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 Швецов Григорий Алексеевич
фамилия, имя, отчество

Группа ФН2-32Б

Тип практики: Ознакомительная практика

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

Студент _____ Швецов Г.А.
подпись, дата *фамилия и.о.*

Руководитель практики _____ Попов А.Ю.
подпись, дата *фамилия и.о.*

Оценка _____

2021 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 Швецов Григорий Алексеевич
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 – Зайделя построения выпуклой оболочки.

Дата выдачи задания «20» сентября 2021 г.

Руководитель практики _____ Попов А.Ю.
подпись, дата *фамилия и.о.*

Студент _____ Швецов Г.А.
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 – Зайделя (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. Постановка задачи. Общие соображения.

Понятие выпуклой оболочки можно определить следующим образом:

Определение 1.1. Область F, принадлежащая плоскости, будем называть выпуклой, если для любой пары точек f_1 и f_2, принадлежащих F, отрезок $f_1 f_2$ целиком принадлежит F.

Определение 1.2. Выпуклой оболочкой множества точек S, принадлежащих плоскости, называется граница наименьшей выпуклой области плоскости, которая охватывает S.

Выпуклая оболочка множества точек S обычно обозначается как $CH(S)$.

Задача построения выпуклой оболочки на плоскости обычно ставится в двух вариантах.

Задача 1.1. В плоскости задано множество S, содержащее n точек. Требуется определить те из них, которые являются вершинами выпуклой оболочки $CH(S)$.

Задача 1.2. В плоскости задано множество S, содержащее n точек. Требуется построить их выпуклую оболочку (т.е. найти полное описание границы $CH(S)$). Ответом этой задачи является упорядоченный список граней выпуклой оболочки.

В двумерном случае гранями являются вершины и ребра оболочки.

Задача 1.1 обычно возникает в теории, а на практике она используется редко ввиду низкой практической полезности результатов решения этой задачи. В данной работе рассматривается именно задача 1.2.

На рис. 1 проиллюстрирован пример задачи построения выпуклой оболочки на плоскости. Множество S состоит из тринадцати точек. Результатом работы алгоритма построения выпуклой оболочки должен быть список $(p_1, p_2, p_3, p_4, p_5, p_6)$. Стоит отметить, что список, содержащий вершины выпуклой оболочки, может начинаться с любой из вершин, входящих в нее. К примеру, список $(p_3, p_4, p_5, p_6, p_1, p_2)$ тоже является правильным результатом [1].

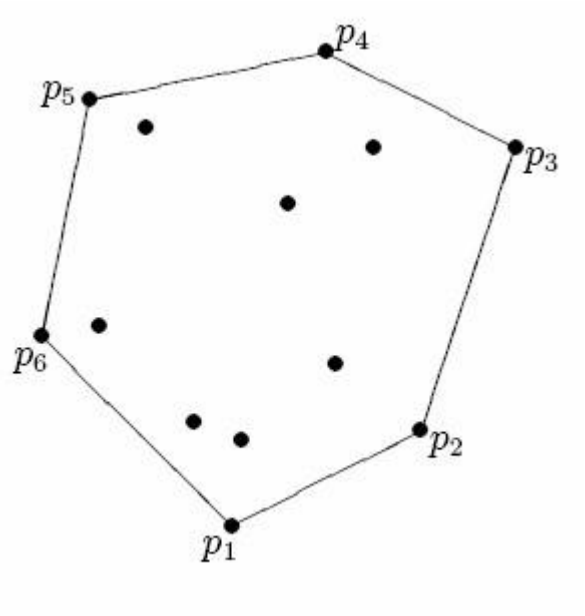

Рис. 1. Тринадцать точек, шесть из которых являются вершинами выпуклой оболочки.

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. Также алгоритмы построения выпуклых оболочек применяются в следующих областях:

- СУБД. Для выполнения запросов на экстремальность данных по нескольким параметрам;
- Робототехника. Для определения оптимальных путей обхода препятствий;
- Материаловедение. В широко используемом методе конечных автоматов;
- Кристаллография. Для облегчения изучения поверхностей кристаллов;
- Компьютерные сети. Выпуклые оболочки используются в алгоритмах маршрутизации.

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. Первую точку возьмем самую крайнюю (например, точку с минимальной X-координатой P_{last}), которая записывается, как первая точка выпуклой оболочки. Для поиска следующих точек вводится точка $P_{candidate}$ (точка-кандидат). С помощью псевдоскалярного (косого) произведения определяем положение остальных точек относительно прямой $P_{last}P_{candidate}$. Если все остальные точки лежат по одну сторону от прямой $P_{last}P_{candidate}$, то кандидат $P_{candidate}$ записывается в массив точек выпуклой оболочки; координатам точки P_{last} присваиваются значения координат точки $P_{candidate}$; 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.

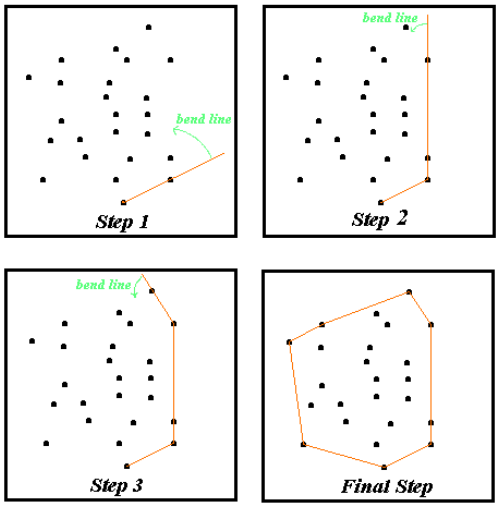

Рис. 2. Принцип метода перебора.

3. Алгоритм Киркпатрика – Зайделя

3.1. История алгоритма

Алгоритм Киркпатрика – Зайделя – 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 Зайделем (статья D.G.Kirpatrick & R.Seidel «THE ULTIMATE PLANAR CONVEX HULL ALGORITHM?» (SIAM Journal on Computing, 15(2) 1986) [2]). Приобрел известность благодаря своему быстродействию.

3.2. Стандартная реализация

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 – Зайделя 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. Результатом исполнения алгоритма является список точек, образующих выпуклую оболочку.

Определение 3.2.1. «Bridge» или «Мостом» будем называть такую прямую, пересекающую медиану точек по X-координате и образованную из двух точек исходного множества, что остальные точки лежат ниже этой прямой для верхней оболочки, или выше – для нижней оболочки (рис. 3).

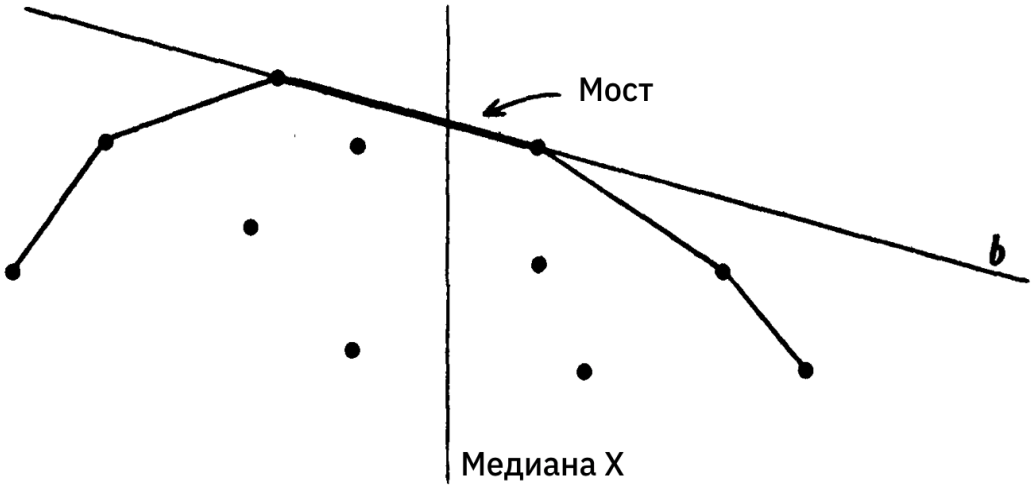

Рис. 3. Построение «моста».

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 – с y_{min} (рис. 4). Далее приведем пример для поиска верхней части оболочки, потому что для нижней части алгоритм применяется по аналогии.

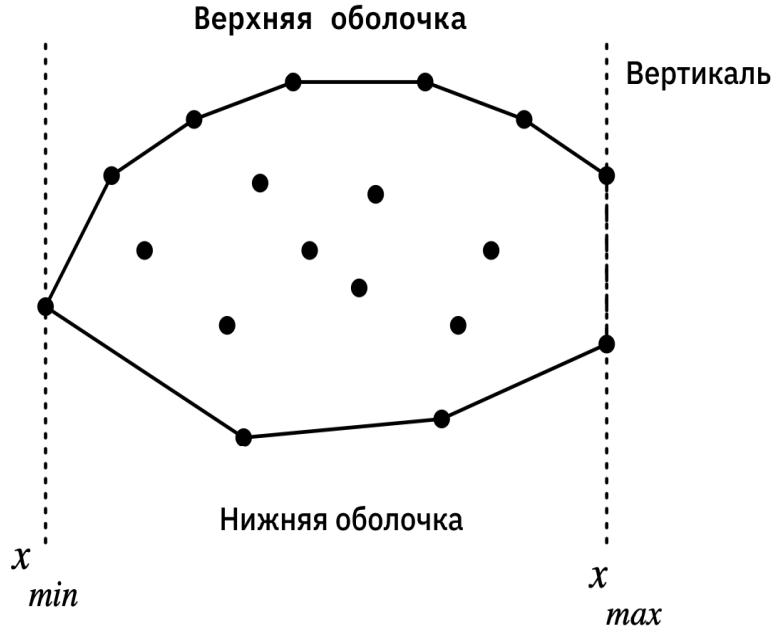

Рис. 4. Верхняя и нижняя оболочка.

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.

Поиск «Моста» опирается на следующие леммы, верные для верхней оболочки. Приведем их без доказательства.

Лемма 3.2.1. Пусть p и q – пара точек множества S. Если эти точки имеют одинаковые «иксы» и $y(p) > y(q)$, тогда q не может быть точкой «Моста».

Лемма 3.2.2. Пусть p и q – пара точек множества S, при этом $x(p) < x(q)$; s_{pq} – наклон прямой h, проходящей через точки p и q, s_b – наклон «Моста». Тогда верны следующие утверждения:

1) Если $s_{pq} > s_b$, тогда p не может принадлежать мосту;
2) Если $s_{pq} < s_b$, тогда q не может принадлежать мосту.

Лемма 3.2.3. Пусть h – опорная линия множества S с наклоном s_h, L – вертикальная прямая проходящая через медиану по всем X-координатам точек множества S. Тогда верны следующие утверждения:

1) $s_h < s_b \Leftrightarrow h$ содержит только такие точки из S, которые строго правее прямой L;
2) $s_h = s_b \Leftrightarrow h$ содержит точку из S, которая находится строго правее L, и точку из S, которая находится слева от L;
3) $s_h > s_b \Leftrightarrow h$ содержит только такие точки из S, которые левее или принадлежат прямой L.

Для того чтобы отсеять на каждой итерации как можно больше точек, целесообразно выбирать в качестве опорной линии h такую прямую, которая имеет наклон, равный медианному наклону K среди пар точек, и проходит через точку p с максимальным значением $y(p) - K * x(p)$. Тогда все точки будут лежать по одну сторону относительно этой опорной прямой.

Если точка p лежит левее прямой L, то очевидно, что $s_h > s_b$. Далее мы, руководствуясь пунктом 1 леммы 3.2.2, исключаем точки, которые точно не являются точками «Моста». Аналогичные рассуждения для случая, когда точка p лежит правее прямой L.

Если же мы вообще не можем определить наклон прямой h, т.е. когда прямая h вертикальна, то мы исключаем точки по лемме 3.2.1 [2].

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 удобства представления точек создаем заголовочный файл Point.h, в котором опишем структуру Point (при инициализации она получает координаты X и Y типа double). Для проверки алгоритма создаем файл – RandomFile.cpp, который создает текстовый файл question.txt и записывает в него случайные координаты точек. Для работы с текстовыми файлами будем использовать две вспомогательные функции: чтение из файла – FileToVector и запись в файл – VectorToFile. Файл с самим «методом перебора» называется Perebor.cpp, в котором и происходит поиск точек выпуклой оболочки.

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 в основном такие команды, как SetDirectory (задание директории) и Import, для работы с List – команды Delete, Rest (удаление первого элемента), AppendTo и PrependTo (добавление в конец и начало соответственно), Length, MemberQ (проверка наличия элемента в List) и др. Для визуализации результата – Graphics, для замера времени – Timing. Кроме того, использовались программные модули (Module) для реализации алгоритма с созданием временных переменных. Также потребовалось создать пользовательскую функцию determinePointPosition, необходимую для определения положения точки относительно прямой. Метод «перебора» реализован в виде пользовательской функции SimpleSearchConvexHull, принимающей на вход список точек points и возвращающей список точек выпуклой оболочки – convexHullPoints.

5. Реализация алгоритма Киркпатрика – Зайделя

5.1. Особенности реализации на языке C++

Реализация алгоритма представлена статической функцией convexHull класса KSAlgorithm, принимающей на вход массив исходных точек. Для представления пар точек воспользуемся стандартной структурой std::pair.

Реализация практически полностью совпадает с псевдокодом алгоритма, указанного в статье [2]. Но благодаря широким возможностям управления памятью в языке C++ удалось увеличить быстродействие и уменьшить потребление памяти. Использование структур и алгоритмов стандартной библиотеки позволило сделать реализацию алгоритма меньшего объема.

Некоторые вспомогательные функции, например, для работы с файлами, были заимствованы из реализации метода перебора на языке C++.

Для поиска медианы среди значений был использован алгоритм под названием Median of Medians, который выполняется за линейное время. Это свойство алгоритма поиска медианы позволяет алгоритму Киркпатрика иметь алгоритмическую сложность $O(n \log h)$.

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. Но также были использованы и другие команды, необходимые для данного метода, например, команда для поиска медианы Median и команда для выбора элементов списка по заданному условию Select.

Как и в первом случае, алгоритм оформлен в виде пользовательской функции – KirkpatrickSeidelConvexHull, принимающая на вход список точек и возвращающая выпуклую оболочку данного множества точек. 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 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.

6. Примеры решения задач

Пример №1:

Входные данные представлены в табл. 1. Результаты работы алгоритмов представлены на рис. 5, все алгоритмы отработали верно.

Таблица 1. Исходные данные первого примера.

№	x	y	№	x	y
1	0,611113	-0,21362	11	0,507503	-0,26102
2	-0,12802	0,286984	12	0,785982	0,595095
3	-0,80514	-0,56272	13	0,910883	0,014836
4	-0,42123	0,652841	14	-0,43634	-0,51012
5	0,292434	-0,39467	15	-0,55261	0,462464
6	0,24509	-0,54447	16	-0,44638	-0,3773
7	-0,58943	-0,7138	17	0,07349	0,006499
8	-0,03811	0,741585	18	0,534173	-0,77109
9	-0,62435	0,925342	19	-0,47837	-0,98827
10	0,627778	-0,42933	20	-0,62777	0,518943

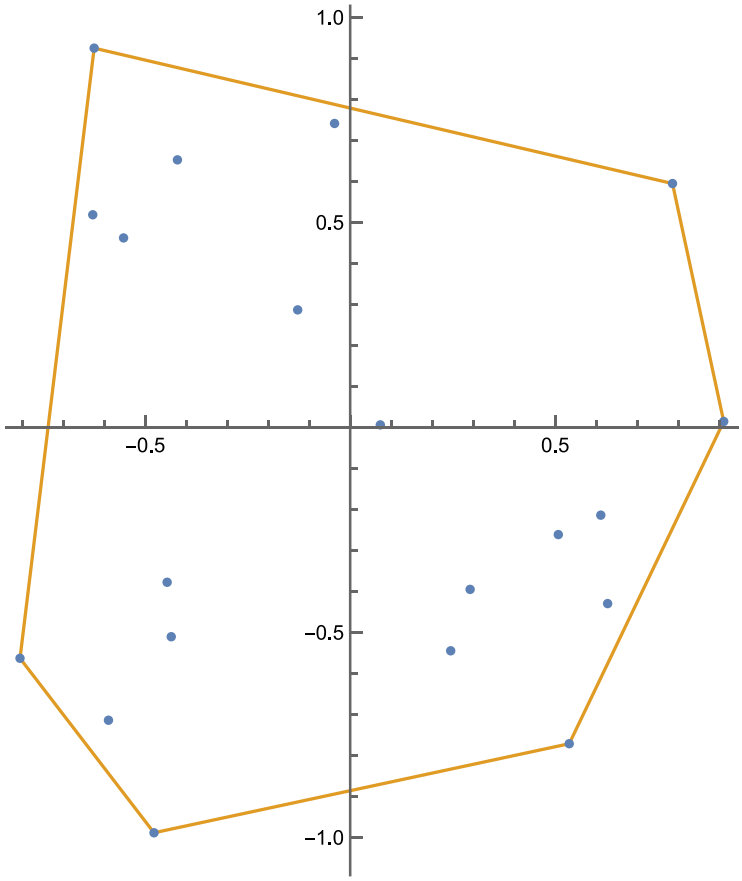

Рис. 5. Результат работы алгоритмов для первого примера.

Пример №2:

На вход алгоритмам были поданы данные, указанные в табл. 2. Результаты отображены на рис. 6.

Таблица 2. Исходные данные второго примера.

№	x	y
1	0,873866	-0,07062
2	-0,54122	0,844815
3	-0,42241	-0,25143
4	0,736873	0,46964
5	-0,31983	0,565911
6	-0,94669	0,565469
7	-0,04132	0,499485
8	-0,91186	0,398588
9	0,942521	0,120616
10	0,623433	-0,75397

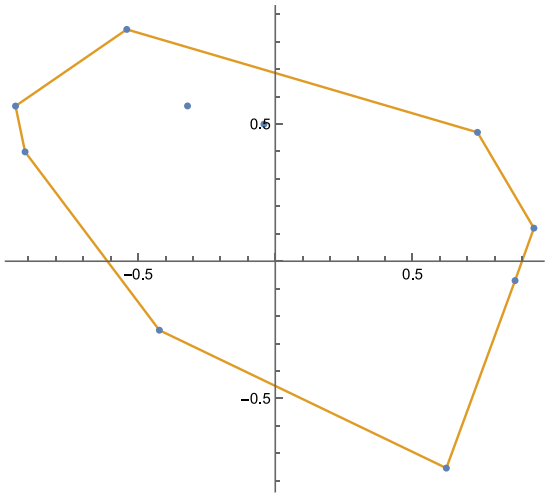

Рис. 6. Результат работы алгоритмов для второго примера.

Таблица 3. Зависимость времени алгоритмов от исходных данных.

Метод	**Метод перебора**		**Алгоритм Киркпатрика**	
Количество точек	**C++**	**WM**	**C++**	**WM**
1 000	0.0014457	2.48438	0.0070000	0.1096375
10 000	0.0143802	25.6094	0.0520000	1.5153563
100 000	0.1548372	-	0.4840000	38.012656
1 000 000	2.9566592	-	2.1185103	-
10 000 000	20.287572	-	12.09003	-

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. Алгоритмы построения выпуклых оболочек и их применение в ГИС иСАПР // Томский Государственный университет.
   URL: http://www.inf.tsu.ru/library/DiplomaWorks/CompScience/2004/Chadnov/diplom.pdf
   (дата обращения 20.12.2021).
2. 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 19.11.2021).