Cartesian trees, contest analysis

Gleb Evstropov
glebshp@yandex.ru

November 14, 2016

Problems are listed in order of increasing estimated difficulty.

1 Oceanic Battle

- Mark all cells that can't contain ship. These are cells inside any of the ships and neighbouring. Each cell will be marked no more than 4 times, thus the total time is $O(nm)$.
- Find the subrectanlge consisting of zeroes with the maximum possible square.
- Use scanline, one-dimensional problem sounds: given an array a_i maximize the value of $\min\limits_{j \in [l,r]}(a_j) \cdot (r - l + 1)$.
- For every i find the nearest element to the left and the nearest element to the right the are strictly lesser. This can be done using the stack-based algorithm, same as building Cartesian tree in a linear time.

2 K-th maximum

Simply implement the Cartesian tree and store the size of the subtree in every node. To answer the query traverse down from the root, going left or right, whether there are enough nodes in the left subtree.

Also, as the problem is offline (i.e. all queries are known at the beginning) we can replace Cartesian tree with the segment tree.

3 Swapper

- Keep two separate Cartesian trees, one for even indices and one for odd.
- To swap even and odd positions, cut away corresponding segments from trees for odd and even, insert the segment from tree of odd to the tree of even and vice versa.

- To access the element just query the corresponding position in the corresponding tree.

4 Range Minimum Query

Implement implicit key Cartesian tree. In each node you should store the size of the subtree and the minimum element in the subtree. To process the query cut away the query segment and print the min field of the root node.

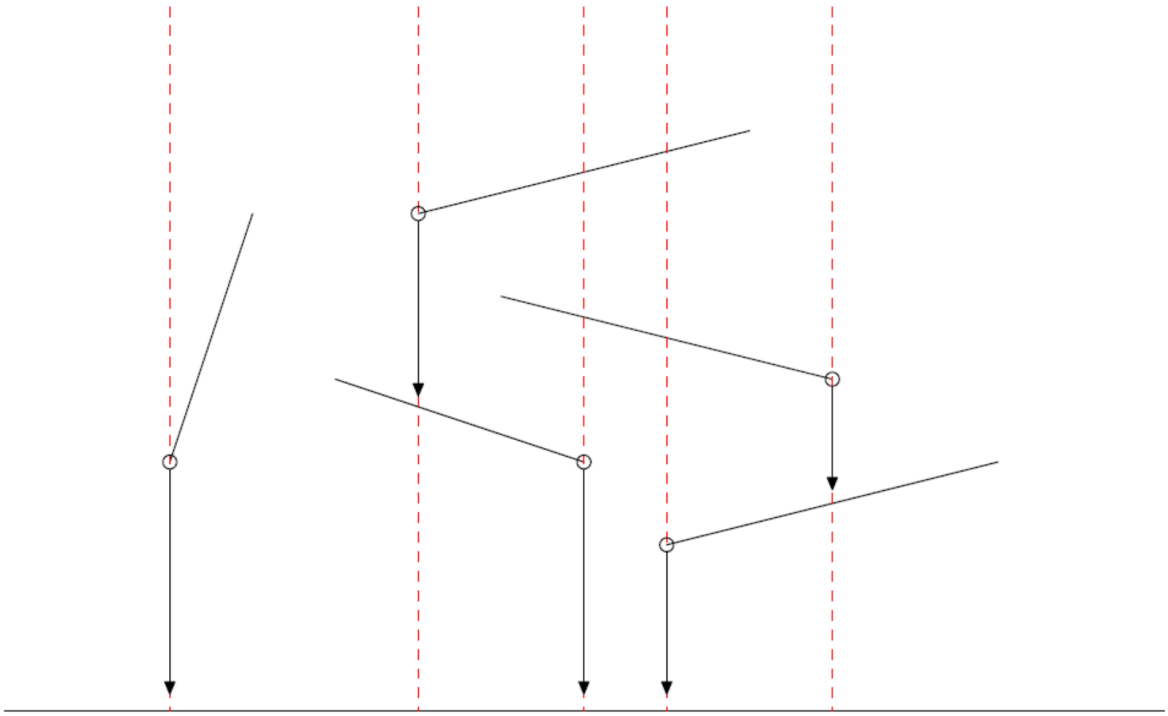
5 Permutations Strike Back

Implement two-dimensional segment tree, i.e. segment tree with a Cartesian tree in each node. The Cartesian tree in the each node of the segment tree stores all elements of the subtree, thus the total size of the segment tree is $O(n \log n)$ (each element present on each level exactly once).

- To change the element consider all Cartesian trees that contain this element. There are $O(\log n)$ such trees, one operation takes $O(\log n)$, thus the total worktime of one change operation is $O(\log^2 n)$.
- To query the number of element between l and r with values between x and y, split the segment (l, r) in fundamental segments (like in any segment tree query). Consider the Cartesian tree in each segments and ask the number of element between x and y. $O(\log^2 n)$ per one query.

6 Amber Ball

- We need to build the graph that for each segment i denotes whether the ball will fall to the ground or the index of first segment on its path after segment i.
- Scanline from left to right, keep the current order of obstacles for the given vertical line.
- Observation: relative order of two obstacles never changes because they do not intersect.
- Observation: while adding an obstacle or removing it from the Cartesian tree we can use the current x position in comparator.
- When we remove an obstacle query the first down to it, thus obtain the outgoing edge in the graph.
- When the position of the scanline matches one of the queries, the first obstacle in the Cartesian tree will be the first on the path of the ball.



7 Database

- Consider the elements in sorted order.
- Operation of the first type is equivalent to flipping the diagram with respect to the line $x = y$.
- Two consecutive operations of type one remain the sequence unchanged.
- Keep two versions of the Cartesian tree, carefully apply operations.