Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского" (ННГУ)

Институт информационных технологий, математики и механики

ОТЧЕТ

«Построение выпуклой оболочки – проход Грэхема»

Выполнил: студент группы 381706-1
Силенко Дмитрий Игоревич
_____ Подпись

Проверил:

_____ Нестеров А. Ю.

Нижний Новгород
2019.

Содержание

1. Введение

Основная цель данной работы – реализовать алгоритм прохода Грэхема, используемый для составления выпуклой оболочки из массива точек на плоскости.

Но для начала необходимо разобраться что же такое выпуклая оболочка.

Выпуклая оболочка множества точек — такое выпуклое множество точек, что все точки фигуры также лежат в нем. Необходимо отметить, что строить мы будем не просто выпуклую оболочку, а минимальную выпуклую оболочку.

Минимальная выпуклая оболочка множества точек — это минимальная по площади выпуклая оболочка.

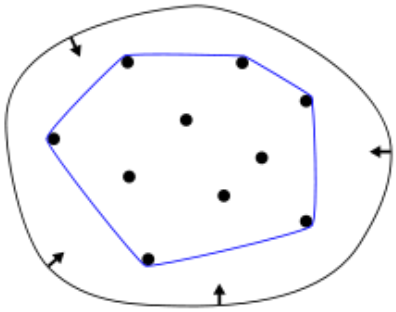

Рисунок 1 Выпуклая оболочка и минимальная выпуклая оболочка

По сути, минимальная выпуклая оболочка это знакомый всем со школы выпуклый многоугольник, состоящий из точек входного множества так, что все остальные точки этого множества лежат внутри него. Поскольку точки в множестве могут быть расположены как угодно, для успешного построения оболочки необходимо не только выбрать начальную вершину, но и отсортировать все остальные относительно нее. Проще всего это делать, оперируя полярными координатами заданных нам точек. Определим линейный порядок, относительно которого в дальнейшем будет производится сортировка. $c_1 \leq c_2$, если либо $\varphi_1 < \varphi_2$, либо $(\varphi_1 = \varphi_2)\&(r_1 \leq r_2)$.

2. Постановка задачи

Для точек a_1, …, a_n, где n≥1, a_i=($a_{i,1}$, $a_{i,2}$) ∈ R^2 при i=1, …, n, указать вершины b_1, …, b_m выпуклой оболочки Conv(a_1, …, a_n) в порядке их встречи при движении по ее границе. Заметим, что в общем случае Conv(a_1, …, a_n) будет многоугольником, а в вырожденных случаях может получиться отрезок или точка. В случае отрезка выходом решающего поставленную задачу алгоритма должны быть две являющиеся его концами точки, а в случае точки - сама эта точка.

Алгоритм построения Conv(a_1, …, a_n) необходимо распараллелить так, чтобы выпуклая оболочка строилась правильно для произвольного числа процессов, выполняющих ее построение.

3. Руководство программиста

3.1. Описание структуры программы

Программа состоит из следующих модулей:

- Модули Convex_Hull_Graham_mpi и Convex_Hull_Graham_mpi_lib. Содержат основные методы для работы с выпуклой оболочкой и множеством точек, в том числе: рандомная генерация заданного количества точек, сортировка точек по их полярным координатам, вычисления определителя и само построение выпуклой оболочки проходом Грэхема
- Модуль main. Содержит набор тестов для проверки работоспособности всех реализованных методов.

3.2. Описание алгоритмов

Построение выпуклой оболочки с помощью прохода Грэхема:

Для решения задачи построения $Conv(a_1, \ldots, a_n)$ мы из точек $a_1, \ldots, a_n$ выберем точки с минимальной первой координатой, среди которых затем найдем точку **c**, имеющую минимальную вторую координату. Таким образом, точка **c** = $lexmin(a_1, \ldots, a_n)$ является лексикографическим минимумом точек $a_1, \ldots, a_n$ и поэтому представляет собой вершину выпуклой оболочки $Conv(a_1, \ldots, a_n)$. Именно с нее мы и начнем обход границы против часовой стрелки, положив $b_1=c$ и осуществив перед этим для удобства промежуточных вычислений параллельный перенос системы координат так, чтобы ее начало совпало с точкой **c**. После такого переноса на точках $a_1, \ldots, a_n$ удается определить такой линейный порядок ($\leq$), что для $c_1, c_2 \in \{a_1-c, \ldots, a_n-c\}$ имеет место $c_1 \leq c_2$, если либо $\det(c_1,c_2)>0$, либо $(\det(c_1,c_2)=0)\&((c_{1,1})^2+(c_{1,2})^2) < (c_{2,1})^2+(c_{2,2})^2)$.

Геометрический смысл такого упорядочения можно проиллюстрировать, если ввести полярную систему координат φ, r ($-\pi<\varphi \leq\pi$, $r\geq0$) с центром в точке **c** и далее положить, что $c_1 \leq c_2$, если (φ_1,r_1) лексикографически не превосходит (φ_2,r_2), где числа φ_i, r_i являются полярными координатами точки c_i, i = 1,2.

Таким образом, $c_1 \leq c_2$, если либо $\varphi_1 < \varphi_2$, либо $(\varphi_1 = \varphi_2)\&(r_1 \leq r_2)$. Как только линейный порядок на элементах (точках) $a_1, \ldots, a_n$ определен, немедленно можем воспользоваться сортировкой, для того чтобы отсортировать эти элементы по не убыванию в соответствии с введенным линейным порядком. После этого мы произведем за время O(n) просмотр отсортированного массива с целью получения итоговых точек $b_1, \ldots, b_m$ исходя из того условия, что точка b_{i+1} располагается строго слева от вектора, идущего из точки b_{i-1} в

точку b_i, что эквивалентно требованию того, чтобы $\det(b_i-b_{i-1},b_{i+1}-b_i)>0$. Рассмотрим этот процесс чуть поподробнее.

Создаем стек и заносим туда две первые точки из нашего множества. Для каждой следующей точки z:

1) Читаем y - верхушку стека и x – предпоследний элемент стека.
2) Если z находится слева от xy ($\det(x-y,z-x)>0$), то добавляем z в стек
3) Если z находится не слева от xy, то удалить y из стека и вернуться на пункт 1) (если в стеке было >= 3 элементов) или удалить y из стека y и добавить туда z (если в стеке было 2 элемента)

Как итог, все элементы, находящиеся в стеке и будут вершинами выпуклой оболочки. Остается лишь выполнить параллельный перенос системы координат обратно в (0;0).

Что касается распараллеливания, то оно происходит и при проведении подготовительных работ, и на моменте начала прохода Грэхема. Каждый процесс берет на себя некоторое количество (= все число точек / количество процессов) точек, высчитывает их полярные координаты, после чего они сливаются в общий массив и сортируются. Далее процессы берут точки из уже отсортированных и на их основе строят свою выпуклую оболочку. Затем отправляют полученный результат на нулевой процесс, который производит формирование уже конечного результата проходя по пришедшим ему точкам и сопоставляя их с текущим результатом.

Быстрая сортировка:

Быстрая сортировка относится к алгоритмам «разделяй и властвуй». Алгоритм состоит из трёх шагов:

1. Выбрать элемент из массива. Назовём его опорным.
2. Разбиение: перераспределение элементов в массиве таким образом, что элементы меньше опорного помещаются перед ним, а больше или равные после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

Для выбора опорного элемента и операции разбиения существуют разные подходы, влияющие на производительность алгоритма. Мы будем брать в качестве опорного элемента середину текущего подмассива.

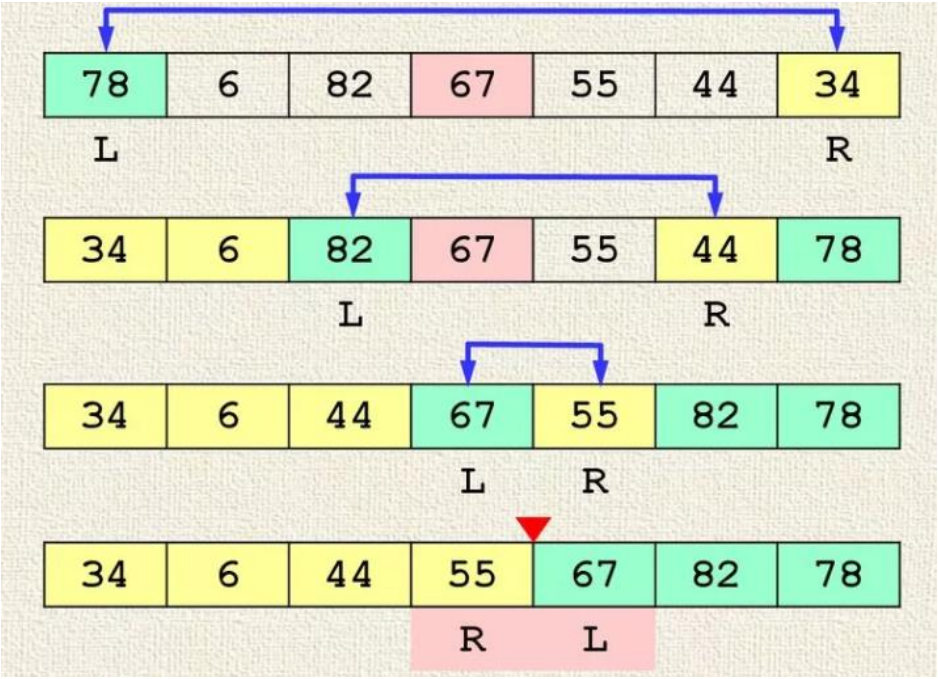

Рисунок 2 Быстрая сортировка, один проход

4. Заключение

Эта лабораторная работа позволила мне лучше разобраться в самом алгоритме построения выпуклой оболочки с математической точки зрения, а также понять, как правильно интерпретировать его в виде кода.

В данной работе мне удалось реализовать алгоритм построения выпуклой оболочки n точек на плоскости с использованием прохода Грэхема. А кроме этого, удалось распараллелить его для успешного построения выпуклой оболочки на любом количестве процессов. Быстрая сортировка используется из-за того, что в среднем она выдает хороший показатель скорости работы (O(log(n)), а так же ее достаточно легко доработать для сравнения в лексикографическом порядке.

5. Литература

1. Груздев Д. В., Таланов В. А. Алгоритмы и структуры данных (лабораторные работы): [https://vk.com/doc51644906_515702336?hash=fc61c591076938632c&dl=0b9843bdc392de49ac], 2004.
2. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение, 1989
3. Википедия: свободная электронная энциклопедия на русском языке: https://ru.wikipedia.org/wiki/Быстрая_сортировка