ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Minimum insertion steps to make a string palindrome(leetcode)»

Выполнил работу

Крапивина Яна

Академическая группа №J3112

Принято

Должность, звание Фамилия Имя преподавателя

Максим Дунаев, практик

Санкт-Петербург

2024

1. Введение

Целью данной работы является изучение и применение алгоритмов динамического программирования (ДП) на практике.

Задачи:

1. Выбрать задачи на сайте Leetcode;
2. Проанализировать их и понять, как применить ДП;
3. Написать хороший код;
4. Проанализировать полученные результаты и составить отчет.
5. Теоретическая подготовка

Динамическое программирование используется для решения задач, которые могут быть разбиты на подзадачи, решая каждую из них один раз и сохраняя результаты для последующего использования.

Используемые типы данных:

• int: целые числа

• string: строка символов

• vector<vector<int>>: двумерный динамический массив целых чисел

• Solution: класс, определяющий функцию minInsertions

1. Реализация

1) Анализ задачи

* Задача требует преобразовать строку в палиндром минимальным числом вставок.
* Палиндром — это строка, которая читается одинаково в обе стороны.
* Основная идея заключается в том, чтобы найти минимальное количество символов, которые нужно добавить в строку так, чтобы она стала палиндромом.

Поскольку задача имеет структуру, где подзадачи перекрываются (например, подстроки строки), очевидным подходом является использование динамического программирования (DP).

2) Применение DP:

* Определяем состояние: pl[i][j] — минимальное количество вставок для превращения подстроки s[i..j] в палиндром.
* Рекуррентные зависимости:
  + Если символы на границах подстроки совпадают (s[i] == s[j]), то:

pl[i][j] = pl[i + 1][j - 1];

* Если символы на границах разные (s[i] != s[j]), то:

pl[i][j] = 1 + min(pl[i + 1][j], pl[i][j - 1]);

* Базовый случай:
* Если длина подстроки равна 1 (i == j), она уже палиндром, поэтому:

pl[i][j] = 0;

* Если длина подстроки равна 2:

Если символы совпадают: pl[i][j] = 0;

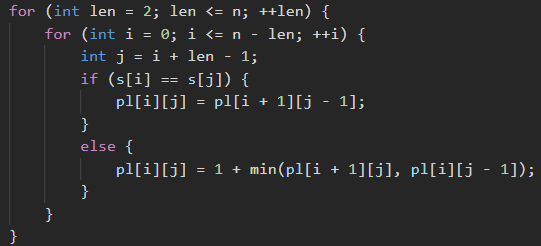
Если не совпадают: pl[i][j] = 1.

3) Реализация алгоритма

Используем двумерный вектор для хранения промежуточных результатов.



Создаем таблицу для хранения промежуточных данных и заполняем



Возврат результата



Используемые библиотеки:

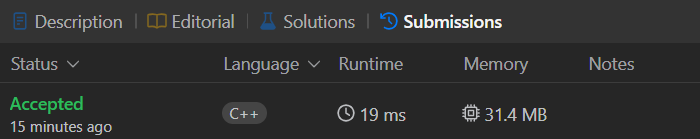
* #include <iostream> — для работы с вводом/выводом.
* #include <vector> — для динамических массивов (vector).
* #include <string> — для работы со строками.
* #include <algorithm> — для функции min

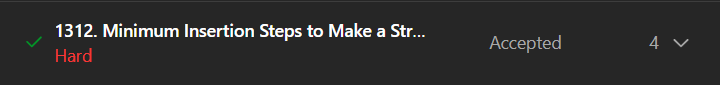
1. Экспериментальная часть

Время работы алгоритма — O(n^2), так как два вложенных цикла проходят по всем возможным подстрокам, а для каждой подстроки выполняется операция с постоянной сложностью.

Пространственная сложность — O(n^2), поскольку необходимо хранить таблицу размером n на n.

Тесты пройдены:





1. Заключение

В ходе выполнения работы был реализован алгоритм минимальных вставок для превращения строки в палиндром с использованием динамического программирования. Цель работы заключалась в разработке эффективного решения для задачи, а также в оценке его времени выполнения. Реализация алгоритма успешно решает задачу, корректно вычисляя минимальное количество вставок для различных строк.

В качестве направления для будущих исследований можно предложить оптимизацию алгоритма с точки зрения использования памяти, поскольку таблица динамического программирования имеет размер O(n^2).

1. Приложения

ПРИЛОЖЕНИЕ А

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Solution {

public:

int minInsertions(string s) {

int n = s.size();

vector<vector<int>> dp(n, vector<int>(n, 0));

for (int len = 2; len <= n; ++len) {

for (int i = 0; i <= n - len; ++i) {

int j = i + len - 1;

if (s[i] == s[j]) {

dp[i][j] = dp[i + 1][j - 1];

}

else {

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

}

}

}

return dp[0][n - 1];

}

};