Московский авиационный институт (национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа по курсу "Дискретный анализ" №7

Студент: Ахметшин Б. Р.

Преподаватель: Макаров Н. К.

Группа: М8О-303Б-22

Дата: _____

Оценка: _____

Подпись: _____

Москва 2024 г.

Оглавление

Постановка задачи

Задана строка *S* состоящая из *n* прописных букв латинского алфавита. Вычеркиванием из этой строки некоторых символов можно получить другую строку, которая будет являться палиндромом. Требуется найти количество способов вычеркивания из данного слова некоторого (возможно, пустого) набора таких символов, что полученная в результате строка будет являться палиндромом. Способы, отличающиеся только порядком вычеркивания символов, считаются одинаковыми.

Формат ввода

Задана одна строка *S* ($S \le 100$).

Формат вывода

Необходимо вывести одно число – ответ на задачу. Гарантируется, что он $\le 2^{63} - 1$.

Алгоритм решения

Динамическое программирование:

Пусть dp_{ij} — это количество палиндромных подпоследовательностей для подстроки, начинающейся с позиции *i* и заканчивающейся позицией *j* в строке *S*.

Инициализация:

Все одиночные символы являются палиндромами сами по себе, поэтому для каждого символа в строке *S* мы устанавливаем $dp_{ij} = 1$.

Заполнение таблицы:

Затем алгоритм постепенно заполняет таблицу для подстрок длиной 2 и более.

Для каждой подстроки s_{ij} с начальной позицией i и конечной позицией j, возможны два случая:

- **Символы совпадают** ($s_i = s_j$): Тогда, кроме палиндромов, которые содержатся в подстроках $s_{i+1}...s_j$ и $s_i...s_{j-1}$, можно добавить еще один палиндром, состоящий из символов s_i и s_j. Таким образом, $dp_{ij} = dp_{i+1,j} + dp_{i,j-1} + 1$.
- **Символы не совпадают** ($s_i \neq s_j$): Тогда результат будет суммой количества палиндромов в подстроках $s_{i+1}...s_j$ и $s_i...s_{j-1}$, исключая общие палиндромы из подстроки $s_{i+1}...s_{j-1}$ (так как они были учтены дважды). Таким образом, $dp_{ij} = dp_{i+1,j} + dp_{i,j-1} — dp_{i+1,j-1}$.

Результат:

После заполнения таблицы результатом будет значение $dp_{0,n-1}$, где n — длина строки.

Сложность работы алгоритма — $O(n^2)$.

Исходный код

```
#include <iostream>
#include <cinttypes>
#include <vector>
#include <string>

namespace lab {
	using u64 = uint64_t;
	template <class T>
	using vec = std::vector<T>;

	u64 countPalindromeSubsequences(std::string const& s) {
		u64 n = s.size();
		vec<vec<u64>> dp(n, vec<u64>(n, 0));
		for (u64 i = 0; i < n; i++) {
			dp[i][i] = 1;
		}

		for (u64 len = 2; len <= n; len++) {
			for (u64 i = 0; i <= n - len; i++) {
				u64 j = i + len — 1;

				if (s[i] == s[j]) {
					dp[i][j] = dp[i + 1][j] + dp[i][j - 1] + 1;
				} else {
					dp[i][j] = dp[i + 1][j] + dp[i][j - 1] - dp[i + 1][j - 1];
				}
			}
		}

		return dp[0][n - 1];
	}
}
```

```
int main() {
	std::string s;
	std::cin >> s;

	auto result = lab::countPalindromeSubsequences(s);
	std::cout << result << std::endl;

	return 0;
}
```

Тесты

Input	Output
abac	6

Input	Output
aaa	6

Вывод

В ходе лабораторной работы я научился решать задачи методом динамического программирования.