Национальный исследовательский университет ИТМО
(Университет ИТМО)

Факультет систем управления и робототехники

Дисциплина: Алгоритмы и структуры данных
Отчет по практической работе (1401 задача).

Студент:
Евстигнеев Дмитрий
Группа: *R3242*
Преподаватель:
Тропченко Андрей Александрович

Санкт-Петербург
2021

Цель: написать программу для решения задачи №1401 на сайте Timus Online

Задача:

1401. Игроки

Ограничение времени: 2.0 секунды

Ограничение памяти: 64 МБ

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!

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 2^n строчек по 2^n 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

Принято системой (JUDGE_ID: 231802FR):

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
9275452	02:16:03 21 мар 2021	Dmitry Evstigneev	1401. Игроки	G++ 9.2 x64	Accepted		0.015	1 604 КБ

Решение на языке C++:

```
#include <iostream>

using namespace std;
```

```
class Gamers {
private:
	int N;
	int size;
	int** matrix;

	void add_figure(int i, int j) {
		int tmp_i = i % 2;
		int tmp_j = j % 2;

		if (tmp_i == 1 && tmp_j == 1) {
			matrix[i - 1][j - 1] = matrix[i - 1][j] = matrix[i][j - 1] = N++;

		}
		else if (tmp_i == 0 && tmp_j == 0) {
			matrix[i + 1][j + 1] = matrix[i + 1][j] = matrix[i][j + 1] = N++;

		}
		else if (tmp_i == 0 && tmp_j == 1) {
			matrix[i + 1][j] = matrix[i][j - 1] = matrix[i + 1][j - 1] = N++;

		}
		else if (tmp_i == 1 && tmp_j == 0) {
			matrix[i - 1][j] = matrix[i][j + 1] = matrix[i - 1][j + 1] = N++;

		}
	}

	void main_solve(int rows_start, int cols_start, int i, int j, int size) {
		if (size == 2) {
			add_figure(rows_start + i % 2, cols_start + j % 2);
			return;
		}

		int new_size = size / 2;

		if (i >= new_size && j >= new_size) {
			main_solve(rows_start + new_size, cols_start + new_size, i -
new_size, j - new_size, new_size);

			matrix[rows_start + new_size - 1][cols_start + new_size - 1] =
				matrix[rows_start + new_size][cols_start + new_size - 1] =
				matrix[rows_start + new_size - 1][cols_start + new_size] =
N++;

			main_solve(rows_start, cols_start + new_size, new_size - 1, 0,
new_size);
			main_solve(rows_start, cols_start, new_size - 1, new_size - 1,
new_size);
			main_solve(rows_start + new_size, cols_start, 0, new_size - 1,
new_size);

		}
		else if (i < new_size && j < new_size) {
			main_solve(rows_start, cols_start, i, j, new_size);

			matrix[rows_start + new_size][cols_start + new_size] =
				matrix[rows_start + new_size][cols_start + new_size - 1] =
				matrix[rows_start + new_size - 1][cols_start + new_size] =
N++;

			main_solve(rows_start + new_size, cols_start + new_size, 0, 0,
new_size);
```

```
                        main_solve(rows_start, cols_start + new_size, new_size - 1, 0,
new_size);
                        main_solve(rows_start + new_size, cols_start, 0, new_size - 1,
new_size);

                }
                else if (i < new_size && j >= new_size) {
                        main_solve(rows_start, cols_start + new_size, i, j - new_size,
new_size);

                        matrix[rows_start + new_size][cols_start + new_size] =
                                matrix[rows_start + new_size][cols_start + new_size - 1] =
                                matrix[rows_start + new_size - 1][cols_start + new_size - 1] =
N++;

                        main_solve(rows_start, cols_start, new_size - 1, new_size - 1,
new_size);
                        main_solve(rows_start + new_size, cols_start + new_size, 0, 0,
new_size);
                        main_solve(rows_start + new_size, cols_start, 0, new_size - 1,
new_size);

                }
                else {
                        main_solve(rows_start + new_size, cols_start, i - new_size, j,
new_size);

                        matrix[rows_start + new_size][cols_start + new_size] =
                                matrix[rows_start + new_size - 1][cols_start + new_size] =
                                matrix[rows_start + new_size - 1][cols_start + new_size - 1] =
N++;

                        main_solve(rows_start, cols_start, new_size - 1, new_size - 1,
new_size);
                        main_solve(rows_start + new_size, cols_start + new_size, 0, 0,
new_size);
                        main_solve(rows_start, cols_start + new_size, new_size - 1, 0,
new_size);
                }
        }

public:
        Gamers(int** matrix, int size) {
                this->matrix = matrix;
                this->size = size;
                N = 1;
        }

        void solve(int i, int j) {
                main_solve(0, 0, i, j, size);
        }
};

int main() {
        int degree = 0, i = 0, j = 0;

        cin >> degree >> i >> j;

        int count = 1 << degree;
        int** matrix = new int*[count];


        for (int i = 0; i < count; i++) {
                matrix[i] = new int[count];
                for (int j = 0; j < count; j++)
```

```
                matrix[i][j] = 0;
        }

        Gamers gamers(matrix, count);

        gamers.solve(i - 1, j - 1);

        for (int i = 0; i < count; i++) {
                for (int j = 0; j < count; j++)
                        cout << matrix[i][j] << " ";
                cout << endl;
        }

        return 0;
}
```

Суть алгоритма:

Задача сводится к решению методом динамического программирования. Основная суть в том, что мы делим каждый раз нашу задачу на 4. То есть делим поле на 4 части, пока размер одной такой части не будет 2 x 2.

Сначала доходим до части, где стоит ноль, и зарисовываем остальные три точки. После (когда возвращаемся из квадрата 2 x 2 в исходный квадрат 4 x 4), зарисовываем три точки, в зависимости от того, где находился наш квадрат 2 x 2. Далее зарисовываем все остальные.

Примеры работы программы:

```
2
1
1
0 1 4 4
1 1 2 4
5 2 2 3
5 5 3 3

Process returned 0 (0x0)   execution time : 10.316 s
Press any key to continue.
```

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

Process returned 0 (0x0)   execution time : 2.481 s
Press any key to continue.
```