Тураев Тимур, 504 (SE)

02.03.2014

**3.1. Задача про Гарри Поттера (acm.timus).**

Уже сдал в прошлой домашней работе.

**3.2.1. TIMUS\_1521 (Военные сборы-2) – Splay-tree**

В терминах сплей-дерева эта задача объясняется чуть проще, поэтому начну именно с этого.

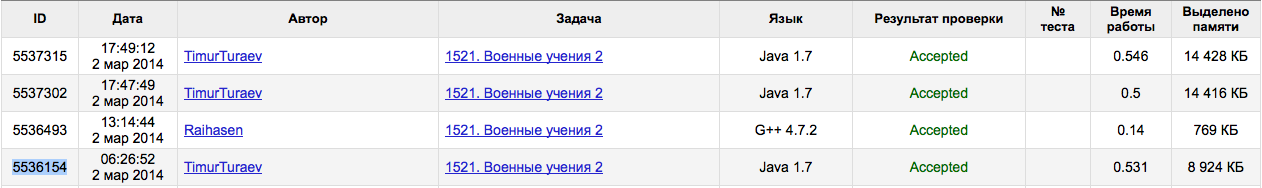
Заведем дерево, закинув туда все числа от 1 до N, причем дерево на неявном ключе. Затем удалим из дерева k-ый элемент: в этот момент вспомним как работает удаление элемента в сплей-дереве: сначала в дереве ищется и вытягивается наверх этот элемент, удаляется и два его поддерева смёрживаются. Во время удаления запомним что за элемент мы нашли (k-ый-то он по неявному ключу, то есть по порядку в массиве, а мы запомним какой номер в нем стоял – и выведем его в ответ), и еще одно число: размер левого поддерева удаляемого элемента. Это число будет означать сколько осталось неудаленных элементов, меньшего удаляемого. Обозначим его за А.

А зная это число легко вычислить где будет следующий k-ый элемент после удаленного: это А + k, по модулю длины оставшего массива. Возвращаемся в начало и повторяем все действия и так пока не кончится массив.

Удаление (и вставка) работает за O(logN), значит общая сложность O(NlogN).

***Код в приложении к письму в файле 1521\_splay.java***

Сплей-дерево в первой строке:



**3.2.2. TIMUS\_1521 (Военные сборы-2) – Treap**

Заведем декартово дерево на неявном ключе и закинем все наши элементы туда (можно даже не за линию, а путем N последовательных вставок, такое заходит).

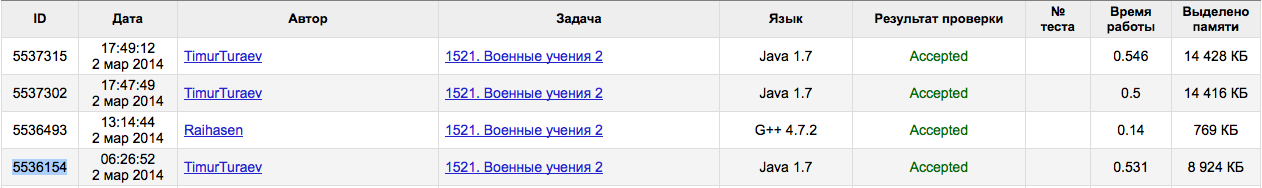
Идея та же: хочется при удалении знать сколько осталось слева живых элементов. А в декартовом дереве это тоже можно просто сделать, ведь как работает удаление? Сначала делаем сплит по «position – 1», затем по «первому» элементу (т.к. дерево на неявном ключе), получаем 3 элемента: левое поддерево left, искомый элемент и правое поддерево right. Затем склеиваем получившиеся деревья.

Искомое число А – размер left.

Сложность та же.

***Код в приложении к письму в файле 1521\_treap.java***

Решение декартовым деревом в последней строке:



**3.3. Swapper – Treap**

Как и хотел, я все-таки сдал эту задачу, написав декартово дерево на неявном ключе.

Теоретическое решение ровно такое же, какое я описывал в предыдущей домашней работе: заводим 2 дерева на четные и нечетные позиции, затем аккуратно определяем границы для каждого дерева и в зависимости от запроса делаем либо два сплита + merge, меняя местами деревья, либо выводим сумму на отрезке, путем опять же сплитов.

***Код в приложении к этому письму, в файле swapper\_treap.java***

Посылка <http://www.e-olimp.com/solutions/1372111>