Лабораторная работа №1 по курсу "Алгоритмы вычислительной геометрии"

Выпуклые оболочки и геометрический поиск

Самохина Алина, 904а

16 ноября 2019 г.

1 Постановка задачи

Дано: Файл исходных данных: два облака точек на евклидовой плоскости.
Требуется: Реализовать программу, в результате работы которой выполняется:

1. Построение выпуклых оболочек обоих множеств
2. Отбор точек обоих множеств, лежащих в пересечении их выпуклых оболочек
3. Подсчёт количества точек, лежащих в пересечении выпуклых оболочек
4. Подсчёт количества точек каждого облака, не попавших в пересечение выпуклых оболочек
5. Визуализация выпуклых оболочек и точек попавших или не попавших в их пересечение

1.1 Описание данных

Исходные данные задаются в текстовом файле.
В первой строке файла указано число точек n. Далее, в каждой строке записаны координаты точек x y и номер облака (0 или 1), разделенные пробелом.
Ограничения на входные данные:

1. Координаты точек – целые числа
2. Максимальное число точек $n = 10^3$

2 Алгоритм решения

2.1 Описание алгоритма

В рамках данной задачи рассматривается несколько подзадач вычислительной геометрии:

1. Построение выпуклой оболочки множества
2. Построение пересечения двух выпуклых оболочек
3. Определение принадлежнсти точки выпукому многоугольнику

Рассмотрим алгоритмы решения и их сложность для каждой из указанных подзадач

2.1.1 Построение выпуклой оболочки множества. Алгоритм Грэхема

1. Пусть p_0 — точка из множества $\mathcal{Q}$ с минимальной координатой y или самая левая из таких точек при наличии совпадений.
   Поиск минимальной точки - $O(n)$.
2. Пусть $p_1, p_2, \ldots, p_m$ — остальные точки множества $\mathcal{Q}$, отсортированные в порядке возрастания полярного угла, измеряемого против часовой стрелки относительно точки p_0 (если полярные углы нескольких точек совпадают, то сортировка происходит по расстоянию до точки p_0).
   Сортировка массива (в данной реализации использовалась быстрая сортировка) - $O(nlogn)$

3. Добавляем p_0 в S, где S - стек.
   $O(1)$
4. Добавляем p_1 в S,
   $O(1)$
5. Цикл. Для точек p_i, $i = 2 \dots m$:
   - Пока угол между NextToTop(S) (предпоследняя точка в стеке),Top(S) (последняя точка в стеке) и p_i, образуют не левый поворот - удаляем последнюю точку в стеке.
   - Добавляем в стек p_i

   Проход по циклу - O(n)
6. Возвращаем S

Полученная асимптотика алгоритма - $O(nlogn)$.

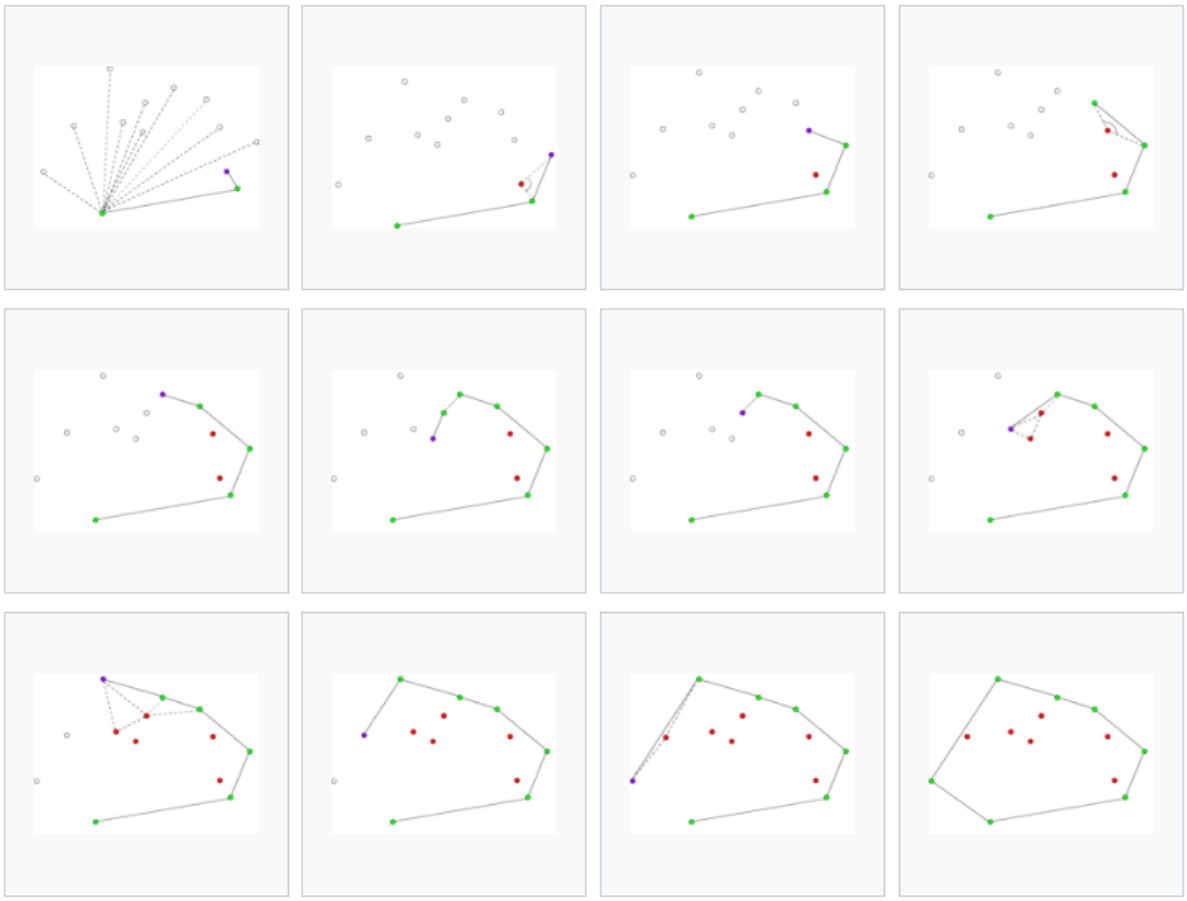

Рис. 1: Пример работы алгоритма Грэхема

2.1.2 Построение пересечения выпуклых оболочек. Алгоритм О'Рурка

Основная идея алгоритма О'Рурка состоит в следующем:

1. Пусть, даны два выпуклых многоугольника P и Q с L и M вершинами соответственно, заданные с помощью списков своих вершин, упорядоченных в порядке обхода против часовой стрелки ($(p_1, p_2, \dots p_L$ и $q_1, q_2, q_M)$).

2. Для каждого многоугольника объявим текущие вершины p_i и q_i, этими точками заканчиваются рассматриваемые рёбра. Идея заключается в том, чтобы не двигаться дальше по тому многоугольнику, текущее ребро которого еще может содержать необнаруженную точку пересечения.

3. Существует четыре возможных варианта взаимного расположения рёбер. Оставшиеся варианты совпада.т с риведёнными с точностью до перестановки p и q. В одном из этих вариантов (рис. 2, а) мы продвинемся по P, так как текущее ребро из Q может содержать необнаруженную точку пересечения; по тем же причинам в случае (рис.2, б) мы продвинемся по границе Q. Если все пересечения на текущем ребре из P уже обнаружены, в то время как текущее ребро из Q все еще может содержать необнаруженное пересечение, мы продвигаемся вдоль P (рис. 2, в); кроме этого, здесь мы должны вычислить точку пересечения ребер. В последнем случае (рис. 2, г) выбор произволен и можно выбрать, например, Q.

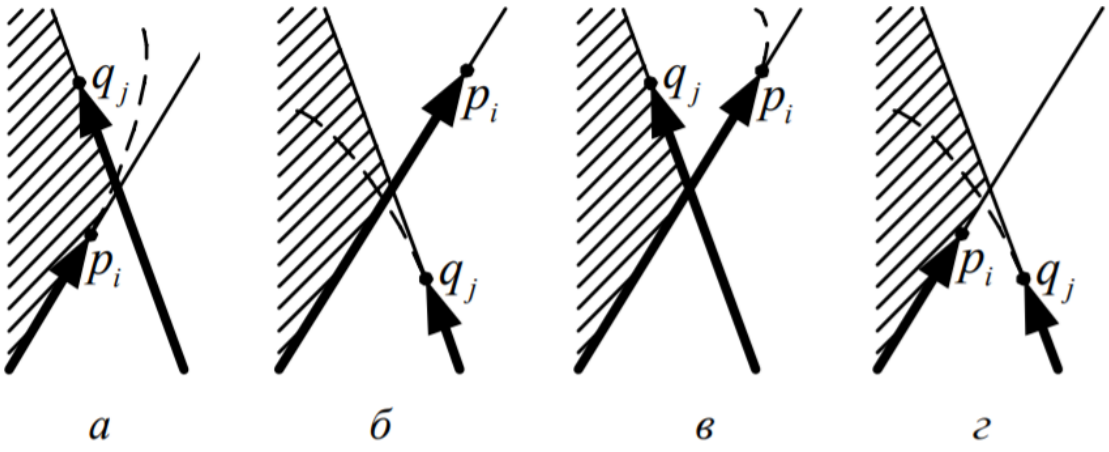

Рис. 2: Варианты взаимного расположения рёбер для алгоритма О'Рурка

Сам алгоритм выглядит следующим образом: Пока не пройдёт максимальное количество итераций (доказывается, что оно не больше 2*(n + m). Оставим подробности в книге О'Рурка по вычислительной геометрии), выполняем следующие инструкции:

1. Рассматриваем текущие рёбра первого и второго многоугольников (начиная, для простоты, с ребер, заканчивающихся в нулевой вершине)

2. Если ребра пересекаются — добавляем точку в список вершин многоугольника пересечения, если её там ещё нет. Если же это первая точка из нашего списка - завершаем выполнение алгоритма.

3. Продвигаемся по многоугольникам на одно ребро в зависимости от результата функции "Движение":

   (a) Определяем взаимное расположение текущих рёбер

   (b) Перемещаемся по одному из многоугольников на одно ребро вперёд, если текущее ребро направлено на ребро другого многоугольника (правила из описания выше и рисунка 2.

   (c) Записать в пересечение многоугольников вершину текущего ребра, если ребро находится "внутри"

Если на выходе получено пустое множество - проверить не лежит ли один многоугольник внутри другого. Это можно сделать, взяв одну из точек многоугольника, и проверив, лежит ли она внутри второго с помощью следующего алгоритма.

2.1.3 Определение принадлежности точки многоугольнику пересечения

Алгоритм:

1. Для каждой интересующей нас точки:

   - для каждого ребра многоугольника:

     (a) проверить лежит ли точка левее если нет:

         i. точка снаружи

         ii. прервать цикл.

3 Программная реализация решения

3.1 Описание

Указанные выше алгоритмы были реализованы на языке программирования Python. Создан модуль utils.py, содержащий все функции, используемые для реализации алгоритмов: сами алгоритмы, вспомогательные функции (такие, как сортировка, определение пересечения отрезков, ввод данных и т.д.). Также создан модуль main.py, содержащий основную последовательность исполнения алгоритмов, вывод данных и их визуализацию.

3.2 Эксперименты

Были проведены эксперименты по измерениию времени исполнения алгоритмов в зависимости от объёма входных данных

кол-во точек, шт.	Алгоритм Грэхема, с.	Алгоритм Рурка, с.	Точки в пересечении, с.
10	0.001	0.001	0.001
100	0.004	0.001	0.023
250	0.009	0.001	0.048
500	0.027	0.002	0.052
750	0.029	0.003	0.069
1000	0.031	0.003	0.083

3.3 Инструкция по работе с программой

1. Запустить main.exe
2. Для того, чтобы запустить программу с автоматически сгенерированными данными для демонстрации введите 0
3. Если есть файл данных - нажмите 1 и вставьте путь файла данных
4. Появляется окно с визуализацией результатов и файл output.txt в папке с исполняемым файлом, который содержит координаты обеих выпуклых оболочек и точек, находящихся в их пересечении (включая точки, лежащие на границах)

4 Выводы

В данной работе бвли реализованы алгоритмы, позволяющие построить пересечение выпуклых оболочек двух облаков точек за $O(nlogn)$. А затем, наивным алгоритмом, найти точки, принадлежащие этому пересечению за $O(n^2)$. Алгоритмы Грэхема и О'Рурка являются наиболее наглядными алгоритмами с лучшей скоростью среди алгоритмов, решающих похожие задачи. В асимптотике сложности данных алгоритмов удалось убедиться на практике.