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, $m \leq 2n$ (т.к. точки могут повторяться). Также создадим массив map длины n, каждый элемент i которого - это пара вида $(left, right)$, где $left$ и $right$ - индексы левой и правой точки отрезка для операции i в массиве p (временная сложность $O(n)$). 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 i 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 заранее созданного массива map ($l_i = map[i]_{left}$, $r_i = map[i]_{right}$), 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 потребоваться $O(n)$ 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. Построим массив map длины n - массив соответствия концов отрезка операции i, $i \in [0, n-1]$ координатам точек в массиве p.

4. Из полученных точек найдем длины всех идущих подряд отрезков s, $s_i = p_{i+1} - p_i, i \in [0, m-2]$.

5. Из полученных отрезков построим дерево отрезков $stree$:

   (a) $current.len_{cur} = 0$;
   $current.buf = 0$;

   (b) если $left == right$, то:
   $current.len_{max} = s[left]$
   иначе $middle = (left + right)/2$;

(c) рекурсивно (b) для $left = left, right = middle$;
рекурсивно (b) для $left = middle + 1, right = right$;

(d) $current.len_{max} = child_l.len_{max} + child_r.len_{max}$;

6. для $op = ops[i]$, $i \in [0, n-1]$:

   (a) Найдем левый отрезок в $stree$ $s_{left} = map[i]_{left}$, правый отрезок $s_{right} = map[i]_{right} - 1$

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

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

   - если $to < left$ или $right < from$, то:
     - если $current.buf == 0$, то return $current.len_{cur}$
     - иначе return $current.len_{max}$;
   - если $from \leq left$ и $right \leq to$, то:
     $current.buf++$
     - если $current.buf == 0$, то return $current.len_{cur}$
     - иначе return $current.len_{max}$;
   - $middle = (left + right)/2$
     $current.len_{cur} = stree.add(from, to, 2 * current, left, middle) +$
     $stree.add(from, to, 2 * current + 1, middle + 1, right)$
     return $current.len_{cur}$

   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) $print(stree.sum())$,
   где $stree.sum()$ - общая длина объединения всех отрезков в множестве:

   - если $current.buf == 0$, то return $current.len_{cur}$
   - иначе return $current.len_{max}$,
     где $current$ - корень дерева отрезков

2 Доказательство корректнсти

Предположим, что алгоритм работает корректно для операции n. Докажем, что он работает корректно для операции $n+1$. При добавлении (удалении) диапозона отрезков из дерева отрезков возможны три случая:

- диапозон отрезков для добавления полностью содержится в диапозоне, покрываемом текущей вершиной дерева. В этом случае значение буфера увеличивается на 1, а при подсчете суммы в текущем диапозоне возвращается максимальное значение длины этого диапозона, что очевидно является корректным результатом. Следует заметить, что такой вершиной будет также являться любой лист дерева отрезков, для которого данное поведение также корректно (т.к. в данном случае максимальное значение длины диапозона равно длине самого отрезка).

- текущий диапозон отрезков для добавления частично содержится в диапозоне, покрываемом текущей вершиной дерева. В этом случае значение буфера не изменяется, а операция добавления рекурсивно вызывается для правого и левого поддерева, каждая из которых также удовретворяет одному из трех случаев и доказывается аналогично. При подсчете суммы в текущем диапозоне текущяя длина отрезков в это диапозоне вычисляеся как сумма длин в левом и правом поддереве текущей вершины, что очевидно является корректным результатом (исходя из структуры дерева отрезков), а возвращается максимальное значение длины этого диапозона, если значание буфера не равно 0 или текущяя длина отрезков в данном диапозоне, если буфер равен 0, что также является корректным результатом.

- диапозон отрезков для добавления не содержится в диапозоне, покрываемом текущей вершиной дерева. В этом случае ни значание буфера, ни текущяя длина отрезков в данном диапозоне не меняются, что очевидно является корректным поведением. При подсчете суммы в текущем диапозоне возвращается максимальное значение длины этого диапозона, если значание буфера не равно 0 (т.е. ранее добавлялся отрезок, полностью содержащийся в этом диапозоне), или текущяя длина отрезков в данном диапозоне, если буфер равен 0 (ранее не добавлялось отрезоков, полностью содержащихся в этом диапозоне), что так же является корректным результатом.

При удалении диапозона отрезков доказательство аналогичное, за исключением того, что из текущего значения буфера вычетается 1, следовательно значене буфера может стать нулевым или отрицательным. В первом случае при подсчете суммы в текущем диапозоне возвращается текущяя длина отрезков в данном диапозоне (т.к. буфер равен 0, то ранее не добавлялось отрезоков, полностью содержащихся в этом диапозоне). Получаем корректный результат. Во втором случае (отрицательный буфер) получаем некорректный результат. Однако, это противоречит условию задачи о том, что удаляться будут только те отрезки, которые перед этим были добавлены во множество. Следоватально, операция удаления также является корректной.

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

map длины n, 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), выполнение n операций добавления в и удаления из дерева отрезков ($O(\log n)$, как было описано выше).