МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический
университет Петра Великого»
Институт Компьютерных Наук и Кибербезопасности
Высшая Школа Технологий Искусственного Интеллекта
Направление: 02.03.01 Математика и компьютерные науки

Теория алгоритмов

Отчёт по лабораторной работе №1

вариант 11

Студент,
группы 5130201/20102 _____ Гаар В.С.

Преподаватель,
к.т.н., доц. _____ Востров А.В.

«_____»_____ 2024 г.

Санкт-Петербург, 2024

Содержание

Введение

Данный отчёт содержит в себе описание реализации лабораторной работы №1 по дисциплине «Теория алгоритмов», направленной на изучение аспектов работы клеточных автоматов. Содержание лабораторной работы представлено ниже:

1. Реализовать двумерный клеточный автомат с окрестностью фон Неймана в соответствии с полученным номером № = номер варианта * 11 * год рождения * день * месяц.
2. Граничный условия предложить самостоятельно (торроидальные, нулевые, единичные).
3. Пользователь определяет ширину поля и количество итераций.
4. Необходимо учесть возможность ввода различных начальных условий (как вручную, так и случайным образом) по выбору пользователя.
5. Реализация возможна в консоли или в графике.
6. Проанализировать свой клеточный автомат в отчете (его поведение, паттерны, «сходимость» и т.д.).

1 Математическое описание

1.1 Клеточные автоматы

Клеточный автомат представляет собой двусторонне бесконечную ленту, каждая ячейка которой может находиться в некотором состоянии. Множество состояний Q, обозначим состояние ячейки i как $s[i]$. Изначально все ячейки находятся в состоянии $B \in Q$, кроме ячеек с номерами от 1 до n. Ячейка с номером i, где $1 \leq i \leq n$ находится в состоянии x_i, где x – входное слово (будем считать, что $\Sigma \subset Q, B \notin \Sigma$).

Правила работы клеточного автомата такие: задано число d и функция $f : Q^{2d+1} \to Q$. За один шаг все клетки меняют состояние по следующему правилу: новое состояние клетки i равно $f(s[i-d], s[i-d+1], \ldots, s[i+d-1], s[i+d])$. Если клетка с номером 0 переходит в состояние Y, то автомат допускает слово x.

Клеточным автоматом (КА) (англ. *cellular automaton*) A размерности d называется четверка $< Z^d, S, N, \delta >$, где

- S – конечное множество, элементы которого являются состояниями A.
- N – конечное упорядоченное подмножество Z^d, $N = \{n_j | n_j = (x_{1_j}, \ldots, x_{d_j}), j \in \{1 \ldots n\}\}$, называемое **окрестностью** (англ. *neighborhood*) A. В данном определении полагается, что клетка всегда принадлежит своей окрестности.
- $\delta : S_n \to S$ – функция перехода для A. [1]

Окрестность фон Неймана ячейки — совокупность ячеек в сетке (двумерном паркете, трёхмерном Евклидовом пространстве, разбитом на равновеликие кубы), имеющих общую сторону (грань) с данной ячейкой. [2]

Пример двумерной окрестности фон Неймана порядка 1 представлен на рисунке № 1.

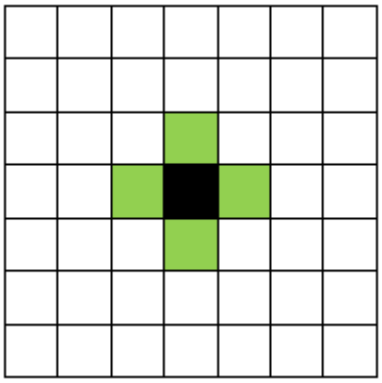

Рис. 1. Двумерная окрестность фон Неймана порядка 1

Двумерный клеточный автомат можно определить как множество конечных автоматов на плоскости, помеченных целочисленными координатами (i, j), каждый из которых может находиться в одном из состояний

$$\sigma_{i,j} : \sigma_{i,j} \in \Sigma \equiv \{0, 1, 2 ... k-1, k\}.$$

Изменение состояний автоматов происходит согласно правилу перехода

$$\sigma_{i,j}(t+1) = \phi(\sigma_{k,l}(t) | (k,l) \in \mathcal{N}(i,j)),$$

где $\mathcal{N}(i,j)$ – некоторая окрестность точки (i,j). К примеру, окрестность фон Неймана определяется как

$$\mathcal{N}_N^1(i,j) = \{(k,l)|\ |i-k| + |j-l| \leq 1\},$$

а окрестность Мура

$$\mathcal{N}_M^1(i,j) = \{(k,l)|\ |i-k| \leq 1, |j-l| \leq 1\}.$$

Число всех возможных правил перехода определяется числом состояний σ и количеством соседей n и составляет

$$N_r = \sigma^{\sigma^n}[3]$$

1.2 Классификация клеточных автоматов

1.2.1 Классификация по типам поведения

Стивен Вольфрам в своей книге A New Kind of Science предложил 4 класса, на которые все клеточные автоматы могут быть разделены в зависимости от типа их эволюции. Классификация Вольфрама была первой попыткой классифицировать сами правила, а не типы поведения правил по отдельности. В порядке возрастания сложности классы выглядят следующим образом:

- Класс 1: Результатом эволюции начальных условий является быстрый переход к гомогенной стабильности. Любые негомогенные конструкции быстро исчезают.
- Класс 2: Результатом эволюции начальных условий является быстрый переход в неизменяемое негомогенное состояние либо возникновение циклической последовательности. Большинство структур начальных условий быстро исчезает, но некоторые остаются. Локальные изменения в начальных условиях оказывают локальный характер на дальнейший ход эволюции системы.
- Класс 3: Результатом эволюции почти всех начальных условий являются псевдо-случайные, хаотические последовательности. Любые стабильные структуры, которые возникают почти сразу же уничтожаются окружающим их шумом. Локальные изменения в начальных условиях оказывают неопределяемое влияние на ход эволюции системы.
- Класс 4: Результатом эволюции являются структуры, которые взаимодействуют сложным образом с формированием локальных, устойчивых структур. В результате эволюции могут получаться некоторые последовательности Класса 2, описанного выше. Локальные изменения в начальных условиях оказывают неопределяемое влияние на ход эволюции системы. Некоторые клеточные автоматы этого класса обладают свойством универсальности по Тьюрингу, что доказано для Правила 110 и игры «Жизнь».

1.3 Тоталистичные клеточные автоматы

Существует специальный класс клеточных автоматов, называемых тоталистичными. На каждом шаге эволюции клеточного автомата значение клетки равно

какому-либо целому числу (обычно выбираемого из конечного множества), а новое состояние клетки определяется суммой значений клеток-соседей и, возможно, предыдущим состоянием клетки. Если состояние клетки на новом шаге зависит от её предыдущего состояния, то такой клеточный автомат называется внешним тоталистичным. Игра Жизнь является примером внешнего тоталистического клеточного автомата с набором значений ячеек $0, 1$.

Термин тоталистичный происходит от английского *totalistic*. В свою очередь *total* может быть переведено как сумма, что и отражено в принципе действия этого типа автоматов, когда новое значение клетки зависит от суммы значений других клеток.

1.4 Представление функции с данным №

Номер функции был вычислен по следующему выражению: № = номер варианта * 11 * год рождения * день * месяц, где номер варианта = 11. Получилось значение 2'424'840, которое далее было представлено в двоичном виде и дополнено до 32 символов незначащими ведущими нулями. Далее использовался получившийся вектор-столбец

$$f(s_0, s_1, s_2, s_3, s_4) = (00000000001001010000000000001000),$$

где переменные s_0, s_1, s_2, s_3, s_4 соответствуют следующим клеткам в окрестности фон Неймана, представленным на рисунке № 2:

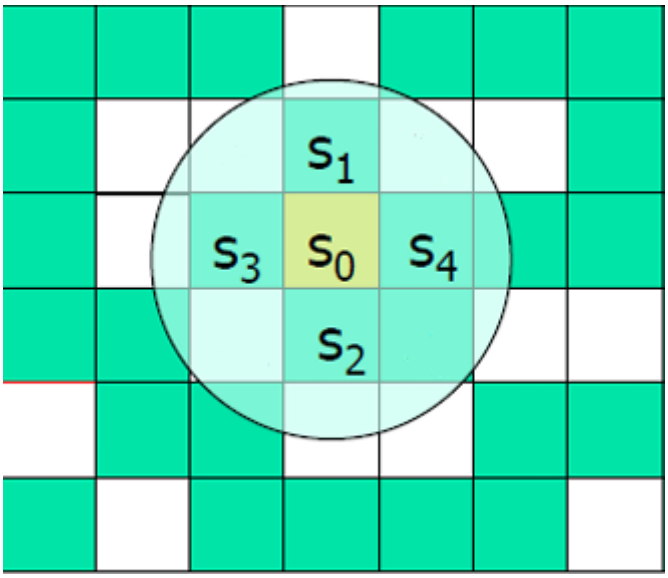

Рис. 2. Нумерация клеток в окрестности фон Неймана

2 Особенности реализации

Сам двумерный клеточный автомат реализован в виде класса `CellularAutomaton` со следующими характеристиками:

- **Поля**:
  - `std::mt19937 rng`: переменная, используемая для генерации случайных чисел;
  - `int width, height`: переменные для хранения ширины и высоты поля соответственно;
  - `std::string rule`: переменная, используемая для хранения вектора-столбца значений двоичной функции $f(s_0, s_1, s_2, s_3, s_4)$ в виде строки;
  - `std::string boundaryCondition`: переменная для хранения формата граничных условий - «zero», «one», «toroidal»;
  - `std::vector<std::vector<int» field`: контейнер для хранения текущего состояния клеточного поля;
  - `std::vector<std::vector<int» nextField`: контейнер для хранения следующего состояния клеточного поля, сдвинутого на один такт эволюции вперёд.
- **Методы**:
  - `void evolve()`: метод, выполняющий сдвиг клеточного поля на один такт эволюции вперёд;
  - `void display() const`: метод для отображения текущего состояния клеточного поля на консоль;
  - `void setInitialState()`: метод для задания исходного состояния клеточного поля пользователем;
  - `void generateInitialState()`: private-метод для случайной генерации исходного состояния клеточного поля;
  - `unsigned int countAliveNeighbors(int x, int y) const`: private-метод для подсчёта количества «живых» клеток, граничащих с клеткой с координатами (x, y);
  - `int getCell(int x, int y) const`: метод для получения значения клетки, расположенной по координатам (x, y) с учётом заданных граничных условий в поле boundaryCondition.

Код создания данного класса приведён на рисунке № 3.

```
1 class CellularAutomaton
2 {
3     std::mt19937 rng;
4
5     int width, height;
6     std::string rule;
7     std::string boundaryCondition;
```

```
8       std::vector<std::vector<int>> field;
9       std::vector<std::vector<int>> nextField;
10
11  public:
12      CellularAutomaton(int width, int height, std::string rule, std::
            string boundaryCondition, bool randomInit = false);
13
14      void evolve();
15
16      void display() const;
17
18      void setInitialState();
19
20  private:
21      void generateInitialState();
22
23      unsigned int countAliveNeighbors(int x, int y) const;
24
25      int getCell(int x, int y) const;
26  };
```

Рис. 3. Код создания класса `CellularAutomaton`

2.1 Функция получения значения клетки

Функция принимает на вход два параметра – координаты клетки (x, y), значение которой Необходимо получить. Далее функция обрабатывает каждый из возможных вариантов граничных условий в поле `boundaryCondition` и возвращает соответствующее значение. Код реализации метода `getCell` представлен на рисунке № 4.

`Вход: координаты клетки x, y типа int.`
`Выход: значение клетки поля типа int в соответствии с её координатами и заданным граничным условием.`

```
1   int CellularAutomaton::getCell(int x, int y) const {
2     if (boundaryCondition == "toroidal") {
3       x = (x + width) % width;
4       y = (y + height) % height;
5       if (x < 0  x >= width  y < 0  y >= height) {
6         return field[y][x];
7       }
8       else {
9         return field[x][y];
10      }
11    }
12    else if (boundaryCondition == "zero") {
13      if (x < 0  x >= width  y < 0  y >= height) {
14        return 0;
15      }
16      return field[x][y];
17    }
18    else if (boundaryCondition == "one") {
```

```
19      if (x < 0  x >= width  y < 0  y >= height) {
20        return 1;
21      }
22      return field[x][y];
23    }
24    return 0;
25  }
```

Рис. 4. Код реализации метода `getCell`

2.2 Функция подсчёта количества живых клеток

3 Результаты работы программы

Заключение

Список источников

[1] ИТМО. Линейный клеточный автомат, эквивалентность МТ [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=Линейный_клеточный_автомат,_эквива (дата обращения 20.11.2024).

[2] ИТМО. Модели клеточных автоматов [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=Модели_клеточных_автоматов (дата обращения 20.11.2024).

[3] A.G.Hoekstra, J.Kroc, P.Sloot. Simulating complex systems by cellular automata. Springer, 2010. ISBN 978-3-642-12202-6