Алгоритмы. Домашнее задание №12

Горбунов Егор Алексеевич

22 декабря 2015 г.

Задача №1 (Вертикальные и горизонтальные прямые)

Условие: даны n вертикальных и горизонтальных отрезков (всего n). Нужно посчитать число пересечений этих отрезков за $\mathcal{O}(n \log n)$.

Решение: пусть горизонтальные отрезки обозначены так: $(x_i, x_i'; y_i)$, где x_i — левая абсцисса горизонтального отрезка, x_i' — правая, а y_i — ордината, на которой отрезок находится. Вертикальные отрезки пусть заданы аналогично: $(y_j, y_j'; x_j)$. Разобъём каждый горизонтальный отрезок $(x_i, x_i'; y_i)$ на такие пары: $(x_i; y_i)$ и $(x_i'; y_i)$, где пара $(x_i; y_i)$ обозначает, что в точке x_i по оси абсцисс открывается отрезок на высоте y_i, а пара $(x_i'; y_i)$ обозначает, соответственно, что в точке x_i' по оси абсцисс такой отрезок закрывается. Отсортируем все полученные пары и вертикальные отрезки по x-координате, т.е. $(y_j, y_j'; x_j) < (x_i; y_i) \iff x_j < x_i$. Теперь будем перебирать пары и вериткальные отрезки в отсортированном порядке, т.е. будем просто идти слева направо по оси абсцисс сканирующей прямой, перпендикулярной оси абсцисс. Обходя таким образом мы будем последовательно натыкаться либо на штуки вида $(x_i; y_i)$ либо $(x_i'; y_i)$ либо $(y_j, y_j'; x_j)$. Сейчас мы скажем, что будем делать в каждом случае и сразу увидим, что задача решена. Заметим: 1) чтобы посчитать число всех пересечений нам нужно посчитать отдельно число пересечений каждого вертикального отрезка с горизонтальными 2) пускай идя сканирующей прямой мы наткнулись на вертикальный отрезок $(y_j, y_j'; x_j)$, тогда ясно, что число его пересечений с горизонтальными отрезками равно число открытых горизонтальных отрезков (тех отрезков, до концов которых сканирующая прямая ещё не дошла) на высоте от y_j до y_j'. Научимся быстро отвечать на вопрос о таком количестве: будем поддерживать сбалансированное дерево T поиска с операцией $numKeys(a, b)$ нахождения числа вершин (ключей) в диапазоне от a до b, мы умеем это делать быстро, т.е. запрос о числе вершин в диапазоне занимает $\mathcal{O}(\log n)$. Таким образом предлагается следующая тактика действий при обходе сканирующей прямой:

1° Встретили $(x_i; y_i)$ (отрезок открывается), тогда выполняем $T.add(y_i)$

2° Встретили $(y_j, y_j'; x_j)$, тогда выполняем $sum = sum + T.numKeys(y_j, y_j')$

3° Встретили $(x_i'; y_i)$ (отрезок закрывается), тогда выполняем $T.delete(y_i)$

Видим, что число открытых в данный момент отрезков на высоте от a до b равно числу ключей в дереве T в диапазоне от a до b, т.к. ключи в дерево добавляются только, когда встречаем открытие отрезка, а удаляются сразу, как встречаем точку, закрывающую отрезок. В конце концов (по окончании обхода сканирующей прямой) будем иметь ответ на вопрос задачи, записанный в переменной sum.

Сложность = время на сортировку + (число границ отрезков) * (время операции $T.add$ или $T.delete$ или $T.numKeys$) = $\mathcal{O}(n \log n)$ ■

Задача №2 (Число различных на отрезке)

Дан массив $A[1..n]$ нужно отвечать на запросы о количестве различных чисел на отрезке $[l..r]$

(a) Отсортируем все запросы по правому концу r. Будем теперь двигаться по массиву слева направо параллельно поддерживая следующий массив: $B[1..n]$: $B[i] = 1$, если справа от i нет элемента равного $A[i]$, иначе $B[i] = 0$. Заметим, что при переходи от элемента i к $i+1$ в массиве B происходят следующие изменения: $B[i+1] = 1$ и, если $A[k], k < i+1$ — это последнее появление элемента равного $A[i+1]$ в $A[1..i]$, то $B[k] = 0$. Таким образом, чтобы поддерживать массив B нам нужно дополнительно хранить дерево L (конечно, сбалансированное!), ключами в котором будут значения массива A, а значениями — индекс последней встречи ключа к текущему моменту. Заметим следующее: пусть мы в данный момент прошли по A префикс $A[1..i]$ и у нас есть запрос на число различных элементов в подмассиве $A[l..i]$, тогда ответом на него будет *число единиц в* $B[l..i]$. Это верно, т.к. чило различных элементов в $A[l..i]$ можно посчитать идя по $A[l..i]$ с конца и прибавлять 1 к ответу каждый раз как встречаем новенькое число, легко видеть, что прибавлять единички мы будем именно в тех местах, где они стоят в $B[l..i]$. Итого будем строить сбалансированное T дерево по массиву B последовательно, по мере обхода A. Ключами в этом дереве будет i, а значениями $B[i]$, соответственно $sum(B[l..i]) = T.sumPrefix(keyFrom = l)$. Последний запрос умеем выполнять за $\log n$. Итого:

   (a) Сортируем запросы по r

   (b) $for\ i\ from\ 1\ to\ n$: — $\mathcal{O}(n)$

   (c) $B[i] = 1, T.add(key = i, value = 1)$
   $k = L.getValByKey(A[i]),\ B[k] = 0,$
   $T.setValue(key = k, val = 0), L.setValByKey(A[i], i)$
   — $\mathcal{O}(\log n)$
   (последние 3 присваивания происходят только если такой k нашёлся)

   (d) запросы просматриваются вместе с проходом по A (это не ломает линейной асимптотики обхода), если запрос $[l, r]$ таков, что $r == i$, то отвечаем на него:
   $T.sumPrefix(keyFrom = l)$ — $\mathcal{O}(\log n)$. $sumValues$ в данном случае эквивалентно

суммированию на подотрезке в массиве B или подсчёт числа единиц :).

Видно, что асимптотика на запрос получилась логарифмическое, т.е. $\mathcal{O}(\log n) = \mathcal{O}(\log^2 n)$ ■

(b) См. пункт (a) ■

(c) Для того, чтобы решить задачу $online$ прибегнем к персистентности. Построим за $\mathcal{O}(n \log n)$ n сбалансированных деревьев T_r, что запросы вида $[l, r]$ будут обрабатываться так: $T_r.sumPrefix(l)$. Т.е. мы, как и в пред. пунктах, пройдёмся по массиву на каждом шаге выделяя новое дерево. На предподсчёт потратится именно $\mathcal{O}(n \log n)$ в силу сбалансированности дерева. ■

Задача №3 (Котики)

Дан набор из отрезков и котики...

(a) Будем хранить в сбалансированном дереве T по ключу i время, когда котик, сидящий на i-ой клетке ушёл. У нас есть n клеток и пусть вначале в этом дереве по всем ключам $1..n$ значения -1 — кот не уходил.

1) Пусть котики уходили с 1 по m моменты времени.
будем хранить $ANS[k]$ — число отрезков, которые стали свободны после ухода котика в момент времени k

2) Поддержим операцию $T.min(a, b)$ и $T.max(a, b)$, ведь мы умеем это делать. И заметим, что если в какой-то момент $T.min(a, b) = -1$, то на отрезке $[a..b]$ сидит котик. Если же это число не равно -1, то $T.max(a, b) = t'$ — время, когда с отрезка ушёл последний котик

3) Теперь для каждого отрезка $[l..r]$ найдём время t' (при поищи T), в которое с отрезка ушёл последний котик и сделаем $ANS[t']++$ (если $t' \neq -1$)

Таким образом видим, что в ANS у нас записан ответ на вопрос задачи в каждый момент времени. Первым этапом мы строим дерево T по запросам: $\mathcal{O}(n \log n)$
Далее мы проходимся по отрезкам и выполняем с каждым действие за $\mathcal{O}(\log n)$

Итого время: $\mathcal{O}(n \log n + m \log n)$, где m — число отрезков. Но время на запрос (обработку отрезка) равно $\mathcal{O}(\log n)$ ■

Задача №4 (k-статистика (числа различны))