УТВЕРЖДАЮ
Профессор кафедры
ИАНИ ННГУ, д.т.н.

_____ Н.В. Старостин
«____»_____2021 г.

Научно-технический отчет

РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

«Разработка и реализация программного обеспечения для выполнения логических операций над множествами ортогональных многоугольников (ПО «SOR»)»

2021

Оглавление

ВВЕДЕНИЕ

Рассматривается проблема разработки программного средства для выполнения логических операций над множествами многоугольников.

В рамках данного проекта проведены следующие работы:

Разработка алгоритмов логического объединения, пересечения и вычитания множеств ортогональных многоугольников с учетом требований к размерам задачи (порядок множества входных данных: 10^5) и быстродействия программы (время выполнения расчета программой: не более 30 секунд).

1. Содержательная постановка задачи

Рассматривается задача выполнения логических операций над множествами ортогональных многоугольников. Имеются два множества ортогональных многоугольников. Требуется получить новое множество ортогональных многоугольников в результате выполнения логических операций объединения, пересечения, вычитания над исходными множествами.

2. Входные данные

Входные данные считываются из файла формата .txt. Пример входных данных приведён на рисунке 1.

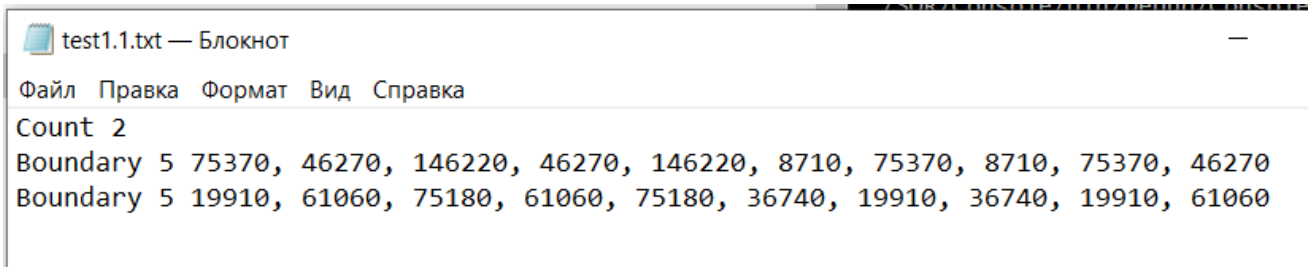

Рисунок 1- Пример тестовых данных

где, «Count 2» - количество многоугольников, «Boundary 5» - объявление многоугольника и количества его вершин, далее следуют координаты вершин многоугольника.

Тестовые данные корректно считаются при разделительных знаках: «, » «; » «. » «: » или « ».

Программа будет корректно работать при указании координат вершин многоугольника при любом порядке обхода и с любой вершины.

3. Выходные данные

К выходным данным ПО «SoR» относятся:

- решение;
- протокол работы системы.

3.1. Решение

Решением является множество многоугольников, полученных в результате выполнения логических операций.

ПО «SoR» осуществляет выгрузку полученного решения в файл формата .txt. Формат файла решения совпадает с форматом файла входных данных и описан в разделе 2.

3.2. Протокол работы системы

Вместе с решением в качестве входных данных система генерирует лог-файл (документ формата .txt), содержащий протокол работы системы.

Структура сообщений в лог-файле:

logger->WriteLog(LOGGER::LogLevel::Data, Time, MessageType, Message)

Аргумент MessageType – тип сообщения. Аргумент может принимать следующие значения:

- Error – сообщение об ошибке.

Аргумент Message – непосредственно сообщение.

Пример сообщений:

- Data\Time\Read first file – successfully (unsuccessfully);
- Data\Time\Read second file - successfully (unsuccessfully);
- Data\Time\Merge – successfully (unsuccessfully);
- Data\Time\Read file with answer - successfully (unsuccessfully);
- Data\Time\Intersect – successfully (unsuccessfully);
- Data\Time\Subtrac – successfully (unsuccessfully);
- Data\Time\Accepted – successfully (unsuccessfully);
- Data\Time\Write answer- successfully (unsuccessfully);
- Data\Time\Error Not enough input data;
- Data\Time\Error Incorrect path to first file;

- Data\Time\Error Incorrect path to second file;
- Data\Time\Error Incorrect operation;
- Data\Time\Error Incorrect path to file with answer;
- Data\Time\Error Wrong Answer.

Так же регистрируются время начала вызова программы из консоли и завершение работы программы.

Пример лог-файла приведен на рис. 2.

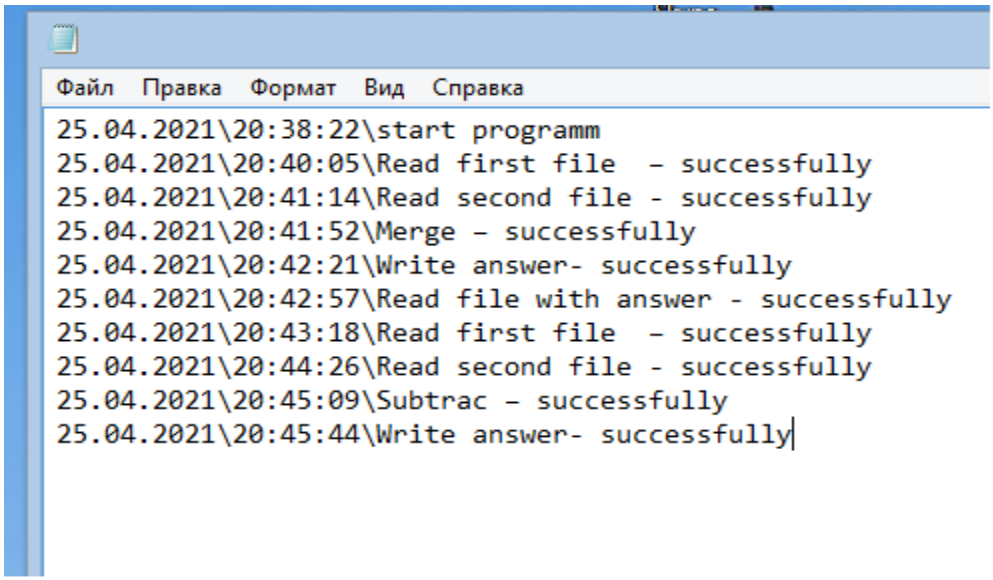

Рисунок 2 – Лог-файл

4. Разработка основных алгоритмов решения задачи

4.1. Алгоритм нахождения точек пересечения

Для каждого горизонтального ребра многоугольников из входных множеств создать тройку $(x_l, y, start), (x_r, y, end)$.

Пройти по множеству вертикальных ребер в порядке неубывания x-координаты, добавить в множество точек пересечения все

$$(x', y) | y_l \le y \le y_r, \sum_{(x,y,start)|x \le x'} 1 - \sum_{(x,y,end)|x < x'} 1 > 0$$

4.2. Утверждение 1

Вершины на одной вертикальной прямой, соседние после сортировки в порядке неубывания, первая из которых находится на нечетной позиции, меняют раскраску (принадлежность многоугольникам) ребра, соответствующего этим вершинам, на противоположную.

4.3. Алгоритм построения обходов по множеству результирующих вершин

Пока множество не пусто: выбрать минимальную лексикографически вершину, начать движение вверх. Пока не вернемся в начальную вершину: взять следующую вершину в направлении движения, выбрать новое направление движения согласно маске найденной вершины. Маска вершины – число, характеризующее принадлежность ортогональных четверть-плоскостей множеству многоугольников в окрестности данной вершины.

4.4. Алгоритм присоединения вырезов

Будем поддерживать дерево отрезков на принадлежность точек конкретному обходу. Пройти по обходам в лексикографическом порядке. Если обход внешний, то сохранить его вертикальные ребра, иначе для всех ребер не правее текущей x-координаты установить в дереве на соответствующем отрезке принадлежность обходу, содержащему это ребро, проверить какому обходу принадлежит текущая вершина, добавить текущий обход к найденному.

4.5. Алгоритмы объединения/пересечения/вычитания множеств многоугольников

Будем поддерживать деревья отрезков, поддерживающие раскраску слева и справа от вертикальной прямой, для обоих входных множеств. Пройти по всем различным x-координатам вершин, принадлежащим исходным множествам. Изначально ни одна точка не принадлежит многоугольникам. При переходе к новой прямой необходимо изменить раскраску справа от прямой согласно п. 4.2. Затем для всех точек пересечения получить маски их раскраски из деревьев, провести соответствующую логическую операцию над масками, если

результирующая маска получилась маской вершины, добавить эту вершину в множество результирующих вершин. Перед переходом к новой прямой необходимо изменить раскраску слева от прямой согласно п. 4.2. Построить обходы многоугольников по множеству результирующих вершин, добавить обходы вырезов к обходам многоугольников, в которые они вложены.

5. Результаты тестирования

5.1. Тестовые данные

Тестовые данные считываются из файла формата .txt. Формат файла решения совпадает с форматом файла входных данных и описан в разделе 2.

5.2. Возможные ошибки в тестовых данных

Если ошибка связана с конкретным многоугольником, в сообщении лога об ошибке указывается название файла и номер строки входного файла, где найдена ошибка. Если ошибка связана только с файлом, в сообщении лога об ошибке указывается название данного файла. Список ошибок в тестовых данных приведён в таблице 1.

Таблица 1 – Возможные ошибки в тестовых данных

Ошибка	Описание ошибки
"Count" needed in file: test1.1.txt	Не прописан «Count»
The number of boundaries needed in file: test1.1.txt	Не прописано количество многоугольников
"Boundary" needed in file: test1.1.txt	Заявлено больше многоугольников, чем описано
The number of points needed in 2 line in file: test1.1.txt	Не указано количество точек многоугольника
Wrong number of points in 2 line in file: test1.1.txt	Количество точек многоугольника должно быть больше 5 и нечётным
Error in coordinates in 3 line in file: test1.1.txt	В координатах присутствуют не только числа и разделительные знаки после координаты
Sudden EOF: test1.1.txt	Недостаточное количество координат прописано
The first and last point of boundary in 2 line are not the same in file: test1.1.txt	Первая и последняя точки многоугольника должны совпадать
Orthogonality problems with boundary in 2 line in file: test1.1.txt	Указанный многоугольник не ортогонален

5.3. Генератор тестовых задач

5.3.1. Требования к генератору тестов

Генератор тестов запускается из командной строки, создает и записывает в файл множество непересекающихся ортогональных многоугольников.

5.3.2. Требования к входным данным

Создание теста производится запуском Generator.exe из командной строки с указанием необязательных параметров:

1) path/filename.txt – файл для записи теста.

Значения: [100, INT_MAX (2147483647)]

2) count – количество многоугольников.

Значения: [1, 100 000]

3) max_points - максимальное количество точек в генерируемых многоугольниках.

Значения: [4, 10000]

4) min_points - минимальное количество точек в генерируемых многоугольниках.

Значения: [4, 10000]

5) max_width – максимальная ширина (расстояние между точками многоугольника, наиболее удаленными друг от друга по оси абсцисс) многоугольников.

Значения: [1, area_size - 4]

6) min_width – минимальная ширина (расстояние между точками многоугольника, наиболее близкими друг к другу по оси абсцисс) многоугольников.

Значения: [1, area_size - 4]

7) max_height – максимальная высота (расстояние между точками многоугольника, наиболее удаленными друг от друга по оси ординат) многоугольников.

Значения: [1, area_size - 4]

8) min_height – минимальная высота (расстояние между точками многоугольника, наиболее близкими друг к другу по оси ординат) многоугольников.

Значения: [1, area_size - 4]

9) max_distance – максимальное расстояние между многоугольниками.

Значения: [1, area_size - 8]

10) min_distance – минимальное расстояние между многоугольниками.

Значения: [1, area_size - 8]

11) gap_count – количество многоугольников с зазорами.

Значения: [0, count]

При этом входные данные должны удовлетворять наличию допустимого решения. В случае если создание множества многоугольников с заданными параметрами невозможно, в консоль выводится сообщение об ошибке.

Если значение необязательного параметра не передается, оно случайным образом генерируется программой в фиксированном диапазоне.

5.4. Требования к выходным данным

Генератор сохраняет созданный тест в файл формата .txt в следующем виде: количество многоугольников, далее для каждого многоугольника количество его вершин и их координаты в порядке обхода по часовой стрелке. В конце еще раз указывается первая точка, она учитывается при подсчете количества вершин многоугольника.

ЗАКЛЮЧЕНИЕ

Была поставлена задача разработки и реализации программного обеспечения для выполнения логических операций над множествами ортогональных многоугольников. В ходе работы поставленная задача была решена. Разработанное ПО выполнило все тесты, получив корректные результаты и затратив время меньше 30 секунд.