ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«1526. Minimum Number of Increments on Subarrays to Form a Target Array»

Выполнил работу

Бровкин Аким

J3110

Принято

Ментор, Вершинин Владислав

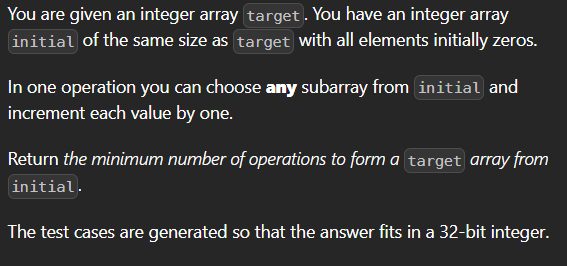
Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Цель работы: вычислить минимальное количество операций для формирования целевого массива из начального.



Задачи:

— Решить задачу используя жадный алгоритм.

— Проанализировать затрачиваемую память и время работы программы.

— Объяснить, зачем здесь используется жадный алгоритм.

1. Теоретическая подготовка

Использовались следующие типы данныx:

int - базовый целочисленный тип по умолчанию

vector<int> - динамический массив, обеспечивающий быстрое добавление новых элементов в конец и меняющий свой размер при необходимости

Также были использован такой метод как size()

1. Реализация

­- Задача заключается в том, чтобы найти минимальное количество операций для превращения массива, состоящего из нулей, в массив target, используя при этом операции инкрементации на подмассивах.

- Начальная операция заключается в установке первого элемента массива target в переменную res, так как это минимальное количество операций, необходимое для достижения первого элемента.

Пробегая по массиву target от второго элемента до конца, нужно добавлять к результату res разницу между текущим и предыдущим элементом, если эта разница положительная. Это позволяет учесть только те изменения, которые требуют дополнительных операций.

Функция max(0, target[i] - target[i - 1]) используется для выбора только положительных изменений, требующих инкрементации.

Использование этой функции – это по своей сути и есть жадный выбор (алгоритм), который вносит вклад в решение только в случае необходимости увеличения, игнорируя те случаи, когда элемент не требует изменений или требует уменьшения, что невозможно в данной задаче.

Преимущества жадного подхода:

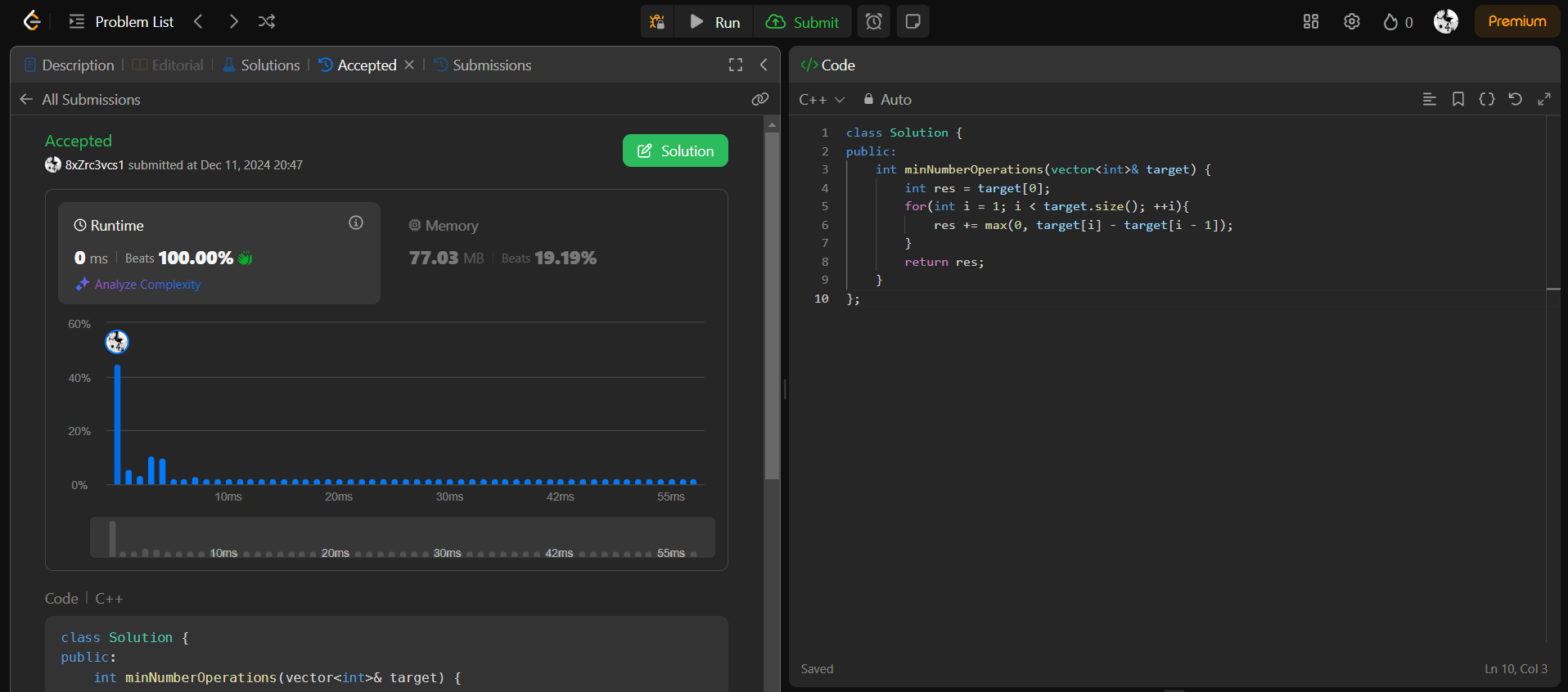
1. Простота: Решение становится интуитивно понятным и простым в реализации, так как в каждом шаге мы делаем очевидный выбор.
2. Эффективность: Жадный алгоритм позволяет достичь решения за линейное время, что особенно важно для работы с большими данными.

3. Оптимальность: В данной задаче жадный подход дает оптимальное решение, так как каждое локальное улучшение (инкрементация только необходимой части массива) ведет к глобальному минимуму операций.

1. Экспериментальная часть

В данной задаче было использован жадный алгоритм, потому что, выбирая на каждом шаге наилучшее локальное решение, мы достигаем глобально оптимального решения. Так же жадный подход здесь оказывается оптимальным, поскольку задача не содержит сложных зависимостей или условий, которые могли бы сделать такой метод неэффективным. Все это позволяет решить задачу за линейное время O(n), n – размер массива target.

1. Заключение



1. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода (с комментариями)

class Solution {

public:

    // Функция minNumberOperations:

    // Память:

    // - Локальные переменные (res, i): 4 \* 2 = 8 байт

    // - Переданный по ссылке target: 24 + 4 \* n байт

    // Общая память: O(n)

    // Асимптотика:

    // - Итерация по массиву target: O(n)

    // Общая асимптотика: O(n)

    int minNumberOperations(vector<int>& target) {

        // Инициализация результата с первым элементом массива target

        int res = target[0];

        // Проход по остальным элементам массива

        for(int i = 1; i < target.size(); ++i) {

            // Добавление разницы между текущим и предыдущим элементом, если она положительна

            res += max(0, target[i] - target[i - 1]);

        }

        // Возвращаем общее количество операций

        return res;

    }

};