«Основные алгоритмы»

задачи, указания, решения и критерии проверки

семестровой контрольной 2018

Задача 1 (2).

1 ⟨i⟩. На вход подаётся число n и координаты точек на плоскости (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$. Необходимо найти максимальную площадь треугольника, вершинами которого являются некоторые точки из списка, причём одна из сторон лежит на оси Oy.

1 ⟨ii⟩. На вход подаётся число n и координаты точек на плоскости (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$. Необходимо найти квадрат со сторонами, параллельными осям координат, минимальной площади, который содержит все перечисленные точки (если точка лежит на границе, то она входит в квадрат).

Решение. **1** ⟨i⟩. Жадный однопроходный алгоритм. Алгоритм ищет максимальный и минимальный y среди точек вида $(0, y)$ и точку с максимальным $|x|$. Для этого используем три переменны $y_{\min}, y_{\max}, |x|_{\max}$, $y_{\min}$ и $y_{\max}$ инициализируются после обработки первой точки вида $(0, y)$ и приравниваются y, $|x|_{\max}$ инициализируется после обработки первой точки вида $(x, y), y \neq 0$ и приравнивается $|x|$. Далее значение переменных пересчитывается после считывания каждой точки (x_i, y_i) (если $x_i = 0$ и $y_i > y_{\max}$, то $y_{\max} = y_i$ и т.д.). После обработки всех точек, алгоритм выводит значение площади $\frac{|x|_{\max}}{2} \times (y_{\max} - y_{\min})$. **Корректность.** Площадь

треугольника равна максимальна когда максимально основание и проведённая к нему высота. Взяв за основание сторону на оси Oy получаем, что она имеет максимальную длину в случае точек $(0, y_{\max}), (0, y_{\min})$, высота к основанию максимальна когда третья вершина имеет максимальную по модулю координату x. **Сложность.** Алгоритм проходит вход один раз, нигде не сохраняя считанные точки, отсюда получаем $O(1)$ по памяти. Все операции во время прохода стоят $O(1)$, получаем сложность $O(n)$ по времени.

1 ⟨ii⟩. Жадный однопроходный алгоритм. Аналогично первому варианту ищем максимальные и минимальные x и y среди всех точек. Вычисляем длину стороны квадрата $a = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min})$. Искомый квадрат имеет координату левой нижней точки $(x_{\min}, y_{\min})$ и сторону a. **Корректность**. Все перечисленные точки лежат прямоугольнике с вершинами

$$(x_{\min}, y_{\min}), (x_{\min}, y_{\max}), (x_{\max}, y_{\max}), (x_{\max}, y_{\min}).$$

Длина стороны искомого квадрата не может быть меньше сторон прямоугольника: если $a < x_{\max} - x_{\min}$, то никакая точка вида $(x_{\min}, y)$ не попадёт в квадрат, даже если точка с координатой $x_{\max}$ лежит на границе. Точка $(x_{\min}, y_{\min})$ лежит в квадрате, поскольку есть точки на прямых $x = x_{\min}$ и $y = y_{\max}$, взяв эту точку крайней нижней и построив от неё квадрат со стороной a получим, что все точки прямоугольника лежат в квадрате. **Сложность.** Алгоритм проходит вход один раз, нигде не сохраняя считанные точки, отсюда получаем $O(1)$ по памяти. Все операции во время прохода стоят $O(1)$, получаем сложность $O(n)$ по времени.

Критерии. Задача теряет 50% от стоимости, если

- Алгоритм использует $\Omega(n)$ по памяти или сам алгоритм явно не описан, поэтому из решения нельзя заключить, что алгоритм использует $O(1)$ памяти.
- В первом варианте ищется $x_{\max}$ вместо $|x_{\max}|$ или задача решается для оси Ox. Во втором варианте ищется прямоугольник содержащий все точки, а не квадрат.

Оба пункта независимы — стоимость может снизиться на 50% два раза. Если при решении задачи используется сортировка, задача оценивается из 0,3 очков.

Отсутсвие обоснования корректности и оценки сложности стоит по 25% в стандартных случаях и до 50% в вопиющих случаях (например, просто приведён код безо всякого обоснования корректности).

От 25% до 50% случай «алгоритм плохо обоснован». В решении указываются ключевые шаги, но сам алгоритм не описан. Например, рассказано, как получить треугольник максимальной площади, но про алгоритм поиска координат его вершин ничего не сказано.

Задача 2 (3). Имеется набор из n блоков высоты 1 и ширины $w_i \in [1, 2, .., 100]$, задающиеся входным массивом $W = \{w_i\}$. Требуется построить из них башню максимальной высоты. Каждый этаж башни содержит ровно один блок. Причем, чтобы башня была устойчивой, ширина ее $k-$го этажа должна быть как минимум в полтора раза больше чем $(k+1)-$го. Постройте алгоритм, решающий данную задачу.

Решение. Сортируем массив сортировкой подсчетом. Башня будет храниться в виде массива $Tower := \varnothing$. Число текущих этажей - $Levels := 0$. Помещаем на вершину башни, т.е. в первый слой, минимальный элемент W: $Tower[Levels++] := W[1]$. Поднимаемся по массиву вверх до тех пор, пока не встретим блок, который был бы хотя бы в полтора раза шире чем предыдущий:

$$\textbf{if } W[Count] < 1.5 \cdot Tower[Levels] \textbf{ then } Tower[Levels++] := W[Count].$$

Алгоритм работает до тех пор, пока $Count$ не станет равен n, после чего возвращает массив $Tower[1..Levels]$.

Корректность. От противного - положим, что существует башня $Tower' = T'$ строго выше чем башня $Tower = T$ полученная в результате работы алгоритма выше. В силу того, что T строилась жадно, то для всякого $k \in [1..Levels]$ верно: $T[k] \leq T'[k]$. По предположению, у башни T' существует слой $Levels + 1$, которого нет в башне T. Однако, т.к. $T'[Levels+1] \geq 1.5 \cdot T'[Levels] \geq 1.5 \cdot T[Levels]$, то блок $T'[Levels]$ должен, по построению, лежать в башне T на $(Level + 1)$-ом слое. Противоречие.

Время работы. Сортировка подсчетом работает за $\Theta(n)$ действий. На каждой итерации цикла **repeat-until** алгоритм совершает максимум константное число действий определяемое сравнение **if**, а т.к. число итераций равно в точности n, то получаем итоговое время работы $\Theta(n)$;

Критерии.

- Нет доказательства корректности (в т.ч. и простой устойчивости башни) – 0 баллов
- Нет доказательства того, что алгоритм строит башню максимальной высоты -1.5 баллов
- Решение за $O(n \log n)$: -0.5 балла
- Решение за $O(n^2)$: -1 балл

Задача 3 (3).

3 $\langle$i$\rangle$. Известно, что $f(n) = O\left(\dfrac{g(n)}{f(n)}\right)$, где $f, g : \mathbb{N} \to \mathbb{R}_+$. Следует ли отсюда, что $f(n) = O((g(n))^{1-\varepsilon})$ для некоторого $\varepsilon > 0$?

3 $\langle$ii$\rangle$. Известно, что $f(n) = \Omega\left(\dfrac{g(n)}{f(n)}\right)$, где $f, g : \mathbb{N} \to \mathbb{R}_+$. Следует ли отсюда, что $g(n) = O((f(n))^{1-\varepsilon})$ для некоторого $\varepsilon > 0$?

Решение. **3** $\langle$i$\rangle$. $f(n) = O\left(\dfrac{g(n)}{f(n)}\right) \Rightarrow f^2(n) = O(g(n))$. В силу непрерывности и монотонности квадратного корня, $f(n) = O(\sqrt{g(n)})$. Утверждение верно.

3 $\langle$ii$\rangle$. Утверждение неверно. Контрпример: $f(n) = n, g(n) = n^2$; условие задачи выполняется: $n = \Omega(\frac{n^2}{n}) = \Omega(n)$, но $n^2 \neq O(n^{1-\varepsilon})$ ни для какого $\varepsilon > 0$.

Критерии.

+1 Получено, что **3** $\langle$i$\rangle$. $f^2(n) = O(g(n))$; **3** $\langle$ii$\rangle$. $f^2(n) = \Omega(g(n))$.

+0.5 (для первого варианта) после первого пункта приведены разумные доводы, приближающие к доказательству.

2 (для второго варианта) в качестве контрпримера указаны верные функции, но не обоснованно, что они действительно образуют контрпример.

Задача 4 (2+2+2). Найдите лучшие верхние и нижние оценки на функции, считая, что при малых n, $T(n) = O(1)$:

4 ⟨i⟩. а) $T(n) = 4T(\lfloor \frac{n}{4} \rfloor) + \frac{n^2}{n+\log^2 n}$; **б)** $f(n) = \sum\limits_{i=1}^{n} \sqrt{\frac{n}{i}}$;

в) $T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + \frac{n^2}{\log_2 n}$;

4 ⟨ii⟩. а) $T(n) = 3T(\lfloor \frac{n}{3} \rfloor) + \frac{n^2+\log^2 n}{n}$; **б)** $f(n) = \sum\limits_{i=1}^{n} \sqrt[3]{\frac{n}{i}}$

в) $T(n) = 6T(\lfloor \frac{n}{5} \rfloor) + n^2 \cdot \log_2 n$;

Указания. Для первого варианта. Для второго варианта аналогично. **а)** $\frac{n^2}{n+\log^2 n} = \Theta(n)$, отсюда $T(n) = \Theta(n \log n)$ по основной теореме.

б)

$$f(n) = \sum_{i=1}^{n} \sqrt{\frac{n}{i}} = \sqrt{n} \sum_{i=1}^{n} \frac{1}{\sqrt{i}} = \Theta(n).$$

Последний переход сделан по упражнению с семинара:

$$\sum_{i=1}^{n} \frac{1}{\sqrt{i}} = \Theta(\sqrt{n}).$$

в) По третьему случаю основной теоремы $T(n) = \Theta(\frac{n^2}{\log_2 n})$. Важно не забыть проверить условие регулярности!

Критерии.

- За верхнюю и нижнюю оценку по 1 баллу
- Не проверили условие регулярности в пункте в: -1 балл
- В пункте а показали, что $f(n) = \Omega$ в первом варианте и $f(n) = \emptyset$ во втором варианте (вместо $f(n) = \Theta$) – 1 балл. Если наоборот – 0 баллов
- За прочие неточности снималось от 0.5 до 1 балла

Задача 5 (2).

5 ⟨i⟩. Найдите обратный остаток к 127 по модулю 353, используя расширенный алгоритм Евклида.

5 ⟨ii⟩. Найдите обратный остаток к 129 по модулю 349, используя расширенный алгоритм Евклида.

Указания. Задача сводится к решению диофантова уравнения

$$127x + 353y = 1.$$

Критерии.

- Неверный ответ: 0 баллов
- Неверный знак у правильного ответа: 1 балл
- Верный ответ: 2 балла

Задача 6 (3). На плоскости даны n точек с координатами

$$\{(x_1, y_1), \ldots, (x_n, y_n)\},\ (x_i, y_i) \in \mathbb{Z}^2.$$

Привести алгоритм для нахождения пути, проходящего через n точек без самопересечения, считая, что точки соединяются отрезком прямой.

Решение. Сортируем массив точек $Dots[1, .., n]$ по x, после чего каждый класс эквивалентности по x сортируем по y. Итоговый путь $Path = (1, .., n)$ будет иметь вид упорядоченной последовательности точек длины n. После сортировки определим путь: $\forall i,\ Path[i] := Dots[i]$. Данный путь и будет являться искомым: $returnPath$.

Корректность. По построению алгоритма, путь последовательно соединяет соседние точки в отсортированном массиве $Dots$. Т.е. это действительно путь. Покажем, что это путь без самопересечений. Без ограничения общности будем считать отсортированный массив $Dots$ как конкатенацию классов эквивалентности E_i, на которые разбиваются точки, где отношение эквивалентности - равенство координаты x. Осталось заметить, что в ходе работы алгоритма отрезком прямой соединяются точки либо из двух соседних классов эквивалентности - $d_1 \in E_i, d_2 \in E_{i+1}$ - либо точки из одного класса эквивалентности E_i. Допустим два отрезка пересекаются в

точке не лежащей в $Dots$. Но это возможно только для пары отрезков вида $(d_1, d_2), (d_3, d_4)$, $d_j \in E_i,\ j \in [1, .., 4]$. Это возможно тогда и только тогда когда $d_1.y < d_3.y < d_2.y < d_4.y$. Но т.к. алгоритм сортировал каждый класс эквивалентности по y и последовательно соединял соседние точки, то т.к. d_1 соединен с d_2, получается $d_2.y < d_3.y$. Противоречие.

Время работы. Весь алгоритм - сортировка по x, затем сортировка каждого класса эквивалентости по y, что может быть реализованно BalancedQuickSort за $O(n \log n)$.

Критерии.

- 3 балла - решение полностью верное, алгоритм верный, доказана корректность, верное время работы.
- Баллы снижаются вдвое при отсутствии доказательства корректности либо если оно неверное.
- Снимается от 0.5 до 1 баллов если нет сортировки по второй координате (там где это существенно)
- Снимается от 1 до 2 баллов, если время работы неверно посчитано или алгоритм неоптимален.

Задача 7(3). **7** ⟨i⟩. Приведите как можно более быстрый алгоритм, который находит в куче (Heap) с максимальным свойством второй максимум ($(n-1)$-ю порядковую статистику).

7 ⟨ii⟩. Приведите как можно более быстрый алгоритм, который находит в куче (Heap) с минимальным свойством второй минимум (вторую порядковую статистику).

Решение. **7** ⟨i⟩. Второй максимум не может находится ниже второго уровня в куче с максимальным свойством: иначе между максимумом и вторым максимумом будет ещё один элемент. Нужно вернуть максимум из двух сыновей корня.

Комментарии проверяющего и критерии.

Некоторое количество студентов, к сожалению, перепутали (или просто не знали), что бывают кучи на максимум и кучи на минимум. Они,

естественно, различаются тем, больше (или равен) родитель своего ребёнка или меньше (или равен). Неиспользование правильного свойства кучи приравнивалось к неверному решению.

Отдельно хочется заметить: частая и очень существенная ошибка утверждение следующего вида «в куче родитель больше ребёнка, так что **все** элементы на втором уровне больше **всех** элементов на последующих уровнях». Это неверное и означает, что человек не понимает, как устроена куча, в чём её свойства, преимущества и недостатки. Подобное обоснование считалось неверным, однако в большинстве случаев оно сопровождалось разумными рассуждениями о поиске максимума (минимума) из детей корня.

- -3 балла за неиспользование свойств кучи (сюда шли все поиски с помощью сортировок, линейного поиска по массиву, путаница с мнимальным и максимальным свойствами и т.д.) – незнание важного понятия курса
- -2 балла за сортировку кучи и/или последующий $\Theta(n \log n)$ алгоритм – непонимание того, что построенная куча уже имеет все нужные свойства
- -1.5 балла за решение без обоснования, либо с неправильным обоснованием, использующим, тем не менее, свойства кучи
- -1 балл за ссылку на свойство кучи без обоснования, каким образом оно связано с задачей e.g. «поскольку куча на максимум, второй максимум нужно искать среди детей корня»
- -1 балл за решение с помощью двойного извлечения максимума/-минимума из кучи с однократным окучиванием за $\Theta(\log n)$
- до -1 балла за прочие недочёты, проблемы с равными элементами и т.д.

Задача 8 (4). **8 ⟨i⟩**. Докажите, что чтобы найти все порядковые статистики $a_{(3)}, a_{(6)}, ..., a_{(3i)}$ массива $a[1, .., n]$ необходимо $\Omega(n \log n)$ сравнений.

8 ⟨ii⟩. Докажите, что чтобы найти все порядковые статистики $a_{(4)}, a_{(8)}, ..., a_{(4i)}$ массива $a[1, .., n]$ необходимо $\Omega(n \log n)$ сравнений.

Решение. Пусть достаточно совершить меньшее число сравнений. Тогда отсортируем массив $[b_1, \ldots, b_n]$ найдя все $3i$-порядковые статистики массива $[b_1, b_1, b_1, \ldots, b_n, b_n, b_n]$ асимптотически меньше, чем за $\Omega(n \log n)$, что невозможно.

Критерии. Критерии снижения (при условии применимости критерия к представленному решению):

- Не показано, как найти $a_{(2)}, \ldots, a_{(3i-1)}$, имея процедуру нахождения $a_{(3)}, \ldots, a_{(3i)}$: -1 балл
- Не сформулировано противоречие с нижней оценкой на количество сравнений: -1 балл
- Сказано, что статистик $\log n$ штук: -2 балла
- нет решения: -4 балла

Задача 9 (5). На шоссе из города A в город B расположено n отелей. Вам известно, что они находятся на расстояниях $0 < x_1, x_2, \ldots, x_n$ от города A. Вы хотите построить на шоссе бензоколонку таким образом, чтобы суммарное расстояние от нее до всех отелей было минимальным. Придумайте оптимальный алгоритм, который находит (в терминах расстояния до города A) оптимальное местоположение для бензоколонки. Докажите корректность и оптимальность алгоритма.

Решение.

Пусть $f(x)$ сумма расстояний до отелей в точке x. Тогда f - кусочно-линейная функция с точками перегиба в отелях. На каждом отрезке между отелями функция убывает, если справа больше точек, чем слева и наоборот. Значит функция унимодальна вниз с минимумом в точке слева и справа от которой отелей поровну - это медиана.

Критерии.

- Неточное решение (с ε-точностью): 0.5 балла
- Нет решения за исключением фразы «$\frac{n}{2}$ статистика» или подобного: 0.5 балла
- Неоптимальное решение: 1 балл

- Оптимальное решение: 2 балла

- Выписана и обоснована корректная асимптотика оптимального алгоритма: +1 балл
- Корректность алгоритма доказана с неточностями: +1 балл
- Корректность алгоритма доказана: +2 балла

Задача 10 (6). Имеется алгоритм, время работы которого оценивается следующей рекуррентой (n — длина входа):

10 ⟨i⟩.

$$T(n) = \begin{cases} C, & n < A,\ A, C - \text{некоторые константы} \\ 3 \cdot T(\lfloor \frac{n}{2} \rfloor) + n, & n - \text{полный квадрат} \\ 3 \cdot T(\lfloor \frac{n}{2} \rfloor) + n^2, & \text{иначе} \end{cases}$$

10 ⟨ii⟩.

$$T(n) = \begin{cases} C, & n < A,\ A, C - \text{некоторые константы} \\ 4 \cdot T([\frac{n}{3}]) + n, & n - \text{полный куб} \\ 4 \cdot T([\frac{n}{3}]) + n^2, & \text{иначе} \end{cases}$$

Дать наилучшие нижнюю и верхнюю оценки.

Решение. Отметим, что рекуррента $T(n) \leq T'(n) = 3 \cdot T(\lfloor \frac{n}{2} \rfloor) + n^2$. По основной теореме о рекурсии, $f(n) = n^2 = \Omega(n^{\log_2 3 + \varepsilon})$, $\varepsilon = 0.01 > 0$ и таком образом $T(n) = O(n^2)$ при любых n. Заметим, что пока мы не приводим пример входа на котором данная оценка достигается (и нам он не понадобится).

Посмотрим на наилучшую нижнюю оценку. Лучший из возможный сценариев - $T(n) = 3 \cdot T(\lfloor \frac{n}{2} \rfloor) + n$, при том условии, что при каждом рекурсивном вызове $\lfloor \frac{n}{2^k} \rfloor$ является полным квадратом. Покажем, что данная ситуация не может быть реализована ни при каких $n > 1$. Рассмотрим некоторый полный квадрат $n = m^2$, $m \in \mathbb{N}$. При четном $m = 2k$ после рекуррсивного вызова получим $n = 4k^2 \to \lfloor \frac{n}{2} \rfloor = 2k^2$. Но $2k^2$ не является полным квадратом в кольце целых чисел. При нечетном $m = 2k+1$ имеем

$n = 4k^2 + 4k + 1 \to \lfloor \frac{n}{2} \rfloor = 2k^2 + 2k = 2k(k+1)$. Последнее число можно переписать как $4 \cdot \binom{k+1}{2}$. В данной ситуации имеется две возможности - в первой, $4 \cdot \binom{k+1}{2}$ не является полным квадратом и тогда следующий рекурсивный вызов будет стоить $\lfloor \frac{n}{2} \rfloor^2$. Во втором случае, $\binom{k+1}{2}$ является полным квадратом и тогда $\lfloor \frac{n}{2} \rfloor$ также полный квадрат, но уже при следующем переходе мы получаем, что $n \to \lfloor \frac{n}{2} \rfloor \to \lfloor \frac{n}{2}/2 \rfloor = 2 \cdot \binom{k+1}{2}$ и аналогично рассуждениям выше, последнее число не является полным квадратом. Таким образом, $T(n)$, в наилучшем случае, определяется как сумма $\sum_{k=0}^{\log(n)} g(\frac{n}{2^k})$, причем в этой сумме как минимум одни из любых трех последовательных слагаемых имеет степень 2. Таким образом, эта сумма оценивается снизу как

$$\begin{aligned} T(n) &= \sum_{k=0}^{\log(n)} g(\frac{n}{2^k}) \\ &\geq \sum (\ldots + \lfloor \lfloor \frac{n}{4} \rfloor \rfloor^2 + \ldots) + (\ldots + \lfloor \lfloor \frac{n}{4 \cdot 8} \rfloor \rfloor^2 + \ldots) + \ldots \\ &\geq \frac{n^2}{16} \end{aligned}$$

То есть, наилучшая нижняя оценка - $\Omega(n^2)$.

В купе со сказанным выше о верхней оценке, получаем итоговую оценку работы рекурренты $\Theta(n^2)$.

Критерии.

- 3 балла - полностью доказана верхняя оценка (либо как в решении, либо приведен конкретный пример и *доказано*, что на нем оценка достигается)
- 3 балла - полностью доказана нижняя оценка - показано, что для всякого n сумма по дереву рекурсии не меньше $c \cdot n^2$. При этом полностью обосновано почему то или иное число не является квадратом.

Задача 11(7). На плоскости даны n точек с координатами $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $(x_i, y_i) \in \mathbb{Z}^2$, не лежащие на одной прямой. Привести алгоритм для нахождения замкнутого пути (цикла), проходящего через n точек без самопересечения, считая, что точки соединяются отрезком прямой.

Решение.

Сведем решение второго пункта к первому. Для этого определим множество точек E_y^{max} с максимальной координатой y. Если таких точек несколько, то выберем из них две точки d_s, d_f - первая имеет минимальную координату x, вторая - максимальную. Если такая точка одна, то $d_s = d_f$. Произведем теперь разбиение множества точек $Dots' = Dots\ E_y^{max}$ следующим образом. Определим для каждого $d \in Dots'$ функцию $k(d) : \mathbb{R}^2 \to \mathbb{R} * \cup\{\infty\}$ определяемую как

$$k(d) = \begin{cases} \frac{d_s.y - d.y}{d_s.x - d.x}, & \text{если} d_s.x \neq d.x \\ \infty, & \text{иначе} \end{cases}$$

Соответственно, определим массив K как массив пар $(k(d), d)$. Отсортируем массив K по убыванию по k, а каждый класс эквивалентности (точек с одинаковыми значениями k) отсортируем по $d.y$. После чего применим алгоритм из прошлого пункта к массиву K. Получили путь $Path_1 = (d_first, .., d_last)$. Наконец, последовательно соединим точки из отсортированного массива E_y^{max} - получаем путь $Path_2 = (d_s, .., d_f)$ - и завершаем алгоритм соединением пар точек (d_{first}, d_s), (d_f, d_{last}). В случае, если значения k одинаковы, а $|E_y^{max}| = 1$, то решения нет. Во всех остальных случаях, получаем цикл $Cycle$, который является искомым.

Корректность. Поясним смысл алгоритма. В первом пункте вы в качестве заметающей плоскость прямой выбрали ось Oy, двигающуюся слева направо, которая последовательно соединяла точки, через которые она проходит. Идея, лежащая в основе этой задаче: чтобы получить цикл, выберем в качестве заметающей прямую, поворачивающуюся вокруг некоторой фиксированной точки. При заметании мы также разбиваем точки на классы эквивалентности, поэтому доказательство корректности аналогично пункту выше. Единственный момент, который стоит пояснить и который нельзя напрямую объяснить первым пунктом это тот факт, что решение не всегда существует. Действительно, если все точки лежат на одной прямой, то цикла без самопересечений, очевидно, не существует. Во всех остальных случаях, заметающая прямая делит точки как минимум на два класса эквивалентности и, в силу рассуждения из первого пункта, существует путь, проходящий через все точки кроме фокусной без самопересечений и замыкающийся в цикл соединением краев с фокусом.

Время работы. Вычисление и сортировка E_y^{max} - $O(n \log n)$ в худшем случае. Вычисление массива K и его сортировка - $const \cdot n + O(n \log n) =$

$O(n \log n)$. Склейка путей в цикл - $O(n)$. Итого - $O(n \log n)$

Критерии.

- 7 баллов - решение полностью верное, алгоритм верный, доказана корректность, верное время работы, решение оптимально.
- Баллы снижаются вдвое от максимального (-3.5) либо при отсутствии доказательства корректности, либо если оно неверное, либо верное решение ссылается на 6, которая решена неверно или необоснованно.
- 0 баллов если алгоритм не выдает цикл.
- 0 баллов, если решение было написано столь плохо (непонятно, нечитаемо, неисполняемо и т.д.).
- Снимается от 2 до 7 баллов, если время работы неверно посчитано или алгоритм неоптимален (по степени неадекватности).