ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Смирнов Олег

Академическая группа J3110

Принято

Вершинин Владислав

Санкт-Петербург

2024

1. Задача

Найти минимальную по лексикографическому порядку строку большую данной, которая не содержит палиндромов и состоит из k первых символов латиницы. Входная строка палиндромов не содержит.

1. Решение

Жадный шаг 1: заметим, что если строка не содержит палиндромов длины 2 или 3, то не содержит палиндромов вовсе.

Жадный шаг 2: минимальная строка будет получаться, если мы последовательно начнём увеличивать её символы слева-направо, то есть сначала переберём все варианты с изменённым только последним символом, потом с двумя и так далее.

Жадный шаг 3: нет смысла проходить всю строку для проверки на содержание палиндромов, просто будем проверять не портит ли всё наша заменённая подстрока.

1. Реализация

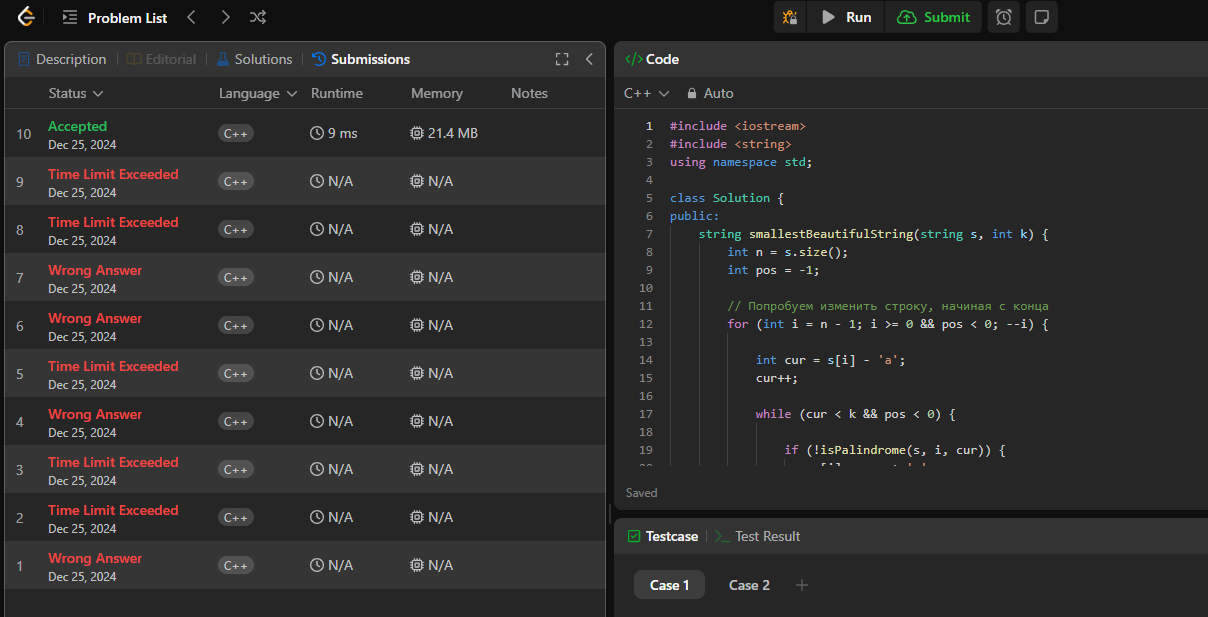
Запускаем цикл по индексу элемента, с которого будет начинаться замена, подбираем минимальный допустимый символ в этой позиции, заполняем остальные места.

1. Память и время

Кроме счётчиков, длины строки, символа в текущей позиции и индекса позиции ничего не храним. Получается дополнительные затраты по памяти O(1).

Время: O(n) перебор индекса для начала подстроки под замену в худшем случае, вложенный цикл O(k) – перебор символов на текущую позицию. То есть O(k\*n). Заполнение пропусков O(k\*n) в худшем случае.

1. Скрин с тестами и код



#include <iostream>

#include <string>

using namespace std;

class Solution {

public:

    string smallestBeautifulString(string s, int k) {

        int n = s.size();

        int pos = -1;

        // Попробуем изменить строку, начиная с конца

        //O(n)

        for (int i = n - 1; i >= 0 && pos < 0; --i) {

            int cur = s[i] - 'a';

            cur++;

            //O(k)

            while (cur < k && pos < 0) {

                if (!isBeautiful(s, i, cur)) {

                    s[i] = cur + 'a';

                    pos = i;

                }

                cur++;

            }

        }

        // Если позиция для изменения не найдена, возвращаем пустую строку

        if (pos < 0){

            return "";

        }

        // Заполняем оставшуюся часть минимальными символами

        //O(k\*n)

        fillRemaining(s, pos + 1, k);

        return s;

    }

private:

    // Проверяем, не будет ли текущий символ формировать палиндром

    bool isBeautiful(const string &s, int i, int cur) {

        if (i - 1 >= 0 && (s[i - 1] - 'a') == cur){

            return true; // Палиндром длины 2

        }

        if (i - 2 >= 0 && (s[i - 2] - 'a') == cur){

            return true; // Палиндром длины 3

        }

        return false;

    }

    // Заполняем строку, начиная с позиции start, минимальными символами

    void fillRemaining(string &s, int start, int k) {

        int n = s.size();

        for (int i = start; i < n; ++i) {

            for (int j = 0; j < k; ++j) {

                if (!isBeautiful(s, i, j)) {

                    s[i] = j + 'a';

                    break;

                }

            }

        }

    }

};

//memory O(1) ничего лишнего не храним: счётчики, индексы, текущий символ

//time O(k\*n)