МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №5

По дисциплине: «***Алгоритмы и структуры данных***»

Тема: «***Поиск с возвратом (Backtracking)***»

**Выполнил:** Антонюк Н.А.

**Группа:** ПО-11

**Проверила:** Глущенко Т.А.

Брест 2023

***Задание.***

Написать программу решения задачи поиска с возвратом на выбор:

• задача о лабиринте ( входные данные – булева матрица размером 6×9

или 9×9, см. материал лекции. Предусмотреть ситуацию отсутствия выхода из лабиринта).

***#include <iostream>***

***using namespace std;***

***const int ROWS = 9;***

***const int COLS = 9;***

***int maze[ROWS][COLS] = {***

***{1, 1, 1, 1, 1, 1, 1, 1, 1},***

***{1, 0, 1, 0, 0, 0, 1, 0, 1},***

***{1, 0, 1, 1, 1, 0, 1, 0, 1},***

***{1, 0, 0, 0, 1, 0, 1, 1, 1},***

***{1, 1, 1, 0, 1, 0, 0, 0, 1},***

***{1, 0, 1, 0, 1, 1, 1, 0, 1},***

***{1, 0, 1, 0, 0, 0, 0, 0, 1},***

***{1, 0, 1, 1, 1, 1, 1, 0, 1},***

***{1, 1, 1, 1, 1, 1, 1, 1, 1}***

***};***

***int solution[ROWS][COLS];***

***bool FindPath(int row, int col) {***

***// Проверка, достигли ли мы выхода из лабиринта***

***if (row == ROWS - 2 && col == COLS - 2) {***

***solution[row][col] = 1; // Помечаем последнюю ячейку в решении***

***return true;***

***}***

***// Проверка, что мы находимся в пределах лабиринта и текущая ячейка не стена***

***if (row >= 0 && col >= 0 && row < ROWS && col < COLS && maze[row][col] == 0 && solution[row][col] == 0) {***

***solution[row][col] = 1; // Помечаем текущую ячейку в решении***

***// Рекурсивный поиск вверх, вниз, влево и вправо***

***if (FindPath(row - 1, col) || FindPath(row + 1, col) || FindPath(row, col - 1) || FindPath(row, col + 1)) {***

***return true;***

***}***

***// Если не удалось найти путь, отмечаем текущую ячейку как недоступную***

***solution[row][col] = 0;***

***}***

***return false;***

***}***

***void PrintSolution() {***

***for (int i = 0; i < ROWS; i++) {***

***for (int j = 0; j < COLS; j++) {***

***cout << solution[i][j] << " ";***

***}***

***cout << endl;***

***}***

***}***

***int main() {***

***setlocale(LC\_ALL, "rus");***

***if (FindPath(1, 1)) {***

***cout << "Решение найдено:" << endl;***

***PrintSolution();***

***}***

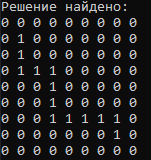
***else {***

***cout << "Выхода из лабиринта нет." << endl;***

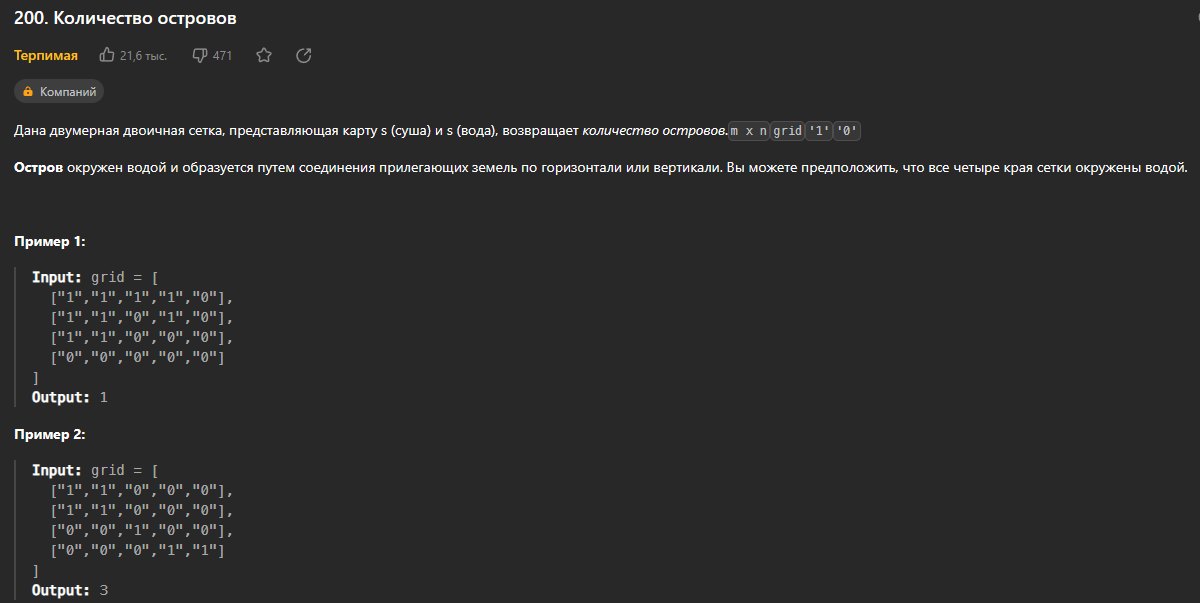
***}***

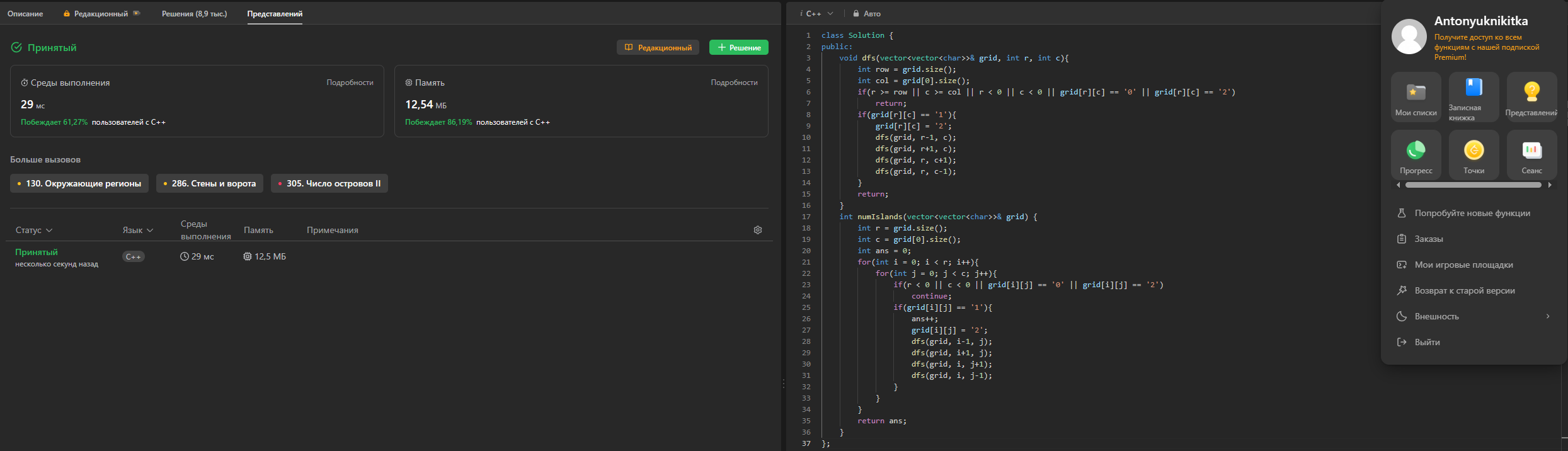
***return 0;***

}



1. **Задача о количестве островов:**





**Алгоритм задачи:**

class Solution {

public:

    void dfs(vector<vector<char>>& grid, int r, int c){

        int row = grid.size();

        int col = grid[0].size();

        if(r >= row || c >= col || r < 0 || c < 0 || grid[r][c] == '0' || grid[r][c] == '2')

            return;

        if(grid[r][c] == '1'){

            grid[r][c] = '2';

            dfs(grid, r-1, c);

            dfs(grid, r+1, c);

            dfs(grid, r, c+1);

            dfs(grid, r, c-1);

        }

        return;

    }

    int numIslands(vector<vector<char>>& grid) {

        int r = grid.size();

        int c = grid[0].size();

        int ans = 0;

        for(int i = 0; i < r; i++){

            for(int j = 0; j < c; j++){

                if(r < 0 || c < 0 || grid[i][j] == '0' || grid[i][j] == '2')

                    continue;

                if(grid[i][j] == '1'){

                    ans++;

                    grid[i][j] = '2';

                    dfs(grid, i-1, j);

                    dfs(grid, i+1, j);

                    dfs(grid, i, j+1);

                    dfs(grid, i, j-1);

                }

            }

        }

        return ans;

    }

};

**3. Описать алгоритм решения задачи о *8 ферзях*.**

Задача о 8 ферзях — это задача размещения восьми ферзей на шахматной доске 8x8 так, чтобы ни один из ферзей не находился под атакой другого. Атака включает в себя вертикальные, горизонтальные и диагональные направления. Задача известна как NP-полная и может быть решена с использованием метода поиска с возвратом (backtracking). Вот алгоритм для решения задачи о 8 ферзях:

Инициализация:

Создать шахматную доску размером 8x8.

Установить счетчик ферзей в 0.

Поместить ферзя:

Если счетчик ферзей равен 8, то все ферзи успешно размещены, задача решена, завершить алгоритм.

Для каждой клетки в текущей строке:

Если клетка свободна (не находится под атакой других ферзей), то поместить ферзя в эту клетку.

Увеличить счетчик ферзей.

Рекурсивно вызвать шаг 2 для следующей строки.

Если вызов вернул true, то ферзи успешно размещены, вернуть true.

Если вызов вернул false, вернуться к предыдущему состоянию, убрав ферзя из текущей клетки.

Проверка безопасности:

Для проверки, находится ли ферзь под атакой, нужно проверить все вертикальные, горизонтальные и диагональные направления от текущей клетки.

Если ферзь находится под атакой, вернуть false.

Если все ферзи размещены безопасно, вернуть true.

Возврат:

Если текущая строка стала больше 8 (выход за границы доски), вернуть false.

Если текущая строка еще не достигла 8, и ни одна из клеток текущей строки не подходит, вернуть false.

Если необходимо вернуться на предыдущий уровень (например, если текущая строка завершилась безуспешно), вернуться и попробовать следующую клетку в предыдущей строке.

**4. Письменно подробно ответить на вопросы.**

***1. Временная сложность полного перебора (brute force):***

- Для задачи о 8 ферзях: В данном случае, если применять полный перебор для каждой клетки на доске, время выполнения может быть экспоненциальным, близким к O(N!). Это связано с тем, что для каждой из 64 клеток на доске нужно рассмотреть возможные комбинации размещения 8 ферзей.

***2. Поиски на графах в данных задачах:***

- В задачах о лабиринте, 8 ферзях и количестве островов применяется поиск с возвратом (backtracking), который можно рассматривать как форму поиска на графе. Граф в этих случаях представляет возможные состояния или шаги в решении задачи.

***3. Временная сложность поиска в глубину:***

- Временная сложность поиска в глубину (DFS) зависит от размера графа и числа ребер. В худшем случае она может быть O(V + E), где V - количество вершин, E - количество ребер.

***4. Временная сложность задачи о количестве островов:***

- Предполагая, что задача о количестве островов решается с использованием поиска в глубину (DFS), временная сложность может быть O(M \* N), где M - количество строк, N - количество столбцов в матрице.

***5. Выбор варианта выхода в задаче о лабиринте:***

- Если в задаче о лабиринте есть несколько выходов, выбор варианта выхода может определяться различными критериями, такими как кратчайший путь к выходу, определенные правила (например, всегда идти влево при наличии выбора).

***6. Альтернативная формулировка задачи о 8 ферзях и количество решений:***

- Альтернативная формулировка: Расставить 8 ферзей на шахматной доске 8x8 так, чтобы ни один из ферзей не находился в атакованной другими ферзями клетке. Задача имеет 92 уникальных решения.