

FINAL EXAM 2011. – DAY #2

ALGORITHM DESCRIPTIONS

FINAL EXAM 2011. – DAY #2	Task RINGIŠPIL

Before we describe the main algorithm two observations are given:

Observation #1

Given a sequence of K rotations, final array can be created in $O(K^2)$ complexity. This reduces to simple interval handling.

Observation #2

Given a sequence of L rotations and position p , we can determine which element will be at position p after all L rotations are made. This is done by going through the rotations list in reverse order. Instead of pushing a number and changing its position while going forward, we can take a position and push it over the numbers while going backward - in that way we can determine which element 'lands' in position p .

Following the two observation we can create the whole algorithm:

While we are going through the sequence of operations and we determine a rotation operation, the only thing we do is that we add it to a list of pending rotations. When we encounter a print operation, we can determine the element to be printed from observation #2. However, naive implementation of this idea is of $O(q \cdot n)$ complexity, which is not fast enough. We can optimize in the following manner - after the list of pending rotations grows to \sqrt{N} size, using observation #1 we can 'push' the array of number to a new start state. Complexity of this step is $O((\sqrt{N})^2) = O(N)$. Total complexity is then $O(N \cdot \sqrt{N})$ (the array is rebuilt every \sqrt{N} steps) + $Q \cdot \sqrt{N}$ (for answering each query)).

Necessary knowledge:

Interval handling

Category

Data structures, ad hoc

FINAL EXAM 2011. – DAY #2	Task OGRLICA

At the beginning we can find all the important X and Y coordinates. Those are all coordinates which are set as coordinates for some rectangle vertices. In that set lets add -100000 and 100000. If now we observe only important coordinates, we get network in coordinate system.

To find distinces from entrance point of the park to any point in network, it is enough to move only on netork edges (edges that connects two X important coordinates and two Y important coordinates) and watch not to pass through any obstacle. That way we can find distances to every point in network from park entrance and park exit.

Now, it is necessary to find area of space by which dog could have moved. This is done by observing every rectangle in network of important coordinates (which doesn't contain obstacle). In every rectangle we find in which part of it dog have moved and that area is added to solution.

How is this calculated? For beginning, lets observe case when $T = 0$. It is easy to check if the dog could have moved through every vertex of rectangle as to look at calculated distances from every vertex to park entrance and park exit. Simple rule is applied: if dog could have been at every vertex of rectangle then area of rectangle is added to solution, otherwise he couldn't pass through current rectangle and adding area is 0.

More complicated case is when T isn't 0. In this case, problem is easier to be solved as to calculate area through dog could have not pass. That area is some kind of convex polygon. In the beginning for polygon whole rectangle is taken (-100000, 100000). Then, we see is dog could have moved through every vertex of rectangle. If he could then we'll calculate how far he could go from this vertex and by one line cut of part of polygon in vertex nearness. Similar, we see if dog could have move along some edge of rectangle and the apply same rule – cut by line. After all the cutting, we calculate area of left polygon by standard formula.

There are few details about lines of cutting left. These are calculated by little geometry. Whole complexity is $O(N^2 \log N)$.

Knowledge needed:

Geometry basics, Graph theory

Category

Geometry, graphs