Структуры

Задача 1. Дан квадрат с крайними точками в $(0;0)$ и в $(C;C)$, а также n точек внутри него. Найти максимальный квадрат, находящийся внутри этого квадрата, в котором не лежит ни одна из данных точек.

а) $\mathcal{O}(n \log n \log C)$;

б) $\mathcal{O}(n \log n)$.

Задача 2. Дано n прямоугольников со сторонами, паралелльными осям координат. Прямоугольники не пересекаются, но могут вкладываться. Вы можете перемещаться по плоскости, но, пересекая границы прямоугольника i, необходимо заплатить налог c_i. Необходимо в онлайне ответить на q запросов: «Вы начинаете в точке $(x_1; y_1)$, хотите попасть в точку $(x_2; y_2)$. Сколько вам придётся заплатить?». $\mathcal{O}((n+q)\log n)$.

Задача 3. Дано n прямоугольников со сторонами, паралелльными осям координат. Прямоугольники не пересекаются, но могут вкладываться. У прямоугольника i есть цвет c_i. Необходимо ответить в онлайне на q запросов, каждый из которых бывает двух видов:

1) Изменить какое-то c_i;

2) Найти количество различных цветов прямоугольников, вложенных в i-й.

За такую асимптотику:

а) $\mathcal{O}((n+q)\log^2 n)$;

б) $\mathcal{O}((n+q)\log n)$.

Задача 4. Дан *отсортированный* массив длины n. Ответить в онлайн на q запросов, каждый из которых бывает двух типов:

1) Сделать $a_i = \min(a_i, x)$ для всех $i \le r$;

2) Найти сумму на отрезке $[l; r]$.

Задача 5. Дан массив длины n. Отвечать в онлайн на q запросов, каждый из которых бывает двух типов:

1) Сделать $a_i = \min(a_i, x)$ для всех $l \le i \le r$;

2) Найти сумму на отрезке $[l; r]$.

Задача 6. Дан массив a из n чисел. Рекорд — позиция i такая, что для всех $j < i$: $a_j < a_i$. Поступают q запросов: установить значение i-го элемента равным x. После каждого запроса необходимо узнать количество рекордов. Ответить на запросы за время $\mathcal{O}((n+q)\log^2 n)$.

Задача 7. Дано дерево на n вершинах. Требуется выполнить некоторый предподсчёт за $O(n \log n)$, после чего требуется обрабатывать запросы следующего вида: даёт некоторое множество из $2k$ вершин (k может быть различное для разных запросов), требуется определить, можно ли удалить из дерева одно ребро таким образом, чтобы в каждой из образовавшихся двух компонент связности было ровно k вершин данного множества. Время ответа на запрос должно составлять $O(k \log k)$.

Задача 8. Имеется n урн, изначально в каждой урне находится по одному шару. Последовательно выполняется n шагов, на i-м шаге выбираются некоторые три числа a_i, b_i и l_i, такие что $a_i + l_i \le n$ и $b_i + l_i \le n$. Затем, одновременно для всех x, таких что $a_i \le x \le a_i + l_i$, шары из урны x перекладываются в урну $b_i + (x - a_i)$. Для каждого шара определите, в какой урне он в итоге окажется за время $O(n \log n)$.

Задача 9. Рассмотрим уже хорошо известную нам задачу. Дана перестановка $p_1, p_2, \ldots, p_n$ и двумерные запросы «количество элементов на отрезке от i до j со значениями от x до y». Пусть теперь также имеются запросы обмена местами двух соседних элементов перестановки p_i и p_{i+1}. Требуется выполнить некоторый предподсчёт за время $O(n \log n)$ и отвечать на оба типа запросов за время $O(\log n)$.

Красивые теоретические задачи

Которые вы будете решать когда пойдёте в одно Высшее учебное заведение

Задача 10. Имеется последовательность $a_1, a_2, \ldots, a_n$. Также есть m запросов прибавления арифметической прогрессии. Каждый запрос характеризуется четырьмя числами k, x, l и r, что означает, что к a_l прибавляется x, к a_{l+1} прибавляется $k + x$, и так далее до a_r, к которому прибавляется $k \cdot (r - l) + x$. Все запросы прибавления известны заранее, найдите итоговую последовательность за время $O(n + m)$.

Задача 11. (Задача специально для Максима Деб Натха, обязательно скажите ему, что она очевида)

Имеется n объектов, каждому присвоен некоторый вес $a_i > 0$. Также вам доступен генератор случайных равномерно распределённых целых чисел в произвольном диапазоне. Требуется выполнить некоторый предподсчёт за $O(n)$, после чего выполнять выбор одного случайного элемента с вероятностью, пропорциональной весам, за время:

a) $O(1)$ ожидаемое;

b) $O(1)$ гарантированное.

Задача 12. (Задача настолько же красивая, насколько и бесполезная)

Вспомните, как решать известную вам задачу LA (k-й предок у вершины в дереве) за $O(n \log n)$ предподсчёта и $O(1)$ на запрос. Тепрь на не используя идею метода 4 русских и Фараха Колтона Бендера на основе этого решите задачу за:

a) $O(\log(\log n))$ на запрос и $O(n \log(\log n))$ предподсчёта;

b) $O(1)$ на запрос и $O(n \log(\log n))$ предподсчёта;

Функция $\alpha(n)$ равна числу раз, которое надо взять логорифм числа, чтобы получить 1. (Т.е. сколько вложенных логорифмов вида $\log(\log(\log \ldots \log(n) \ldots))$ надо написать, чтобы получить в результате 1.

c) $O(1)$ на запрос и $O(n \cdot \alpha(n))$ предподсчёта;

d) $O(\alpha(n))$ на запрос и $O(n)$ предподсчёта;

(В этих двух пунктах не нужен СНМ!)

e) (Я не умею это решать, но вдруг есть решение) $O(1)$ на запрос и $O(n)$ предподсчёта;

Задача 13. Предложите алгоритм нахождения k-й порядковой статистики за $O(n)$:

a) Ожидаемого времени работы;

b) Гарантированного времени работы.

Задача 14. Мы умеем строить мин. остов за $O(m \log m)$. Тепрь рассмотрим другую задачу, нам надо построить любое остовное дерево, в котором вес максимального ребра минимален. Решите её за $O(m)$. (Гарантируется что граф связный).

Задача 15. (Крутая задача, куплю шоколадку тому, кто решит)

Представим себе, что числа имеют не 32 или 64 бита, а n бит. При этом все операции с числами (арифметические, сдвиги, побитовые) выполняются за $O(1)$. Научитесь искать число единичных бит в числе за $O(\log n)$.

Задача 16. Давайте посортируем числа. Мы все знаем стандартные алгоритмы, котороые работают за $O(n \log n)$ времени. Более того можно доказать, что бытсрее это сделать нельзя. Попробуйте и вы доказать это.

Задача 17. Давайте на время забудем про предыдущую задачу. Тогда у нас есть сортировка подсчётом, работающая за $O(n+C)$, где C — верхнее ограничение на числа. Прдумайте, как её ускорить в корень раз, и отсортировать n чисел за $O(\sqrt{C}+n)$. А как это сделать за $O(n \log_n C)$?

Задача 18. Сортировать n чисел, неограниченных по размеру, мы не можем быстрее, чем за $O(n \log n)$. Однако мы можем построить кучу на n числах за $O(n)$. Предложите какой-нибудь алгоритм, как это сделать.