«Основные алгоритмы»

задачи, указания, решения и критерии проверки

семестровой контрольной 2019

Задача 1 (3). На вход программы поступает последовательность из n целых положительных чисел, все числа в последовательности различны. Рассматриваются все пары различных элементов последовательности, находящихся на расстоянии не меньше чем 4 (разница в индексах элементов пары должна быть 4 или более, порядок элементов в паре неважен). Необходимо определить количество таких пар, для которых произведение элементов делится на 29. Постройте онлайн-алогритм, решающий задачу.

Решение.

Будем хранить в памяти: n - число обработанных элементов последовательности без трех последних, k - число чисел кратных 29 в обработанной части последовательности также без последних трех ее элементов, три бита a_1, a_2, a_3, обозначающие кратность 29 последних трех элементов обработанной последовательности, и текущий ответ s - число пар элементов последовательности, находящихся на расстоянии не меньше 4, произведение которых кратно 29.

Заметим, что смещение n на 3 единицы влево не играет роли. Чтобы вычислить ответ для всего хода, достаточно обработать три фиктивных элемента (и потом заменить их на настоящие, если ещё не конец хода).

При обработке нового элемента проверяем, делится ли он на 29. Если делится, то обновляем значение s текущего ответа, прибавив к нему число обработанных элементов последовательности n, если не делится, то прибавляем к нему число обработанных элементов последовательности, кратных 29, k. Производим сдвиг в битах a_1, a_2, a_3: элемент, соответствующий a_1 становится «обработанным», увеличивая значение k на единицу, если делится на 29, a_1 принимает значение a_2, a_2 принимает значение a_3, a_3 теперь соответствует новому элементу. Увеличиваем значение n на единицу.

Корректность: пусть в какой-то момент все значения вычислены верно и на вход поступает новый элемент. Если он делится на 29, то он может составить пару, удовлетворяющую условиям задачи с каждым из элементов

последовательности, кроме последних трех, т.е. с n элементами. Если он не делится на 29, то только с теми элементами, которые кратны 29 и не находятся среди последних трех, т.е с k элементами. Число пар, в которые новое число не входит уже посчитано и равно s, таким образом с учетом нового элемента во всей последовательности будет $s+n$ или $s+k$ пар в зависимости от кратности элемента. Число, соответствующее биту a_1, при поступлении следующего элемента будет находиться от него на расстоянии 4, и может составить с ним пару. Тогда следует пересчитать значение k, в зависимости от a_1 и увеличить n. При этом элемент кратности a_2 будет находиться от следующего элемента на расстоянии 3, a_3 на расстоянии 2, а только что обработанный элемент - на расстоянии 1. Таким образом на новом шаге все переменные (включая переменную, хранящую ответ) будут переопределены корректно.

При обработке каждого нового элемента исполняется $O(1)$ операций: максимум одно произведение, три суммирования, три присваивания. При обработке поледовательности длины n получаем сложность по времени $O(n)$. При работе алгоритма хранятся значения шести переменных, поэтому сожность по памяти составляет $O(1)$

Критерии.

1. При отсутствии доказательства корректности снимается 1 балл
2. При отсутствии оценки сложности снимается 0.5 балла
3. При ошибках в построении алгоритма снимается до $1,5$ балла за задачу
4. Построен не онлайн алгоритм: задача оценивается из 1 балла
5. Построеный алгоритм осуществляет перебор — 0 баллов за задачу

Задача 2 (2). Известно, что $f(n) = O(ng(n))$ и $g(n) = O\left(\sqrt{f(n)}\right)$, где $f, g : \mathbb{N} \to \mathbb{R}_+$. Найдите лучшую верхнюю оценку для функции $f(n)$.

Решение. Оба вариантов:

$$\begin{cases} f(n) = O(ng(n)) \\ g(n) = O\left(\sqrt{f(n)}\right) \end{cases} \Rightarrow \begin{cases} \exists c_1, N_1 : \forall n > N_1 \to f(n) \leqslant c_1 n g(n) \\ \exists c_2, N_2 : \forall n > N_2 \to g(n) \leqslant c_2\sqrt{f(n)} \end{cases} \Rightarrow$$

$$\Rightarrow \exists N = max(N_1, N_2) : \forall n > N f(n) \leqslant c_1 c_2 n\sqrt{f(n)} \Rightarrow$$
$$\Rightarrow \sqrt{f(n)} \leqslant c_1 c_2 n \Rightarrow f(n) \leqslant c_1^2 c_2^2 n^2 \Rightarrow f(n) = O\left(n^2\right)$$

Далее надо показать, что оценка наилучшая, т.е. нельзя предоставить оценку лучше для функций f и g, которые удовлетворяют условиям задачи. Функции $f(n) = n^2$ и $g(n) = n$ удовлетворяют условиям задачи и оценка на $f(n) = O\left(n^2\right)$ для данных функций неулучшаема. Значит данная оценка является лучшей.

Критерии.

$-0,5$ Отсутствие доказательства того, что оценка лучшая.

$-0,5$ Отсутствие использования определения O-большого для доказательных выкладок.

$+2$ Получен верный ответ законными или полузаконными способами.

Задача 3 (4). Найдите Θ-асимптотику для $T(n)$, считая что $T(n) = O(1)$ при малых n.

3 ⟨i⟩.

1. $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$. 2. $T(n) = 16T(\frac{n}{3}) + n^2$.

3. $T(n) = 125T(\frac{n}{5}) + \frac{n^4}{\sqrt{n}\log n + \sin^2 n}$.

3 ⟨ii⟩.

1. $T(n) = 50T(\frac{n}{7}) + (\sqrt{n})^3$. 2. $T(n) = 15T(\frac{n}{2}) + \frac{n^5 + \cos^2 n}{\sqrt{n}\log n}$.

3. $T(n) = 729T(\frac{n}{9}) + n^3$.

Указания. Для решения задачи достаточно правильно воспользоваться основной теоремой о рекурсии. Каждый пункт задачи соответствует случаю теоремы.

Критерии. Критерии проверки (штрафы суммируются):

- Задача решается через Master Theorem, но ни слова не сказано, что студент ее использует: -0,1 балла.
- В задаче неверное решение и дан неверный ответ: 0 баллов.
- Решение целиком верное, но ответ почему-то написан неверный: от -0,5 до -1 балла.
- Ответ получен из неверной предпосылки: -0,5 балла.
- Если решение основывалось на оценке дерева рекурсии и оценка неверная: -0,5 балла.
- Математическая ошибка: от -0,2 до -0,5 в зависимости от степени грубости и влияния на решение.
- Нет проверки на регулярность там, где это необходимо: -1 балл.
- Проверка на регулярность дана как факт либо включает существенные ошибки: -0,8 балла.

- Арифметические ошибки в проверке на регулярность: от -0,2 до -0,5 в зависимости от грубости.
- В проверке на регулярность не говорится про то, что работает для достаточно больших n (из-за чего вполне просто приводится контрпример): -0,2 балла.
- В проверке на регулярность константа больше либо равна 1: от -0,5.

Задача 4 (2). Деком называют модифицированный стек, в который можно добавлять и извлекать элементы как сверху, так и снизу. То есть, все операции стека принимают дополнительный параметр: front или back; например, push_front добавляет элемент на верхушку стэка, а push_back добавляет элемент со стороны дна стека. Опишите реализацию дека с помощью двусвязных списков.

Критерии.

1. Приведено полное решение: 2 балла.
2. Приведено решение без реализации одного из push или pop: 1,5 балла.
3. Код без объяснений понижает оценку на 1 балл.
4. Нет реализации ни push_front, ни push_back: 0 баллов.
5. Нет решения: 0 баллов.

Решение. Будем поддерживать указатели $front$ и $back$ на начало и конец двусвязного списка соотетственно. Для определённости опишем операции для $front$, операции для $back$ описываются симметрично.

Операция push_front(x) состоит из создания нового элемента списка с ключём x, записи в ссылку вперед элемента по указателю $front$ указателя на новый элемент, и записи указателя на новый элемент во $front$.

Операция pop_front состоит из создания вспомогательного указателя $x = front$, перемещения указателя $front$ по ссылке назад, удаления элемента по указателю x и присваивания $null$ указателю на следующий элемент для элемента по индексу $front$.

Задача 5 (4). Решите систему уравнений в целых числах:

5 ⟨i⟩.

$$\begin{cases} 5x + 7y = 3 \\ 11y + 17z = 19 \end{cases}$$

5 ⟨ii⟩.

$$\begin{cases} 3x + 23y = 11 \\ 31y + 37z = 17 \end{cases}$$

Решение. Вариант 1:

$$5x + 7y = 3$$

$$5a + 2y = 3 \ //a = x + y$$

$$a = 1, y = -1$$

$$a = x + y => x = 2$$

$$x = 2 + 7k, y = -1 - 5k \quad (+\mathbf{1} \text{ балл})$$

$$11y + 17z = 19$$

$$11b + 6z = 8 \ //b = y + z - 1$$

$$6c - b = 2 \ //c = z + 2b - 1$$

$$c = 1, b = 4$$

$$z + 2b - 1 = c => z = -6$$

$$y + z - 1 = b => y = 11$$

$$y = 11 + 17m, z = -6 - 11m \quad (+\mathbf{1} \text{ балл})$$

Для того, чтобы решить систему, нужно приравнять решения для y:

$$-1 - 5k = 11 + 17m => 5k + 17m = -12 \quad (+\mathbf{1} \text{ балл})$$

Решаем получившееся уравнение:

$$k = 1, m = -1$$

$$k = 1 - 17l, m = -1 + 5l$$

Подставляя в получившиеся ранее решения, получаем:

$$\begin{cases} x = 2 + 7(1 - 17l) & = 9 - 119l \\ y = -1 - 5(1 - 17l) & = -6 + 85l \\ z = -6 - 11(-1 + 5l) & = 5 - 55l \end{cases} \quad (+\mathbf{1} \text{ балл})$$

□

Вариант 2:

$$3x + 23y = 11$$

$$3a + 2y = 2 \ //a = x + 7y - 3$$

$$a = 0, y = 1$$

$$a = x + 7y - 3 => x = -4$$

$$x = -4 + 23k, y = 1 - 3k \quad (+\mathbf{1} \text{ балл})$$

$$31y + 37z = 17$$

$$31b + 6z = 17 \ //b = y + z$$

$$6c + b = -1 \ //c = 5b + z - 3$$

$$c = 0, b = -1$$

$$5b + z - 3 = c => z = 8$$

$$y + z = b => y = -9$$

$$y = -9 + 37m, z = 8 - 31m \quad (+\mathbf{1} \text{ балл})$$

Для того, чтобы решить систему, нужно приравнять решения для y:

$$1 - 3k = -9 + 37m => 3k + 37m = 10 \quad (+\mathbf{1} \text{ балл})$$

Решаем получившееся уравнение:

$$k = -9, m = 1$$

$$k = -9 + 37l, m = 1 - 3l$$

Подставляя в получившиеся ранее решения, получаем:

$$\begin{cases} x = -4 + 23(-9 + 37l) & = -211 + 851l \\ y = 1 - 3(-9 + 37l) & = 28 - 111l \quad (+\mathbf{1} \text{ балл}) \\ z = 8 - 31(1 - 3l) & = -23 + 93l \end{cases}$$

□

Возможны другие варианты решения, в частности, после решения первого уравнения, можно подставить получившийся y во второе уравнение и выразить z и k через l, что уже приведёт к решению. Также возможны варианты с рассмотрением обоих уравнений по модулю коэффициентов перед x и z соответственно, после чего находится y, а с его помощью x и z. В ходе проверки таких решений было меньше, а их содержание было в основном верным, поэтому баллы за них выставлялись в зависимости от декватности происходящего (4 балла за верное решение, 1 за некоторое продвижение, 0 за неправильные рассуждения).

Критерии.

1. Найдено только частное решение ⇒ 0 баллов
2. Сделан неверный переход от общего решения $ax + by = 1$ к общему решению $ax + by = c$ ⇒ -0,5 баллов за секцию

3. Не подошло частное решение из общего решения ⇒ 0 баллов за соответствующую часть решения
4. Арифметическая ошибка ⇒ -0,2 баллов
5. Какая-то идея решения без попыток решения ⇒ 0 баллов
6. Правильный ответ при отсутствии расписанного решения ⇒ 0 баллов.
7. Правильный ответ при абсолютно непонятном решении ⇒ 0 баллов
8. Проблемы с написанием общего решения в конкретной секции[1] при правильном нахождении частного ⇒ -1 балл или -0,5 баллов за секцию
9. Сняты баллы за первые 2 секции не за арифметические ошибки ⇒ третья секция оценивается из 0,5 баллов
10. Есть ошибки в первых 3 секциях ⇒ последняя секция оценивается из 0,5 баллов
11. Ответ не приведён к разумному виду ⇒ до -2 балла в случае, когда не сделаны последние 2 шага, до -1 балла, если 3 и 4 шаги были в некотором виде реализованы.

Задача 6(3). Какое минимальное число сравнений может сделать алгоритм сортировки сравнениями в результате своей работы?

Решение. Ответ: $n-1$ сравнение.
Нужно привести пример алгоритма и входа, который сортирует массив за данное количество сравнений. И нужно показать, что не существует алгоритма сортировки сравнениями, сортирующего массив за меньшее количество сравнений.
Пример: алгоритм сортировки пузырьком на отсортированном массиве. Алгоритм сравнит 1 и 2 элемент, потом 2 и 3 и т.д. до $n-1$ и n. Оценка: если алгоритм работает не более, чем за $n-2$ сравнения, то он не сравнивает между собой какие-то 2 соседних элемента в отсортированном массиве. Тогда если мы на входе поменяем их местами, то алгоритм сделав те же самые сравнения отработает некорректно.

Критерии.

- Пример: 0.5 балла за вход, 0.5 балла за алгоритм.
- Оценка: доказательство оценки стоит 2 балла, разные помарки стоят -1 балл.

[1] Авторское решение разбито на секции, за каждую секцию даётся балл. Решение студента делится на секции схожим образом, если это возможно.

Задача 7 (3). Элементы двоичной кучи с максимальным свойством образуют массив $[a_1, a_2, \ldots, a_n]$; все элементы попарно различны. Найдите минимальную глубину, на которой может находиться порядковая статистика массива **7** ⟨i⟩. $a_{(\lfloor n/8 \rfloor)}$. **7** ⟨ii⟩. $a_{(\lfloor n/16 \rfloor)}$. Корень дерева имеет глубину 0.

Решение. Сразу хочу сказать, что задача получилась гораздо сложнее, чем предполагалось. Для простой задачи нужно было добавить условие $n \geqslant 100$. Без этого приходится разбирать много крайних случаев при малых n.

Ответ:
1 вариант. При $n < 8$ такой порядковой статистики нет. При $8 \leqslant n \leqslant 13$ и $24 \leqslant n \leqslant 27$ минимум – второй уровень. При прочих n минимум – третий уровень.
2 вариант. При $n < 16$ такой порядковой статистики нет. При $16 \leqslant n \leqslant 29$ и $48 \leqslant n \leqslant 55$ минимум – третий уровень. При прочих n минимум – четвёртый уровень.

Решение приводится для первого варианта, для второго решение аналогичное.

В куче с максимальным свойством где все элементы различны потомки строго меньше своего родителя. Значит в поддереве с корнем $a_{(\lfloor n/8 \rfloor)}$ не более $\lfloor n/8 \rfloor$ вершин.

Покажем, что этот элемент не может находиться на первом уровне. Когда в одной из веток находится максимальная доля элементов массива? Тогда, когда эта ветка заполнена до глубины h, а другая только до $h-1$. Это бывает если в куче $n = 2^h - 1 + 2^{h-1}$ элементов. Тогда в максимально заполненной ветке находится $2^h - 1 = \frac{2n-1}{3}$ элемент[2]. Тогда всего в куче получается максимум $\frac{2n-1}{3} + \left\lfloor \frac{n}{8} \right\rfloor + 1$ элементов, что меньше n, поскольку

$$\frac{2n-1}{3} + \frac{n}{8} + 1 = \frac{19n+16}{24} \leqslant \frac{19n+2n}{24} < n.$$

Значит такой кучи быть не может.

Пример выше был приведён для того, чтобы показать некорректность типичной оценки, встречавшейся в работах студентов, когда писали что в левой ветви двоичной кучи находится не более $\left\lceil \frac{n-1}{2} \right\rceil$ элемента. В действительности же там может находиться до $2^h - 1 = \frac{2n-1}{3}$ вершин. Также неполными без детального рассмотрения округлений и различного количества вершин на последнем уровне были доказательства через разницу высот, оцененных через логарифмы.

Приведу теперь полностью корректное решение. Посчитаем какое минимальное количество элементов может находиться в подкуче с корнем на глубине $d \geqslant 1$. На последнем уровне кучи находится не более 2^h элементов. Если на последнем уровне кучи $c' \leqslant 2^h - 2^{h-d}$ элементов, то в подкуче

[2] $n = 3 \times 2^{h-1} - 1$, $\frac{n+1}{3} = 2^{h-1}$, $\frac{2n+2}{3} = 2^h, 2^h - 1 = \frac{2n-1}{3}$.

элементы могут присутствовать лишь на $h-d-1$ уровнях в количествах $2^0+2^1+\ldots+2^{h-d-1}=2^{h-d}-1$. Иначе в подкуче $2^{h-d}-1+c$ элементов, где $1 \leqslant c \leqslant 2^{h-d}$; а на последнем уровне кучи $2^h-2^{h-d}+c$ элементов.

Итак, если порядковая статистика $a_{(\lfloor n/8 \rfloor)}$ находится на глубине d, то в первом случае должно выполняться неравенство

$$2^{h-d}-1 \leqslant \left\lfloor \frac{n}{8} \right\rfloor = \left\lfloor \frac{2^h-1+c'}{8} \right\rfloor, 1 \leqslant c' \leqslant 2^h-2^{h-d}. (1)$$

Поскольку

$$\left\lfloor \frac{2^h-1+c'}{8} \right\rfloor \geqslant \left\lfloor \frac{2^h}{8} \right\rfloor = 2^{h-3},$$

неравенство (1) справедливо при $d=3$ и произвольном n.

Посмотрим теперь неравенство (1) при $d=2$ и $h=3$. ($2^{3-2}-1=1$.)

$$1 \leqslant \left\lfloor \frac{2^3-1+c'}{8} \right\rfloor \leqslant \left\lfloor \frac{2^3+2^3-2^1-1}{8} \right\rfloor = 1.$$

Видим, что при $d=2$ и $h=3$ неравенство (1) справедливо при любом $1 \leqslant c' \leqslant 2^h-2^{h-d}$, т.е. при $8 \leqslant n \leqslant 13$.

Посмотрим теперь неравенство (1) при $d=2$ и $h=4$. ($2^{4-2}-1=3$.)

$$3 \leqslant \left\lfloor \frac{2^4-1+c'}{8} \right\rfloor \leqslant \left\lfloor \frac{2^4+2^4-2^2-1}{8} \right\rfloor = 3.$$

Видим, что при $d=2$ и $h=4$ неравенство (1) справедливо при $2^4-2^3+1 \leqslant c' \leqslant 2^4-2^2$, т.е. при $24 \leqslant n \leqslant 27$.

Теперь покажем, что неравенство (1) невыполнимо при $d=2$ и $h>4$.

$$\left\lfloor \frac{2^h-1+c'}{8} \right\rfloor \leqslant \left\lfloor \frac{2^h+2^h-2^{h-2}-1}{8} \right\rfloor = 2^{h-2}-2^{h-2-3}-1 < 2^{h-2}-1.$$

Рассмотрим коротко второй случай. Для него необходимо проверить неравенство

$$2^{h-d}-1+c \leqslant \left\lfloor \frac{2^h-1+2^h-2^{h-d}+c}{8} \right\rfloor, 1 \leqslant c \leqslant 2^{h-d}. (2)$$

Тут нужно как раньше посмотреть что происходит при $h=3, h=4, h>4$ и убедиться, что неравенство (2) невыполнимо при $d=2$ и выполнимо при $d=3$ независимо от h и c. Оставляю это упражнение студентам. Всё равно никто не сделал ничего подобного на контрольной.

Критерии.

- Пример достижимости 2 уровня (3 уровня для 2 варианта) при малых n стоит 1 балл.
- Оценка: доказательство оценки стоит 2 балла, разные помарки стоят -1 балл.
- Если оценка оставлена в виде разности логарифмов, не доведена до численного ответа, это ещё -0,5 балла.

Задача 8 (4). Структура данных «таблица» представляет собой числовую матрицу размера $m \times n$ (двумерный массив), элементы которой в каждой строке и каждом столбце отсортированы по возрастанию. В случае, если в таблице меньше mn элементов, в незаполненных клетках написано ∞. Покажите как вставить новый элемент в частично заполненную таблицу за $O(m+n)$.

Решение. Будем решать задачу рекурсивно. Заметим, что если таблица является строкой или столбцом, то достаточно добавить элемент в её конец и двигать влево (вниз) до тех пор, пока элемент не окажется на своём месте.

В общем случае добавляем элемент x в правый нижний угол таблицы (там записано ∞, поскольку таблица заполнена частично). Далее запускаем описанный ниже процесс в ходе которого будет поддерживаться свойство таблицы (упорядоченность по возрастанию в строках и столбцах) для всех элементов кроме, может быть, x. Процесс: если x оказался больше левого и верхнего соседа, то x на своём месте и мы заканчиваем, если нет, то найдём максимум среди соседей x слева и сверху и поменяем местами x с этим максимумом. Пусть, не ограничивая общности, мы поменяли x с соседом слева y. В результате этой перестановки свойство таблицы может сломаться только для последнего столбца, в остальные строки и столбцы ничего нового, кроме x не добавится. В последнем стоблце после перестановки достаточно проверить выполнение условия для элемента над y (он меньше, т.к. y был максимальным из соседей x), элемента под y (он больше, т.к. до перестановки находился правее и ниже y в таблице, следовательно он больше по свойству таблицы)

Каждая перестановка x реализуется за $O(1)$. Поскольку в результате перестановки одна из координат таблицы уменьшается на 1, то в результате алгоритм совершает $O(m+n)$ операций.

Критерии.

1. Любое рассуждение, не указывающее, что делать с элементом a_{ij}, вместо которого в таблице оказался x – 0 баллов.
2. После применения алгоритма структура данных перестала обладать основным свойством – 0 баллов.
3. Асимптотика алгоритма есть $\Omega(mn)$ – 0 баллов.

4. Отсутствие обоснования корректности – 0 баллов.

5. Частичное решение, правильно реализующее вставку, возможно, с огрехами реализации – до 2 баллов.

Задача 9 (4). На вход алгоритма подаётся массив натуральных чисел A и число x. Постройте алгоритм, работающий за $O(n \log n)$, который проверяет, есть ли в массиве A два элемента,

9 ⟨i⟩. сумма которых равна x.

9 ⟨ii⟩. разность которых равна x.

Решение. Алгоритм сортирует массив за $O(n \log n)$, например, с помощью MergeSort, далее для каждого элемента массива $a[i]$ ищет бинарным поиском в массиве элемент $x - a[i]$ (для разности $x + a[i]$).

Корректность: в массиве есть два элемента сумма которых равна x тогда и только тогда для некоторого элемента $a[i]$ в массиве есть элемент $x - a[i]$.

Сложность: Сортировка стоит $O(n \log n)$, бинарный поиск стоит $O(\log n)$ и запускается не более n раз. Итого $O(n \log n)$.

Критерии.

1. Правильное применение бинарного поиска в отсортированном массиве, либо правильное применение указателей при, в целом, неправильном решении — до 1 балла за задачу.

2. Использование сортировки подсчётом — 0 баллов (вследствие плохой асимптотики).

3. Асимптотика алгоритма есть $\Omega(n^2)$ не вследствие используемой быстрой сортировки — 0 баллов.

4. Код без пояснений снижает оценку на 1 балл.

5. Быстрая сортировка за $n \log n$ без комментариев снижает оценку на 0.5 балла.

Задача 10 (5). Дан массив пар чисел (x_i, w_i), будем называть $w_i > 0$ весами соответствующих элементов. Пусть сумма $\sum w_i = W$. Нужно найти взвешенную медиану этого массива m, это такое число, что сумма весов элементов массива с $x_i \leqslant m$ и сумма весов элементов массива с $x_i \geqslant m$ обе больше или равны $\frac{W}{2}$. Предложите линейный по числу арифметических операций алгоритм.

Критерии.

- Решения с асимптотикой хуже $O(n)$ стоят 0 баллов, включая $O(n \log n)$ решения.

- Рандомизированный линейный алгоритм оценивается из 4 баллов.

Решение.

Рассмотрим алгоритм решения более общей задачи (можно назвать ее задачей нахождения взвешенной порядковой статистики) нахождения минимальной координаты X, такой что суммарный вес элементов с $x_i \leqslant X$ больше или равен Y ($Y \leqslant W/2$). Исходная задача решится в этом случае вызовом алгоритма от $Y = W/2$. Будем строить рекурсивный алгоритм, если $n = 1$, то вернем x_1, иначе найдем за $O(n)$ обычную медиану m массива x_i и сделаем по ней Partition, то есть разделим все x_i на меньшие m (первая группа), большие m (вторая группа), равные m (третья группа). Посчитаем сумму весов W_1 в первой группе и сумму весов в третьей группе W_3. Если $W_1 \geqslant Y$, рекурсивно вызовем алгоритм от первой группы с тем же Y, потому что искомый $X < m$. Если сумма весов в первой группе $W_1 < Y$, то проверим верно ли, что ответом является m — это так, если $W_1 + W_3 \geqslant Y$. Иначе вызовем рекурсивно алгоритм от второй группы с новым $Y = Y - W_1 - W_3$, потому что искомый $X > m$. Тогда можно оценить асимптотику алгоритма с помощью рекуренты $T(n) \leqslant T(\lfloor n/2 \rfloor) + O(n)$. По мастер-теореме $T(n) = O(n)$.

Также эту задачу можно было решить адаптировав любой из алгоритмов поиска обычной медианы массива, отличие лишь в том, что надо делать Partition по указанной выше схеме.

Задача 11 (7). На вход подается массив целых чисел $A = [a_1, a_2, \ldots, a_n]$, предложите эффективный алгоритм нахождения непрерывного подмассива $a_i, a_{i+1}, \ldots, a_j$ с максимальным произведением количества элементов в подмассиве и минимума по подмассиву.

Критерии.

- $O(n^2)$ решение оценивается из 2 баллов
- $O(n \cdot max(x_i))$ решение оценивается из 1 балла
- Алгоритм работает корректно только на массиве из различных элементов $\Rightarrow$ -0.5 балла

Решение. Решение 1: Построим рекурсивный алгоритм по схеме разделяй-и-властвуй. Если длина массива $n = 1$, то вернем величину его единственного элемента. Иначе разделим массив поровну на две части. Оптимальный подмассив либо лежит слева, либо справа, либо накрывает середину массива. В левой и в правой части запустимся рекурсивно. Теперь научимся за линейное время находить максимум целевой функции по подмассивам накрывающим середину. Переберем два симметричных случая: когда минимум подмассива

находится слева и когда справа. Разберем подробно случай, когда он слева. Будем идти от середины массива налево и перебирать тем самым левый конец подмассива, поддерживая минимум m уже пройденной части. Минимум уже пройденной части является минимумом подмассива, поэтому правый конец оптимально поставить перед первым элементом в правой половине массива, который меньше m. Если искать этот правый конец перебором, то эта часть будет работать за $O(n^2)$ (для каждого левого конца ищем правый за линейное время), нас это не устроит. Но можно заметить, что при рассмотрении каждого нового левого конца значение m может только уменьшаться, поэтому каждый новый правый конец всегда будет не левее предыдущего, значит мы можем искать его вторым указателем за $O(n)$ в сумме. Асимптотику алгоритма можно через рекуренту $T(n) = 2T(\lfloor n/2 \rfloor) + O(n)$ оценить по мастер-теореме как $O(n \log n)$.

Решение 2: Сортировкой массива пар (a_i, i) за $O(n \log n)$ найдем индексы на которых стоят все порядковые статистики массива a_i (автоматически равные элементы отсортируются по индексу). Далее запустим некоторый процесс, который отработает за $O(n)$ и найдём по пути всё, что нам нужно. Вначале создадим двусвязный список из всех пар (a_i, i) в порядке возрастания i и для каждого элемента массива сохраним ссылку на его положение в этом списке. У крайних элементов поставим ссылки на 0 и n индекс соответственно. Будем идти по убыванию значений a_i, находить максимальный подмассив в котором этот a_i является минимумом и удалять его из двусвязного списка (его соседи в двусвязном списке при этом станут соседями между собой), все это будем делать за $O(1)$. С удалением из двусвязного списка всё ясно, оно работает за $O(1)$ из коробки. Осталось заметить, что перед моментом удаления (a_i, i) из двусвязного списка, максимальный подмассив, в котором он является минимумом «зажат» ровно между индексами двух его соседей в двусвязном списке, потому что они оба его меньше (раз мы начали удалять его до них), а все элементы между ними его больше (потому что мы их уже удалили). Значит нужно умножить a_i на разность индексов соседей минус 1. Суммарная асимптотика $O(n \log n)$.

Решение 3: С помощью стека можно решить эту задачу за $O(n)$. Опишем это решение кратко. Идея будет, как и в предыдущем алгоритме, в том, чтобы найти для каждого a_i длину максимального подмассива, в котором он является минимумом. Добавим в правый конец массива минус бесконечность. Пойдём по массиву слева направо, будем при этом поддерживать в стеке отсортированную по возрастанию часть уже прочитанных элементов, а именно: первый элемент стека a_{i_1} будет минимумом уже прочитанной части, второй элемент стека a_{i_2} является минимумом всего прочитанного после a_{i_1}, третий элемент стека a_{i_3} является минимумом всего прочитанного после a_{i_2}, и т.д. Легко видеть, что каждый новый прочитанный элемент a_i выкинет из стека все, что больше или равно его. Осталось заметить, что при выкидывании элемента из стека максимальный подмассив, в котором он является минимумом, «зажат» между предыдущим элементом стека и текущим a_i, который его выкидывает. Поскольку в конце массива лежит минус бесконечность, все элементы исходного массива из стека точно будут

выкинуты, и в момент выкидывания мы вычислим для них нужное значение, и возьмём по всем этим значениям максимум.