ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Название лабораторной работы (Задача на жадный алгоритм на leetcode)»

Выполнила работу

Смирнова Анна

Академическая группа №J3112

Принято

Преподаватель: Дунаев Максим Владимирович

Санкт-Петербург

2024

1. Введение

Сопоставление подстановочных карт (wildcard matching).

Цель: Учитывая входную строку и шаблон, реализовать сопоставление шаблонов с подстановочными знаками с поддержкой, где s = “\*”, p = “?”.

Задача: задача должна быть решена с помощью жадного алгоритма.

1. Теоретическая подготовка

- Знание языка программирования C++

- Понимание жадного алгоритма для решения задачи

3. Процесс этапа выполнения работы:

1. Постановка задачи:

Необходимо реализовать функцию, которая проверяет соответствует ли строка s шаблону p.

1. Анализ поставленной задачи:

Для решения необходимо использовать жадный алгоритм.

Принципы жадного алгоритма в коде:

* Сразу же обрабатывает символы, если они совпадают
* Когда встречается символ “\*”, алгоритм запоминает его позицию и текущее положение строки

1. Проектирование алгоритма:

bool isMatch(string s, string p) {

    int len\_s = s.length();

    int len\_p = p.length();

    int index\_s = 0;

int index\_p = 0;

    int last\_index\_star = -1;

    int match\_index = 0;

    while (index\_s < len\_s) {

        // Если символы совпадают или шаблон - '?'

        if (index\_p < len\_p && (s[index\_s] == p[index\_p] || p[index\_p] == '?')) {

            index\_s++;

            index\_p++;

        }

        // Если встречаем '\*', запоминаем его индекс

        else if (index\_p < len\_p && p[index\_p] == '\*') {

            last\_index\_star = index\_p; // Запоминаем позицию '\*'

            match\_index = index\_s; // Запоминаем текущее положение строки

            index\_p++; // Переходим к следующему символу шаблона

        }

        // Если символы не совпадают

        else if (last\_index\_star != -1) {

            // Возвращаемся к позиции после последнего '\*'

            index\_p = last\_index\_star + 1; // Переходим к следующему символу после '\*'

            index\_s = ++match\_index; // Увеличиваем индекс строки

        }

else {

            return false;

        }

    }

    // Проверяем оставшуюся часть шаблона

    while (index\_p < len\_p && p[index\_p] == '\*') {

        index\_p++;

    }

    return index\_p == len\_p;

};

4. Экспериментальная часть

Подсчёт по памяти:

Переменные: 6 \* 4 байта = 24 байта

Строки: 2 \* 24 байта = 48 байт

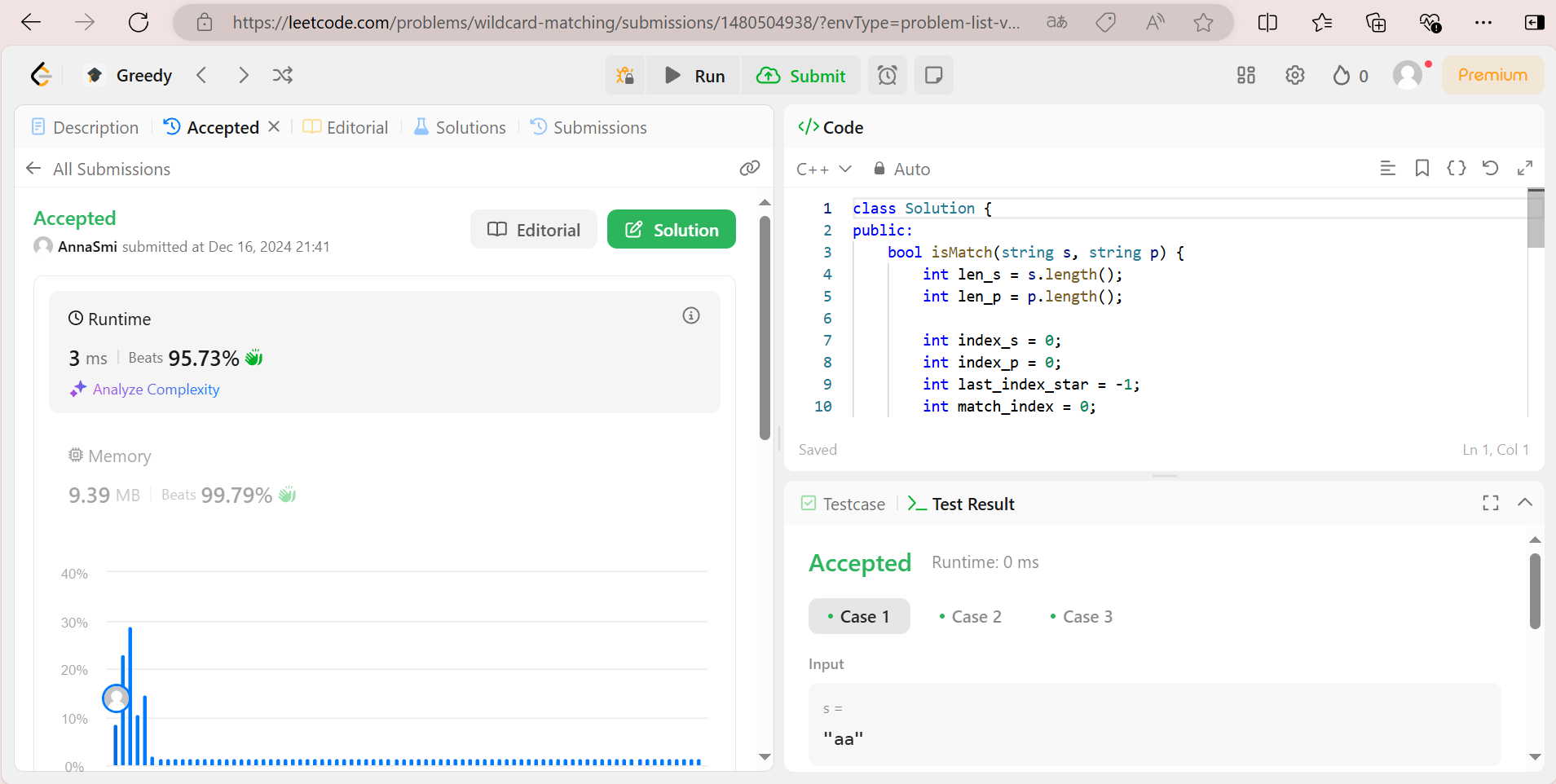
Общая память: 24 + 48 = 72 байт

Пространственная сложность: O(1)

Подсчет асимптотики (только для циклов и сложных структур):

Временная сложность: O(1)

Тестирование:



5. Заключение

В ходе выполнения работы мною был реализован алгоритм, позволяющий, проверить соответствует ли строка s шаблону p. Цель работы была достигнута путём тестирования алгоритма для разных строк и шаблонов. Использование жадного алгоритма в данной задаче эффективно благодаря тому, что он обрабатывает символы строки и шаблона последовательно, обеспечивая линейную временную сложность. Жадный алгоритм в данной задаче заключается в том, что он сразу обрабатывает совпадающие символы и использует “\*” для перехода к следующему символу, что в результате приведёт к соответствию всей строки шаблону. Он не рассматривает все возможные варианты, а использует уже найденные совпадения.

6. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода файла lab7.cpp

#include <iostream>

#include <string>

using namespace std;

bool isMatch(string s, string p) {

    int len\_s = s.length();

    int len\_p = p.length();

    int index\_s = 0;

    int index\_p = 0;

    int last\_index\_star = -1;

    int match\_index = 0;

    while (index\_s < len\_s) {

        if (index\_p < len\_p && (s[index\_s] == p[index\_p] || p[index\_p] == '?')) {

            index\_s++;

            index\_p++;

        }

        else if (index\_p < len\_p && p[index\_p] == '\*') {

            last\_index\_star = index\_p;

            match\_index = index\_s;

            index\_p++;

        }

        else if (last\_index\_star != -1) {

            index\_p = last\_index\_star + 1;

            index\_s = ++match\_index;

        }

else {

            return false;

        }

    }

    while (index\_p < len\_p && p[index\_p] == '\*') {

        index\_p++;

    }

    return index\_p == len\_p;

}

int main() {

    string s = "aa", p = "\*";

    if (isMatch(s, p)) {

        cout << "true" << endl;

    } else {

        cout << "false" << endl;

    }

    return 0;

}