# Алгоритмы. Теор.листок 4

Артемий Клячин

1 декабря 2021 г.

## Задача 1.

Рассмотрим две окружности с центрами в A и B и радиусами r и R соответсвенно, тогда множество точек, из которых они видны под один углом это окружность с центром в точке $A + \overline{(A-B)} * (r^2/(R^2 - r^2))$ и радиусом $|A - B| * (rR/(R^2 - r^2)$.

По условию даны три окружности. Для двух пар окружнотей строим окружности, из которых они видны под один углом. Любоя точка пересечения двух полученных окружностей и будет искомой.

## Задача 2.

Считаем, что острова не имеют общих точек, иначе бы их не назвали бы островами. Т.е. корабль радиуса 0 гарантировано проходит. С другой стороны корабль размера $\frac{L}{2}$ (где $L$ – ширина реки) гарантировано не проходит. Будем искать максимальную ширину корабля двоичным поиском. Тогда нам потребуется провести $\log_2(\frac{L}{2\epsilon}) = O(\log(\frac{1}{\epsilon}))$ испытаний. Покажем, что каждое испытание можно провести за $O(n^2)$. Для этого прономеруем центры островов числами от 1 до $n$. На одном береге поставим точку 0, на другом точку $(n + 1)$. Пусть мы проверяем походит ли корабль радиуса $x$. Увеличим радиусы всех островов на $x$. При этом некоторые могут перекрыть друг друга. Также сдвинем каждый берег реки на  в направлении другого берега. Если существует путь из точки 0 в точку $(n+1)$ полностью по суше, то точечный корабль не сможет пройти по новой реке, соответственно корабль радиуса $x$ не сможет пройти по исходной реке. Обратное тоже верно. Для поиска пути по суше нам нужно построить граф, состоящий из пар островов (или острова и берега) и провести ребро между точками, соответствующими этим двум объектам, если они перекрылись. Построение графа займёт $O(n^2)$ времени, поиск пути из вершины $O$ в $(n + 1)$: $O(n)$ времени. Таким образом искомое время работы достигнуто.

## Задача 3.

Отсортируем точки отдельно по координате $x$ и по координате $y$ за $O(nlogn)$. Будем использовать метод “разделяй и властвуй”. Разделим точки вертикальной прямой  на 2 примерно разных множества. Точки на прямой также

поделим так, чтобы размены множеств отличались не более, чем на один. Пусть для каждого множеств мы решили задачу за время $T(\frac{n}{2})$. Обозначим периметр минимального треугольника через $P$. Если для исходного множества существует треугольник с периметром меньшим $P$, то его вершины будут относиться к разным множествам, но при этом расстояние между любой парой вершин будет меньше, чем $\frac{P}{2}$. Т.е. все вершины будут находится в вертикальной полосе ширины $P$, с прямой  посередине. Будем двигаться по списку точек в направлении возрастания $y$ (мы уже отсортировали точки) и рассматривать только точки из указанной полосы. Для каждой текущей точки $A$, пара других вершин, лежащих выше и образующих совместно треугольник с периметром меньшим $P$, может находится только в прямоугольники высотой $\frac{P}{2}$ и шириной $P$, параллельными осям координат, симметричном относительно прямой , и с точкой $A$ на нижней границе. В левой стороне прямоугольника (размером $\frac{P}{2} \cdot \frac{P}{2}$ может находится максимум 8 точек, так чтобы они не образовывали треугольник с периметром меньшим $P$). Аналогично, и в правой половине квадрата не более 12 точек. Т.е. для каждой точки $A$ мы должны проверить константное число ближайших соседей (по $y$) из каждого из множеств. Т.е. за линейное время $O(n)$ мы проверим все возможные треугольники с периметром меньшим $P$ в полосе ширины $P$. Таким образом время работы алгоритма будет определяться рекуррентной формулой $T(n) = 2 * T(n/2) + O(n)$, из которого получается $T(n) = O(nlogn)$. Доказательство факта, что в квадрате $\frac{P}{2} \cdot \frac{P}{2}$ может располагаться не более 8 точек так, чтобы не было треугольника с периметром меньшим $P$. Разобьём квадрат на 4 маленьких квадратика $\frac{P}{4} \cdot \frac{P}{4}$. Пусть у нас 9 точек, удовлетворяющих условию, и по принципу Дирихле в один из квадратиков попадает 3 точки. Квадратик вписан в окружность радиуса $\frac{P}{8} \cdot \sqrt{2}$. Для окружности радиуса $R$, максимальный периметр имеет правильный треугольник и этот периметр равен $3 \cdot \sqrt{3} \cdot R = 3 \cdot \sqrt{6} \cdot \frac{P}{8} < 0.92P$. Противоречие.

## Задача 4.

Прямоугольник – выпуклая фигура, поэтому, если он содержит все $n$ точек, то он содержит также выпуклую оболочку на этих точках. Поэтому начнём с построения выпуклой оболочки для n точек алгоритмом Грэхэма-Эндрю за $O(n \log n)$. За $O(n)$ найдём самую левую, правую, верхнюю и нижнюю точки. Построим через низ прямые параллельные осям координат, мы получим прямоугольник, который содержит все точки. Будем одновременно (все прямые поворачиваются на одинаковый угол) вращать прямые, на которых лежат стороны прямоугольника так, чтобы все точки всегда оставались внутри прямоугольника. В какие-то моменты стороны выпуклой оболочки будут совпадать с частью прямой и у нас будет меняться точка, относительно которой вращается прямая. Но таких моментов будет не более $4n$. Между двумя моментами площадь прямоугольника описывается простой тригонометрической формулой, для которой легко найти максимум аналитически.

## Задача 5.

Строим выпуклую оболочку за O(n log n). Возможно, что она получится вырожденной. Прямая будет разделяющей, если она пересекает выпуклую оболочку. Отдельно храним уравнения прямых соответствующих верхним и нижним частям выпуклой оболочки в виде уравнений y = $m_i * x + b_i$.$m_i$., $O(1).y = Mx+B.O(logn), Mm_im_{i+1}.$., ıjM()i−(i+1)., ıj.(1)ıjıj.$O(logn).-$ , $qO(qlogn)$..

## Задача 6.

Для удобства вычислим частичные суммы с конца:

$$p_k = \sum_{i=k}^{n} b_i,$$

тогда стоимость массива будет равна:

$$S = \sum_{i=1}^{n} p_i,$$

и

$$b_k = p_k - p_{k+1}.$$

Пусть мы переставляем $k$-тый элемент в $m$-тую позицию. Рассмотрим случай $m > k$ (случай $m < k$ рассматривается аналогично). Тогда изменение стоимости массива составит:

$$\delta_{k,m} = (m-k)b_k - \sum_{i=k+1}^{m} b_i = (m-k)(p_k - p_{k+1}) - (p_{k+1} - p_{m+1}).$$

Для каждого $m > 1$ и каждого $1 \le k < m$ нужно вычислить $\delta_{k,m}$ и выбрать максимальное значение. Тривиальный алгоритм вычислит это за $O(N^2)$. Чтобы вычислить за меньшее время, воспользуемся приёмом динамического программирования, называемым Convex Hull Trick.

Для каждого $m > 1$ будем искать максимум среди семейства линейных функций ($1 \le k < m$):

$$y_k(x) = (x-k)(p_k - p_{k+1}) - p_{k+1}$$

в точке $x = m$. Это можно сделать за $O(\log m)$, если у нас есть уже построенная верхняя огибающая. Но огибающая у нас достраиваится постепенно в процессе увеличения $m$ и полное построение огибающей занимает $O(n)$. Итак за $O(n \log n)$ мы можем вычислить:

$$\Delta_{k<m} = \max_{1<m\le n, 1\le k<m} \delta_{k,m} = \max_{1<m\le n} (p_{m+1} + \max_{1\le k<m} (y_k(m))).$$

Аналогично рассматривается случай $k > m$, который даёт максимальное изменение стоимости $\Delta_{k>m}$ и ответ на задачу:

$$S + \max\{\Delta_{k<m}, \Delta_{k>m}\}$$

получен за $O(n \log n)$.