Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет систем управления и робототехники

Алгоритмы и структуры данных

Отчёт по лабораторной работе №5 (1401. Игроки)

Преподаватель: Тропченко А. А.

Выполнил: Марухленко Д. С.

Группа: R3235

Санкт Петербург, 2021г.

1. Цель работы

Решить задачу №1401на платформе Timus Online Judge
https://acm.timus.ru/problem.aspx?space=1&num=1401

2. Задача

Условие

Известно, что господин Чичиков зарабатывал свой капитал и таким способом: он спорил со всякими недотёпами, что сможет доказать, что квадратную доску размера 512 × 512 нельзя замостить следующими фигурами:

X XX X XX
XX X XX X
и всегда выигрывал. Однако один из недотёп оказался не так уж глуп, и сказал, что сможет замостить такими фигурами доску размера 512 × 512 без правой верхней клетки. Чичиков, не подумав, ляпнул, что он вообще может любую доску размера $2^n \times 2^n$ без одной произвольной клетки замостить такими фигурами. Слово за слово, они поспорили. Чичиков чувствует, что сам он не докажет свою правоту. Помогите же ему!

Ограничение времени: 3.0 секунды
Ограничение памяти: 64 МБ

Исходные данные

В первой строке записано целое число n ($1 \le n \le 9$). Во второй строке через пробел даны два целых числа x, y: координаты «выколотой» клетки доски ($1 \le x, y \le 2^n$), x — номер строки, y — номер столбца. Левый верхний угол доски имеет координаты (1, 1).

Результат

Ваша программа должна выдать 2n строчек по 2n чисел в каждой строке. На месте выбитой клетки должно стоять число 0. На месте остальных клеток должны стоять числа от 1 до $(2^{2n} - 1) / 3$ — номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, выведите «−1».

Пример

Исходные данные	Результат
2 1 1	0 1 3 3 1 1 4 3 2 4 4 5 2 2 5 5

3. Материалы работы

3.1. Объяснение алгоритма

После считывания проверяется условие, необходимое для требуемого размещения (оно выполнится при любых входных данных, но всё же проверка добавлена, т.к. по заданию «Если же такую доску нельзя покрыть фигурами, выведите «−1».»).

Поле представляется в виде двумерного массива, в который записываются числа по следующему алгоритму: Данное квадратное поле делится на 4 новых поля поменьше. На стыке тех полей, где не должно остаться «дырки» размещается фигура. Далее эта операция продолжается до тех пор, пока размер квадратов не станет 2*2 (клетка, уже занятая размещением предыдущей фигуры, считается локальной «дыркой» в квадрате, полученном делением). В конце выводится результат размещения.

Пример работы программы с входными данными 3 3 3

```
 3  3  4  4  8  8  9  9
 3  2  2  4  8  7  7  9
 6  2  0  5 11 11  7 10
 6  6  5  5  1 11 10 10
18 18 19  1  1 13 14 14
18 17 19 19 13 13 12 14
21 17 17 20 16 12 12 15
21 21 20 20 16 16 15 15
```

3.2. Код программы.

```
1.  #include <iostream>
2.  #include <cmath>
3.
4.  using namespace std;
5.  int table[512][512];
6.
7.  void paint(int x, int y, int i, int j, int n) {
8.      static int c;
9.      ++c;
10.     n = n / 2;
11.     if ((x - j) >= n && (y - i) >= n) {
12.         table[i + n - 1][j + n - 1] = c;
13.         table[i + n][j + n - 1] = c;
14.         table[i + n - 1][j + n] = c;
15.         if (n>1){
16.         paint(j + n - 1, i + n - 1, i, j, n);
17.         paint(j + n, i + n - 1, i, j + n, n);
18.         paint(x, y, i + n, j + n, n);
19.         paint(j + n - 1, i + n, i + n, j, n);
20.         }
21.     } else if ((x - j) >= n && (y - i) < n) {
22.         table[i + n - 1][j + n - 1] = c;
23.         table[i + n][j + n - 1] = c;
24.         table[i + n][j + n] = c;
25.         if (n>1){
26.         paint(j + n - 1, i + n - 1, i, j, n);
27.         paint(x, y, i, j + n, n);
28.         paint(j + n, i + n, i + n, j + n, n);
29.         paint(j + n - 1, i + n, i + n, j, n);
30.         }
31.     } else  if ((x - j) < n && (y - i) < n) {
32.         table[i + n - 1][j + n] = c;
33.         table[i + n][j + n - 1] = c;
34.         table[i + n][j + n] = c;
35.         if (n>1){
```

```
36.         paint(x, y, i, j, n);
37.         paint(j + n, i + n - 1, i, j + n, n);
38.         paint(j + n, i + n, i + n, j + n, n);
39.         paint(j + n - 1, i + n, i + n, j, n);
40.         }
41.     } else if ((x - j) < n && (y - i) >= n) {
42.         table[i + n - 1][j + n - 1] = c;
43.         table[i + n][j + n] = c;
44.         table[i + n - 1][j + n] = c;
45.         if (n>1){
46.         paint(j + n - 1, i + n - 1, i, j, n);
47.         paint(j + n, i + n - 1, i, j + n, n);
48.         paint(j + n, i + n, i + n, j + n, n);
49.         paint(x, y, i + n, j, n);
50.         }
51.     }
52.     return;
53. }
54.
55. int main() {
56.     int n, x, y;
57.     cin >> n >> y >> x;
58.     n = 1 << n;
59.     if ((int) (pow(n, 2) - 1) % 3 != 0) {
60.         cout << -1;
61.         return 0;
62.     }
63.
64.
65.     --x;
66.     --y;
67.     paint(x, y, 0, 0, n);
68.
69.     for (int i = 0; i < n; ++i) {
70.         for (int j = 0; j < n; ++j)
71.             cout << table[i][j] << " ";
72.         cout << endl;
73.     }
74.     return 0;
75. }
```

4. Результат выполнения и ссылка на репозиторий GitHub

9304679	12:41:29 5 апр 2021	Daniil Marukhlenko	1401	G++ 9.2 x64	Accepted		0.015	1 412 КБ

https://github.com/japersik/algorithms_and_data_structures/

5. Вывод

Работа выполнена, задача решена, решение удовлетворят требованиям.