Разные приколы в геометрии

pakhomovee

Compiled: 7 августа 2023 г.

1 Про выпуклую оболочку

Задача 1.

Дано n точек. Найдите их выпуклую оболочку (выпуклый многоугольник минимальной площади, внутри которого лежат все точки).

1.1 Алгоритм Джарвиса

Выберем какую-нибудь точку p_0, которая гарантированно попадёт в выпуклую оболочку. Например, нижнюю, а если таких несколько, то самую левую из них.

Дальше будем действовать так: найдём самую «правую» точку от последней добавленной (то есть точку с минимальным полярным углом) и добавим её в оболочку. Будем так итеративно добавлять точки, пока не «замкнёмся», то есть пока самой правой точкой не станет p_0.

1.2 Алгоритм Грэхема

Алгоритм Грэхема — это оптимизация алгоритма Джарвиса, основанная на следующем наблюдении: если отсортировать все точки по полярному углу относительно точки p_0, то выпуклая оболочка будет какой-то подпоследовательностью такого отсортированного массива точек.

Алгоритм последовательно строит выпуклые оболочки для каждого префикса этого отсортированного масива. При добавлении i-й точки в оболочку нужно удалить сколько-то последних добавленных точек, которые не будут входить в новую оболочку. Чтобы это делать эффективно, мы можем хранить выпуклую оболочку в стеке и в цикле while мотреть на три последние точки и проверять, образуют ли они правый поворот. Если это так, то среднюю следует удалить — мы нашли треугольник $(p_0, p_i, p_{i-2}$, который содержит p_{i-1}, значит её можно удалить.

Каждая точка будет добавлена один раз удалена не более одного раза, что занимает константное количество операций. Соответственно, время работы будет упираться во время работы сортировки, то есть $O(n \log n)$.

2 Про СНТ

Задача 1.

Нужно обрабатывать запросы:

- добавить в набор прямую вида $(f, xy) = ax + by$
- по паре (x, y) найти минимум $f(x, y)$ по прямым в наборе

Просто уберем лишнюю переменную:

$$ax + by = x\left(a + b\frac{y}{x}\right)$$

Получили функцию от одной переменной, значит, нам нужно самое обычное СНТ.

Задача 2.

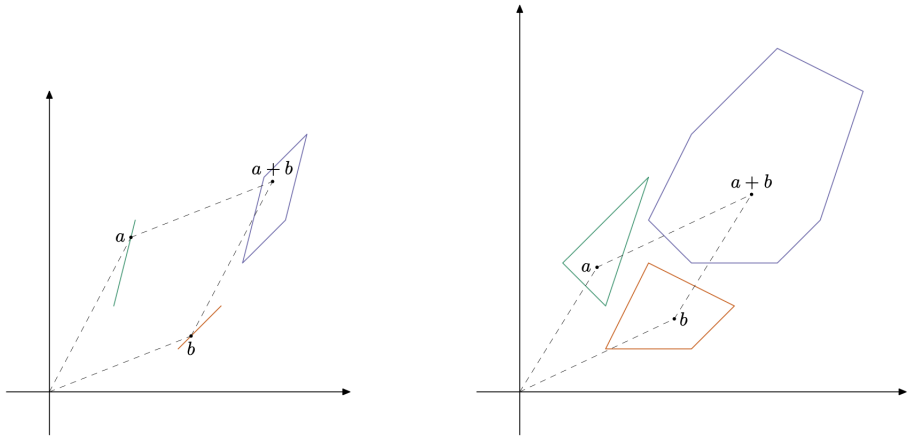
Найдите сумму двух нижних огибающих.

Метод 1. Выпишем все точки перегиба, удалим дубликаты. Померджим списки прямых. Перестроим огибающую с нуля. Все операции можно выполнить за $O(n)$.

Метод 2. Можно сделать два указателя и реализовать это in-place.

3 Про сумму Минковского

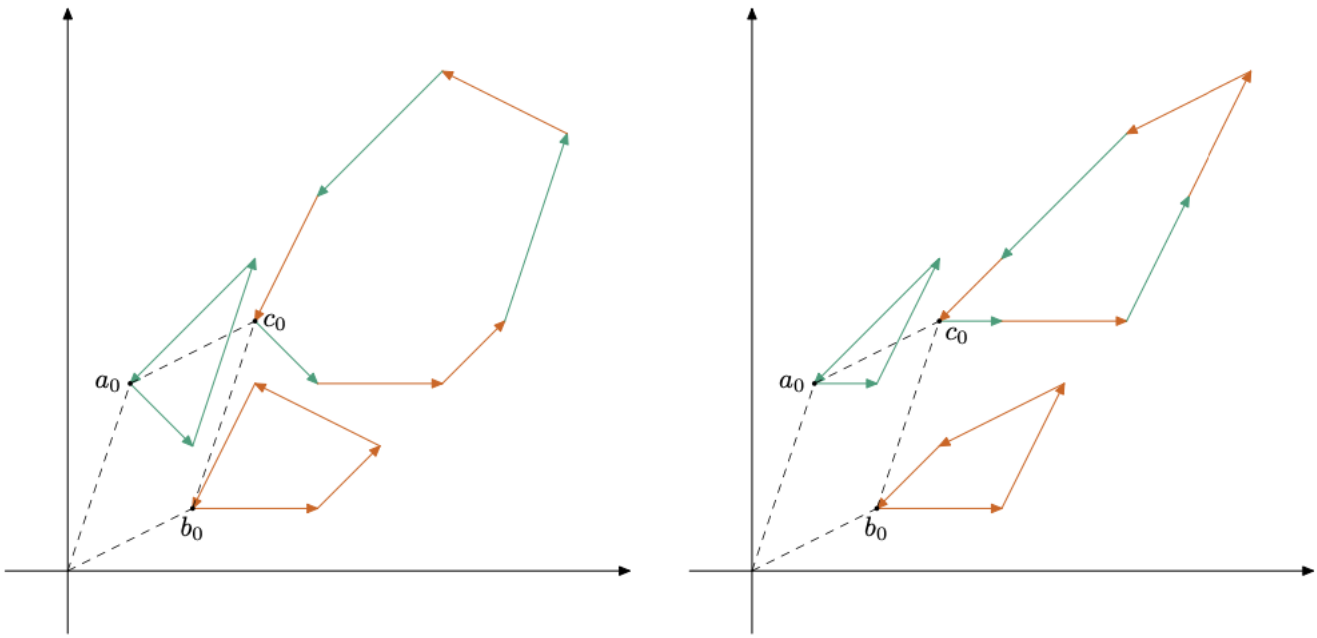
Суммой Минковского двух множеств A и B называется множество $C = \{a + b : a \in A, b \in B\}$.



(a) Сумма Минковского двух отрезков (b) Сумма двух двух выпуклых многоугольников

Рис. 1: Примеры суммы Минковского

Несложно показать, что сумма Минковского выпуклых многоугольников также является выпуклым многоугольником, а количество сторон этого многоугольника равно сумме количество сторон исходных. Если ориентировать стороны всех многоугольник в направлении обхода против (или по) часовой стрелке, то множество векторов, получившихся из сторон Ñ, совпадет с объединением множеств векторов, получившихся из сторон À и Â. Исключением является случай, когда в множестве векторов сторон À и Â есть колинеарные вектора. В таком случае, в множестве сторон Ñ будет находиться сумма этих двух колинеарных векторов. Можно не рассматривать этот случай, как исключение, а сказать, что таком случае, Ñ может получиться не строго выпуклым, т.е. иметь несколько подрядидущих вершин на одной прямой.



(a) Сумма двух выпуклых многоугольников (b) Пример с колинеарными векторами

Рис. 2: Нахождение суммы Минковского

Отсюда следует алгоритм нахождения суммы Минковского двух выпуклых многоугольников. Найдем в каждом многоугольнике вершину, крайнюю в выбраном направлении. v Направление v нужно выбрать таким, чтобы в каждом многоугольнике крайней оказалась одна точка, а не сторона. Например, можно выбрать в многоугольниках минимальные в лексикографическом порядке точки, тогда $v = (-\infty, -c_1)$. Пусть, в A это будет вершина a_0, а в B — b_0. Тогда $c_0 = a_0 + b_0$ является крайней вершиной в C в направлении v. Теперь положим вектора сторон многоугольников A и B в массив, и отсортируем по углу (конечно же, если точки имеют целые координаты, компаратор для сортировки тоже нужно писать в целых числах). Будем откладывать вектора в порядке сортировки, начиная от точки c_0.

Это можно реализовать in-place при помощи двух указателей.

Применения

- Пересечение выпуклых многоугольников
- Расстояние между выпуклыми многоугольниками

Рассмотрим их по отдельности.

Пересечение

Даны два выпуклых многоугольника A и B. Нужно проверить пересекаются ли они.

$$A \cap B = \varnothing \Leftrightarrow \exists p, p \in A, p \in B$$

Пусть B' — многоугольник B, который отразили относительно начала координат.

$$\exists, p \in A, p \in B \Leftrightarrow \exists p, p \in A, (-p) \in B \Leftrightarrow 0 \in A + B'$$

Таким образом, нужно просто проверить лежит ли точка $(0, 0)$ внутри суммы Минковского многоугольников A и B'.

Расстояние

Расстояние между многоугольниками определяется следующим образом:

$$dist(A, B) = \min_{a \in A, b \in B} dist(a, b)$$

Введем тот же B', что и в случае пересечения. Тогда:

$$dist(A, B) = \min_{a \in A, b' \in B'} |a + b'| = \min_{c \in (A+B')} |c|$$

Таким образом, чтобы найти расстояние между двумя выпуклыми многоугольниками A и B, нужно найти расстояние от начала координат до суммы Минковского A и B'. Растояние от точки до многоугольника равно минимуму из расстояний от точки до его сторон (либо 0, если точка лежит внутри многоугольника).

4 Про касательные

Мы хотим научиться легко искать касательную к выпуклому многоугольнику из точки за $O(\log n)$. Воспользуемся следующим алгоритмом:

Пусть $k = \lceil \log_2 n \rceil$, $i = 0$, A —наш многоугольник, а P_0 —точка из каторой мы ищем левую касательную.

$k + 1$ раз делаем следующее:

1. пусть $p = i - 2^k, q = i + 2^k$ (оба числа считаем по модулю n)
2. рассмотрим прямые по точкам $(P_0, A_i), (P_0, A_p), (P_0, A_q)$
3. пусть t — значение второй точке в паре, соответствующей самой левой прямой из этих трех. Сделаем $i := t$.
4. $k := k - 1$

5 Про каллиперы

Задача 1.

Есть множество точек на плоскости. Нужно найти две самые удалённые из них.

Очевидно, что диаметр множества соединяет какие-то две точки на выпуклой оболочке.

Definition 5.1

Прямая L называется опорной прямой (англ. line of support) для многоугольника P, если его внутренность лежит по одну сторону от L, при этом L проходит хотя бы через одну из вершин P.

Пусть L_1, L_2 —две параллельные опорные прямые фигуры Φ, расстояние между которыми имеет максимальное значение. A_1, A_2 —граничные точки фигуры Φ, принадлежащие соответственно прямым L_1 и L_2. Тогда отрезок A_1A_2 перпендикулярен обеим прямым.

Доказательство. Предположим, что это не так. Тогда расстояние между прямыми L_1, L_2 меньше, чем отрезок A_1A_2, то есть меньше, чем расстояние между опорными прямыми L_1', L_2' фигуры Φ, перпендикулярными A_1A_2. Противоречие. 😎

Theorem 5.2

Диаметр выпуклого многоугольника равен максимальному расстоянию между параллельными опорными прямыми.

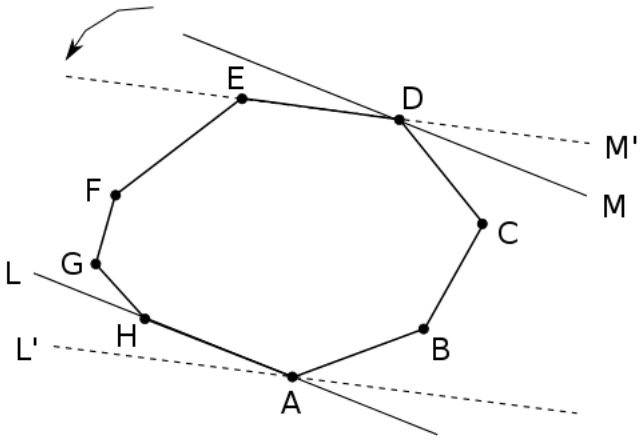
Доказательство. Пусть Φ — выпуклая фигура, L_1 иL_2 — параллельные опорные прямые, расстояние между которыми имеет наибольшее возможное значение d,A_1 и A_2 — общие точки фигуры Φ и прямых L_1 и L_2 соответственно. По предыдущей теореме A_1A_2 перпендикулярен к прямым L_1, L_2, следовательно, его длина равна d. Докажем, что расстояние между любыми двумя точками фигуры Φ не преводходит d. Действительно, если B и C — какие-либо две точки фигуры Φ, а m и n — опорные прямые, перпендикулярные к отрезку BC, то отрезок BC не превосходит расстояния между прямыми m и n, которое в свою очередь не превосходит d. Следовательно, длина BC не может быть больше d. 😎

Заметим, что параллельные опорные прямые можно провести не через любую пару точек.

Definition 5.3

Точки, через которые можно провести параллельные опорные прямые, будем называть противолежащими (англ. antipodal).

Благодаря предыдущей теореме нам нужно рассмотреть только противолежащие точки. Задача в том, чтобы рассмотреть их, не перебирая все пары точек.



Рассмотрим рисунок справа. L и M — опорные прямые, проходящие через вершины A и D соответственно. Значит, (A, D) — противолежащая пара точек. Если начать вращать прямые против часовой стрелки вокруг данных точек, они будут оставаться опорными прямыми, пока одна из прямых не станет накладываться на сторону многоугольника. В нашем примере, при вращении M к позиции M,M коснётся точки E раньше, чем L коснётся B, поэтому (A, E) становится новой парой противолежащих точек. Теперь M будет вращаться вокруг E, а L продолжит вращаться вокруг A, и следующей парой противолежащих точек станет (B, E). Продолжая таким образом, мы сгенерируем все пары противолежащих точек, так как параллельные линии пройдут под всеми возможными углами. Определение новой пары противолежащих точек требует только одного сравнения углов, которое можно выполнить с помощью предиката поворота.

Данный алгоритм можно реализовать следующим образом: хранить указатели на противолежащие вершины, и на каждой итерации алгоритма увеличивать либо один из данных указателей, либо сразу оба (когда обе прямые проходят через сторону многоугольника), и заканчивать работу, когда опорные прямые сделают полный круг. Таким образом, каждая из вершин будет посещена каждой из прямых не более двух раз.

6 Про рандом

6.1 Поиск пары ближайших точек

Задача 1.

Даны n точек на плоскости. Требуется найти такую пару точек, что расстояние между ними минимально среди всех расстояний между всеми парами данных точек.

Будем добавлять точки по одной в случайном порядке и поддерживать пару ближайших точек. Пусть на текущем шаге алгоритма расстояние между ближайшей парой точек равно d. Разобьем плоскость на квадраты $d \times d$, для каждого из них сохраним точки внутри. Так как d — наименьшее возможное расстояние между парой точек, в одном квадрате может лежать не более четырех точек.

Если при добавлении точки минимальное расстояние изменилось, то есть точка на расстоянии менее d от нее. Эта точка будет находиться либо в том же квадрате, либо в соседнем.Значит, нам нужно перебрать не более 36 точек, чтобы убедиться, что точка не меняет минимальное расстояние. Если она его поменяла, то полностью перестраиваем нашу структуру.

Заметим, что перестройка структуры происходит, если текущая точка одна из наиболее близких на префиксе. Вероятность такого события $\frac{2}{k}$ (если есть несколько одинаковых минимальных расстояний, то либо до добавления этой точки расстояние уже было таким, и ничего перестраивать не надо, либо же во всех таких минимальных расстояниях одной из точек является наша новая точка, и тогда вероятность еще меньше — $\frac{1}{k}$). Структуру мы перестроим за $O(k)$. Итоговая асимптотика равна $O(\sum\limits_{k=1}^{n} \frac{2}{k} \times k) = O(n)$.

Легко обобщить этот алгоритм для бóльших размерностей. Нужно побить пространство на кубы со стороной d и искать пару для новой точки во всех соседних кубах. Для фиксированной размерности алгоритм будет все еще линейным.

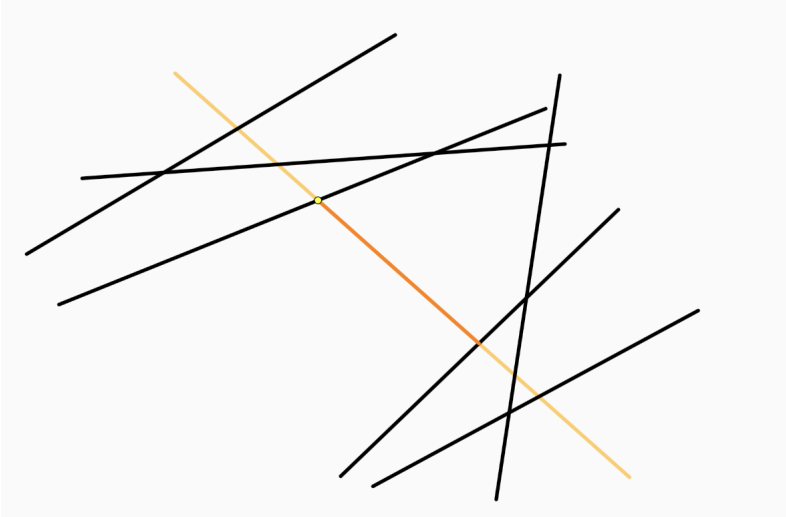
6.2 Есть ли точка в пересечении полуплоскостей

Задача 2.

Даны n полуплоскостей. Требуется найти точку, лежащую во всех этих полуплоскостях, либо сказать, что такой нет.

Задачу пересечения полуплоскостей можно решить за время $O(n \log n)$ различными способами. Они находят не только одну точку, но все множество пересечения. Мы же рассмотрим рандомизированный алгоритм, работающий за ожидаемое время $O(n)$. Давайте будем добавлять полуплоскости по одной в случайном порядке, поддерживая самую высокую точку в множестве пересечения уже добавленных полуплоскостей (Если таких точек несколько, то самую левую из них. При этом изначально ограничим все квадратом $[-10^9, -10^9] \times [10^9, 10^9]$ при помощи четырех полуплоскостей, чтобы не было проблем с тем, что эта точка бесконечно удалена).

Тогда если при добавлении новой полуплоскости эта точка лежит в очередной полуплоскости, то ничего не поменяется, потому что множество пересечения могло только уменьшиться, но при этом его верхняя точка сохранилась. Если же эта точка не лежит в новой полуплоскости, то придется заново искать самую верхнюю точку. Заметим, что эта новая точка обязана лежать на прямой, высекающей новую полуплоскость. То есть нам надо найти самую высокую точку на прямой, лежащую в пересечении полуплоскостей. Но ведь все остальные полуплоскости высекают на этой прямой какие-то лучи. Поэтому нам надо найти самую высокую точку в пересечении лучей. Для этого надо отдельно посмотреть на все лучи, смотрящие вниз, и взять из них самый нижний (начинается в точке A). Аналогично взять все лучи, смотрящие вверх, и взять из них самый верхний. Проверить, что эти два луча пересекаются, и в таком случае взять точку A. Это можно сделать за линейное время.



Вероятность, что выбранная прямая лежит на краю равна $\frac{2}{k} \Rightarrow$ асимптотика равно $O(\sum\limits_{k=1}^{n} \frac{2}{k} \times k) = O(n)$.

6.3 Минимальная покрывающая окружность

Задача 3.

Даны n точек на плоскости. Требуется найти окружность минимального радиуса, такую что все данные точки лежат внутри или на границе этой окружности.

В отличие от других рассмотренных рандомизированных геометрических алгоритмов, у которых есть общеизвестные детерминированные аналоги, работающие в $\log n$ раз дольше, в данном случае рандомизированное решение будет единственным применимым с практической точки зрения.

Есть два варианта того, как будет выглядеть минимальная покрывающая окружность: либо это описанная окружность какой-то тройки из данных точек, либо окружность, построенная на отрезке между двумя какими-то точками как на диаметре, потому что во всех других случаях ее радиус можно немного уменьшить так, чтобы эта окружность все еще содержала в себе все данные точки.

Давайте по очереди добавлять точки в случайном порядке, поддерживая минимальную покрывающую окружность текущего набора точек. Если новая точка лежит внутри текущей окружности, то ничего менять не надо, а в противном случае нужно перестроить окружность. Стоит заметить, что если новая точка не лежит в старой окружности, то она обязана лежать на новой минимальной окружности. На окружности лежит 3 точки, поэтому вероятность такого события — $\frac{3}{k}$, где k — индекс новой точки (если на окружности лежит больше 3 точек, то эта окружность была минимальной еще до добавления текущей точки, а если окружность построена на двух точках как на диаметре, то вероятность еще меньше — $\frac{2}{k}$). Теперь если мы построим линейный алгоритм, который находит минимальную покрывающую окружность, проходящую через новую точку, то итоговая асимптотика алгоритма будет $O(\sum\limits_{k=1}^{n} \frac{3}{k} \times k) = O(n)$.

Алгоритм нахождения минимальной покрывающей окружности, проходящей через новую точку аналогичен алгоритму, который мы только что рассмотрели. Мы перебираем все остальные точки в случайном порядке, и если новая точка лежит вне текущей окружности, то эту окружность нужно перестроить. Вероятность того, что новая точка лежит на минимальной окружности не больше $\frac{2}{k}$, поэтому алгоритм будет линейным.

Остается последний этап: зафиксированы уже две точки и нужно построить минимальную покрывающую окружность, проходящую через них. Алгоритм аналогичен. Начинаем с окружности, построенной на отрезке между этими двумя точками как на диаметре, а затем добавляем остальные точки в случайном порядке. Вероятность, что окружность нужно перестроить не больше $\frac{1}{k}$, при этом если зафиксированы уже три точки на окружности, то окружность определяется однозначно.

7 Про полуплоскости, если есть точка внутри пересечения

Пусть точка P лежит СТРОГО в пересечении полуплоскостей. Переместим P в начало координат. Все полуплоскости сдвинем на вектор $-P$. еперь начало координат лежит в пересечении полуплоскостей. Давайте все прямые вида $ax + by + c = 0$ заменим на точки $\left(\frac{a}{c}, \frac{b}{c}\right)$. Теперь мы перешли к двойственной задаче, и у полученного набора точек нужно найти выпуклую оболочку, что уже намного проще. После этого мы найдем список индексов прямых, которые образуют пересечение полуплоскостей.

8 Про вращающийся сканлайн

Задача 1.

Даны n точек на плоскости. Есть q прямых. Для каждой прямой найдите расстояние до ближайшей точки.

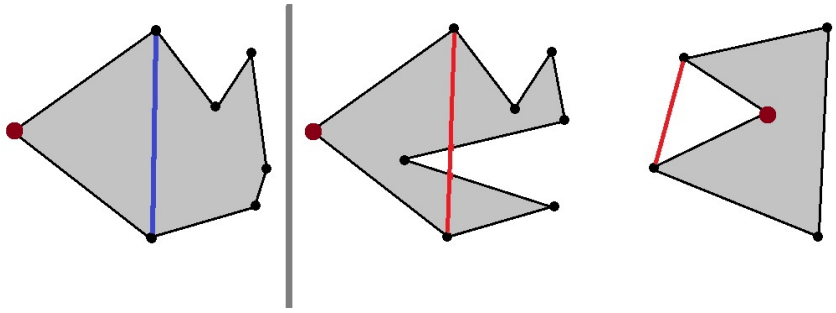
Рассмотрим все прямые между парами точек (с ориентированными напрявляющими векторами). Прямые-запросы добавим в этот массив тоже и отсортируем всё по полярному углу. Будем идти по прямым (по тем, которые не являются запросами) и для текущей прямой поддерживать точки в порядке сортировке по ОРИЕНТИРОВАННОМУ расстоянию до неё. Начать проще всего с горизонтальной прямой. Заметим, что порядок точек изменится только в момент, когда мы будем перебирать прямую через эти две точки. Таким образом, для каждой прямой мы научились поддерживать точки в порядке сортировки за $O(n^2)$ суммарно. В момент, когда к нам приходит прямая-запрос у нас уже есть сортировка для параллельной ей прямой, поэтому ответить на запрос легко за $O(\log n)$ при помощи бинпоиска.

$O(n^2 + q \log n)$.

9 Триангуляция за $O(n^2)$

Задача 1.

Дан произвольный многоугольник. Найдите его триангуляцию.



Theorem 9.1

У любого простого n-вершинного многоугольника P всегда существует два не пересекающихся между собой уха.

Проверку на то, является ли вершина ухом реализуем за $O(n)$. В начале найдем все уши и положим их в очередб. Также будет хранить двусвязный список текущих вершин (и массив, в котором можно посмотреть лежит ли уже вершина в очереди). На очередном шаге алгоритма будем брать первую вершину из очереди и удалять её. После этого ухом могли стать только её соседи (любое новое ухо точно содержит одного из её соседей), поэтому, если её соседи еще не в очереди, проверим, стали ли они ушами и, если надо, добавим их в очередь.

Самая долгая часть алгоритма — очередь. Заметим, что при каждом удалении вершины мы делаем $O(1)$ проверок на ухо. Удалений $O(n)$. Итоговая асимптотика $O(n^2)$.