ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		В. А. Кузнецов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ДОПОЛНИТЕЛЬНОМ ЗАДАНИИ № 1

БАКТЕРИИ

по курсу:

ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Исходные данные задачи

Дано поле размером N на N. Значение N выбирается не более 12113 (в процессе отладки рекомендовано использовать значение до 100).

1. В течение первых D дней на поле каждый день появляется случайное число от K1 до K2 бактерии в случайных позициях поля.

2. На следующий и каждый день после появления бактерии в соседних ей ячейках (8 соседних) поля с вероятностью P+ = 5% появляется по бактерии в каждой незанятой ячейке.

3. Через T дней после появления бактерия перестает размножаться. 4. Каждый день с вероятностью P- = 2% бактерия со сроком жизни более T/2 дней умирает, занятое ей поле не освобождается.

Задачи для решения

1. Определить среднее, максимальное и минимальное количество дней, за которое бактерии заполнят все заданное поле для нескольких вариантов заданных N, K1, K2. При D = 7, T = 14. Определить вероятность того, что поле не будет заполнено для каждого набора N, K1, K2.

2. При T = ∞, D = 1, K1 = K2, P+ = 100% и P- = 0% и некоторых произвольно заданных K1, N определить позиции поля, в которых появляются бактерии на шаге 1, при которых поле будет заполнено максимально быстро.

2 Полное описание реализованных функций

2.1 is_filled

Функция is_filled проверяет, заполнено ли все поле бактериями. Возвращает true, если все клетки поля заняты бактериями, иначе false. Работа функции происходит следующим образом: проходит через каждую клетку поля и проверяет, жива ли в ней бактерия. Если находит хотя бы одну пустую клетку, возвращает false. Если все клетки заняты, возвращает true.

2.2 spread

Функция spread осуществляет размножение бактерий на поле. Ничего не возвращает. Работа функции происходит следующим образом:

1. Проходит через каждую клетку поля.
2. Если клетка занята бактерией, проверяет все 8 соседних клеток.
3. Если соседняя клетка свободна, с вероятностью p_plus в неё помещается новая бактерия.
4. Новые бактерии временно сохраняются в векторе new_bacteria.
5. После завершения обхода всех клеток новые бактерии добавляются на поле.

2.3 simulate

Функция simulate запускает симуляцию размножения бактерий до тех пор, пока всё поле не будет заполнено. Принимает следующие аргументы:

1. initial_positions - вектор начальных позиций бактерий.

Возвращает количество дней, за которые поле полностью заполнится бактериями. Работа функции происходит следующим образом:

1. Инициализирует поле бактериями на указанных начальных позициях.
2. Запускает цикл, который выполняется до тех пор, пока всё поле не будет заполнено.
3. В каждом цикле вызывается функция spread(), и счётчик дней увеличивается.
4. Когда поле заполнено, возвращает количество прошедших дней.

2.4 find_best_positions

Функция find_best_positions находит начальные позиции бактерий, при которых поле будет заполнено максимально быстро. Возвращает вектор позиций начальных бактерий, при которых поле заполняется быстрее всего. Работа функции происходит следующим образом:

1. Выполняет 1000 итераций поиска оптимальных начальных позиций.
2. В каждой итерации случайным образом генерирует начальные позиции для K1 бактерий.
3. Очищает поле перед каждой новой симуляцией.
4. Запускает симуляцию и определяет, сколько дней потребуется для полного заполнения поля.
5. Если текущая симуляция завершилась быстрее предыдущих, сохраняет эти начальные позиции как лучшие.

2.5 main

1. Инициализирует генератор случайных чисел.
2. Вызывает функцию find_best_positions(), чтобы найти лучшие начальные позиции для бактерий.
3. Выводит найденные лучшие позиции на экран.

3 Листинг программы

Листинг 1

```
#include <iostream>
#include <vector>
#include <climits>

struct Bacteria {
    bool alive;
};

const int N = 100;
const int K1 = 5;
const double p_plus = 1.0;
const double p_minus = 0.0;

std::vector<std::vector<Bacteria>> field(N, std::vector<Bacteria>(N,
{false}));

bool is_filled() {
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            if (!field[i][j].alive) return false;
    return true;
}

void spread() {
    std::vector<std::pair<int, int>> new_bacteria;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            if (field[i][j].alive) {
                for (int delta_x = -1; delta_x ≤ 1; ++delta_x) {
                    for (int delta_y = -1; delta_y ≤ 1; ++delta_y) {
                        if (delta_x == 0 && delta_y == 0) continue;
                        int new_x = i + delta_x, new_y = j + delta_y;
                        if (new_x ≥ 0 && new_x < N && new_y ≥ 0 && new_y < N
&& !field[new_x][new_y].alive) {
                            if ((rand() % 100) < (p_plus * 100)) {
                                new_bacteria.push_back({new_x, new_y});
                            }
                        }
                    }
                }
            }
        }
    }
    // Заполнение поля новыми бактериями
    for (auto &position : new_bacteria)
        field[position.first][position.second].alive = true;
}

int simulate(std::vector<std::pair<int, int>> initial_positions) {
    for (auto &position : initial_positions)
        field[position.first][position.second].alive = true;
    int days_passed = 0;
    while (!is_filled()) {
        spread();
```

Продолжение листинга 1

```
        days_passed++;
    }
    return days_passed;
}

std::vector<std::pair<int, int>> find_best_positions() {
    std::vector<std::pair<int, int>> best_positions;
    int min_days_passed = INT_MAX;

    // Генерация случайных начальных позиций
    for (int _ = 0; _ < 1000; _++) {
        std::vector<std::pair<int, int>> initial_positions;
        while (initial_positions.size() < K1) {
            int x = rand() % N;
            int y = rand() % N;
            initial_positions.push_back({x, y});
        }

        for (int x = 0; x < N; ++x)
            for (int y = 0; y < N; ++y)
                field[x][y].alive = false;

        int days_passed = simulate(initial_positions);
        if (days_passed < min_days_passed) {
            min_days_passed = days_passed;
            best_positions = initial_positions;
        }
    }

    return best_positions;
}

int main() {
    srand(time(nullptr));
    std::vector<std::pair<int, int>> best_positions = find_best_positions();
    std::cout
            << "Позиции поля изначальных бактерий, при которых поле будет
заполнено максимально быстро:"
            << std::endl;
    for (auto &positions : best_positions)
        std::cout << "(" << positions.first << ", " << positions.second << ")"
<< std::endl;
    return 0;
}
```

4 Результаты тестирования программы

```
Позиции поля изначальных бактерий, при которых поле будет заполнено максимально быстро:
(21, 68)
(86, 66)
(99, 48)
(16, 21)
(76, 21)

Process finished with exit code 0
```

Рисунок 1 - N=100, K1=5

```
Позиции поля изначальных бактерий, при которых поле будет заполнено максимально быстро:
(101, 108)
(8, 158)
(166, 39)
(190, 74)
(48, 111)
(174, 156)
(67, 179)
(179, 54)
(50, 53)
(55, 31)

Process finished with exit code 0
```

Рисунок 2 - N=200, K1=10

```
Позиции поля изначальных бактерий, при которых поле будет заполнено максимально быстро:
(49, 24)
(2, 25)

Process finished with exit code 0
```

Рисунок 3 - N=50, K1=2