Добрый день/вечер!

1 Слайд

Нашей задачей являлась реализация двух алгоритмов построения выпуклой оболочки с помощью языка программирования C++ и системы компьютерной алгебры Wolfram Mathematica​. Задача была решена двумя методами:

1. Методом перебора, т.е. последовательно находя такие прямые, проходящие через пары точек, что все остальные точки лежат по одну сторону от этих прямых;
2. Метод Киркпатрика – Зайделя

2 слайд

Сначала дадим несколько базовых определений, чтобы иметь представление, что нам нужно найти.

Пусть на плоскости задано конечное множество точек A.

Оболочкой этого множества называется любая замкнутая линия H без самопересечений такая, что все точки из A лежат внутри этой кривой.

Множество называется выпуклым, если оно вместе с любыми своими точками содержит весь отрезок, соединяющий эти точки.

Если кривая H является выпуклой, то соответствующая оболочка также называется выпуклой.

Минимальной выпуклой оболочкой (МВО) называется выпуклая оболочка минимальной длины (минимального периметра).

3 слайд

Метод перебора

Этот алгоритм достаточно прост в реализации, потому что самая сложная его часть – определение, по какую сторону точка лежит относительно других двух точек. Однако эффективность данного алгоритма низка и в худшем случае достигает сложности, равной , что делает его использование практически непригодным для решения объемных задач.

Метод перебора заключается в последовательном нахождении таких прямых, проходящие через пары точек, что все остальные точки лежат по одну сторону от этих прямых.

Первую точку возьмем самую крайнюю (например, точку с минимальной X-координатой Plast), которая записывается, как первая точка выпуклой оболочки. Для поиска следующих точек вводится точка Pcandidate (точка-кандидат). С помощью псевдоскалярного (косого) произведения определяем положение остальных точек относительно прямой PlastPcandidate. Если все остальные точки лежат по одну сторону от прямой PlastPcandidate, то кандидат Pcandidate записывается в массив точек выпуклой оболочки; координатам точки Plast присваиваются значения координат точки Pcandidate; затем происходит переход к следующей итерации. В противном случае (если нашлась точка, лежащая по другую сторону от прямой), производится переход к следующей итерации без каких-либо присваиваний (рис. 2)(?). Алгоритм подходит к концу, когда найденная на некотором шаге крайняя точка является точкой, полученной при первой итерации.

4 - 6 слайды

Алгоритм Киркпатрика – Зайделя

Алгоритм основан на известном методе под названием «разделяй и властвуй», когда исходная задача разбивается на подзадачи, затем с помощью рекурсии решаются подзадачи, после чего каждый такой результат объединяется в общее решение – ответ на исходную задачу.

Однако алгоритм Киркпатрика – Зай-деля отличается тем, что эти действия происходят в обратном порядке: сначала необходимо определить, как решения подзадач будут объединяться, а лишь потом решить эти подзадачи. Алгоритм имеет сложность 𝑂(𝑛 log ℎ) (𝑛 – количество исходных точек, ℎ – количество точек выпуклой оболочки), то есть время выполнения зависит как от входных параметров, так и от результата.

4 слайд:

1 картинка – Сначала разбиваем множество точек на два по медиане по координатам икс. Это нужно для достижения алгоритмической сложности .

2 картинка – Делим точки произвольно на пары. Если количество точек нечетно, то оставшуюся точку кладем в список возможных кандидатов.

5 слайд:

1 картинка – Для каждой прямой, проходящей через пару точек, считаем наклон этой прямой. Находим среди всех наклонов медианное значение.

По этой медиане находим точку или несколько точек с максимальным значением , где – то самое медианное значение. Мы получим прямую, которая лежит выше всех точек. Если на этой прямой несколько точек из исходного множества и одна из них левее «медианной прямой», а другая – правее, то мы нашли мост. Он состоит из этой пары точек.

Если же этого не произошло, то, опираясь на леммы, указанные в работе, мы «выкидываем» точки, которые точно не являются точками моста. Рассматриваем таким же образом оставшихся кандидатов, пока не найдем мост. На каждой итерации уйдет около 25% точек.

2 картинка – После получения моста делим множество исходных точек на 2 подмножества, одно из них состоит из точек, которые левее левой точки моста, и самой левой точки моста; второе – из точек, которые правее правой, и правая точка. Далее проводим такие же действия для каждого подмножества.

6 слайд:

Важно заметить, что поиск оболочки в работе Киркпатрика и Зайделя сформулирован только для верхней оболочки. Но, как сами авторы отмечают, абсолютно аналогичные рассуждения можно провести для нижней оболочки.

7 слайд

На слайде представлен пример решения задачи нахождения выпуклой оболочки. Исходные данные: 20 точек. Результат: 9 точек.

8 слайд

На данном слайде представлены различные результаты работы алгоритмов в зависимости от исходных данных. Как видно из рисунков количество точек выпуклой оболочки совпадает для разных алгоритмов. Оба алгоритма отработали верно.

Время алгоритмов метода перебора и метода Киркпатрика – Зайделя, конечно, отличается и совпадает соответственно с их сложностью.

9 слайд

На данном слайде представлена наглядная зависимость времени исполнения обоих методов от количества точек (С++).  
10 - 11 слайды

На слайде представлены результаты работы алгоритмов и наглядный рисунок построения выпуклой оболочки в системе компьютерной алгебры Wolfram Mathematica.

12 слайд

Вывод

В ходе нашей совместной учебно-ознакомительной практики были изучены два метода решения задачи о построении выпуклой оболочки множества точек на плоскости.  
 Под конец перефразируем пословицу: «Семь раз подумай, один раз реализуй алгоритм»