Домашнее задание №5

Д.А. Першин

12 декабря 2014 г.

1 Словесное описание алгоритма

Пусть имеется входной массив операций длины n, каждый элемант которого представляет из себя тройку следующего вида: $(sign, l, r)$, где $sign$ - операция (1 - «+» , 0 - «-»), l и r - правая и левая координаты отрезка соответственно ($|l, r| < 1.000.000.000$). Необходимо после каждой операции вставки или удаления отрезка вывести общую длину объединения всех отрезков, лежащих на данный момент в множестве.

Для решения данной задачи выделим из входного массива все уникальные точки (левые и правые) и отсортируем их с помощью алгоритма $qsort$, полученный массив назовем p, его длина m, $m \leq n$. Далее из полученных точек найдем длины всех идущих подряд отрезков, имеющихся во входном множестве (т.к. точки отсортированы, просто найдем массив расстояний между соседними точками), временная сложность $O(n)$, назовем массив s, длина массива - $m - 1$. По полученным отрезкам построем дерево отрезков $stree$, каждая вершина которого будет состоять из трех чисел (len_{cur} - текущая длина отрезков в данном диапозоне без учета наложений; len_{max} - максимально возможная длина отрезков (требуется для отложенных обновлений), buf - буфер, будут описан далее), временная сложность $O(n)$. Дерево будем хранить слудующим образом: создадим массив, в котором для любой вершины с индексом i ее сыновьями будут вершины $2i$ и $2i+1$. Несложно заметить, что в данном случае в худшем случае может потребоваться массив длины $4n$ (кол-во листьев необходимо округлить до ближайшей степени двойки).

Для каждой операции из входного массива найдем диапозон отрезков в дереве отрезков, над которыми нужно совершить операцию $sign$. Для этого найдем координаты левой l и правой r точек в массиве p с помощью алгоритма $bsearch$, при этом координаты левого и правого отрезков диапозона, над которыми нужно совершить операцию $sign$ в дереве отрезков $stree$ равны l и $r - 1$ соответственно.

Так как для выполнения операции над диапозоном отрезком в дереве отрезков в худшем случае может потребовоться $O(n)$ операций ($O(n^2)$ для всех точек), будем использовать отложенные обновления: если необходимо совершить операцию над диапозоном отрезков и текущая вершина дерева отрезков входит в этот диапозон целиком, то при добавлении увеличим счетчик buf в вершине на 1, при удалении - уменьшим на 1, если нет - рассчитываем длину для текущего отрезка как сумму длин его левого и правого подотрезков, получая их длину рекурсивно, описаным выше способом. Т.к. в условии сказано, что удаляться будут только отрезки, которые перед этим были добавлены во множество, данный алгиритм будет работать корректно.

Можно заметить, что любой диапозон может быть разбит на три части: центральный, который полностью входит в один из диапозонов на текущем уровне в дереве и для рассчета суммарной длины которого потрубуется $O(1)$ операций, левый и правый на которые потрубуется не более $O(\log n)$ операций в худшем случае. В итоге получаем временную сложность $O(\log n)$ на одну операцию и $O(n \log n)$ на все операции вставки и удаления.

При получении общей длины объединения всех отрезков, лежащих на данный момент в множестве, будем использовать следующий алгоритм: если в вершине дерева отрезков счетчик $buf \neq 0$, то просто возвращаем максимальную длину len_{max} для данного диапозона отрезков, если нет - возвращаем len_{cur}.

Алгоритм:

1. Каждый входной элемент представим в виде тройки $(sign, l, r)$, где $sign$ - операция (1 - «+» , 0 - «-»), l и r - правая и левая координаты отрезка соответственно, запишем их в массив ops длины n.
2. Построим массив p уникальных точек длины m, $p = qsort(uniq(ops.l \cup ops.r))$.
3. Из полученных точек найдем длины всех идущих подряд отрезков s, $s_i = p_{i+1} - p_i, i \in [0, m-2]$.
4. Из полученных отрезков построим дерево отрезков $stree$:
    - (a) $current.len_{cur} = 0;$
$current.buf = 0;$
    - (b) если $left == right$:
$current.len_{max} = s[left]$
иначе $middle = (left + right)/2;$
    - (c) рекурсивно (b) для $left = left, right = middle;$
рекурсивно (b) для $left = middle + 1, right = right;$
    - (d) $current.len_{max} = child_l.len_{max} + child_r.len_{max};$
5. $\forall op \in ops$:

(a) Найдем левый отрезок в $stree\ s_{left} = bsearch(p, op.left)$, правый отрезок $s_{right} = bsearch(p, op.right) - 1$

(b) если $op.sign == 1$: $stree.add(s_{left}, s_{right}, 1, 0, m - 1)$
если $op.sign == 0$: $stree.sub(s_{left}, s_{right}, 1, 0, m - 1)$

где:

$stree.add(from, to, current, left, right)$:

- если $to < left$ или $right < from$
  то return $max(current.buf * current.len_{max}, current.len_{cur})$
- если $(from \leq left$ и $right \leq to)$
  то $current.buf + +$
  return $max(current.buf * current.len_{max}, current.len_{cur})$
- $middle = (left + right)/2$
  $current.len_{cur} = stree.add(from, to, 2 * current, left, middle) +$
  $stree.add(from, to, 2 * current + 1, middle + 1, right)$
  return $max(current.buf * current.len_{max}, current.len_{cur})$

где $current$ - текущая вершина, $left$, $right$ - левая и правая граница текущего поддерева; $from$, to - границы отрезков

для $stree.sub(from, to)$ аналогично, только $current.buf - -$

(c) $print(stree.sum())$

2 Доказательство корректнсти

Предположим, что при удалении отрезка значени буфера стало отрицательным, но это противоречит условию задачи о том, что удаляться будут только отрезки, которые перед этим были добавлены во множество.

3 Асимптотические оценки

В результате получаем сложность по памяти $O(n)$, так как мы используем входной массив длины n, массив точек, максимально возможная длина которого $2n$, массив отрезков длиной $2n - 1$ в худшем случае, дерево отрезков, кол-во вершин которого не превосходит $4n$, а также рекурсивные алгоритмы для построения дерева отрезков и выполнения операций над ним, глубина рекурсии не больше $\log n$. Сложность по времени равна в среднем $O(n \log n)$, так как мы используем алгоритм быстрой сортировки ($O(n \log n)$ в среднем, $O(n^2)$ в худшем случае), алгоритм бинарного поиска ($O(\log n)$) и выполнение n операций добавления в и удаления из дерева отрезков ($O(\log n)$, как было описано выше).