ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу

Ларкин Григорий

Академическая группа №J3113

Принято

Должность, звание Фамилия Имя преподавателя

Санкт-Петербург

2024

1. Введение

В этом разделе необходимо описать цель работы и задачи, которые вам предстоит решить. Важно, цель у работы всегда одна, а задач может быть несколько.

1. Реализация

Задача заключается в определении, соответствует ли заданная строка s заданному шаблону p, где шаблон может содержать специальные символы:

. (точка) — соответствует любому одиночному символу.

\* (звездочка) — соответствует нулю или более повторениям предыдущего символа.

Алгоритм (динамическое программирование):

Создание таблицы mTable: создается двумерная булева таблица mTable размером (длина(s) + 1) x (длина(p) + 1). mTable[i][j] будет true, если подстрока s, начинающаяся с позиции i, соответствует подстроке p, начинающаяся с позиции j.

Базовый случай: mTable[длина(s)][длина(p)] = true (пустая строка соответствует пустому шаблону).

Заполнение таблицы: Алгоритм итерируется по таблице справа налево и снизу вверх, заполняя ячейки на основе следующих правил:

p[j] — обычный символ: mTable[i][j] = true, если s[i] == p[j] и mTable[i+1][j+1] == true.

p[j] — '.': mTable[i][j] = true, если mTable[i+1][j+1] == true.

p[j] — '\*': mTable[i][j] = true, если выполняется одно из условий:

Ноль повторений: mTable[i][j+2] == true.

Одно или более повторений: s[i] == p[j-1] (или p[j-1] — '.') и mTable[i+1][j] == true.

Результат: mTable[0][0] содержит окончательный результат — true, если вся строка s соответствует шаблону p, и false в противном случае.

1. Динамической программирование

Динамическое программирование позволяет избежать повторных вычислений. В данной задаче могут возникать перекрывающиеся подзадачи (например, проверка соответствия подстрок). Таблица mTable хранит результаты решений подзадач, и алгоритм использует эти результаты для решения более крупных задач. Это значительно повышает эффективность по сравнению с рекурсивным решением, которое может экспоненциально увеличивать время выполнения из-за повторных вычислений. Динамическое программирование снижает временную сложность до O(длина(s) \* длина(p)).

1. Листинг кода

class Solution {

public:

bool isMatch(string s, string p) {

int textLen = s.size(); // O(1) - получение размера строки

int patternLen = p.size(); // O(1) - получение размера строки

// O(textLen \* patternLen) - память под двумерный вектор

vector<vector<bool>> mTable(textLen + 1, vector<bool>(patternLen + 1, false));

mTable[textLen][patternLen] = true; // O(1) - доступ к элементу вектора

bool charMatch = false; // O(1) - память под bool переменную

bool starP = false; // O(1) - память под bool переменную

bool dotP = false; // O(1) - память под bool переменную

char patternChar; // O(1) - память под char переменную

// O(patternLen) - внешний цикл

for (int patternIndex = patternLen - 1; patternIndex >= 0; patternIndex--) {

patternChar = p[patternIndex]; // O(1) - доступ к элементу строки

if (patternChar == '\*') { // O(1) - сравнение символов

starP = true; // O(1) - присваивание

continue; // O(1) - переход к следующей итерации

}

dotP = (patternChar == '.'); // O(1) - сравнение символов и присваивание

// O(textLen) - внутренний цикл

for (int textIndex = textLen; textIndex >= 0; textIndex--) {

if (textIndex < textLen) { // O(1) - сравнение чисел

charMatch = (dotP || (patternChar == s[textIndex])); // O(1) - сравнение символов, логические операции, присваивание

} else {

charMatch = false; // O(1) - присваивание

}

if (starP) { // O(1) - проверка условия

if (mTable[textIndex][patternIndex + 2]) { // O(1) - доступ к элементу вектора

mTable[textIndex][patternIndex] = true; // O(1) - доступ к элементу вектора, присваивание

} else if (charMatch) { // O(1) - проверка условия

if (textIndex < textLen) { // O(1) - сравнение чисел

mTable[textIndex][patternIndex] = mTable[textIndex + 1][patternIndex]; // O(1) - доступ к элементам вектора, присваивание

}

}

} else { // O(1) - проверка условия

if (charMatch) { // O(1) - проверка условия

if (textIndex < textLen) { // O(1) - сравнение чисел

mTable[textIndex][patternIndex] = mTable[textIndex + 1][patternIndex + 1]; // O(1) - доступ к элементам вектора, присваивание

}

}

}

}

starP = false; // O(1) - присваивание

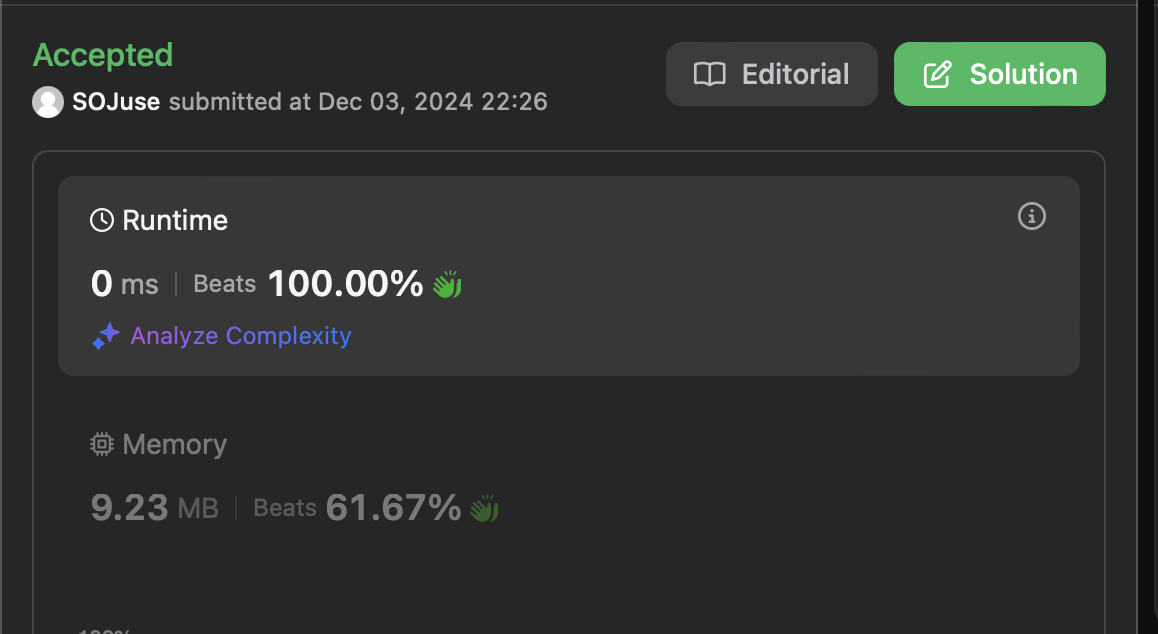
}

return mTable[0][0]; // O(1) - доступ к элементу вектора

}

};

1. Фото



Изображение 1: Скрин пройденных тестов