Алгоритмы и структуры данных
Контрольная работа №1
Вариант 26

Иван Алексеев М3139

17.11.2020

Задача 1

По определению:

$$f = \Omega(g) \Leftrightarrow \exists N > 0, C > 0 \;\; \forall n \geqslant N : f(n) \geqslant C \cdot g(n)$$

По условию $f(n) = 13n^2 - 16n\sqrt{n}\log^2 n, \;\; g(n) = n \log n.$

$$13n^2 - 16n\sqrt{n}\log^2 n \geqslant C \cdot n \log n \Leftrightarrow 13n - 16\sqrt{n}\log^2 n \geqslant C \cdot \log n$$

$$\frac{16\sqrt{n}\log^2 n + C \cdot \log n}{13} \leqslant n$$

$$\frac{16\sqrt{n}\log^2 n + C \cdot \log n}{13} \leqslant \frac{16\sqrt{n}\log^2 n + C \cdot \sqrt{n}\log^2 n}{13} = \frac{(16 + C)\sqrt{n}\log^2 n}{13}$$

Пусть $C = 10$:

$$\frac{(16 + C)\sqrt{n}\log^2 n}{13} = 2\sqrt{n}\log^2 n \leqslant n \Leftrightarrow 4\log^4 n \leqslant n$$

Но из матанализа, мы знаем, что $\exists N > 0 : \forall n > N : 4\log^4 n \leqslant n$ (значит, что $\frac{16\sqrt{n}\log^2 n + C \cdot \log n}{13} \leqslant n$ тоже выполняется при таком N).

Значит при $C = 10$ $\exists N > 0 : \forall n > N :$

$$13n^2 - 16n\sqrt{n}\log^2 n \geqslant C \cdot n \log n$$

, ч.т.д.

Задача 2

Докажем по индукции, что $T(n) \leqslant 6n$.

База:

$$T(0) = 0 + 0 + 0 \leqslant 6 \cdot 0 = 0$$

$$T(1) = 0 + 0 + 1 = 1 \leqslant 6 \cdot 1 = 6$$

Пусть для $\forall x < n : T(x) \leqslant 6x$. Докажем, что верно и для n.

Тогда $T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \leqslant 6 \cdot \frac{n}{2} + 6 \cdot \frac{n}{3} + n = 3n + 2n + n = 6n$

Также легко заметить, что так как $T(n) \geqslant 0$, то:

$$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \geqslant n$$

Значит $T(n) = \Theta(n)$.

Задача 3

Давайте сначала рассмотрим задачу попроще – нахождение подотрезка массива с максимальной суммой (фактически изначальная задача, но без ограничений на длину подотрезка).

Давайте предпосчитаем префиксные суммы для данного массива (массив s). Рассмотрим r – правую границу подотрезка. Зафиксируем её. Пусть это правая граница нашего подотрезка. Тогда, чтобы добиться максимальности суммы на подотрезке $[l; r]$ нам нужно чтобы

$s[r] - s[l-1]$ было максимальным, но это фактически минимизация $s[l-1]$. Проитерировав по r и взяв максимум от всех максимумов на итерируемых подотрезках мы найдем ответ.

Однако у нашей задачи есть ограничение на размер подотрезка. Давайте предподсчитаем минимум в плавающем окне размера $b - a + 1$ для всего массива (это делается с помощью двух стеков с поддержкой минимума).

Теперь ещё раз зафиксируем r – правую границу подотрезка. Заметим, что искомая левая граница $(l-1)$ не может лежать на расстоянии ближе a влево (из-за наших ограничений на минимальный размер a). Также она не может находиться дальше чем b (из-за ограничений на максимальный размер b). Здесь мы можем воспользоваться заранее предподсчитанным минимумом в окне, решая задачу. Задача решена.

(Единственным нюансом будет случай, когда длина отрезка $s[0; r]$ меньше длины плавающего окна, но этот случай решается простым предподсчетом минимума на префиксе.)

Задача 4

При выборе не центрального элемента при разделении в бинпоиске мы будем делать в худшем случае больше чем $\log_2 n$ сравнений (при выборе большей области). Значит для поиска верхнего или нижнего вхождения x нам понадобится $\log_2 n$ сравнений.

При поиске первого индекса (i или j) мы хотя бы раз воспользуемся бинпоиском, так как он оптимальный. Теперь, в худшем случае нам понадобиться $\log_2 n$ сравнений для поиска второго индекса. Учтем, что число n может быть не степенью двойки, и нам понадобиться какая-то малая константа. Тогда количество операций для поиска i и j – $2\log_2 n + O(1)$.

Задача 5

Тонкое дерево – дерево, полученное из биномиального удалением у некоторых вершин ребенка максимального ранга. Обозначим за $D(x)$ количество детей вершины x, за $R(x)$ ранг вершины x.

Проведя аналогию с Фибоначчиевыми кучами, мы можем рассмотреть следующие свойства тонких деревьев:

1) Любой узел x дерева либо тонкий ($D(x) = R(x) - 1$), либо не тонкий ($D(x) = R(x)$).

2) Корень не тонкий.

3) Для любого узла x ранги его детей справа налево равны соответственно $0, 1, 2, \ldots, D(x) - 1$.

4) Узел x тонкий $\Leftrightarrow R(x)$ на 2 больше, чем ранг его самого левого сына, или его ранг равен 1, и он не имеет детей.

Теперь приступим к самой операции DecreaseKey. При уменьшении ключа мы можем нарушить свойтства кучи, тогда давайте перенесем поддерево с корнем в уменьшаемом элементе в корневой список. Обновим минимум в тонкой куче.

Теперь у нас останутся нарушения только двух типов:

1) Нарушение третьего свойства тонкого дерева.

2) Нарушение первого или второго свойства тонкого дерева.

Назовем узел x узлом первого нарушения среди детей узла p, если $R(x)$ отличается от ранга его ближайшего правого брата на 2, либо он не имеет правого брата и его ранг равен 1.

Пусть x — узел первого нарушения.

1) Если x не тонкий, тогда поместим поддерево с корнем в самом левом сыне узла x на место пропущенного в списке его братьев. Тогда x станет тонким.

2) Если x тонкий, тогда уменьшим $R(x)$ на 1. Узлом Локализации Нарушения будет либо родитель x, либо левый брат. Нарушение станет станет нарушением второго типа.

Теперь у нас остались только нарушения второго типа. Попробуем от них избавиться.

Назовем узел x узлом второго нарушения, при выполнении одного из условий:

1) $R(x)$ на три больше, чем ранг его самого левого сына.

2) $R(x) = 2$, и x не имеет детей.

3) x – тонкий корень дерева.

Пусть x — узел второго нарушения, узел p – его родитель. Переместим поддерево с корнем в x в корневой список и уменьшим $R(x)$.

1) Если x не старший брат, перейдем к его левому брату (Нарушение станет нарушением второго типа).

2) Если x был старшим братом возможно два случая. Если p не тонкий, пометим его тонким. Иначе p – новый узел второго нарушения. Перейдем к нему.

Данный алгоритм или остановится, или дойдет до корня дерева. В это случае сделаем ранг корня на 1 больше ранга его самого левого сына.

С каждым шагом алгоритма мы уменьшаем количество тонких узлов на 1 и добавляем не больше одного дерева в корневой список.

В этом случае потенциал уменьшается минимум на 1 и амортизированная стоимость становится $O(1)$.

Так как мы перемещаемся либо вверх либо влево по дереву, операций будет выполнятся не более $O(\log n)$.

Задача 6

Пункт 1

v	0	1	2	3	4	5	6
p_v	0	3	3	3	4	3	3
r_v	0	0	0	2	0	0	0

Пункт 2

v	0	1	2	3	4	5	6	7	8	9
h_v	0	1	3	4	3	6	7	7	7	5

Пункт 3

Ответ: b, e, f, g, h

Пункт 4

Ответ: нет правильных ответов.