Тураев Тимур, 504 (SE)
19.02.2014

1.4*. Задано множество точек, нужно уметь определять уровень точки относительно этого множества за O(logN), при этом точка во множество не добавляется.

Очень хочется попробовать свести эту задачу к первой, где мы умеет для «некоторого» множества определять уровень точки, относительно этого множества. [потом, правда, мы добавляли точку во множество, но теперь это делать не будем].

Предположим, нам поступила на вход новая точка T.

Возникают следующие проблемы: множество может быть какое угодно сложное, и хочется **выделить** из него такое, каким оно было бы, если бы мы решали первую задачу. Что это значит? Значит, хочется отсортировать все точки так как на тимусе: сначала по y-координате, затем (внутри группы) по x-координате. Далее, нужно выбрать то множество точек, которое уже было бы обработано перед оцениванием уровня новой точки T: это довольно просто сделать, используя дважды бинарный поиск (за O(logN)): сначала находим самую верхнюю из всех точек не выше T, затем (если их несколько) самую правую – получим точку X и после этого формируем новое множество S’ из всех точек, что строго ниже или левее точки X (назовем эту точку крайней). Таким образом задача будто бы свелась к первой: дано множество S’, отсортированное по y, затем по x, дана точка T и нужно определить уровень этой точки относительно множества S’. Как ее решать, мы знаем: строим AVL-дерево и имитируем вставку в это дерево новой точки, но по сути не вставляем, ибо таково условие.

Рассмотрим следующую точку: с ней нужно сделать тоже самое, со следующей также и так далее. Хочется не строить, а как-то уметь быстро находить то дерево, которое построилось бы по каждому множеству S’.

В прошлом семестре (да и на прошлом занятии) звучали слова «персистентные структуры». Так как наше множество S фактически никогда не меняется, можно отсортировать точки в порядке y-x (как в первой задаче) и строить из них персистентное дерево, постепенно добавляя вершины в это дерево в порядке сортировки. Персистентность обеспечивает сохранение всех версий дерева: а именно указатель на корень каждого AVL-дерева в каждый момент времени: так что если мы знаем крайнюю точку X (а мы ее найдем за логарифм), то получив указатель на корень дерева в момент, когда эта крайняя точка X была фактически “последней” добавленной, спуском по дереву мы найдем ответ на искомый вопрос. Спуск (в худшем случае) тоже отработает за O(logN), итоговое время – **O(logN)**.