ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Задача на динамическое программирование (Wildcard Matching)»

Выполнил работу

Исмагилова Юлия

Академическая группа №J3110

Принято

практик, Вершинин Владислав

Санкт-Петербург

2024

Содержание

[Содержание 2](#_Toc187271621)

[Введение 3](#_Toc187271622)

[Теоретическая подготовка 4](#_Toc187271623)

[Реализация 5](#_Toc187271624)

[Экспериментальная часть 6](#_Toc187271625)

[Заключение 7](#_Toc187271626)

[Приложения 8](#_Toc187271627)

Введение

**Цель работы:** разработать алгоритм на языке С++ для решения следующего технического задания: "Для входной строки (s) и шаблона (p) реализуйте сопоставление шаблонов с подстановочными знаками с поддержкой '?' и '\*’, где: '?' - соответствует любому отдельному символу, а '\*' - соответствует любой последовательности символов (включая пустую последовательность). Сопоставление должно охватывать всю входную строку (не частично)." Алгоритм должен использовать для решения задачи динамическое программирование.

**Задачи:**

1. Реализовать функцию, которая принимает входную строку и шаблон и возвращает булево значение, указывающее, соответствует ли строка шаблону.
2. Реализовать функцию, которая принимает входную строку и шаблон и возвращает булево значение, указывающее, соответствует ли строка шаблону.

Теоретическая подготовка

**Типы данных**

1. <bool>: используется для хранения логических значений.
2. vector<vector<bool>>: двумерный массив (матрица) логических значений, используется для хранения промежуточных результатов динамического программирования.

**Алгоритмы**

1. Динамическое программирование: является основным подходом, используемым в коде. Необходимо для создания матрицы для хранения подстрок и шаблонов.

**Динамическое программирование**

Динамическое программирование (DP) — это метод решения задач, который использует предыдущие результаты для вычисления новых результатов. В данном случае DP можно использовать для проверки возможности разделения массива на две части с равной суммой.

Реализация

**Инициализация:**

Определяются длины строк s и p. (рис. 1)

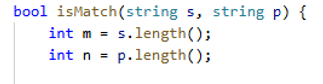


Рисунок 1 – пример использования в коде

**Обработка шаблонов, начинающихся с '\*':**

Происходит заполнение первой строки матрицы для шаблонов, которые начинаются с '\*' (рис. 2)

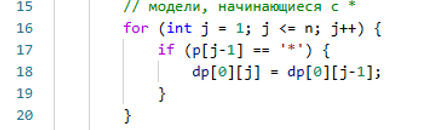


Рисунок 2 – пример использования в коде

**Заполнение матрицы ДП:**

Происходит заполнение матрицы для каждого символа строки s и шаблона p. (рис. 3)

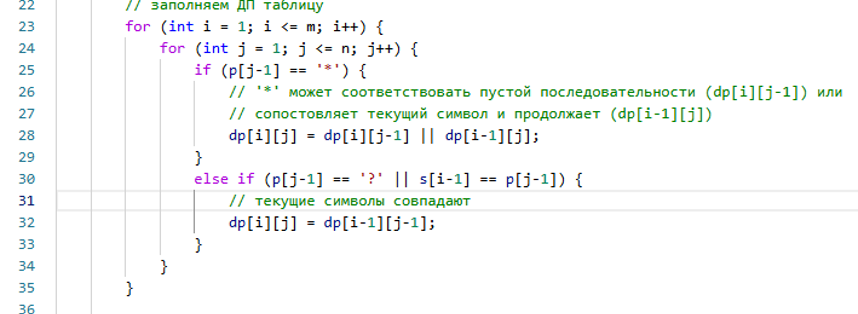


Рисунок 3 – пример использования в коде

Экспериментальная часть

**Подсчёт памяти:**

Основной потребитель памяти в данном случае — это матрица, поэтому занимаемая память будет зависеть от её размеров. Для матрицы с длиной строки 5 символов и шаблона в 4 символа примерный подсчёт памяти будет рассчитываться по следующей формуле: (m + 1) × (n + 1) × размер одного элемента (1 байт), и будет равен: 6 × 5 = 30 байт.

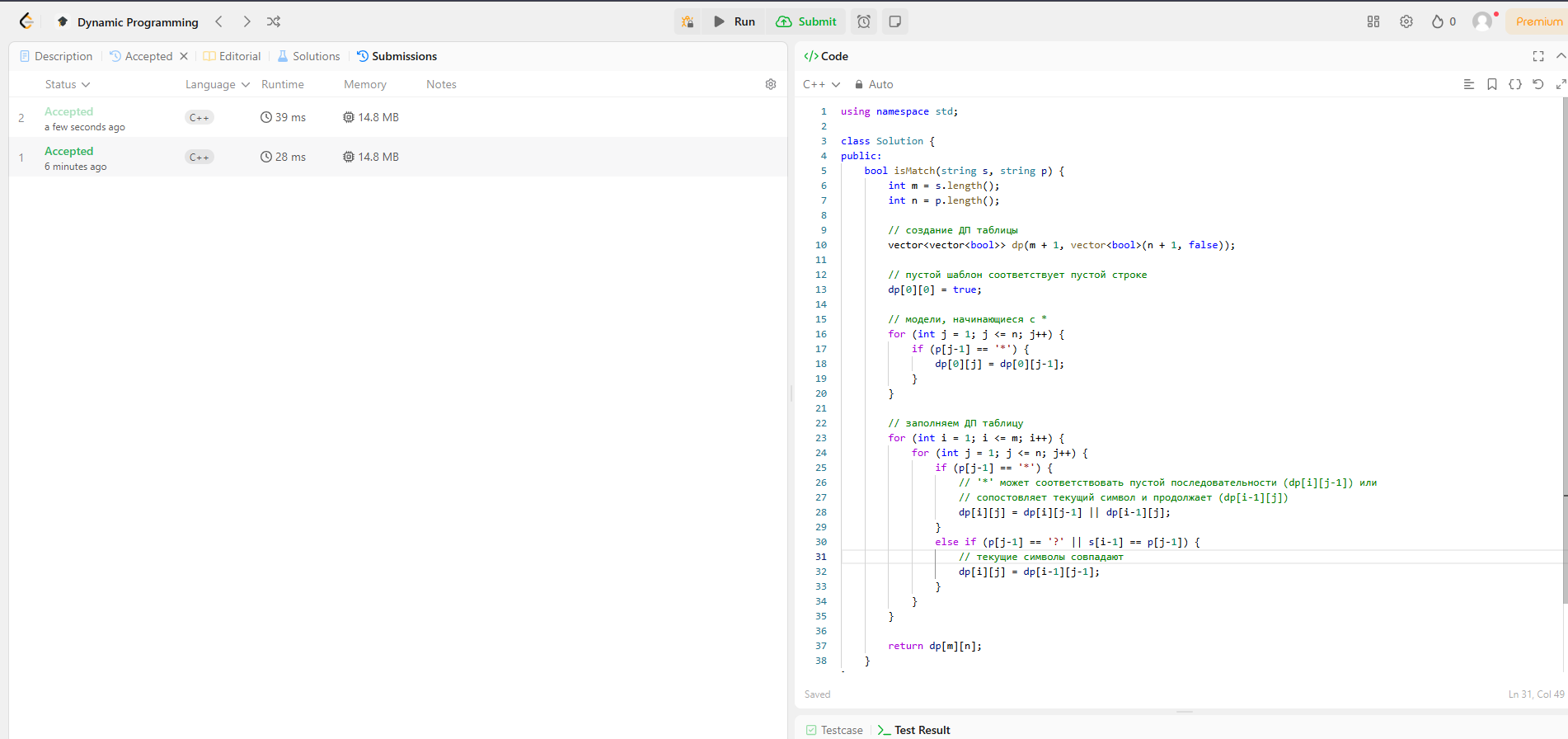
**Подсчёт асимптотики:**

1. Временная сложность: O(m×n).
2. Пространственная сложность: O(m×n).

**Почему необходимо динамическое программирование?**

Динамическое программирование позволяет эффективно решать задачу сопоставления шаблонов с подстановочными знаками, используя рекурсивную природу задачи, перекрывающиеся подзадачи и оптимальные подструктуры. Это делает алгоритм значительно быстрее и эффективнее по сравнению с простым рекурсивным подходом.

**Скриншот с leetcode:**

****

Заключение

В ходе выполнения работы мною был реализован алгоритм сопоставления строки с шаблоном с использованием динамического программирования. Цель работы была достигнута путём тестирования на различных строках и шаблонах, включая случаи с использованием специальных символов '?' и '\*'. Полученные результаты подтвердили корректность работы алгоритма и совпадают с теоретическими оценками сложности.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения уменьшения затрат использования памяти, а также рассмотреть параллельные версии алгоритма для работы с большими строками и шаблонами.

Приложения

ПРИЛОЖЕНИЕ А

Листинг кода:

