ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу

Старцев Иван 467585

Академическая группа №J3110

Принято

Практик, Владислав Вершинин

Санкт-Петербург

2024

1. Введение

Цель работы: решить проблему с leetcode используя Динамическое программирование (далее ДП).

Задачи:

* Выбрать проблему на leetcode
* Понять как и где должно использоваться ДП
* Написать красивый и рабочий код

1. Теоретическая подготовка

Выбранной проблемой стала проблема Dungeon-game. Для ее оптимального решения понадобится ДП и двумерный массив vector<vector<int>>.

1. Реализация

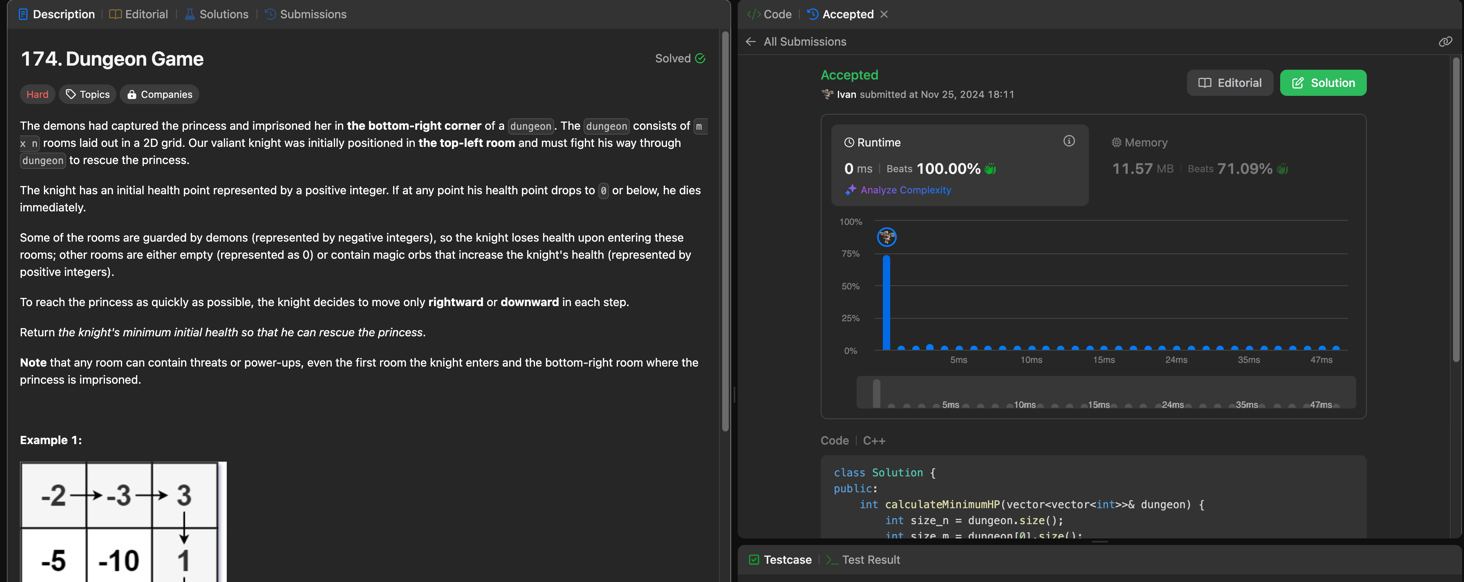
При решении данной задачи с использованием ДП сразу вспомнилась задача из ЕГЭ по информатике, но тут необходим немного другой подход. Если считать минимальное кол-во очков здоровья(далее - ХП) из левой верхней в правую нижнюю, то мы получим такое ХП при котором мы точно сможем выжить в последней клетке, но не на всем маршруте. Поэтому пришла идея считать ХП обратным проходом, где мы будем идти из левой нижней в правую верхнюю клетку считая минимальное ХП в каждой клетке. Таким образом мы разбиваем задачу на меньшие подзадачи для подсчета

1. Экспериментальная часть

Аксиоматика времени данного алгоритма- O(N\*M).

Аксиоматика памяти данного алгоритма – O(N\*M).

Работоспособность алгоритма доказана на изображении №1



Изображение №1 – Вердикт leetcode

1. Заключение

В ходе выполнения работы мною был реализован алгоритм решающий проблему с сайта leetcode.Цель работы была достигнута путем использования Динамического программирования. Дальнейшего развития данного решения я не представляю, так как это очень очевидное и оптимальное решение.

ПРИЛОЖЕНИЕ А

Листинг кода файла main.cpp

#include <iostream>

using namespace std;

class Solution {

public:

int calculateMinimumHP(vector<vector<int> > &dungeon) {

int size\_n = dungeon.size();

int size\_m = dungeon[0].size();

// будем считать необходимое хп обратным проходом, короче как в егэ

vector<vector<int> > dp(size\_n + 1, vector<int>(size\_m + 1, INT\_MAX));

dp[size\_n - 1][size\_m] = 1;

dp[size\_n][size\_m - 1] = 1;

for (int i = size\_n - 1; i >= 0; i--) {

for (int j = size\_m - 1; j >= 0; j--) {

int hp = min(dp[i + 1][j], dp[i][j + 1]) - dungeon[i][j];

if (hp <= 0) {

dp[i][j] = 1;

} else {

dp[i][j] = hp;

}

}

}

return dp[0][0];

}

};