ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу

Дубовик Никита

Академическая группа J3110

Принято

Ментор, Владислав Вершинин

Санкт-Петербург

2024

**Введение:**

Цель работы – Решить hard задачу 2262. на leetcode на динамическое программирование.

Задачи работы:

Определить как в данной задаче применить динамическое программирование;

Реализовать собственный алгоритм решающий задачу на c++.

**Реализация:**

1. Реализация алгоритма

Если в подстроке присутствуют несколько одинаковых символов, то только первый будет влиять на привлекательность. Тогда для определения общей привлекательности всей строки требуется найти для каждого символа во скольких подстроках он будет являться первым таким уникальным. Для этого можно создать массив, где индекс будет обозначать каждую букву в алфавите. В этом массиве будут храниться предыдущие вхождения символа, пока мы проходимся по строке. Мы суммируем общее количество подстрок, где данный символ будет влиять на показатель привлекательности. Это значение можно подсчитать умножив количество левых и правых конечных индексов для подстроки.

class Solution {

public:

    long long appealSum(std::string& str) {

        long long result = 0;

        long long dp[26];

        for (size\_t i = 0; i < str.size(); ++i) {

            const size\_t index\_of\_letter = str[i] - 'a';

            result += (i + 1 - dp[index\_of\_letter]) \* (str.size() - i);

            dp[index\_of\_letter] = i + 1;

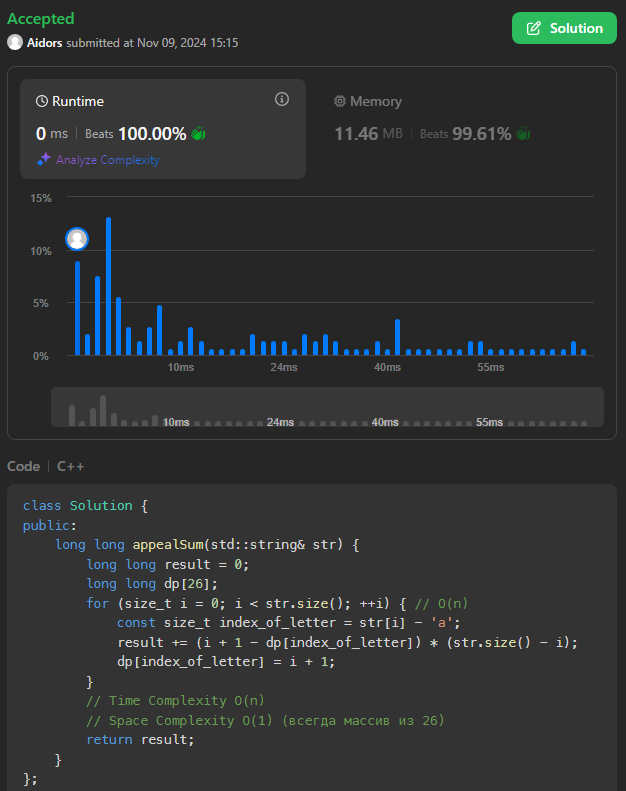
        }

        return result;

    }

};

1. Проверка корректности алгоритма



Изображение №1 – Доказательство корректной работы алгоритма

**Экспериментальная часть:**

Произведу расчёт асимптотики. В цикле будет ровно str.size() операций, а значит асимптотика будет O(n). Мы используем дополнительно константное количество памяти, поэтому пространственную сложность можно считать O(1).

**Заключение:**

В ходе выполнения работы мною был реализован алгоритм решающий поставленную задачу с использование динамического программирования

**Приложения:**

ПРИЛОЖЕНИЕ

Исходный код решения

class Solution {

public:

    long long appealSum(std::string& str) {

        long long result = 0;

        long long dp[26];

        for (size\_t i = 0; i < str.size(); ++i) { // O(n)

            const size\_t index\_of\_letter = str[i] - 'a';

            result += (i + 1 - dp[index\_of\_letter]) \* (str.size() - i);

            dp[index\_of\_letter] = i + 1;

        }

        // Time Complexity O(n)

        // Space Complexity O(1) (всегда массив из 26)

        return result;

    }

};