МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования "Самарский национальный исследовательский университет имени академика С.П. Королева"

(Самарский университет)
Институт информатики, математики и электроники
Факультет математики
Кафедра информатики и вычислительной математики

РЕФЕРАТ
по дисциплине "Теория языков и трансляторов"
по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем
(уровень бакалавриата)

на тему:
"Клеточные автоматы"

Выполнил _____
(подпись)
Студент группы №

Проверил
Зав. кафедрой,
д.ф.-м.н., профессор
Степанов А.Н.

_____
(подпись)
Самара, 2019

Введение

Идея клеточных автоматов появилась в конце сороковых годов 20 века. Она была задумана и сформулирована Джоном фон Нейманом и Конрадом Цусе независимо друг от друга как универсальная вычислительная среда для построения, анализа и сравнения характеристик алгоритмов. "Клеточные автоматы являются дискретными динамическими системами, поведение которых полностью определяется в терминах локальных зависимостей”.

Клеточные автоматы в информатике являются аналогом физического понятия "поля". клеточный автомат может мыслиться как стилизованный(от слова стиль) мир. Пространство представлено равномерной сеткой, каждая ячейка или клетка которой содержит несколько битов данных; время идет вперед дискретными шагами, а законы мира выражаются единственным набором правил, скажем, небольшой справочной таблицей, по которой любая клетка на каждом шаге вычисляет своё новое состояние по состояниям её близких соседей. Таким образом, законы системы являются локальными и повсюду одинаковыми. "Локальный" означает, что для того, чтобы узнать, что произойдет здесь мгновение спустя, достаточно посмотреть на состояние ближайшего окружения: никакое дальнодействие не допускается. "Одинаковость" означает, что законы везде одни и те же: я могу отличить одно место от другого только по форме ландшафта, а не по какой - то разнице в законах."

Следует отметить, что клеточные автоматы – это не просто машины, работающие с разбитым на клетки полем. Область применение клеточных автоматов почти безгранична: от простейших "крестиков - ноликов" до искусственного интеллекта. Тема клеточных автоматов очень актуальна, так как может привести к разгадкам многих вопросов в окружающем мире. Создатель игры "Жизнь" Конуэй, считал, что нашу вселенную можно представить клеточным автоматом, который управляет движением элементарных частиц в соответствии с некоторыми правилами (кстати, сейчас Конуэй придумал еще одно применение клеточных автоматов в этой области: представим себе достаточно большое количество "первичного бульона" из хаотически распределенных клеток, если можно ожидать появления из такого хаоса структур, способных самовоспроизводить себя, то это еще одно подтверждение теории зарождения жизни на Земле). Клеточные автоматы используются для моделирования гидродинамических и газодинамических течений. Уравнения гидродинамики соответствуют математической модели, описывающей поведение решетчатого газа, одного из клеточных автоматов, на макроуровне. Структуры, возникающие в этих клеточных автоматах, похожи на возмущение поведения поверхности потока жидкости механическим препятствием. Примитивные одномерные клеточные автоматы могут моделировать процесс горения различного характера. В настоящее время теория клеточных автоматов наиболее перспективно прилагаема к вопросу о разработке самовосстанавливаемых электронных цепей. Клеточные автоматы применимы не только в математике и физике, а также в биологии, экономике, социологии, информатике и т. д. Их можно использовать при

составлении генетических алгоритмов. Представим себе клеточный автомат, для клеток которого дополнительным условием выживания является выработка некоторой последовательности выходных данных (назовем ее условно реакцией) в ответ на последовательность входных данных (раздражение, являющееся свойством среды), предсказывающее последующее состояние среды. Чтобы такой автомат функционировал, добавляется также механизм случайного изменения правил выработки реакции (мутации) и передачи вновь возникающим клеткам информации о правилах реагирования соседей (наследования).

Помимо исследования условий развития моделей живых систем, такой подход позволяет решать и некоторые практические задачи, в частности поиск кратчайшего пути на графе. Структура графа кодируется некоторым образом в хромосомах клеток. Предполагается, что алгоритмы, приобретенные вследствие мутаций и наследования, будут соответствовать решениям задачи. По своему поведению клеточные автоматы делятся на четыре класса. К первому классу относятся автоматы, приходящие через определенное время к устойчивому однородному состоянию. Автоматы второго класса через некоторое время после пуска генерируют стационарные или периодические во времени структуры. В автоматах третьего класса по прошествии некоторого времени перестает наблюдаться корреляция процесса с начальными условиями. Наконец, поведение автоматов четвертого класса сильно определяется начальными условиями и с их помощью можно генерировать весьма различные шаблоны поведения. Такие автоматы являются кандидатами на прототип клеточной вычислительной машины. В частности, с помощью специфических клеточных конфигураций игры "Жизнь", которая как раз и является автоматом четвертого типа, можно построить все дискретные элементы цифрового компьютера.

Отметим еще одно применение клеточных автоматов в информатике – шифрование и сжатие данных. Клеточные автоматы применимы при реализации эффективной системы распознавания образов. Один из возможных путей ее создания – построение динамической системы, аттракторами которой в ее конфигурационном пространстве были бы типичные картины - образы. Начальные условия всегда окажутся в области притяжения одной из картин, с течением времени система трансформирует начальные параметры, приведя их к наиболее близкой структуре – аттрактору. То есть произойдет автоматическое распознавание образа. Причем можно создавать обучающиеся системы распознавания, в них законы эволюции имеют состояние программирования. Использовать клеточные автоматы можно и при решении оптимизационных задач. Часто в различных сферах деятельности возникают задачи нахождения оптимального варианта из неограниченного числа возможных. Точного решения, как правило, не требуется, но дискретный компьютер не способен даже приблизительно дать оптимальный результат. Таким образом, клеточные автоматы нашли и находят широкое применение во многих сферах человеческой деятельности, многие задачи которых стало возможным решить только с помощью компьютера.

Применение сгенерированной карты

Случайно сгенерированные карты мира нужны для такого жанра игр как roguelike. В данных играх смерть перманентна (т.е. после смерти нельзя начать играть на той же карте).

Главным преимуществом roguelike-игр является непредсказуемость уровня.

Также эта карта может применяться в играх, где главная задача — найти полезные ископаемые. В таких играх игрок под землей в какой-то пещере, ему обычно предоставляются некоторые инструменты для добычи ресурсов.

Текст программы

```
//подключение библиотек
#include <iostream>//io - библиотека
#include <cstdlib>    //srand, rand
#include <ctime>      //time()
#include <cstring>   //библиотека для обработки строк
#define DEBUG_ON//объявление директивы для расширенного вывода
//использование стандартного пространства имен
using namespace std;
//прототипы необходимых функций
inline void fill_random(int w, int h, float chance, bool **matr);
inline void do_step(bool **matr, bool **matr_new, int d_l, int b_l, int h, int w);
inline int neighbours_cnt(bool **matr, int x, int y, int h, int w);

//точка входа
int main(int argc, char **argv)
{
  if(argc < 3) // если юзер не написал ширину и высоту - ответить help-ом
  {
    cout << "Usage:     " << argv[0] << " width height [death_limit] [birth_limit] [steps] [chance]\n" \
        << "Types:   int    int    int    int    int    int    float" << '\n' \
        << "Default: 0      0      0      3      4      10     0.48"  << '\n' \
        << "if count of neighbours > birth_limit->birth()" << '\n'\
        << "if count of neighbours <= death_limit->die()" << '\n';
    return 0;
  }
  srand(time(NULL)); //random seed
  //read_vars{//чтение переменных из аргументов
  int w,h,death_limit,birth_limit,steps; // input vars
  float chance;
  death_limit=3; birth_limit=4; steps=10; chance=0.48; // default values


  switch(argc) {
    case 7: chance = atof(argv[6]);
```

```
    case 6: steps = atoi(argv[5]);
    case 5: birth_limit = atoi(argv[4]);
    case 4: death_limit = atoi(argv[3]);
    default:  //<4 || >7
      h = atoi(argv[2]);
      w = atoi(argv[1]);
  }
  //}
 //alloc arrays{//объявление массивов и их инициализация
  bool **matr,**matr_new; // map
  h+=2;w+=2;//это обрамление единицами реального массива
  /*
  1111
  1da1
  1ta1
  1111
  */
  matr = (bool **)calloc(h, sizeof(bool *));
  matr_new = (bool **)calloc(h, sizeof(bool *));

  for(int i = 0; i < h; i++)
  {
    matr[i] = (bool *)calloc(w, sizeof(bool));
    matr_new[i] = (bool *)calloc(w, sizeof(bool));
    matr[i][0] = 1;//обрамление единицами
    matr[i][w-1] = 1;
    matr_new[i][0] = 1;
    matr_new[i][w-1] = 1;
  }
  w--;//возвращаемся к виртуальным размером(к тем в которые будем записывать значение)
  h--;
  for(int j = 1; j< w; j++)
  {
    matr[0][j] = 1;//обрамление единицами
    matr[h][j] = 1;
    matr_new[0][j] = 1;
    matr_new[h][j] = 1;

  }
  //}
  fill_random(w, h, chance,matr); // random fill matrix

  //do_conversion{//производим преобразования за steps шагов
  for(int i = 0; i < steps; i++)
  {
    do_step(matr, matr_new, death_limit, birth_limit, h, w);
    swap(matr, matr_new);//обновленный массив меняем со старым
```

```
    #ifdef DEBUG_ON
      cout << "processing:" <<  i+1 << "/" << steps << '\n';
    #endif
  }
  #ifdef DEBUG_ON
    cout << '\n';
  #endif
  //}
  //print_out{//вывод результата
  cout << "output map:\n";
  for(int i = 1; i < h; i++)
  {
    for(int j = 1; j < w; j++)
    {
      #ifdef DEBUG_ON
        cout << matr[i][j];
      #else
        cout << ((matr[i][j])?'#':'.');
      #endif
    }
    cout << '\n';
  }
  //}
  return 0;
}

//далее описание функций для которых был описан прототип
inline void fill_random(int w, int h, float chance, bool **matr)
{
  cout << "input random:\n";
  for(int y = 1; y < h; y++)//0-е нулями заполнены
  {
    for(int x = 1; x < w; x++)
    {
      matr[y][x] = (rand() % 100 / 100.0 <= chance); // 0..99 -> 0..0.99
      #ifdef DEBUG_ON
        //cout << "matr[" << i << "][" << j << "]="
        cout << matr[y][x];
      #else
        cout << ((matr[y][x])?'*':' ');
      #endif
    }
    cout << '\n';
  }
  cout << '\n';
}
```

```
inline void do_step(bool **matr, bool **matr_new, int d_l, int b_l, int h, int w)
{
  int n_cnt;
  for(int y = 1; y < h; y++)
    for(int x = 1; x < w; x++)
    {
      n_cnt = neighbours_cnt(matr, x, y, h, w); // считаем соседей
      matr_new[y][x] = (matr[y][x])?(n_cnt >= d_l):(n_cnt > b_l);
      // если клеток вокруг меньше, чем death_limit, текущая клетка умирает(еслы
была жива)
      // если клеток вокруг больше, чем birth_limit, то клетка рождается(если ее не
было)
    }
}

inline int neighbours_cnt(bool **matr, int x, int y, int h, int w)
{
  int cnt = 0, n_x, n_y; // new x,y
  for(int i = -1; i < 2; i++)
    for(int j = -1; j < 2; j++)
    {
      n_x = x + j;
      n_y = y + i;
      if ((i || j) && (matr[n_y][n_x])) cnt++;
      //вышли за границы или это непустая клетка//мы не выходим за границы
реального   массива из-за обрамления
    }
  return cnt;
}
```

Список используемых источников

1)Самарский национальный исследовательский государственный университет им. Чернышевского [Электронный ресурс]. URL: nonlin.sgu.ru/data/papers/Train/CellAutomat.pdf (дата обращения: 30.03.2019)