Segment tree contest
Problem analysis

Gleb Evstropov
glebshp@yandex.ru

November 12, 2015

1 Ski Race

The i-th man will start at the moment i and finish at the moment $i + w_i \cdot L$. We need to find the number of pairs (i, j) such that $i < j$ and $i + w_i \cdot L > j + w_j \cdot L$. The problem is equal to computing the number of inversions in the array.

Counting inversions can be done by modified merge sort, or by adding elements one by one and using binary search + segment tree for prefix sums.

2 Inverse RMQ

Set every element of a_k to the maximum q_i over all queries (l_i, r_i, q_i) such that $l_i \leq k \leq r_i$. This is the minimum possible value of a_k. Now check that all queries are satisfied. If some position k is not covered by any query we may set any value a_k.

For both phases a segment tree can be used. First phase is a set of `MaxOrEqual(l, r, x)` operations, that sets all $a_i = max(a_i, x)$ for $l \leq i \leq r$, while second phase consists of `GetMin(l, r)` operations only.

3 Smart Cheater

Use the linearity of expectation to reduce the problem to the following: get the subsegment of segment (l, r) of array a_i with the maximum possible sum.

Use the segment tree that stores the following values for every node v:

- ans — the best possible answer in the subtree of the node v;
- $pref$ — the maximum prefix sum of the corresponding segment;
- suf — the maximum suffix sum of the corresponding segment;

- sum — sum of all elements of the segment.

All the values can be recalculated using the following formulas:

- $ans(v) = max(ans(left), ans(right), suf(left) + pref(right)$;
- $pref(v) = max(pref(left), sum(left) + pref(right))$;
- $suf(v) = max(suf(right), suf(left) + sum(right))$;
- $sum(v) = sum(left) + sum(right)$.

4 Gym Class

Reformulate the problem: perform a single swap in order to minimize the number of inversions. Make the following observation:

- If $i < j$ and $a_i > a_j$ then j will never be a left element of the swap;
- If $i < j$ and $a_i > a_j$ then i will never be a right element of the swap;
- Find the sequence of possible left ends $pl_1, pl_2, \ldots, pl_k$ and possible right ends $pr_1, pr_2, \ldots, pr_m$;
- Each element k decreases the number of inversions by 2 if we choose $pl_i \le k \le pr_j$ and $a_{pl_i} \ge a_k \ge a_{pr_j}$;
- Every element gives -2 on some rectangle, so we need to find a point covered by the maximum possible number of rectangles.

5 Task Manager

Step 0: represent the tasks as points on the plane get rid of equal x_i and y_i by sorting pairs. Greedy algorithm to build lexmin topology sort of a directed graph:

- Find all sources, i.e. nodes that have no incoming arcs;
- Pick the source with minimal weight;
- Remove this source from the graph, add new sources;
- Time complexity is $O(|V| \cdot \log |V| + |E|)$.

In the given problem graph has $O(|V|^2)$ edges, hence we cannot just apply the above algorithm. What are the sources of the graph? Points (x_i, y_i), such that there is no point j that $x_i < x_j$ and $y_i < y_j$. Let's store all the sources sorted as pairs, i.e. the sequence $s_1, s_2, \ldots, s_k$ with condition $x_{s_i} < x_{s_{i+1}}$ and $y_{s_i} > y_{s_{i+1}}$.

When the minimal source is removed, some new point may be added to the list of sources, but no other points are deleted. To add new point we should answer the query `LeftAndGreater(x, y)` — the point i with maximum x_i that is lesser than x and $y_i > y$. Use this operation until new points are added to the sequence s_i.