Санкт-Петербургский Государственный Университет
Математико-механический факультет
Кафедра системного программирования

Псевдо-прямоугольники и псевдо-треугольники с пустой внутренностью.

Курсовая работа студента 445 группы
Копелиовича Сергея Владимировича

Научный руководитель К.В. Вяткина

Санкт-Петербург
2010

Содержание

1 Введение

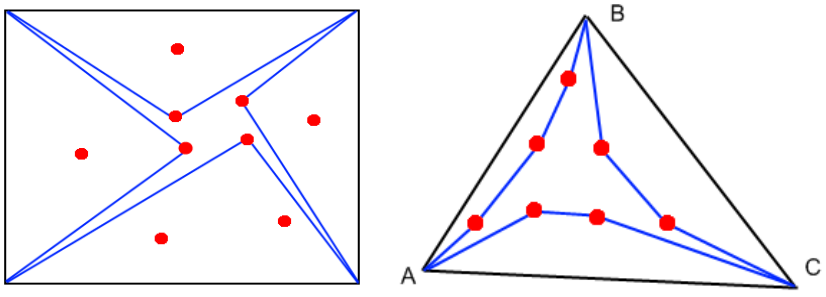
Псевдо-треугольником называется замкнутая ломаная, имеющая ровно три вершины выпуклы. Псевдо-треугольники были введены в рассмотрение при решении задачи планирования движения на плоскости с выпуклыми (например, многоугольными) непересекающимися препятствиями [4, 2]. О других задачах, в которых применяются псевдо-треугольники можно прочитать в [4, 2].

Изучение псевдо-трегуольников и их обобщения — псевдо-прямоугольников и псевдо-четырехугольников может дать более четкое понимание того, как эти объекты утроены и, как следствие, возможность более эффективно решать задачи представленные в [4, 2].

2 Постановка задачи, терминология

На протяжении всей работы мы будем пользоваться терминологией из [5] и [3]. Нам понадобятся такие понятия, как `псевдо-треугольник` (pseudo-triangle), `пустой псевдо-треугольник` (empty pseudo-triangle), `звездный псевдо-треугольник` (star-shaped pseudo-triangle). В длинном названии `звездный псевдо-треугольник` слово `псевдо` мы будем часто опускать. Для краткости мы будем пользоваться обозначением `iff`, которое следует читать как “тогда и только тогда”. Слово `треугольник` мы будем часто заменять более коротким “$\triangle$”.

На плоскости есть треугольник (прямоугольник) и n точек внутри него. Определение псевдо-прямоугольника и треугольника дадим с помощью следующих изображений:



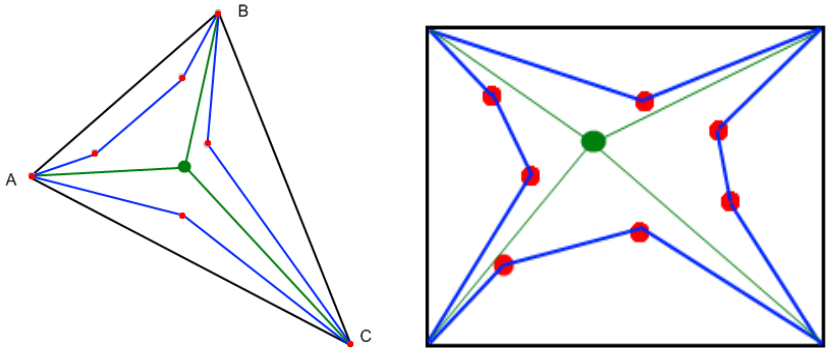
Def: *Псевдо-прямоугольник (треугольник)*
Синия ломаная на картинке (псевдо-прямоугольником называется именно ломаная, т.е. для данного набора “прямоугольник + n точек” можно

построить несколько псевдо-прямоугольников). Ломаная должна обладать следующими свойствами.

1. Простая и замкнутая
2. Содержит все 4 (3) вершины исходного прямоугольника ($\triangle$), которые делят ломаную на 4 (3) части. Часть ломаной, соединяющую вершины A и B, будем обозначать $A - B$ (для любой другой пары смежных вершин — аналогично).
3. Каждая из 4-х (3-х) частей нестрого вогнута внутрь прямоугольника ($\triangle$)

Если так же выполнено свойство "*Многоугольник, ограниченный ломаной, не содержит точек*", псевдо-прямоугольник (треугольник) называется пустым.

По заданному множеству точек интересно определить количество возможных псевдо-прямоугольников (треугольников). Также все эти прямоугольники (треугольники) интересно построить.
Дадим определение частного вида псевдо-прямоугольников (треугольников) — звездных. Опять же прибегнем к помощи изображений.



Def: *Звездный псевдо-прямоугольник ($\triangle$)*
Частный случай псевдо-прямоугольника ($\triangle$). От псевдо-отличается наличием *центра* — точки, лежащий строго внутри многоугольника, из которой видны все 4 (3) вершины исходного прямоугольника ($\triangle$).

Еще один класс задач, рассматриваемый в данной работе, — по заданному множеству точек найти количество пустых звездных прямоугольников (треугольников) и их построить.

3 Результаты

Представим текущие результаты в виде таблицы:

Тип псевдо-многоугольника	Количество сторон	Время построения всех	Время на подсчет количества	Память на подсчет количества	Нижняя оценка на max количество	Верхняя оценка на max количество
Звездный	3	$O(n^2)$	$O(n^2)$	$O(n)$	$n^2/4$	$6n^2 - 4$
Произвольный	3	$O(n^3)$	$O(n^2)$	$O(n)$	$(\frac{n}{3} - 1)^3$	n^3
Звездный	4	$O(n^2)$	$O(n^2)$	$O(n^2)$	$n^2/4$	$12n^2 - 4$
Произвольный	4	$O(n^6)$	$O(n^6)$	$O(n^4)$	$(\frac{n}{6} - 1)^6$	$n^6/2$

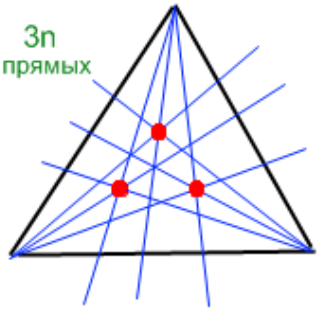
Наиболее значимые новые результаты выделены красным.
Оценки снизу на max количество говорят о том, что **все алгоритмы построения оптимальны**.

4 Звездные треугольники

4.1 Основные утверждения

У нас есть 3 вершины треугольника и n точек внутри. Все $n+3$ точки будем называть *особыми*.

Заметим, что по положению *центра* однозначно можно восстановить все 3 части *звездного псевдо*-$\triangle$. Причем за $O(n)$ (по аналогии с `алгоритмом Грэхема` (Graham)). Чтобы ломаная обладала всеми нужными свойствами, достаточно потребовать, чтобы центр не лежал ни на какой прямой, содержащей хотя бы 2 из $n+3$ особых точек. Рассмотрим корректную *звездную ломаную* и ее *центр*. Центр-точку можно заменить на *центр-шар* маленького радиуса. В шаре найдется новая точка-центр, обладающая нужным нам свойством.

Теперь изучим положение *центра*. Для каждой вершины $\triangle$ рассмотрим n прямых, проходящих через данную вершину и n точек, лежащих внутри. Полученные $3n$ прямых разбивают треугольник на грани. Центр, очевидно, лежит в какой-то из них. Причем, пользуясь рассуждением с *центром-шаром*, можно утверждать, что строго внутри. Осталось увидеть, что если зафиксировать не сам центр, а только грань, в которой он лежит, ломаная все равно восстановится однозначно.	

Отсюда сразу получается алгоритм построения всех различных звездных $\triangle$ за $O(n^3)$, использующий $O(n^2)$ памяти:

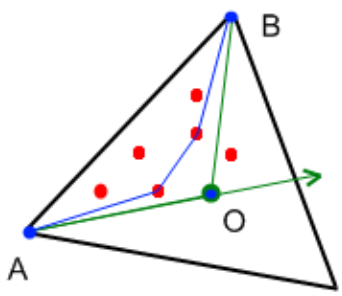
1. Разбить треугольник на грани за $O(n^2)$
2. Для каждого из $O(n^2)$ центров-граней построить за $O(n)$ 3 части ломаной.
3. Получить от построенных конструкций `hash`-значения [1] и за $O(n^2)$ оставить только различные элементы.

[1] Здесь и далее можно применить любую `hash`-функцию **не порождающую коллизии**, значение которой для n-звенной ломаной можно вычислить за $O(n)$. Такие существуют, см. [1].

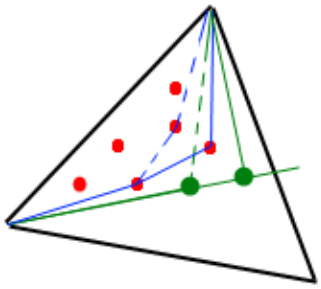
4.2 Алгоритм подсчета количества за $[O(n^2), O(n^2)]$

Чтобы вышеописанный алгоритм улучшить до $O(n^2)$, достаточно вместо того, чтобы каждый раз ломаную строить сначала, получать ее из "предыдущей".

Подсчитаем сперва все $O(n^2)$ ломаных первого из трех типов (т.е. ломаные, соединяющие вершины A и B). Центр получается как пересечение двух лучей. Один луч зафиксируем. Второй будем двигать. Ломаная целиком лежит в $\triangle AOB$, точка O двигается по лучу (см. рисунок) так, что $\triangle$ расширяется.



Если в $\triangle$ появляется новая точка (на картинке изображена красным), из ломаной удаляются несколько вершин достаточно близких к B и добавляется новая. Чтобы в сумме такие операции делались за $O(n)$, будем хранить ломаную стеком.



Для того чтобы добавлять точки в правильном порядке, нужно предварительно отсортировать их относительно каждой из трех вершин A, B, C за воемя $O(n \log n)$.

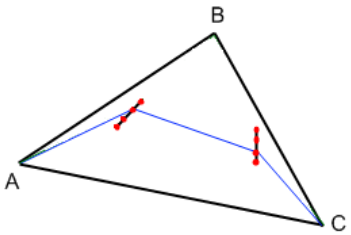
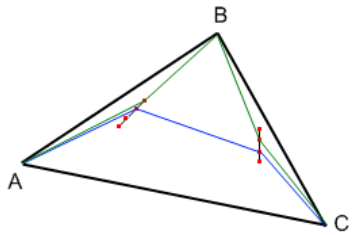
По ходу можно пересчитывать `hash`-значения ломаных. Соответственно, мы научились за $O(n^2)$ получать для каждого центра-грани `hash`-значение ломаной, связывающей A и B. По аналогии найдем два других `hash`-а. Теперь с помощью hash-таблицы за $O(n^2)$ оставим только различные звездные $\triangle$. Мы получили количество звездных треугольников и возможность их восстановить (для каждой ломаной можно хранить кроме `hash`-а $O(1)$ информации — как она получилась из "предыдущей"). Использовали мы $O(n^2)$ времени и $O(n^2)$ памяти.

4.3 Оценка max-количества сверху и снизу

Теорема: *Число звездных треугольников не превосходит* $6n^2 - 4$
Наши $3n$ прямых порождают планарный граф. Число вершин в этом графе не превосходит $3n^2$ (равенство достигается, если точки находятся

в общем положении). Теперь, используя теорему Эйлера $v + f = e + 2$ и неравенство $3f \leqslant 2e$ (f = faces, e = edges, v = vertices), получаем, что $f \leqslant 6n^2 - 4$.

Example: *Число звездных треугольников может быть* $(\frac{n}{2})^2$

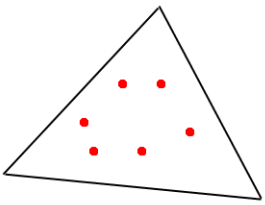
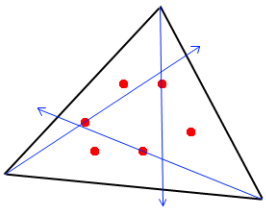
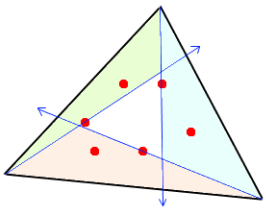
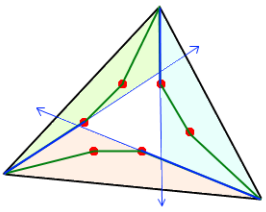
Есть две группы по $n/2$ точек. Каждая пара точек порождает ломаную AC. Для каждой AC строим BC и BA.		

5 Псевдо-треугольники

5.1 Оценка max-количества сверху и снизу. Алгоритм за $O(n^4)$.

Рассуждения, приведенные в этом разделе, во многом повторяют описанные ранее в работах [5] и [3].

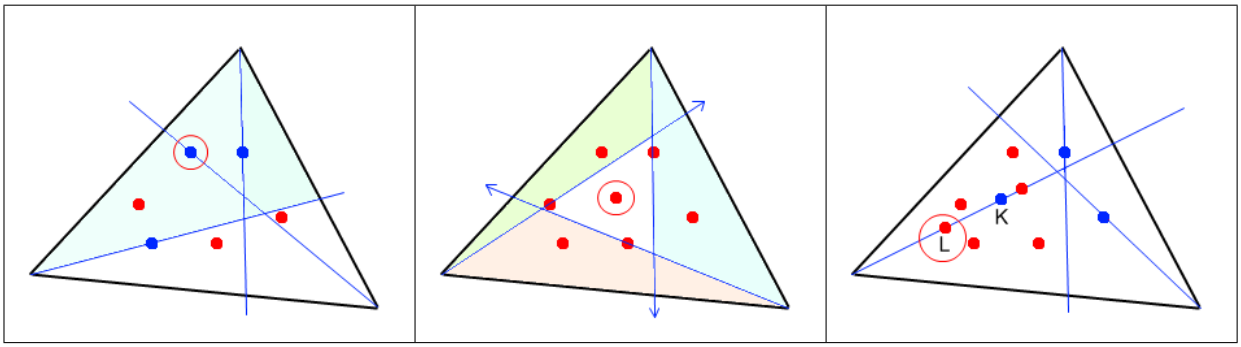
Утверждение 1: *Псевдо-*$\triangle ABC$ *состоит из трех ломаных, соединяющих* A *и* B, B *и* C, C *и* A *соответственно (порядок вершин первая-последняя важен). Если в каждой из этих трех ломаных зафиксировать первое звено, псевдо-*$\triangle$ *восстанавливается однозначно.*
Докажем *Утверждение1*. Для начала поясним, откуда такое предположение берется:

			
Набор точек (1)	Лучи (2)	Области (3)	Ломаные (4)

Как видно на 4-м изображении, из каждой вершины исходят 2 луча. Из двух лучей мы фиксируем левый. Зная лучи (рис.2), получаем области, в

которых должны лежать ломаные (рис.3) и собственно ломаные (рис.4)

Теперь рассмотрим случаи, когда по 3-м левым лучам получаются некорректные ломаные:



Пояснения к рисункам:

1. Точка, через которую хотим провести луч, лежит в запрещенной области.
2. Запрещенный треугольник не пуст.
3. На луче до первой точки лежит еще одна.

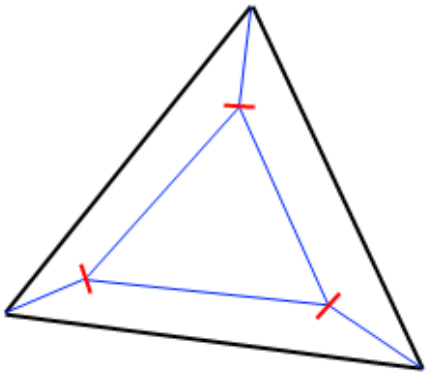
Собственно доказательство:

1. У любого псевдо-$\triangle$ в каждой из 3-х ломаных есть первое звено. Зафиксируем его. Докажем *Утверждение 1.*
2. Рассмотрим одну ломаную. Ее можно замкнуть, добавив ребро исходного $\triangle$. Полученная замкнутая ломаная является границей многоугольника. Все точки, лежащие на границе и внутри этого многоугольника, содержатся в $\cap$ двух полуплоскостей (см. рис. выше). Эту область назовем допустимой для ломаной.
3. Допустимые области не пересекаются (т.к. в набор из 4-х полуплоскостей, определяющий эти две области, обязательно входят две различные полуплоскости, ограниченные одной прямой).
4. Каждая точка лежит в одной из областей $\Rightarrow$ для каждой ломаной однозначно определено множество точек, которое ломаная содержит, или огибает. Зная множество точек, получаем ломаную `алгоритмом Грэхема`.

Утверждение 2: *Количество псевдо-$\triangle$ = количеству троек точек, которые не порождают ни одной из трех проиллюстрированных выше запрещенных ситуаций.*
Следствие: *Получили алгоритм, работающий за $O(n^4)$.*
Алгоритм:

1. Перебираем тройки точек
2. Для каждой за $O(n)$ проверяем, что:
   (a) $\triangle$ не лежащий ни в одной из допустимых областей пуст.
   (b) Все 3 "первых" точки лежат в своих допустимых областях.
   (c) Что на том же луче не лежит ни какая точка до выбранной.

Example: *Число псевдо-$\triangle$ может быть* $(\frac{n}{3}-1)^3$

Красные отрезки означают множества из $n/3$ точек каждое. Из каждой из трех вершин треугольника два луча выходят в две соседние точки ближайшего отрезка, т.е. $\frac{n}{3}-1$ способами.	

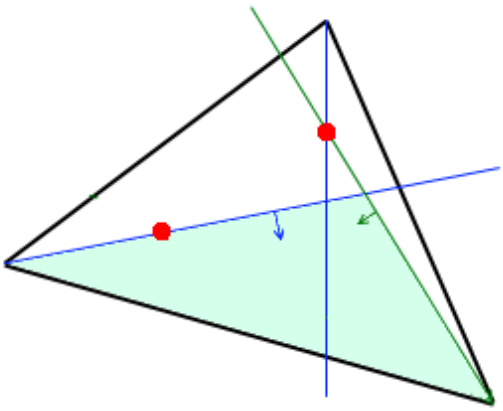
5.2 Количества псевдо-$\triangle$ за $[O(n^2), O(n)]$.

Алгоритм за $O(n^3)$:
При прочтении алгоритма рекомендуем пользоваться рисунками (см. ниже).

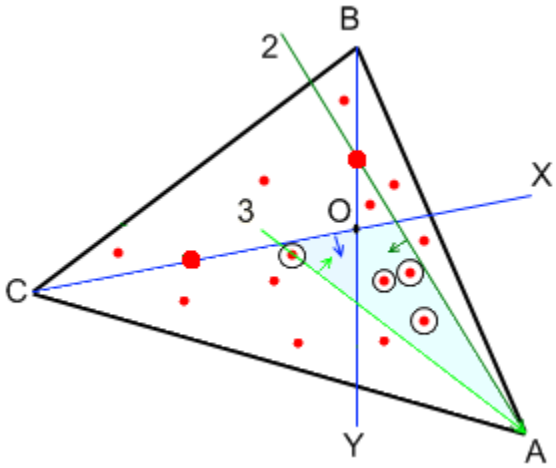
1. Научимся делать следующее: при фиксированных двух лучах говорить, сколько способов выбрать третий так, чтобы получался корректный псевдо-$\triangle$, т.е. не было ни одной их 3-х запрещенных ситуаций. Чтобы выбрать луч, нужно выбрать точку, поэтому будем отвечать на запрос "количество хороших точек".
2. Нам уже известна одна ломаная (заключенная между двумя лучами). При желании мы все за те же $O(n^2)$ можем эту ломаную пересчитывать (тот же метод, что и для звездных $\triangle$).
3. С третьей запрещенной ситуацией побороться просто: для каждой из трех вершин будем перебирать только "хорошие" точки.
4. Первая запрещенная ситуация проверяется для двух уже фиксированных лучей за $O(1)$, а на третий получается ограничение вида: точка, задающая третий луч, должна лежать в пересечении двух полуплоскостей.
5. Вторая ситуация (запрещенный треугольник) дает еще одну полуплоскость, в которой должна лежать точка, задающая третий луч.

6. Полученные 3 полуплоскости дают в $\cap$ $\triangle$ с вершиной в точке A (см. рисунок). Сколько точек лежат в этом $\triangle$ можно посчитать в лоб за $O(n)$.
7. Можно за $O(1)$ (аморатизированное время). Будем эту величину пересчитывать, сдвигая один из двух ранее фиксированных лучей.

Далее следует иллюстрация работы алгоритма и детальное изложение пункта 7.



Первые две полуплоскости (1)



Полная картина мира (2)

Сверху на 2-й картинке отмечены все допустимые точки (обведены окружностями). Это ровно те точки, которые лежат в пересечении 3-х полуплоскостей. Третью полуплоскость порождает самая правая по углу относительно A точка в $\triangle COY$.

Введем новое обозначение: (AB, CD, EF) — $\triangle$, ограниченный соответствующими 3-мя прямыми.

Алгоритм ответа на запрос:

1. Если $\triangle COY$ пуст, ответ = количество точек в углу $(AC, A2, CX)$.
2. Иначе появился луч 3 и ответ = количество точек в углу $(A3, A2, CX)$.

Чтобы не разбирать отдельно эти 2 случая, скажем, что, если $\triangle COY$ пуст, $[A3) = [AC)$.

Как быстро найти луч $[A3)$**?**
Массив P_A — все точки, отсортированные по углу относительно A. Естественно, сортировать по углу мы будем заранее за $O(n \log n)$ и только один раз.

Аналогично P_B и P_C — массивы точек, упорядоченных относительно B и C. При фиксированном $[CX)$ луч $[A3)$ можно хранить и быстро пересчитывать, если точка Y движется против часовой стрелки.

Как подсчитать число точек в углу?
При фиксированном $[CX)$ сделать за $O(n)$ предподсчет частичных сумм на P_A — для точки, попавшей в ту же полуплоскость, что и A относительно $[CX)$, пишем 1, для не попавшей пишем 0. Суммируем числа на отрезке массива, получаем ответ.

Схема алгоритм (`Algo1`):

1. За $O(n \log n)$ получили массивы P_A, P_B, P_C.
2. Фиксируем $[CX)$
3. Делаем предподсчет за $O(n)$ частичных сумм на P_A для точек, лежащих с той же стороны от $[CX)$, что и точка A.
4. Инициализируем луч $[A3)$ равным $[AC)$.
5. Перебираем точки в P_B в порядке против часовой стрелки.
6. Если очередная точка левее выделенной точке на луче $[CX)$, пробуем ее, как точку Y для луча $[BY)$.
7. Если очередная точка лежит с той же стороны от $[CX)$, что и точка A, заменяем луч $[A3)$ на новый.
8. При фиксированных $[CX), [BY)$ у нас уже есть $[A3)$, и мы можем подсчитаем количество точек в P_A между $[A3)$ и $[A2)$ за $O(1)$. Для этого нужно для точек "3" и Y узнать их позиции в массиве P_A и обратиться к предподсчитанному массиву частичных сумм.

Алгоритм представлен. При фиксированном $[CX)$ мы используем всего $O(n)$ дополнительной памяти. Отсюда вывод: алгоритм работает за $[O(n^2), O(n)]$.

5.3 Количество звездных $\triangle$ за $[O(n^2), O(n)]$

Чтобы получить алгоритм, использующий линейный объем памяти, как минимум нужно отказаться от идеи разбивать $\triangle$ на грани, что мы и сделаем. Сперва изучим задачу:

Фиксируем лучи 1 и 3. Точка Y — точка в $\triangle AO3$, минимизирующая угол ZAY. Мы получили луч 2. Чтобы получить луч 4, соединим точки A и C ломаной (часть границы псевдо-$\triangle$). Утверждается, что псевдо-$\triangle$ получится звездным `iff` луч из вершины B, задающий ломаную $B - A$. идет в любую точку из угла PBO лежащую в той же полуплоскости, что и точка B относительно луча $[AZ)$.

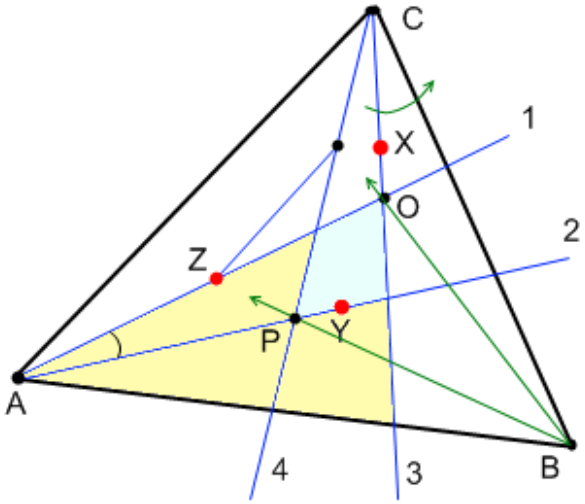


Схема алгоритма:
Алгоритм основан на описанном выше `Algo1`. Здесь мы пользуемся частичными суммами и некоторыми обозначениями взятыми из `Algo1`.

1. Фиксируем $[AZ)$.
2. Перебираем $[CX)$. Точка X перемещается против часовой стрелки.
3. Ломаная AC пересчитывается за суммарное время $O(n)$ для каждого $[AZ)$ (`метод Грэхема`).
4. Точка Y пересчитывается так: луч $[CX)$ повернулся, в $\triangle AO3$ появились новые точки. Выберем из старой точки и новых оптимум. Для каждого $[AZ)$ эти действия будут выполнены за суммарное время $O(n)$.
5. С помощью частичных сумм за $O(1)$ узнаем число точек внутри области OBY.

6 Звездные прямоугольники

Здесь используются те же методы, что и для звездных псевдо-△. Все утверждения и доказательства аналогичны. Приведен только общий план доказательств.

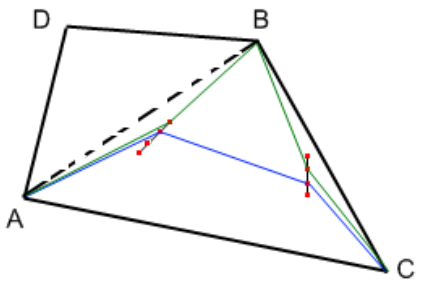
6.1 Алгоритм за $O(n^2)$

Теорема: *Количество звездных прямоугольников* $O(n^2)$
Проведем через каждую пару вершина-точка прямую. Эти $4n$ прямых разбивают исходный прямоугольник на $O(n^2)$ частей. Центр принадлежит одной из граней. По грани, содержащей центр, можно однозначно восстановить ломаную.
Следствие: *Получен алгоритм, за* $O(n^2)$ *вычисляющий для каждого центра* `hash`*-значения трех частей ломаной.*

6.2 Оценка max-количества сверху и снизу

Теорема: *Число звездных прямоугольников не превосходит* $12n^2 - 4$
Наши $4n$ прямых порождают планарный граф. Число вершин в этом графе не превосходит $6n^2$ $(6 = 4{*}3/2)$, значит, число граней не превосходит $12n^2 - 4$. Звездный прямоугольник задается центром, центр задается гранью.

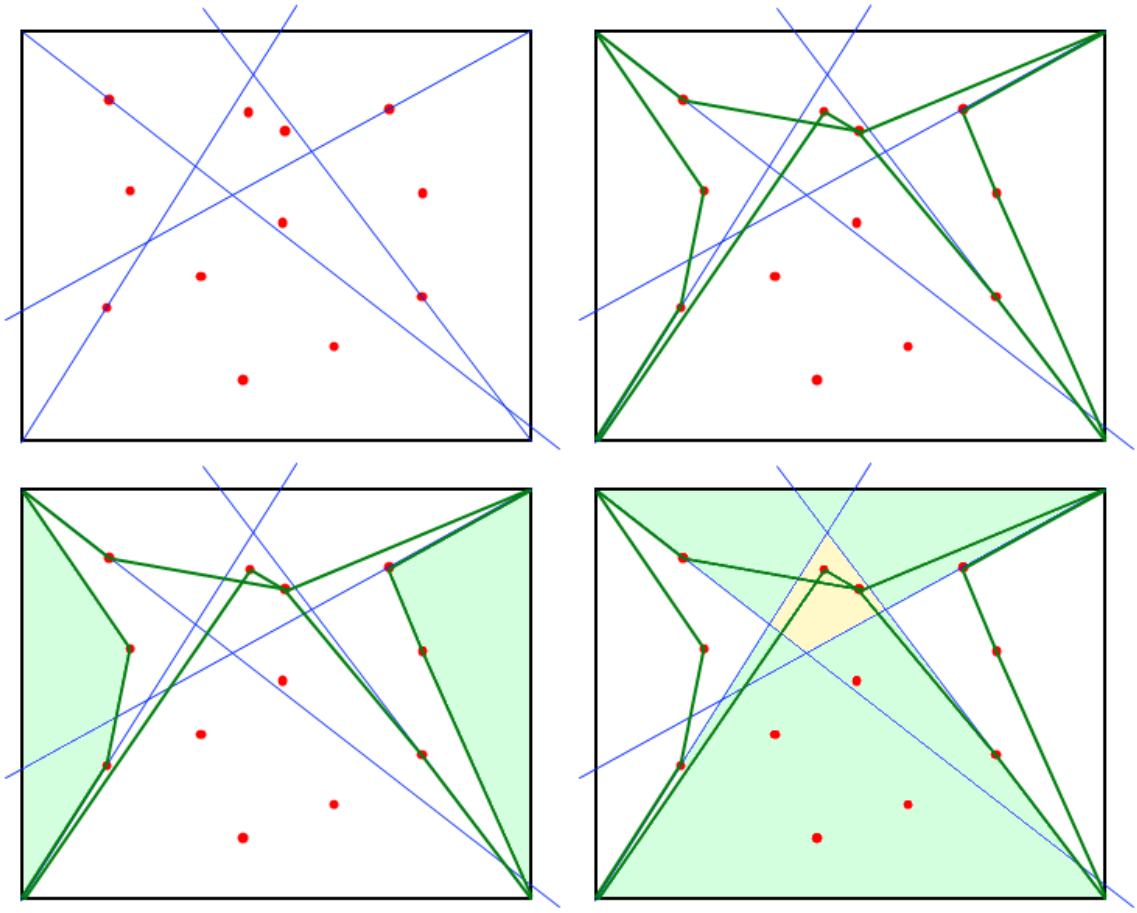
Example:
Число звездных прямоугольников может быть $(\frac{n}{2})^2$
Модифицируем пример для треугольника, добавляя в него новую вершину.



7 Псевдо-прямоугольники

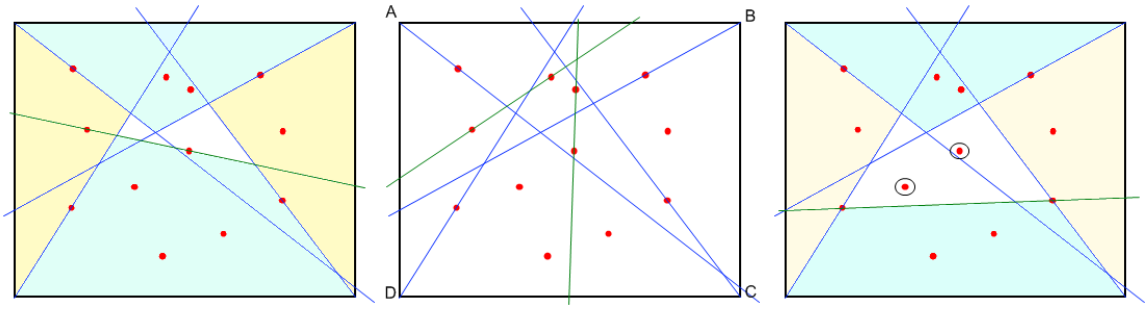
7.1 Отличие от случая △.

В начале попробуем применить тот же способ, что мы использовали для △ — фиксируем 4 луча (для каждой вершины прямоугольника фиксируем левый). Получается следующее:



Допустимая область для участка ломаной, соединяющего две вершины прямоугольника, совпадает с пересечением двух полуплоскостей. В отличие от случая $\triangle$, **допустимые области могут пересекаться**. Поэтому, если мы построим ломаную так же как строили ее для $\triangle$ (используя все особые точки, лежащие внутри или на границе допустимой области) ломаная может иметь самопересечения.

Основная идея, позволяющая побороться с самопересечением: зафиксируем прямую, которая будет разделять два противоположных участка ломаной. Такую прямую всегда можно провести через какие-то две из $N+4$ особых точек.



На первом рисунке проиллюстрирован удачный выбор разделяющей прямой. Чтобы запретить ситуации, изображенные на втором рисунке, введем определение:

Def: *Прямая, разделяющая верхнюю и нижнюю части ломаной* Любая прямая, такая что пары вершин (A и B) и (C и D) лежат в разных полуплоскостях. Аналогично определяется *прямая, разделяющая левую и правую части ломаной.*

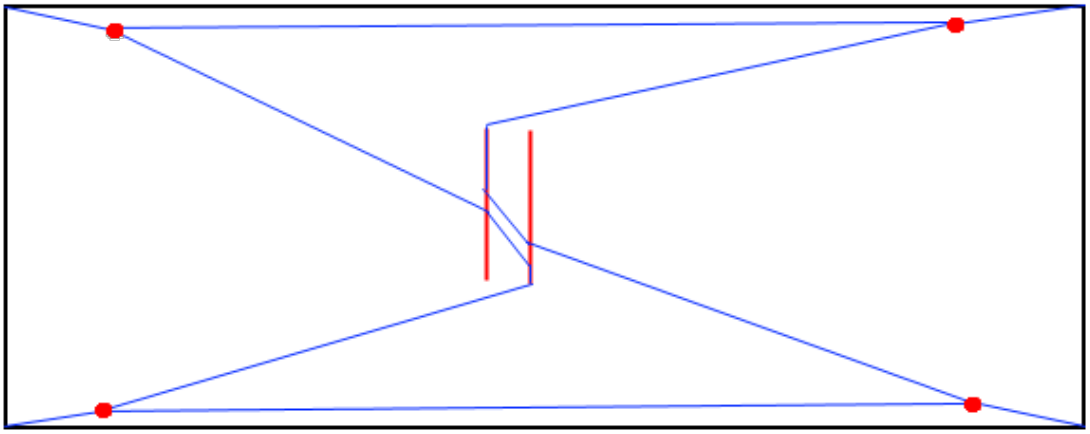
На третьем рисунке показано, что некоторые разделяющие прямые (в комбинации с 4 лучами) порождают **непустой** псевдо-прямоугольник.

7.2 Оценка сверху и снизу на максимальное количество

Теорема: *Количество псевдо-прямоугольников не превосходит* $n^6/2$

1. Рассмотрим ответ. Противоположные части ломанной выпуклы и не пересекаются (даже по вершине), а значит, их можно разделить прямой, проходящей через некоторые 2 точки множества.
2. Так за $O(n^4)$ можно перебрать прямые, разделяющие левую и правую части, а также верхнюю и нижнюю части.
3. Чтобы получить оценку $O(n^8)$, помимо разделяющих прямых зафиксируем 4 луча, направляющих ломаные.
4. По зафиксированным нами объектам ломаная
   (а) восстанавливается однозначно и
   (б) не имеет самопересечений.
5. Чтобы теперь получить оценку $O(n^6)$, заметим, что обе пары противоположных допустимых областей пересекаться не могут. Поэтому достаточно фиксировать только одну прямую. Т.е. после того, как мы зафиксировали 4 луча, остается всего 2 случая:
   (1) ломаная уже не имеет самопересечений или
   (2) одна из двух пар противоположных участков имеет пересечение.
   Фиксируем **одну** разделяющую прямую.
6. Итого: n^4 на выбор 4-х направляющих лучей и $n^2/2$ на выбор (если нужно) разделяющей прямой.

Example: *Число псевдо-прямоугольников может быть* $(\frac{n}{6}-1)^6$



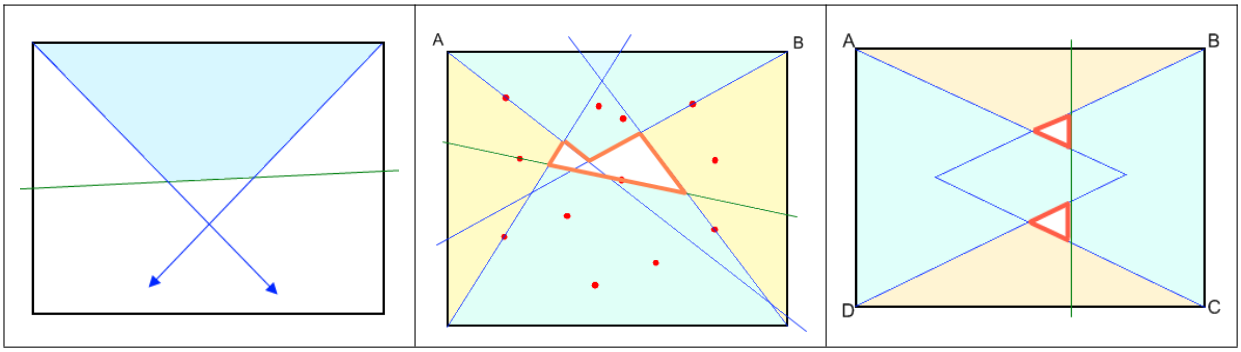
Пояснения к примеру: Красный круг = скопление $n/6$ точек. Красный отрезок = равномерно расположенные $n/6$ точек. В каждом круге и на каждом отрезке нужно выбрать, какие точки принажлежат какой области. В каждом из 6 случаев не менее $n/6$ вариантов.

7.3 Алгоритм построения всех за $[O(n^6), O(n^4)]$

Доказательство предыдущей теоремы уже дает алгоритм, работающий за $O(n^7)$:

1. Перебираем 4 луча за $O(n^4)$.
2. Строим для каждых двух смежных лучей соответствующий участок ломаной за $O(n)$.
3. Если части ломаной не пересекаются (определяем это за $O(n)$), проверяем за $O(n)$, получился ли псевдо-прямоугольник пустым.
4. Если некоторые 2 противоположные части ломаной пересекаются, то другая пара проивоположных фрагментов не имеет общих точек, определяем, какие части пересекаются.
5. За $O(n^2)$ перебираем варианты разделяющей прямой (она обязательно содержит какие-то 2 из $n+4$ особых точек).
6. Строим новые ломаные за $O(n)$ и проверяем, (опять же за $O(n)$), что получился ли новый псевдо-прямоугольник пустым.
7. Некоторые ответы мы могли получить дважды (две разных разделяющих прямых иногда порождают одинаковые ответы). Боремся с этим `hash`-ами `hash`-таблицей, за $O(n^6)$ оставляем только различные объекты.
8. $O(n^4) \times O(n^2) \times O(n) = O(n^7)$

Алгоритм за $O(n^6)$ времени и $O(n^4)$ памяти:
Проблема сейчас в том, что на каждом шаге перебора, работающего за $O(n^6)$, нам нужно построить некоторые объекты за $O(n)$.
Постараемся все предподсчитать заранее:



1. Ломаную можно построить, зная только 2 луча и разделяющую прямую. Этот предподсчет занимает $O(n)$ времени. Одновременно с этим можно вычислить ее `hash`-значение.
2. Область, которая должна быть пуста, определяется также 2-мя лучами и разделяющей прямой. Проверку на пустоту можно выполнить за $O(n)$.
3. Эта область может быть как связной так и не связной (см. рисунки выше).

Итого: Используя $O(n^4)$ памяти и $O(n^5)$ времени, сгенерировали и сохранили всю полезную инофрмацию — собственно ломаные, `hash`-и, флаг пустоты запрещенных областей. После этого за $O(n^6)$ перебираем результирующие объекты и для каждого из них за $O(1)$, используя уже имеющуюся информацию, проверяем корректен ли он и не встречался ли раньше в `hash`-таблице.

Улучшаем оценку на память: Сейчас мы используем $O(n^6)$ памяти на `hash`-таблицу для всех различных псевдо-прямоугольников. И $O(n^4)$ на все предподсчеты. Заметим, что различные 4-ки лучей задают различные псевдо-прямоугольники. Значит, совпадения могут встретиться только среди $O(n^2)$ псевдо-прямоугольников, задаваемых одной и той же четверкой лучей. Вывод:
создаем внутреннюю *hash*-таблицу, в которой храним $O(n^2)$ объектов.

КОНЕЦ

8 Заключение

Сформулируем основные результаты, связанные с прямоугольниками:

1. В данной работе впервые детально описаны объекты `пустой псевдо-прямоугольник` и `пустой звездный псевдо-прямоугольник`.
2. Предложен алгоритм построения всех псевдо-прямоугольников за $O(n^6)$ и получена оценка $\Theta(n^6)$ на их количество в худшем случае.
3. Предложен алгоритм построения всех звездных псевдо-прямоугольников за $O(n^2)$ и получена оценка $\Theta(n^2)$ на их количество в худшем случае.
4. Количественные оценки позволяют заключить, что эти два алгоритма являются оптимальными.
5. Оценка памяти $O(n^4)$ для алгоритма за $O(n^6)$ дает маленькую константу в оценке времени его выполнения на персональном компьютере.

Сформулируем основные результаты, связанные с треугольниками:

1. В [5] и [3] уже были предложены методы разбиения $\triangle$ на грани и фиксации трех лучей, направляющих частей ломаной. С помощью этих методов были даны оценки $O(n^2)$ и $O(n^3)$ на количество звездных псевдо-$\triangle$ и обычных псевдо-$\triangle$ с пустой внутренностью.
2. Алгоритм подсчета количества псевдо-$\triangle$ за $[O(n^2), O(n)]$ является новым и удобным для практической реализации.
3. В [3] рассмотренны задачи минимизации и максимизации периметра и площади многоугольников, ограниченных ломаной псевдо-$\triangle$ (всего 4 задачи). В нашей работе эти задачи не рассмотрены. Тем не менее, стоит заметить, что, умея строить, а следовательно, и перебирать, все псевдо-$\triangle$ за $O(n^3)$, можно получить решения всех 4-х задач за $O(n^3)$.

Последнее, что важно заметить: все результаты, полученные нами для прямоугольника, легко обобщаются на случай выпуклого четырехугольника.

Список литературы

[1] *Алгоритмы. Построение и анализ. Издание 2-е* Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн, 2007

[2] G. Rote, F. Santos, and I. Streinu. *Pseudo-triangulations-a survey.* In J. E. Goodman, J. Pach, and R. Pollack, editors, Surveys on Discrete and Computational Geometry-Twenty Years Later, volume 453 of Contemporary Mathematics. American Mathematical Society, 2008.

[3] Hee-Kap Ahn, Sang Won Bae, Iris Reinbacher *Optimal Empty Pseudo-Triangles in a Point Set* CCCG 2009, Vancouver, BC, August 17-19, 2009

[4] M. Pocchiola and G. Vegter. *Topologically sweeping visibility complexes via pseudo-triangulations.* Discrete Comput. Geom. Dec. 1996.

[5] Marc van Kreveld, Bettina Speckmann *On the Number of Empty Pseudo-Triangles in Point Sets* CCCG 2007, Ottawa, Ontario, August 20–22, 2007