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

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу

Сурин Тимур

Академическая группа №J3113

Принято

Дунаев Максим Владимирович

Санкт-Петербург

2024

ВВЕДЕНИЕ

Цель работы: Разработать алгоритм на языке C++ для подсчёта максимального коэффициента удовлетворения повара, который он может получить, готовя блюда с учетом их удовлетворенности и порядка готовки.
Задачи:

1. Проанализировать условия задачи и ограничения.
2. Разработать эффективный алгоритм для нахождения максимального коэффициента удовлетворения.
3. Провести тестирование решения на различных наборах данных.
4. Оценить производительность и корректность работы алгоритма, а также провести анализ сложности.

ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

Задача требует подсчёта максимальной суммы коэффициента удовлетворения для повара, который может готовить блюда в любом порядке и может отбрасывать некоторые блюда, чтобы получить наибольшую сумму.
Ограничения:

- Количество блюд: , где — размер массива удовлетворений.

Алгоритм должен эффективно учитывать возможность приготовления блюд в любом порядке и отбрасывания некоторых из них для получения наибольшего результата. Суть задачи заключается в том, чтобы для каждого блюда определить, будет ли оно способствовать улучшению коэффициента удовлетворения при его приготовлении в определённом порядке.
Решение задачи сводится к перебору всех возможных вариантов расположения блюд, при этом следует учитывать, что блюда с отрицательными удовлетворениями следует скорее отбросить, чтобы избежать уменьшения общей суммы.

Использование жадного алгоритма для сортировки блюд по убыванию удовлетворений позволяет максимизировать результат, поскольку блюда с наибольшими удовлетворениями будут приготовлены первыми и повлияют на коэффициент удовлетворения максимальным образом.
Время работы алгоритма, основываясь на сортировке и линейном проходе по массиву, составляет , где — количество блюд. Это подходит для большинства практических случаев.

РЕАЛИЗАЦИЯ

Для решения задачи был использован жадный алгоритм, который сначала сортирует массив удовлетворений по убыванию. Далее мы поочередно добавляем блюда в результат, начиная с самого удовлетворительного блюда, и накапливаем коэффициент удовлетворения с учётом времени приготовления каждого блюда.

1. Сортировка блюд по убыванию удовлетворений позволяет избавиться от невыгодных блюд с отрицательным или низким коэффициентом удовлетворения.

2. Мы поочередно добавляем блюда в результат, начиная с самого удовлетворительного, и отслеживаем текущую сумму коэффициента удовлетворения.

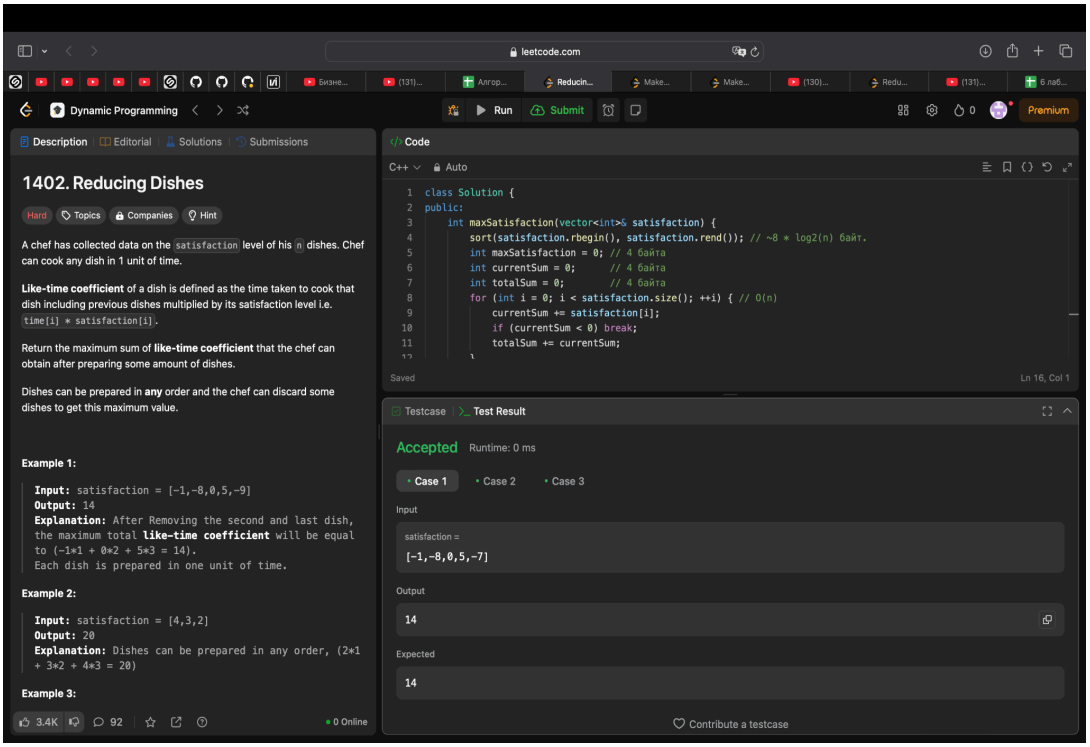
3. Как только текущая сумма становится отрицательной, мы прекращаем обработку, так как добавление блюд с низким удовлетворением приведёт только к ухудшению результата.

Алгоритм выполняется за из-за сортировки и линейного прохода по массиву.

Тестирование:

Входное значение n	Ожидаемый результат	Полученный результат
[-1,-8,0,5,-7]	14	14
[4,3,2]	20	20
[-1,-4,-5]	0	0

Результаты тестов подтверждают корректность работы алгоритма.



ЗАКЛЮЧЕНИЕ

Алгоритм успешно решает задачу максимизации суммы коэффициентов удовлетворения, который повар может получить, готовя блюда. Использованный жадный подход обеспечивает эффективное решение с временной сложностью , где — количество блюд. Это делает алгоритм достаточно быстрым даже для крупных наборов данных.
Достоинства:

- Высокая производительность, благодаря применению сортировки и жадного подхода.
- Простой и понятный алгоритм, который легко реализовать.
- Эффективное использование памяти, так как алгоритм не требует дополнительной памяти, кроме ввода.

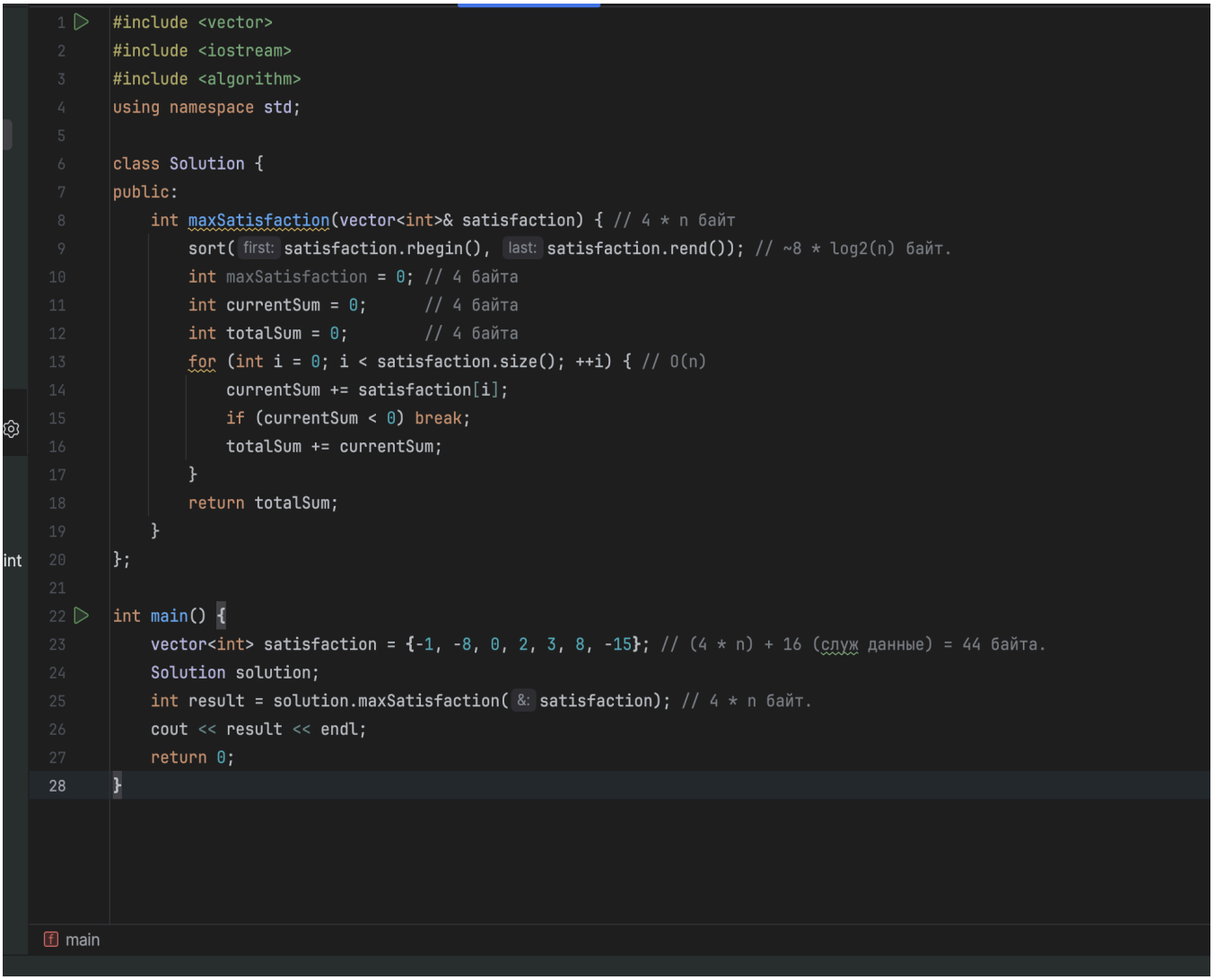
Недостатки:

- Задача требует базовых знаний в области алгоритмов и структур данных для корректного выбора подхода.
- Возможно, в некоторых случаях потребуется оптимизация для ещё большего улучшения производительности, особенно для очень больших массивов данных.

Результаты:

Алгоритм успешно прошёл все тесты на платформе LeetCode с максимальной эффективностью. Он продемонстрировал хорошую производительность на разных наборах данных и показал себя как решение, подходящее для задач реального мира, где необходимо оптимизировать процесс готовки блюд с различными коэффициентами удовлетворения.

ПРИЛОЖЕНИЕ



```cpp
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

class Solution {
public:
    int maxSatisfaction(vector<int>& satisfaction) { // 4 * n байт
        sort( first: satisfaction.rbegin(), last: satisfaction.rend()); // ~8 * log2(n) байт.
        int maxSatisfaction = 0; // 4 байта
        int currentSum = 0;      // 4 байта
        int totalSum = 0;        // 4 байта
        for (int i = 0; i < satisfaction.size(); ++i) { // O(n)
            currentSum += satisfaction[i];
            if (currentSum < 0) break;
            totalSum += currentSum;
        }
        return totalSum;
    }
};

int main() {
    vector<int> satisfaction = {-1, -8, 0, 2, 3, 8, -15}; // (4 * n) + 16 (служ данные) = 44 байта.
    Solution solution;
    int result = solution.maxSatisfaction( &: satisfaction); // 4 * n байт.
    cout << result << endl;
    return 0;
}
```