Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Фундаментальные науки
КАФЕДРА Прикладная математика

ОТЧЕТ ПО ПРАКТИКЕ

Студент Романов Владислав Игоревич
фамилия, имя, отчество

Группа ФН2-32Б

Тип практики: Ознакомительная практика

Название предприятия: Научно-учебный комплекс «Фундаментальные науки» МГТУ им. Н.Э. Баумана

Студент _____ Романов В. И.
подпись, дата *фамилия и.о.*

Руководитель практики _____ Марчевский И. К.
подпись, дата *фамилия и.о.*

Оценка _____

2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Кафедра «Прикладная математика»

З А Д А Н И Е
на прохождение ознакомительной практики

на предприятии Научно-учебный комплекс «Фундаментальные науки» МГТУ им. Н.Э. Баумана

Студент Романов Владислав Игоревич
фамилия, имя, отчество

Во время прохождения ознакомительной практики студент должен

1. Изучить на практике основные возможности языка программирования C++ и систем компьютерной алгебры, закрепить знания и умения, полученные в курсах «Введение в информационные технологии», «Информационные технологии профессиональной деятельности».
2. Изучить способы реализации методов решения задачи построения выпуклой оболочки множества точек на плоскости.
3. Реализовать алгоритм Киркпатрика-Сайделя построения выпуклой оболочки.

Дата выдачи задания «22» сентября 2020 г.

Руководитель практики _____ Марчевский И. К.
подпись, дата *фамилия и.о.*

Студент _____ Романов В. И.
подпись, дата *фамилия и.о.*

Содержание

Задание

Набор точек на плоскости задан парами своих координат. Требуется построить выпуклую оболочку данного множества точек – т.е. выпуклый многоугольник наименьшей площади, содержащий все эти точки. В качестве ответа привести список точек по порядку (по часовой стрелке или против часовой стрелки), задающих многоугольник, являющийся границей выпуклой оболочки.

а) решить задачу «методом перебора», последовательно находя такие прямые, проходящие через пары точек, что все остальные точки лежат по одну сторону от этих прямых;

б) решить задачу эффективно, используя алгоритм Киркпатрика – Сайделя (Kirkpatrick – Seidel algorithm).

Структура исходного файла данных:

n	<< количество точек
x1 y1	<< координаты первой точки
…	
xn yn	<< координаты n-ой точки

Структура файла результата:

q	<< количество точек, задающих многоугольник, являющийся границей выпуклой оболочки
x1 y1	<< координаты первой точки
…	
xq yq	<< координаты q-ой точки

Введение

Основной целью ознакомительной практики 3-го семестра, входящей в учебный план подготовки бакалавров по направлению 01.03.04 – Прикладная математика, является знакомство с особенностями осуществления деятельности в рамках выбранного направления подготовки и получение навыков применения теоретических знаний в практической деятельности.

В ранее пройденном курсе «Введение в специальность» произошло общее знакомство с возможными направлениями деятельности специалистов в области прикладной математики и получен опыт оформления работ (реферата), который полезен при оформлении отчета по практике.

В рамках освоенного курса «Введение в информационные технологии» изучены основные возможности языка программирования C++ и сформированы базовые умения в области программирования на C++. Задачей практики является закрепление соответствующих знаний и умений и овладение навыками разработки программ на языке C++, реализующих заданные алгоритмы. Кроме того, практика предполагает формирование умений работы с системами компьютерной алгебры и уяснение различий в принципах построения алгоритмов решения задач при их реализации на языках программирования высокого уровня (к которым относится язык C++) и на языках функционального программирования (реализуемых системами компьютерной алгебры).

1. Формализация задачи

Сформулируем несколько определений для формализации поставленной задачи в рамках теории графов:

Определение 1.1.1. Множество S на плоскости называется выпуклым, если для любых двух точек $P,Q \in S$ весь отрезок PQ принадлежит S.

Определение 1.1.2. Выпуклой оболочкой CH (S) множества S называется наименьшее выпуклое множество, содержащее S «Наименьшее множество» здесь означает наименьший элемент по отношению к вложению множеств, то есть такое выпуклое множество, содержащее данную фигуру, что оно содержится в любом другом выпуклом множестве, содержащем данную фигуру.

В данной терминологии поставленная задача сводится к задаче о поиске такого подмножества точек из S, что они образуют выпуклый многоугольник, во внутренней области (или на сторонах) которого лежат все точки заданного множества S.

Задача о построении выпуклой оболочки является одной из важнейших задач вычислительной геометрии, она имеет множество приложений, например в распознавании образов, обработке изображений, а также при раскрое и компоновке материала. На данный момент существует множество методов решения данной задачи, различающихся по сложности.

2. 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 $O(n^3)$, что делает его использование практически непригодным для решения объемных задач.

Суть этого метода заключается в поиске точек, образующих такие прямые, что все остальные точки лежат по одну сторону от этой прямой. Для удобства сначала выбирается первая точка с x_{max} или x_{min}, которая записывается как первая точка выпуклой оболочки. Для поиска следующих точек вводятся параметры p_{former} (последняя точка выпуклой оболочки) и p_{latter} (точка-кандидат). Если все остальные точки лежат по одну сторону от прямой $p_{former}p_{latter}$, то кандидат p_{latter} записывается в массив точек выпуклой оболочки; p_{former} присваивается значение p_{latter}; 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.

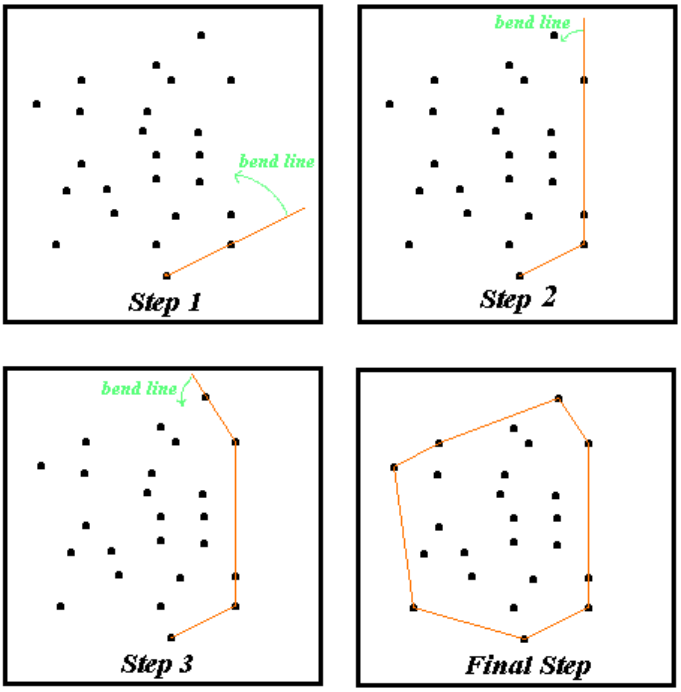

Рисунок 1. Принцип метода перебора.

3. Алгоритм Киркпатрика – Сайделя

3.1. История алгоритма

Алгоритм Киркпатрика – Сайделя – алгоритм, позволяющий построить выпуклую оболочку множества точек на плоскости. В современном виде алгоритм был впервые опубликован в 1986 году американским ученым Дэвидом Киркпатриком и его австрийским коллегой Раймундом Сайделем (статья D.G.Kirpatrick & R.Seidel "The ultimate planar convex hull algorithm" (SIAM Journal on Computing, 15(2) 1986)). Приобрёл известность благодаря своему быстродействию.

3.2. Стандартная реализация

Алгоритм основан на другом известном алгоритме 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–Сайделя 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($nlog(h)$), то есть время выполнения зависит как от входных параметров, так и от результата. Результатом исполнения алгоритма является список точек, образующих выпуклую оболочку.

Определение 3.2.1. "Bridge" или "Мостом" будем называть такую прямую, пересекающую медиану точек по координате X и образованную из двух точек исходного множества, что остальные точки лежат ниже (для верхней оболочки) этой прямой, или выше этой прямой – для нижней оболочки (рис. 2)

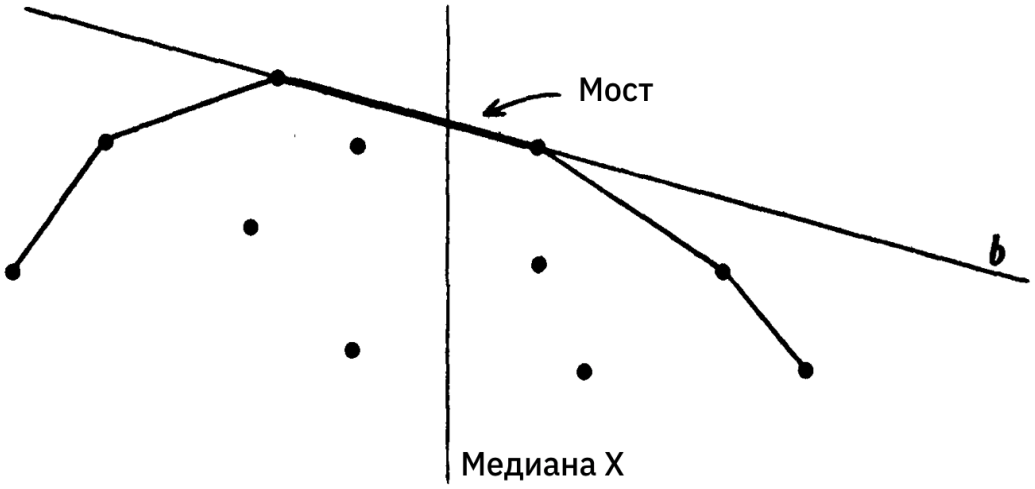

Рисунок 2. Построение "моста".

Сперва необходимо найти точки, с которых алгоритм будет начат, то есть точки с x_{max} и $x_{min} - p_{max}$ и p_{min} соответственно. В том случае, если несколько точек с x_{min} или x_{max} лежат на одной вертикали, то для верхней части оболочки необходимо взять точку с y_{max}, а для нижней – сy_{min} (рис. 3). Далее приведем пример для поиска верхней части оболочки, потому что для нижней части алгоритм применяется по аналогии.

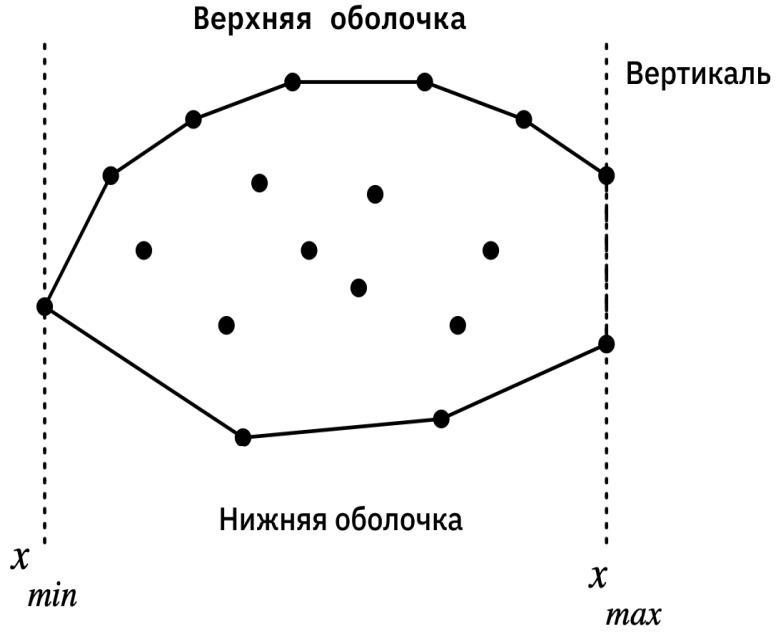

Рисунок 3. Верхняя и нижняя оболочка.

В случае, если $p_{min} = p_{max}$, результатом будет единственная точка – p_{min}. Если точки не равны, то из исходного множества точек рассматриваем такие, что

$$T = \{p_{min}, p_{max}\} \cup \{p \in S | x(p_{min}) < x(p_{min}) < x(p_{max})\}.$$

Далее среди всех X-ых координат точек нового множества необходимо найти медиану. Обозначим ее за x_m – вертикальная прямая, проходящая через эту точку, делит множество T на два примерно одинаковых по мощности подмножества. Затем нужно найти "Мост", пересекающий эту вертикаль, убрать из рассмотрения все точки, лежащие ниже этого моста, а после применить алгоритм рекурсивно к двум подмножествам оставшихся точек слева и справа от вертикали соответственно.

4. Реализация метода перебора

4.1. Особенности реализации на языке C++

Стандартная библиотека C++ предоставляет удобный контейнер для хранения и обработки различных структур данных – std::vector<T>. Для более визуально наглядного представления точек воспользуемся вспомогательной структурой Point (при инициализации получает два значение типа double (координаты X и Y). Для обработки расположения точки относительно прямой введем enum PointPosition с тремя случаями: within (лежит на прямой), oneSide (по одну сторону), otherSide (по другую сторону).

Для заполнения массива исходных точек создадим модуль под названием HandleFiles, в котором опишем два метода: выгрузка из файла – fillVectorWithPointsFromTxt – и загрузка в файл – writeToFile.

4.2 Особенности реализации в системе компьютерной алгебры

Встроенные возможности системы Wolfram Mathematica значительно упрощают решение данной задачи, в особенности при обработке данных. Для работы с файлами использовались такие команды, как SetDirectory (задание директории) и ReadList (считывание в List), для работы с List – команды Delete (удаление), Partition (разбиение данных внутри List), AppendTo и PrependTo (добавление в конец и начало соответственно), Length (определение длины списка), MemberQ (проверка наличия элемента в List) и др., для визуализации результата– Graphics, для замера времени – Timing. Кроме того, использовался программный модуль (Module) для реализации алгоритма с созданием временных переменных, а также потребовалось создать пользовательскую функцию determinePointPosition, необходимую для определения положения точки относительно прямой, проведённой через две другие точки. Метод реализован в виде пользовательской функции IneffectiveConvexHull, принимающей на вход список точек, заданных своими координатами и возвращающей результат работы метода перебора.

5. Реализация алгоритма Киркпатрика – Сайделя

5.1. Особенности реализации на языке C++

Реализация алгоритма представлена функтором KirkpatrickSeidelAlgorithm. Инициализация начинается с формирования массива исходных точек из файла, путь к которому является единственным параметром для конструктора. Для представления пар точек воспользуемся вспомогательной структурой PointPair (при инициализации принимает два параметра типа Point).

Сначала среди исходных точек ищутся точки с x_{max} и x_{min}. Стандарт C++17 дает нам возможность возвращать из функции несколько значений с помощью tuple<T> (кортеж). Именно им можно воспользоваться, чтобы с помощью одной функции получить начальные точки для верхней и нижней оболочек. Далее вызывается функция getPartOfHull для каждой из частей оболочки с этими точками. Потом исходное множество делится на подмножество (в соответствии с описанием стандартной реализации выше) и вызывается рекурсивная функция connect, заполняющая вектор точек выпуклой оболочки.

Внутри функции connect производится вызов другой функции – getBridge.

Для поиска моста необходимо каждый раз делить точки на пары (если их нечетное количество, то первую из них нужно записать в кандидаты, а остальные поделить на пары). Далее для каждой из пар точек высчитываются наклонные, среди которых впоследствии ищется медиана. Потом необходимо разделить пары точек на три типа: small (если их наклонная < медианы наклонных); equal (если равна) и large (если больше). И со всеми вышеперечисленными параметрами вызывается функция continueForUpperBridge или continueForLowerBridge, результатом выполнения которой будет переменная опционального типа, хранящая либо пару точек (точек моста), либо

ничего. Для обработки этой ситуации удобно воспользоваться std::optional. Если из переменной опционального типа можно развернуть значение (пару точек), то работа функции getBridge завершается, а если на выходе был получен nil (no value), то функция вызывает саму себя с новыми кандидатами.

Для поиска медианы среди значений был использован алгоритм под названием Median of Medians, который выполняется за линейное время. Изначально медиана искалась с помощью предварительной сортировки значений, что колоссально увеличивало время исполнения программы.

5.2 Особенности реализации в системе компьютерной алгебры

Как и в случае с методом перебора, встроенные возможности системы очень полезны при выполнении данной задачи. Встроенные команды позволили значительно ускорить написание кода, устранив необходимость в создании большого числа пользовательских. Большая часть команд, указанных в описании реализации метода перебора, нашли применение и в данном алгоритме.

Как и в первом случае, создана пользовательская функция KirkpatrickSeidelConvexHull, принимающая на вход список точек и возвращающая результат работы алгоритма. Реализация алгоритма схожа с описанной ранее для языка C++, подразумевает создание аналогичных пользовательских функций, например рекурсивной функции connect и др. Однако благодаря применению широкого арсенала встроенных команд в Wolfram Mathematica, удалось частично упростить и улучшить читаемость кода, сохранив при этом основные принципы его работы и эффективность, что позволяет сделать выводы о полезности системы Wolfram Mathematica для решения подобных задач.

Заключение

В ходе практики были изучены основные возможности языка программирования C++ и системы компьютерной алгебры «Wolfram Mathematica», закреплены знания и умения полученные в рамках курсов «Введение в информационные технологии», «Информационные технологии профессиональной деятельности». Были изучены методы решения задачи о построении выпуклой оболочки множества точек на плоскости. Были изучены возможности системы компьютерной алгебры «Wolfram Mathematica» в сфере визуализации и построения алгоритмов.

Список литературы

1. The ultimate planar convex hull algorithm // Department of computer science of Princeton University.
   URL: https://www.cs.princeton.edu/~chazelle/temp/451/451-2019/KirkSeidel.pdf
   (дата обращения 24.12.2020).
2. Kirkpatrick-Seidel Algorithm (Ultimate planar convex hull algorithm) // Open-Genious IQ
   URL: https://iq.opengenus.org/kirkpatrick-seidel-algorithm-convex-hull
   (дата обращения 24.12.2020).
3. Алгоритм Киркпатрика // Википедия. Свободная энциклопедия.
   URL: https://ru.wikipedia.org/wiki/Алгоритм_Киркпатрика
   (дата обращения 24.12.2020).