МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

КАФЕДРА КОМПЬЮТЕРНЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ст. препод, канд. техн. наук		
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

РЕАЛИЗАЦИЯ АЛГОРИТМА ЛИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

по дисциплине: Дискретная математика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	Р3122		Л.А. Ушакова
		подпись, дата	инициалы, фамилия

Санкт-Петербург, 2018

Содержание

1 Анализ задачи

Постановка задачи: Алгоритм Ли для поиска пути (гексагональная сетка) на равностороннем четырехстороннем (в виде паралелограмма) дискретном рабочем поле. Длина сторон рабочего поля может варьироваться в пределах [10;100]. Требуется реализовать указанный в теме алгоритм. При этом вывод итоговых и промежуточных результатов должен быть реализованы в виде svg графики.

1.1 Гексагональная сетка

В основе гексагональной сетки (*англ.* hexagon grid) лежит правильный шестиугольник (*англ.* hexagon), внутренние углы которого равны 120°. На гексагональной сетке две параллельные стороны всех шестиугольников могут располагаться либо параллельно оси ординат, либо оси абсцисс. Возьмем первый вариант и выведем математические формулы, которые описывают гексагональную сетку и её элементы на равностороннем четырехстороннем дискретном рабочем поле (*РЧДРП*).

Прежде всего необходимо задать размерность шестиугольника $hexSize$. Она определяет расстояние углов от центра фигуры. Размерность паралелограмма обозначим через $dimension$.

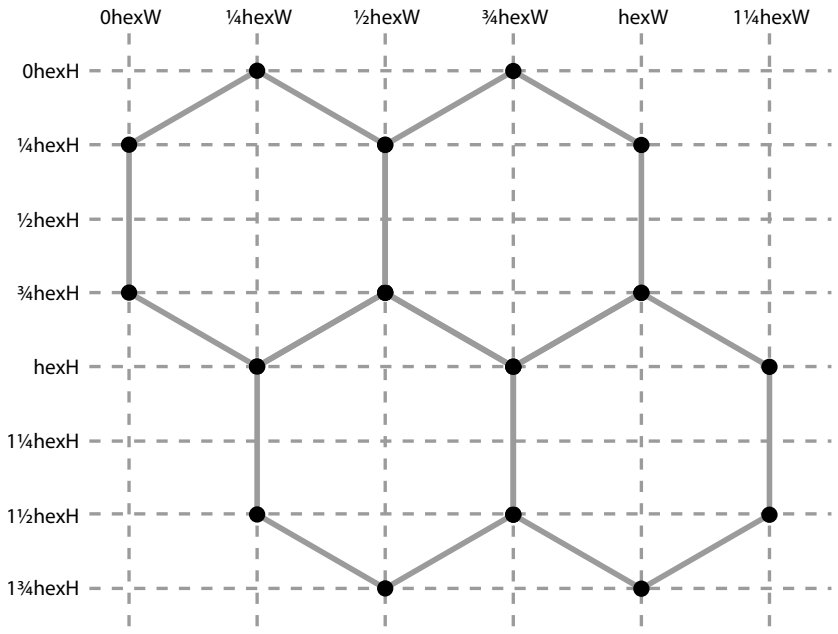

Рисунок 1.1 — Гексагональная сетка.

Высота данного поля $gridHeight$ вычисляется по формуле:

$$gridHeight = hexH(\frac{3}{4}dimension + \frac{1}{4}), \tag{1.1}$$

где hexH — высота шестиугольника, которую можно найти по формуле $2hexSize$. Ширина данного поля $gridWidth$ вычисляется по формуле:

$$gridWidth = hexW \times dimension, \tag{1.2}$$

где hexW — ширина шестиугольника, которая равна $hexSize\sqrt{3}$.

1.2 Алгоритм Ли

Алгоритм Ли — волновой алгоритм поиска пути на карте, алгоритм трассировки. С его помощью можно построить путь, или трассу, между двумя любыми элементами в лабиринте. Из начального элемента распространяется в четырёх (в этой задаче — в 6) направлениях волна. Тот элемент, в который она пришла, образует фронт волны.

Элементы первого фронта волны являются источниками вторичных волн.

Элементы второго фронта генерируют волну третьего фронта и так далее. Процесс заканчивается тогда, когда достигается конечный элемент. На втором этапе строится трасса. Построение производится в соответствии с некоторыми правилами:

а) при построении трассы движение проходит в зависимости от выбранных приоритетов,

б) путевые координаты уменьшаются при переходе от начального элемента к конечному.

Эти приоритеты выбираются в процессе разработки. В зависимости от выбора тех или иных приоритетов получаются различные трассы. Но в любом случае длина трассы остается одной и той же.

Используя волновой алгоритм, можно найти трассу в лабиринте с любым количеством стен. В этом и заключается **преимущество** его использования.

Недостаток алгоритма Ли заключается в том, что при построении трассы требуется большой объем памяти.

2 Проектирование программного инструмента

В качестве системы координат была выбрана классическая прямоугольная система, где одна координата обозначает номер строки в сетке, а вторая номер столбца. у каждой ячейки максимум шесть соседних ячеек, это количество может уменьшаться из-за наличия закрытых ячеек.

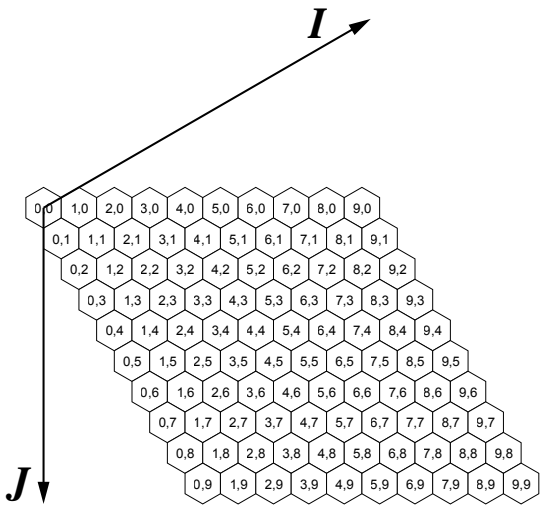

Рисунок 2.1 — Гексагональная сетка с ситемой координат.

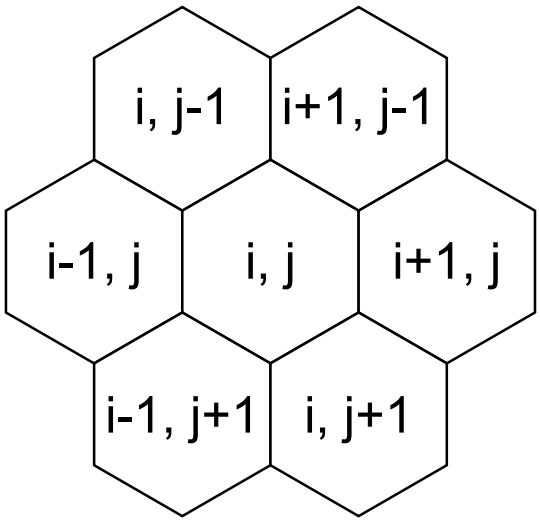

Рисунок 2.2 — Соседние ячейки.

2.1 Описание классов и методов

а) Класс **OffsetPoint**:

1) координата X
2) координата Y
3) перегрузка оператора проверки на равенство(==)
4) перегрузка оператора проверки на неравенство (!=)
5) перегрузка оператора сравнения (<)

б) Класс **HexagonGrid** (основной):

Поля класса:

1) **dimension** - размерность сетки
2) **hexSize** - размер шестиугольника (расстояние от центра до вершины)
3) **gridWidth** - ширина сетки

4) **gridHeight** - высота сетки
5) **hexW** - ширина шестиугольника
6) **hexH** - высота шестиугольника
7) **start** - точка старта (типа OffsetPoint)
8) **finish** - точка финиша (типа OffsetPoint)
9) **fronts** - карта фронтов
10) **path** - множество ячеек, входящих в путь
11) **terminates** - множество закрытых ячеек
12) **neighbours** - множество соседних ячеек к данной (public)

Методы класса:

1) конструктор **HexagonGrid(int dimension, OffsetPoint start, OffsetPoint finish)**
2) **drawSVG(string fileName)** - отрисовка сетки с учетом закрытых ячеек и пути
3) **neighbours(int i, int j)** - алгоритм поиска соседних ячеек к данной Осуществляет проверку всех вершин вокруг данной, и если они открыты, а также не выходят за пределы сетки, то добавляет в список. Возвращает список открытых соседних ячеек.
4) **liAlg()** - реализация алгоритма Ли (пускает волну до нахождения финиша) Перебирает все ячеки, пока не найдет финишную или список открытых соседних ячеек не окажется пуст (в этом случае путь не найден, программа выходит из цикла). Каждая из этих ячеек в случае, если уже не входит в другой фронт волны, помечается номером на единицу больше фронта данной ячейки. В итоге имеем карту, где каждому номеру фронта соответствует множество ячеек.
5) **findPath()** - нахождение пути (двигается от финиша по фронтам волны) Начиная с финишной ячейки, ищет первую соседнюю с фронтом на единицу меньше и добавляет ее в путь, пока не дойдет до ячейки старта.
6) **BuildTerminates()** - построение закрытых ячеек случайным образом (private)

3 Описание работы программы

При запуске файла main.cpp пользователю предлагается ввести несколько значений:

а) dimension - размерность сетки (по условию от 10 до 100)

б) startX, startY - координаты начальной ячейки (не превышают размерности сетки)

в) finishX, finishY - координаты конечной ячейки (не превышают размерности сетки)

Если данные не соответствуют требованиям выводится сообщение "Invalid input".

Далее создается объект класса HexagonGrid, то есть создается сетка с учетом закрытых ячеек. А затем выполняется алгоритм. Для каждого фронта волны создается svg файл, в котором числами отмечены этот и все предыдущие фронты. В конце алгоритма, если путь найден, создается итоговый файл **00_FINAL.svg**, в котором отображен путь. Если в процессе выполнения алгоритма не найдено свободных соседних ячеек, а финиш не достигнут, выводится сообщение "Path is not found".

4 Тестирование

Процесс тестирования необходимо разделить на этапы:

а) Низкая нагрузка — размерность РЧДРП в 10-20 элементов;
б) Стандартная нагрузка — РЧДРП размерностью 30-60;
в) Максимальная нагрузка — РЧДРП размерностью от 80 до 100.

Раздел тестирования построен следующим образом:

а) Указываются входные параметры (размерность сетки);
б) Количество svg-файлов в директории;
в) Время отрисовки всех svg-файлов;
г) Время выполнения алгоритма Ли.

Размерность	**Кол-во svg-файлов**	**Время отрисовки (мс)**	**Время выполнения алгоритма Ли (мс)**
10	18	471.985	2.981
20	32	2865.764	8.98
40	41	12148.226	29.968
50	78	33820.687	57.967
60	117	69962.495	61.962
100	214	336477.874	214.881

Таблица 4.1 — Результаты тестирования

Список использованных источников

1. w3schools [Электронный ресурс] (дата обращения: 03.06.2018). https://www.w3schools.com/graphics/svg_intro.asp.

2. *Academy, Html.* Знакомство с SVG [Электронный ресурс] (дата обращения: 03.06.2018). https://htmlacademy.ru/courses/130/.

3. *Patel, Amit.* Hexagonal Grids [Электронный ресурс] (дата обращения: 03.06.2018). — 2015. https://www.redblobgames.com/grids/hexagons/.

4. *PatientZero.* Создание сеток шестиугольников [Электронный ресурс] (дата обращения: 03.06.2018). — 2017. https://habr.com/post/319644/.

5. *Ф.А., Новиков.* Дискретная математика для программистов / Новиков Ф.А. — Питер, 2013.