments.

We then use the final segments set to calculate answer just as subtask 6.