ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Тищенко Павел

Академическая группа J3112

Принято

Лектор, Ходненко Иван Владимирович

Санкт-Петербург

2024

Структура отчёта:

1. **Введение**

Цель работы

Необходимо решить задачу hard на leetcode с помощью жадных алгоритмов и проанализировать его решение

Задачи:

Изучение теоретических основ жадных алгоритмов

Реализация решения на языке программирования C++

Оптимизация решения

Анализ результатов

2. **Теоретическая подготовка**

Для решения данной лабораторной работы мне было необходимо разобраться с тем, что такое жадные алгоритмы, почему они не щедрые и как их применять в разработке и решении задач. Жадные алгоритмы — это подход к решению задач, где на каждом шаге принимается наиболее оптимальное решение локально, с надеждой, что оно приведёт к глобально оптимальному результату.

Типы данных

- vector: используется для хранения динамических массивов. Позволяет изменять размер в процессе выполнения программы.

-стандартные типы данных: int, bool и тд.

3. **Реализация**

1**. Выбрать задачу на leetcode**

Выбор был очень сложным, реализован с помощью кнопки “Pick One”

2. **Осознание того, что от меня хотят в задаче**

Исходя из условия задачи (и потупив +- 30 минут), я понял какие “блюда” являются “не выгодными” (на основе своих ручных тестов и ихкомпиляции в голове)

3. **Реализация решения на c++**

После выбора и проверки на работоспособность мне осталось лишь переписать все мои мысли на код с использованием стандартных типов данных и std::vector, и циклов.

4. **Accepted и оптимизация**

После того, как я увидел зеленую кнопку “Accepted” я переименовал переменные и сделал код более структурированным и красивым. Однако, это не все! В момент подсчета асимптотики для алгоритма, я осознал, что за счет функции **summ** скорость в худшем случае может быть почти O(N^2), что очень плохо, тогда мне пришло осознание, что создав булеву переменную с дополнительной проверкой, которая будет очень сильно оптимизировать время работы алгоритма, в случае большого количества отрицательных значений в массиве.

4. Экспериментальная часть

Подсчёт по памяти

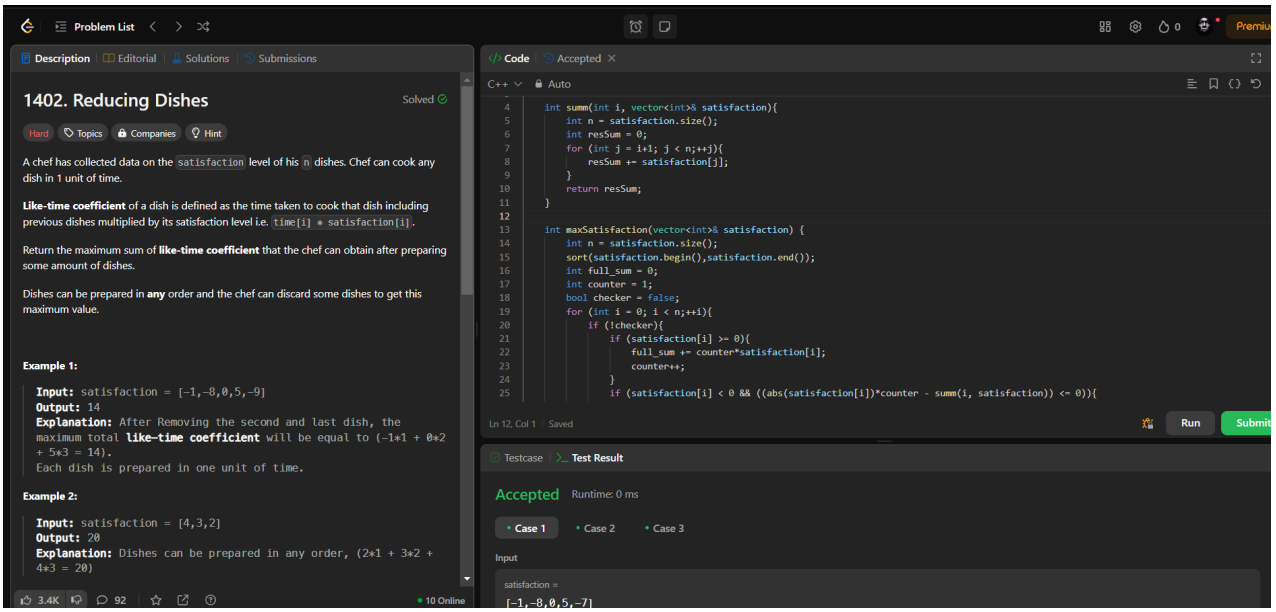
1. “Интовые” переменные— занимает O(1), каждая по 8 bite, 6 штук (и одна булевая переменная стоимость 1bite).
2. Массив lst — хранит промежуточные результаты для каждого элемента массива target, занимает O(n) памяти, (24+8n)bite.

3. Память затрачиваемая для сортировки O(n), создастся +-(2n) подмассивов и -> logn*n новых переменных
3. Итоговая память: O(n), (49 + 8n + 24*(2n) + logn*8)bite.

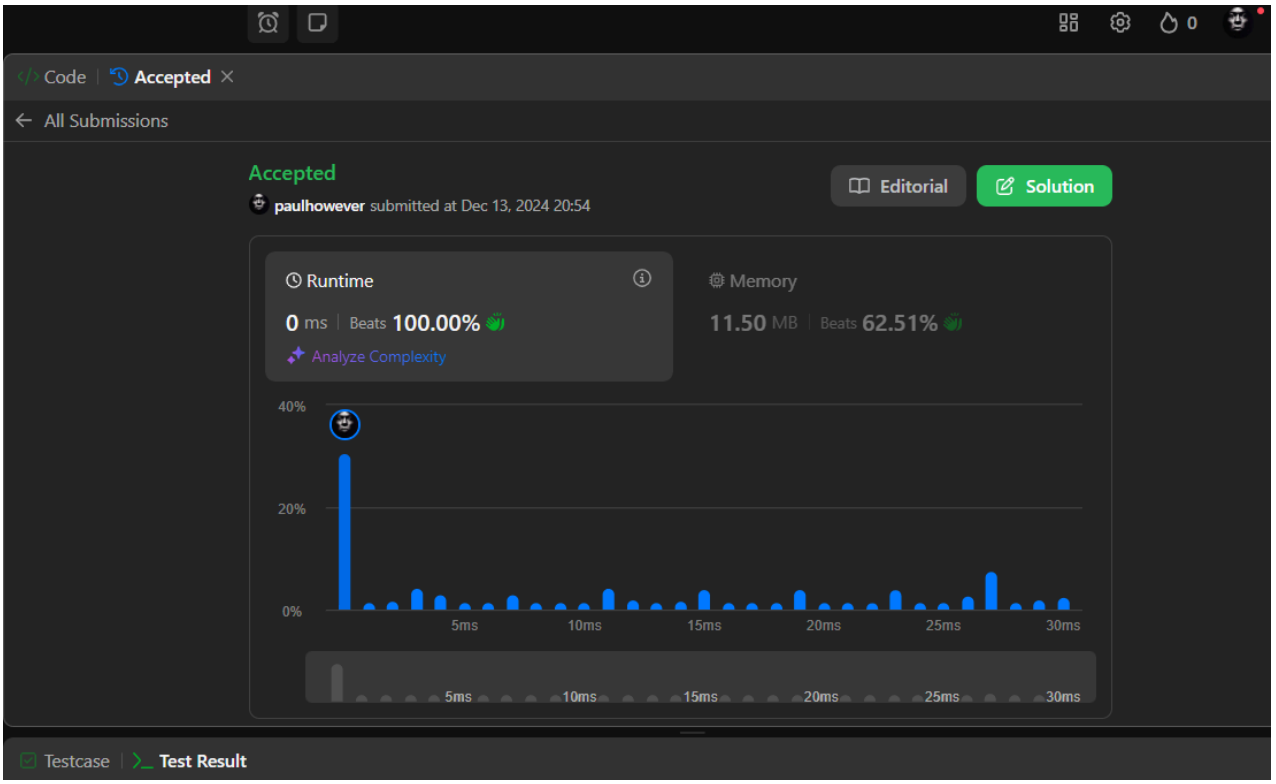
Подсчёт асимптотики

Первым делом происходит сортировка массива, ф-я sort в c++ реализована с помощью алгоритма quickSort, поэтому сортировка занимает O(N*logN). Так как мы проходимся по всему массиву 1м основным циклом и также в некоторых условиях обращаемся к еще одному дополнительному циклу с пробегом почти по всему массиву, то в лучшем случаи (если все эл-ты массива положительные) будет O(n), в худшем соответственно O(N^2). Средний случай O(N*logN)

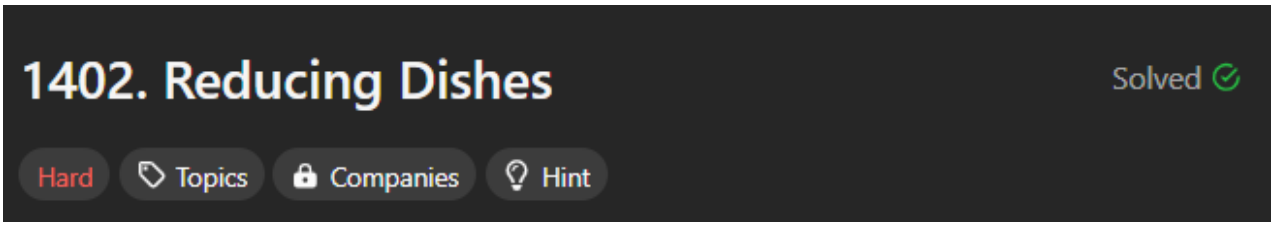
Leetcode Hard task DP Рисунок 1.1



Leetcode Hard task DP Рисунок 1.2



Leetcode Hard task Рисунок 1.3



Эта задача решается жадным алгоритмом, поскольку:
Чтобы максимизировать значение суммы коэффициентов времени (like-time coefficient), более выгодно готовить блюда с высоким уровнем удовлетворённости позже Мы можем игнорировать блюда с отрицательной удовлетворённостью, если их добавление уменьшает общую сумму. Это делается жадным образом: мы проходим по отсортированному массиву, начиная с блюда с максимальной удовлетворённостью, и перестаём добавлять блюда, если общая сумма перестаёт увеличиваться.
На каждом шаге мы принимаем локально оптимальное решение. Это решение оказывается глобально оптимальным благодаря сортировке и линейному обходу.

5. Заключение

В ходе выполнения лабораторной работы были изучены основные принципы жадных алгоритмов. Задача на платформе LeetCode была успешно решена. Также данный алгоритм был протестирован на дополнительных массивах и он показал ожидаемые результаты и по скорости и правильности выполнения. В качестве дальнейшего исследования можно попробовать решить усложненные задачи на данную тематику или придумать усложнения для задачи решаемой в ходе данной лабораторной работы.

6. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода файла leetCode_7lab.cpp

```
class Solution {
public:

    int summ(int i, vector<int>& satisfaction){
        int n = satisfaction.size();
        int resSum = 0;
        for (int j = i+1; j < n;++j){
            resSum += satisfaction[j];
        }
        return resSum;
    }

    int maxSatisfaction(vector<int>& satisfaction) {
        int n = satisfaction.size();
        sort(satisfaction.begin(),satisfaction.end());
        int full_sum = 0;
        int counter = 1;
        bool checker = false;
        for (int i = 0; i < n;++i){
            if (!checker){
                if (satisfaction[i] >= 0){
                    full_sum += counter*satisfaction[i];
                    counter++;
                }
                if (satisfaction[i] < 0 && ((abs(satisfaction[i])*counter -
summ(i, satisfaction)) <= 0)){
                    checker = true;
                    full_sum += counter*satisfaction[i];
                    counter++;
                }
            }
            else{
                    full_sum += counter*satisfaction[i];
                    counter++;
            }
        }
        return full_sum;
    }
};
```