Домашняя работа

Котов Артем, МОиАД2020

26 сентября 2020 г.

Содержание

Task 1 **2**

Task 2 **3**

Task 3 **3**

Task 4 **5**

Task 5 **6**

Task 6 **7**

Task 1

Условие: Оцените время работы детерминированного алгоритма поиска порядковой статистики, если вместо пятерок разбивать элементы на

(a) семерки.

(b) тройки.

Решение.

(a) Рассмотрим для начала некий общий случай разбиения на k штук. Делаем такую же процедуру, как на паре. Может оценить количество элементов в левом блоке как $\geq \frac{k+1}{2}\frac{n}{2k} = \frac{n(k+1)}{4k}$. Тогда в наибольшей оставшейся части можно оценить как $\leq n - \frac{n(k+1)}{4k} = \frac{n(3k-1)}{4k}$

Замечание. *Проверим, что это сходится с полученными оценками для $k = 5$: smaller block $\geq \frac{3n}{10}$, greater block $\leq \frac{7n}{10}$*

Теперь посмотрим такую же оценку для $T(n) \leq Cn + T\left(\frac{n}{k}\right) + T\left(n\frac{3k-1}{4k}\right)$. Попробуем оценить $T(n) \leq AC'n$, где $C' > C$. Так же по индукции будет проверять: рассмотрим переход

$$T(n) \leq Cn + T\left(\frac{n}{k}\right) + T\left(n\frac{3k-1}{4k}\right) \leq Cn + \frac{n}{k}AC' + \frac{3k-1}{4k}nAC' =$$
$$= \left(C + \frac{AC'}{k} + \frac{3k-1}{4k}AC'\right)n \leq \left(1 + \frac{A}{k} + \frac{3k-1}{4k}A\right)C'n =$$
$$= \frac{4k + 4A + (3k-1)A}{4k}C'n = \frac{3kA + 3A + 4k}{4k}C'n$$

Потребуем, чтобы $\frac{3kA+3A+4k}{4k} = A$, тогда $A = \frac{4k}{k-3}$. Таким образом, имеем, что для такой выбранной A оценка $T(n) \leq \frac{4k}{k-3}C'n$ верна, то есть $T(n) = \mathcal{O}(n)$

Замечание. *Проверим, совпадет ли эта константа с док-вом для $k = 5$: $A = 10$.*

Эта оценка подойдет для $k > 3$.

(b) Рассмотрим теперь отдельно случай $k = 3$. Во-первых, из предыдущего можно заметить, что для этого случая оценка $T(n) \leq Bn$ не работает. То есть, можно утверждать, что $T(n) \neq \mathcal{O}(n)$. Также, очевидно, что можно решить задачу за $\mathcal{O}(n \log n)$ (отсортировать исходный массив).

Более того, аналогичными выкладками из предыдущего пункта можно показать, что $T(n) = \Omega(n \log n)$. Комбинируя эти два результата, мы может утверждать, что $T_{k=3}(n) = \Theta(n \log n)$

■

Task 2

Условие: Даны массив из n чисел и m чисел $p_1, p_2, \ldots, p_m$, нужно за $\mathcal{O}(n \log m + m)$ для каждого i найти p_i-ую порядковую статистику.

Решение.

Так как числа $\{p_i\}$ натуральные по своему смыслу, то мы можем за $\mathcal{O}(n + m)$ отсортировать их с помощью `CountSort`. Разобьем этот массив пополам, то есть рассмотрим массив $[p_1, \ldots, p_{m/2}, \ldots, p_m]$. Найдем соответствующую $p_{m/2}$-ую порядковую статистику (обзовем массив из n числе как $\boldsymbol{x}$). Тогда исходный массив разделится на части $[x_1, \ldots, x_{p_{m/2}}, \ldots, x_n]$, с элементами слева от $x_{p_{m/2}}$ меньше, чем этот элемент, а справа — больше. Теперь рекурсивно запустим поиск такой же соответствующей статистики для $[p_1, \ldots, p_{m/2-1}]$ и $[x_1, \ldots, x_{p_{m/2-1}}]$ и для $[p_{m/2+1}, \ldots, p_m]$ и $[x_{p_{m/2+1}}, \ldots, x_n]$:

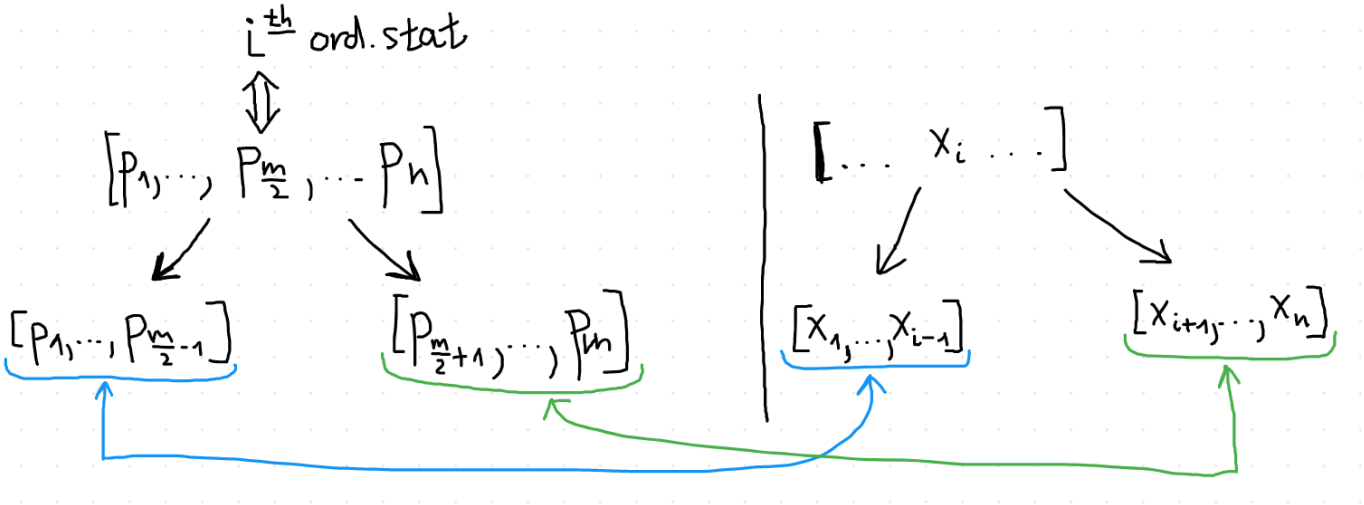

Рис. 1: Поясняющая картинка.

то есть мы сделаем $\mathcal{O}(\log m)$ вызовов этой процедуры, при этом суммарная длина массивов, в которых ищется статистика $\leq n \Longrightarrow \mathcal{O}(n \log m)$.

Таким образом, мы найдем все нужные статистики. Общая сложность сложится из рекурентной схемы и первоначальной сортировки массива $[p]$, то есть

$$\mathcal{O}(n \log m + n + m) = \mathcal{O}(n \log m + m).$$

■

Task 3

Условие: Про поиск ближайших k элементов к медиане

(a) по индексу

(b) по модулю разности значений

Решение.

(a) За линейные времена найдем медиану m и, например, $m - k/2$ статистику.

Замечание. *Где надо, там правильно округляем до целого.*

Затем, выкинем левую часть исходного массива, то есть ту часть, элементы в которой меньше, чем найденная статистика

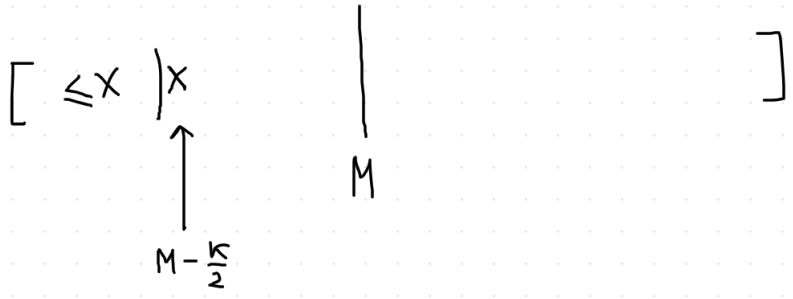

Рис. 2: Художественная картинка, поясняющая слова

и будем рассматривать оставшуюся часть массива. Найдем в ней k-ую статистику (она соответствует $m + k/2$-ой статистике исходного массива). В ответ выводим элементы, от начала до найденной последней статистике.

Сначала также найдем медиану m, $m - k/2$ и $m + k/2$ статистики. Теперь создадим новый массив b, значения элементов которого лежат между $m - k/2$ и $m + k/2$ статистиками, то есть $a_{m-k/2} \leq b_i \leq a_{m+k/2}$. Теперь вычтем из каждого элемента массива b значение медианы, возьмем модуль, то есть $b_i = |b_i - m|$. Эти все действия мы делали за $\mathcal{O}(n)$.

Теперь в массиве b найдем k-ую порядковую статистику (обозначим ее за y), тогда в массиве слева будет блок, элементы которого будут $< y$, берем в ответ начало этого массива до найденной k-ой порядковой статистики.

Замечание. *Имеется в виду, что мы запоминаем в пару индекс элементов из исходного массива, чтобы можно было потом легко восстановить значения самих элементов*

Замечание. *Так как к этому моменту в массиве b_i хранится "расстояние" элементов до медианы, то k-ая порядковая статистка в таком массиве говорит нам, какой элемент находится k-ым по счету по "расстоянию" до медианы.*

■

Task 4

Условие: Даны два массива из положительных чисел a и b, размер обоих равен n. Выбрать массив p из k различных чисел от 1 до n так, чтобы $\frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i} \to \max$. Время $\mathcal{O}(n \log M)$, где $M = \max\limits_i(a_i/b_i)$.

Решение.

Ряд информации, взятой из общего чата: $M = \max\limits_i(a_i,b_i,n)$, числа помимо того, что положительные еще и целые. Рассмотрим $\sum_{i\in I} a_i / \sum_{i\in I} b_i \geq t$, где I некое k-элементное подмножество из $[1,\ldots,n]$. Это отношение можно переписать в виде: $S = \sum_{i\in I} a_i - t\sum_{i\in I} b_i \geq 0$. То есть зададимся вопросом, есть ли такое I, что эта разность верна для какого-то заданного t?

Рассмотрим множество $\{a_i - tb_i\}_{i=1}^{n}$. Найдем $n-k$-ую порядковую статистику в этом множестве (то есть хотим найти элемент, который бы стоял на k-ом месте в отсортированном по убыванию массиве). Возьмем все элементы, начиная с этого и до конца массива вправо в качестве искомых. Если для таких элементов $S < 0$, то для такого t не существует никакого I, чтобы $S \geq 0$, так как все оставшиеся элементы заведомо меньше уже взятых, значит, если бы мы взяли кого-то из них, то результирующая S была бы еще меньше. С другой стороны, если для данного t $S \geq 0$, то мы получили какой-то набор элементов.

Что мы в действительности хотим, так это как можно больше увеличить t. Было бы неплохо понять, какие границы может принимать t: из очевидного, можно за левую границу взять $t = 0$, а за правую $t = \max(a_i/b_i)$.

Замечание. *Рассмотрим, почему такая правая граница подойдет. Рассмотрим индивидуальные дроби $\frac{a_i}{b_i}$. Обозначим наибольшую из них как $\frac{a}{b}$. Посмотрим, как относятся друг с другом $\frac{a}{b} \lesseqgtr \frac{a+x}{b+y}$, где x/y какая-то другая дробь из оставшихся.*

$$\frac{a}{b} - \frac{a+x}{b+y} = \frac{ay - bx}{b(b+y)}$$

Нас интересует знак этого выражения. Так как знаменатель строго положителен, то остается рассмотреть $ay - bx \lesseqgtr 0$. Для этого вспомним, что a/b — наибольшая дробь, то есть, в частности, $\frac{a}{b} > \frac{x}{y}$, следовательно, $ay > bx$. Таким образом, "прибавление" какой угодно дроби к наибольшей уменьшает отношение.

Замечание. *Если рассмотреть S как функцию от t при фиксированном I, то это будет монотонная убывающая функция, что полезно для бинпоиска.*

То есть на данный момент, мы умеем выбирать потенциальный набор I за $\mathcal{O}(n)$ для всякого t. А вот с бинпоиском теперь наступают проблемсы, если для старого M это было бы ответом, так как мы бы за $\mathcal{O}(\log \max(a_i/b_i))$ находили бы нужный t и в результате получили бы сложность $\mathcal{O}(n \log M)$, то с новой M не совсем понятно, откуда здесь взять эти штуки.

Казалось бы, t у нас реально может принимать только рациональные значения, а их на конечном промежутке конечно, можно замапить, видимо, это отношения на числитель/знаменатель/количество элементов. С другой стороны, я понимаю, почему концептуально может не работать старое M (хотя бы потому, что $\log M\Big|_{M=1} = 0$)

Update: Вспомним, что рациональные числа на конечном промежутке конечны. Тогда рассмотрим, какие возможные дроби мы можем получать для исходных a и b. Рассмотрим грубую оценку, когда мы не ограничены только лишь суммами соответствующих пар, то есть всевозможные отношения сумм. Наименьший возможный числитель для таких дробей — это 1, а наибольший $\sum_{i=1}^{n} a_i$, аналогично и для b_i. В случае, когда у нас нет ни одного элемента, превышающего n, то это значит, все a либо принимают значения $[1, \ldots, n]$ либо некоторые (возможно все) из них совпадают, это же верно и для b.

Если же a содержит элемент(ы), превосходящие n, то ничего особо полезного сказать нельзя, разве что числитель индивидуальных дробей ограничен $\max_i(a_i)$.

Теперь составим змейку, пересчитывающую рациональные числа

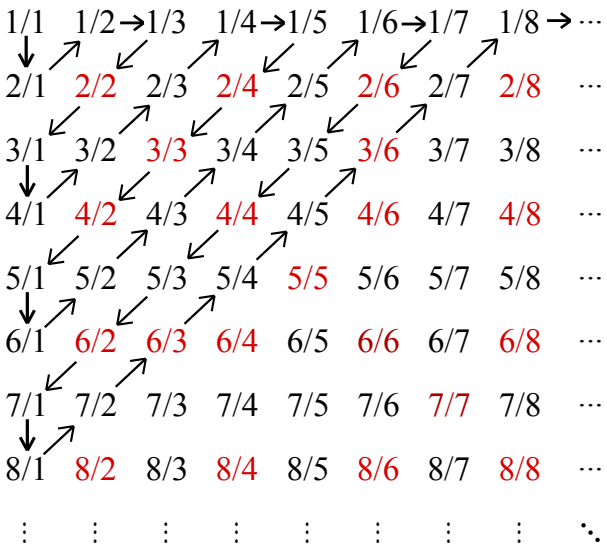

Рис. 3: Пересчет рациональных чисел

Пусть мы отсортировали пары (a_i, b_i) по первому ключу. Тогда, мы можем начинать отсчитывать вертикаль от a_1, а заканчивать вертикаль a_n, горизонталь из b_i сформируем в том порядке, в котором выставлены соответствующие a_i.

1) Тогда, если никакие a_i и b_i не превосходят n, то наибольшее достижимое значение числителя и знаменателя мы можем ограничить как Cn, где C константа. Бинпоиск же интересующего нас t можно организовать среди диагональных элементов такой увеличенной матрицы $Cn \times Cn$, то есть среди Cn штук элементов.

2) Теперь случай когда, у нас могут быть элементы бóльшие n. В этом случае, мы уже говорил, что t реально ограничено $\max(a_i/b_i)$. То есть если мы составим расширенную матрицу размера $max(a_i, b_i) \times \max(a_i, b_i)$ (содержащую ту структуру с отсортированными

элементами a_i), то среди диагональных элементов этой матрицы будет где-то находится искомое t.

Таким образом, реально ведется поиск среди элементов таких диагоналей, а их количество ограничивается (без учета съедаемой слабой асимптотики для случая, когда $n > \max(a_i, b_i)$) $\max(a_i, b_i, n)$. Таким образом, окончательно приходим к ответы для исправленной M.

■

Task 5

Условие: Докажите, что для поиска максимума в массиве различных чисел потребуется как минимум $n - 1$ сравнение.

Решение.

Какое-то очень коротенькое док-во (скорее показательство): рассмотрим задачу нахождения максимума, как турнир среди n участников, где в "дуэли" побеждает больший из пары. Тогда, чтобы определился победитель турнира (то есть нашелся максимум), надо, чтобы остальные участники проиграли как минимум один раз, а таких участников $n - 1$ штука.

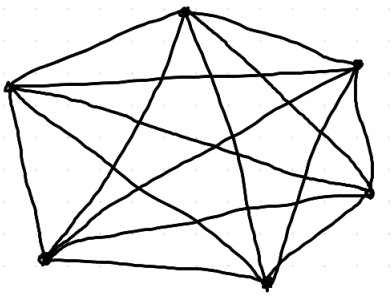

Рис. 4: Турнир, а не пентаграмма (к тому же вершин 6), забыл стрелочки нарисовать :c

■

Решение.

(Второй способ ?) Посмотрим на случай $n = 2$. Очевидно, чтобы понять кто больше, нам необходимо сделать ровно одно сравнение. Пусть, не умаляя общности, $x_1 < x_2$ и заведем $x_{\max} = x_2$. Что происходит, когда к нам приходит еще один эл-т x_3? Нам надо понять, может ли он быть новым наибольшим элементом? Для этого достаточно провести одно сравнение с $x_{\max}$, то есть итого на данном этапе у нас проведено $1 + 1$ сравнение. Если $x_2 < x_3$, то заменим $x_{\max}$ пришедшим эл-том,если же $x_2 \geq x_3$, то просто выбросим x_3 и продолжим "получать" новые элементы. С каждым новым элементом мы будем проводит одно сравнение, то есть `cnt`$+= 1$, а изначально, на первом шаге, `cnt` $= 1$. Таким образом, когда мы расширим количество элементов до n мы проведем $1 + (n - 2) = n - 1$ сравнений, чтобы выяснить кто же является наибольшим. ■

Task 6

Условие: Дана последовательность из n чисел, нужно за один проход и $\mathcal{O}(n)$ времени в среднем найти в ней k минимумов, используя $\mathcal{O}(k)$ дополнительной памяти.

Решение.

Разобьем исходный массив на следующие группы $[2k|k|\dots|\leq k]$. Создадим массив размера $2k$, поместим в него первые $2k$ элементов. Найдем в нем k-ую порядковую статистику (для этого массива это даже медиана), тогда массив разделиться пополам, выкинем правую половину, так как она заведомо больше, чем медиана, помести в эту часть следующие k элементов исходного массива, опять найдем медиану, выкинем правую часть и т.д.

Всего за $\mathcal{O}(k)$ мы каждый раз находим медиану, а кусочков, на которых мы это делаем $\leq \frac{n}{k}$. В конце слева у нас останутся искомые элементы. Общая сложность $\mathcal{O}(k\frac{n}{k}) = \mathcal{O}(n)$. ■