Алгоритмы. Домашнее задание №4

Горбунов Егор Алексеевич

9 октября 2015 г.

Задача №1 (унимодальный массив)

(a) *Поиск пика за $\mathcal{O}(\log n)$ в унимодальном массиве $A[1..n]$*
Рассмотрим элементы $a = A[\frac{n}{2}]$ и $b = A[\frac{n}{2}+1]$. Если $a < b$, то пик находится где-то справа от b или в b, т.к. мы попали на «подъём», иначе, если $a > b$, то мы попали на «спуск» и пик точно находится левее a или в a. По разные стороны от пика 2 соседних элемента a и b находиться не могут. Таким образом можно продолжить поиск пика в одной их выделенных частей массива, размер которой $= \frac{n}{2}$. Поиск завершится тогда, когда будут выбраны последние возможные 2 элемента и из них будет выбран наибольший. Время работы такого поиска: $T(n) = T(\frac{n}{2}) + \mathcal{O}(1) = \mathcal{O}(\log n)$
PS: вообще говоря можно выбирать любые 2 элемента, меньшие n в разы, и сравнивать их. Например $A[\frac{n}{3}]$ и $A[\frac{2n}{3}]$. Ясно, что если первый меньше второго, то пика уже точно нет левее $\frac{n}{3}$, иначе, если второй больше первого, то правее $\frac{2n}{3}$ пика уже точно нет. Таким образом размер задачи уменьшается на треть: $T(n) = T(\frac{2n}{3}) + \mathcal{O}(1) = \mathcal{O}(\log n)$

(b) *Найти минимальный bounding box у выпуклого многоугольника за $\mathcal{O}(\log n)$*
Нам дан массив вершин $V[1..n]$ многоугольника в порядке обхода по часовой стрелке, причём никакие 3 подряд идущие вершины не лежат на одной прямой. Ясно, что чтобы найти минимальный ограничивающий прямоугольник, нужно найти:

$$x_{min} = min(V[i].x),\ x_{max} = max(V[i].x),\ y_{min} = min(V[i].y),\ y_{max} = max(V[i].y)$$

Тогда ответом на задачу будет прямоугольник, построенный на вершинах (x_{min}, y_{min}), (x_{max}, y_{max}), которые задают его диагональ (стороны прямоугольника параллельны осям координат).
Посмотрим на данный нам обход многоугольника по часовой стрелке, но будем рассматривать только координату x каждой точки обхода. Ясно, что по мере обхода, эта координата сначала увеличивается (уменьшается) до самой правой (самой левой) по оси абсцисс точки, а потом начинает уменьшаться (увеличиваться) до самой левой (правой) по оси абсцисс точки, ну а далее опять может начать увеличиваться (уменьшаться), пока обход не приведёт нас к последней точке обхода. Для удобства добавим замкнём обход, поставив в конец первую точку. Удобно представлять обход, как показано на рисунку

ниже: по оси ординат отложено значение $v.x$, а по оси абсцисс (влево) идут индексы точки обхода.

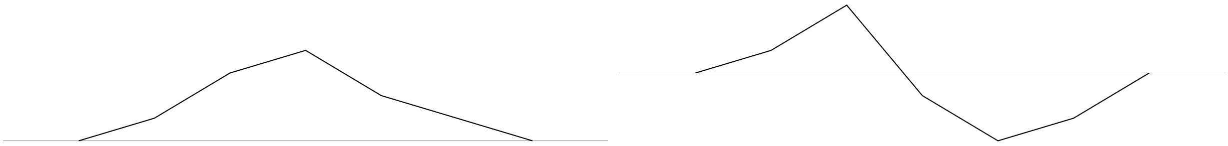

Рис. 1: удобное представление массива обхода многоугольника по координате

Наша задача тогда — найти моды на в таком массиве обхода. В обходе у первой и последней вершины совпадают координаты (т.к. вершины одинаковы).
Делаем так: рассмотрим 3 точки ($a = V[\frac{n}{2}], b = V[\frac{n}{2} - 1], c = V[\frac{n}{2} + 1]$) посередине массива обхода. В силу того, что никакие 3 не лежат на одной прямой найдётся такая точка, что её x-координата отлична от x-координаты начальной (конечной) точки обхода. Пусть это точка b. Если $b.x > V[1].x = V[n].x$. Пик может находиться справа от b или слева от b. В свою очередь точка b находится на «склоне», а значит, если от точки b мы пойдём в сторону «подъёма», то придём к пику уж точно, т.к. где-то нужно будет начать спускаться опять к значению $V[1].x$. Сторону подъёма мы легко можем определить за $\mathcal{O}(1)$ в силу того, что нет 3 точек лежащий на одной прямой. Таким образом мы будем делить задачу на двое и найдём в итоге 1 пик за $\mathcal{O}(\log n)$. Второй пик найдётся элементарно за $\mathcal{O}(\log n)$, т.к. мы его поищем алгоритмом поиска пика в унимодальном массиве слева и справа от уже найденного пика на первом шаге.
Так мы найдём x_{min} и x_{max}. Аналогично ищутся y_{min} и y_{max}. Итоговая сложность $\mathcal{O}(\log n)$
■

(c) *Принадлежность точки выпуклому многоугольнику за* $\mathcal{O}(\log n)$

Задача №2 (Амортизационное время работы счётчика)

Будем обозначать счётчик: cnt

1. Увеличение счётчика на 1
   Введём функцию потенциала: $\phi(cnt)$ = *Число единиц в двоичной записи* cnt
   Пусть длина двоичной записи cnt равна n. И пусть в cnt первые k младших разрядов заполнены единицами ($k+1$ разряд заполнен 0, а остальные какие-то...). Ясно тогда, что число операций, которое нужно затратить на инкремент равно $k+1$ — обнулить первые k младших разрядов и вставить в $k+1$ единицу. Таким образом амортизированное время работы операции инкремента:

$$T_{amortized}(n) = T_{real}(n) + \phi(cnt+1) - \phi(cnt) = k + 1 + \phi(cnt) - k + 1 - \phi(cnt) = 2 = \mathcal{O}(1)$$

2. Уменьшение счётчика на 1
   Аналогично увеличению, только потенциальная функция будет следующей:

$$\phi(cnt) = \textit{Число нулей в двоичной записи } cnt$$

3. Сравнение счётчика с 0

   Т.к. на память ограничений в условии задачи я не вижу, то мы можем эффективно поддерживать индекс самой левой единицы в двоичной записи числа cnt. Т.к. при операциях инкремента и декремента он как максимум может сдвигаться на 1 влево или вправо. Таким образом для сравнения с 0 счётчика достаточно посмотреть на этот индекс и если он невалидный (напрмер, -1), То счётчик нулевой, иначе нет.

$\blacksquare$

Задача №3 (Скошенная система счисления)

(a) *неотрицательное целое число единственным образом записывается скошенной системе счисления*

Пусть это не так и $n \geq 0$ такого, что: $n = \sum_{i=1}^{l_1} a_i(2^i - 1)$ и $n = \sum_{i=1}^{l_2} b_i(2^i - 1)$, пусть $l_1 > l_2$, тогда:

$$\sum_{i=1}^{l_2} (a_i - b_i)(2^i - 1) + \sum_{i=l_2+1}^{l_1} a_i(2^i - 1) = 0$$

Заметим, что $\sum_{i=l_2+1}^{l_1} a_i(2^i - 1) \geq 2^{l_2+1} - 1$ (только $a_{l_2+1} \neq 0$), а $\sum_{i=1}^{l_2} (a_i - b_i)(2^i - 1) \leq \sum_{i=1}^{l_2} (2^i - 1) = 2^{l_2+1} - 2 - l_2$ (тут все $b_i = 0$, а $a_i = 1$, с поправкой на то, что лишь один элемент a_i может быть $= 2$ и хотя бы один b_i не нулевой =)). Видим, что так нуля не может в сумме получиться, а значит $l_1 = l_2$. Т.е. получили, что у любых 2-х записей числа в скошенной системе счисления одинаковая длина! Докажем далее по индукции, что такая запись единственна. База проверяется элементарно для $n = 0, n = 1$. Пускай верно для чисел вплоть до $n - 1$, докажем для n. Опять от противного: пусть есть 2 записи числа (уже одной длины):

$$n = \sum_{i=1}^{l} a_i(2^i - 1) = \sum_{i=1}^{l} b_i(2^i - 1)$$

Если $a_l = 0$, то и b_l равно 0, т.к. мы уже показали, что у чисел одинаковая длина.

Если $a_l = 2$, то $\forall\ i \in \{1, \ldots, l_2 - 1\}\ a_i = 0$, т.е. $n = 2^{l+1} - 2$. Но тогда b_l тоже равно 2, т.к. самое больше число, которое можно получить не приравнивая b_l к 2 это: $\sum_{i=1}^{l} (2^i - 1) = 2^{l+1} - 2 - l$, а это меньше n, что невозможно. Таким образом получили, что $a_l = b_l$. Заметим тогда, что мы получили 2 различных записи числа $n - a_l(2^l - 1) < n$, что противочит индукционному предположению. А значит, запись числа n единственна. $\blacksquare$

(b) *Инкремент скошенного числа за $\mathcal{O}(1)$*

Пусть $n = \sum_{i=1}^{l} a_i(2^i - 1)$, причём a_k — первый не равный 0 элемент. Тогда как найти $n + 1$:

(a) Если $a_k = 1$ и $k > 1$

$$n + 1 = \sum_{i=k}^{l} a_i(2^i - 1) + 1 = a'_1 + \sum_{i=k}^{l} a'_i(2^i - 1)$$

$a'_1 = 1$, а остальные $a'_i = a_i$

(b) Если $a_k = 1$ и $k = 1$

$$n + 1 = a_1 + \sum_{i=2}^{l} a_i(2^i - 1) + 1 = a'_1 + \sum_{i=2}^{l} a'_i(2^i - 1)$$

$a'_1 = 2$, а остальные $a'_i = a_i$. Свойства скошенных чисел не нарушены, т.к. это первая 2 в записи и за нею ничего нет. (Другой двойки быть не могло, т.к. a_k — первый ненулевой знак).

(c) Если $a_k = 2$

Тогда

$$\begin{aligned} n + 1 &= 2(2^k - 1) + a_{k+1}(2^{k+1} - 1) + \sum_{i=k+2}^{l} a_i(2^i - 1) + 1 = \\ &= 2^{k+1} - 2 + 1 + a_{k+1}(2^{k+1} - 1) + \sum_{i=k+2}^{l} a_i(2^i - 1) = \\ &= (a_{k+1} + 1)(2^{k+1} - 1) + \sum_{i=k+2}^{l} a_i(2^i - 1) \end{aligned}$$

Получили корректную запись числа. $a_{k+1} \leq 1$ по свойству скошенного числа.

Итого мы видим, что для операции инкремента нужно в первых двух случаях просто увеличить младший разряд, а в 3-ем случае (если первый ненулевой знак = 2) нужно обнулить бит, где стоит двойка и увеличить следующий за ним бит. Ясно, что это реализуемо за $\mathcal{O}(1)$, если хранить дополнительно индекс двойки в числе, а само число хранить в массиве. На каждый инкремент тогда мы точно знаем что менять и меняем это за $\mathcal{O}(1)$. ∎

Задача №4 (-)

Задача №5 (-)

Задача №6 (Персистентный список символов с доступом, откатом и добавлением $\mathcal{O}(\log n)$)

Такую структуру данных можно реализовать используя любое дерево поиска, которое будет обладать логарифмической сложностью на добавление и поиск элемента. Для каждого момента времени i храним в массиве корень дерева поиска T_i. В корне T_i храним число элементов в дереве $T_i.size$. Каждая вершина дерева T_i представляет из себя пару $(char, index)$, причём ключом дерева T_i, по которому производится поиск, является $index$ — это и есть индекс такого символа в массиве. Пусть вся система находится на моменте времени i. Добавление символа требует пройти по дереву поиска вглубь затратив $\mathcal{O}(\log n)$ операций и, соответственно, изменению подвергается не более $\log n$ вершн. Те вершины, которые во время добавлению изменяются, копируются в новое дерево T_{i+1} с изменениями, а остальные вершины сохраняются и остаются, как в дереве T_i. Таким образом мы затрачиваем на добавление $\mathcal{O}(\log n)$ операций. На вытаскивание i элемента тратится столько же, т.к. дерево поиска нам это обеспечивает, а откат производится за $\mathcal{O}(1)$, т.к. все корни хранятся в массиве. ■