ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Дубовик Никита

Академическая группа J3110

Принято

Ментор, Владислав Вершинин

Санкт-Петербург

2024

**Введение:**

Цель работы – Решить hard задачу 1665 на leetcode используя концепцию жадного алгоритма.

Задачи работы:

Определить как в данной задаче применить жадный алгоритм;

Реализовать собственный алгоритм решающий задачу на c++.

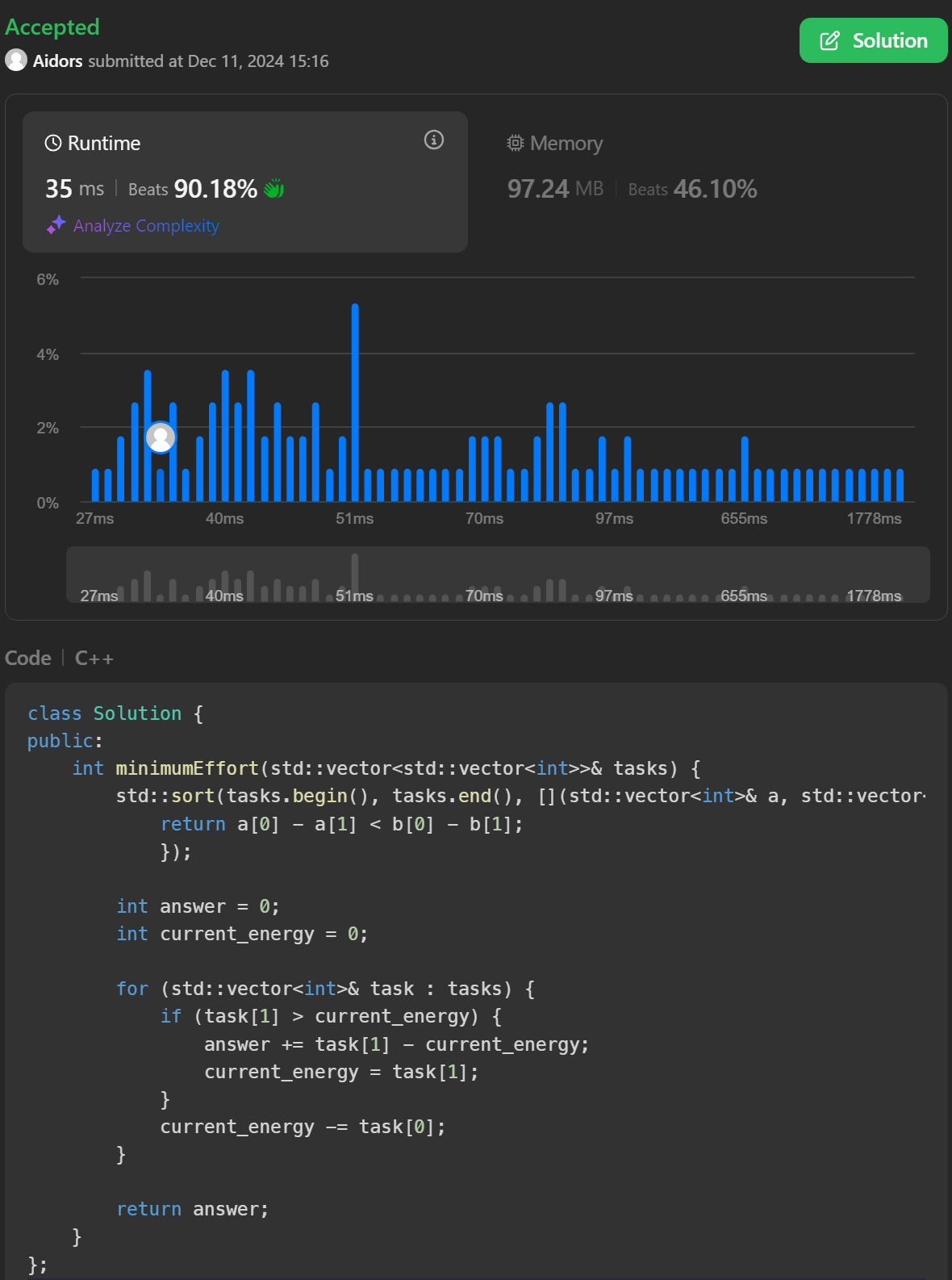
**Реализация:**

1. Реализация алгоритма

Основываясь на метрике minimum – actual я сортирую вектор с задачами используя lambda функцию для вычисления метрики во время сортировки. Далее пробегая по задачам из получившегося вектора в нужном порядке производится проверка, хватает ли нашей оставшейся энергии для получения доступа к задаче, если да, то необходимая энергия вычитается из оставшейся энергии, если нет, то нам нужно пополнить нашу изначальную энергию на необходимую разницу и обновляем значение оставшейся энергии. Поскольку мы судим о задачах только на основе высчитанной нами метрике и из-за этого на каждом шаге делаем локально оптимальный выбор порядка выполнения задач, то данный алгоритм является жадным алгоритмом

1. Проверка корректности алгоритма

Доказательство корректной работы представлены на изображении №1



Изображение №1 – Доказательство корректной работы алгоритма

**Экспериментальная часть:**

Произведу расчёт асимптотики. Space Complexity O(1), так как дополнительная память (кроме констант) использована не была. Асимптотика алгоритма сортировки O(n\*log(n)). Асимптотика цикла будет O(n). Итоговая Time Complexity O(n\*log(n)).

**Заключение:**

В ходе выполнения работы мною был реализован алгоритм решающий поставленную задачу используя жадный алгоритм.

**Приложения:**

ПРИЛОЖЕНИЕ

Исходный код решения

#include <vector>

#include <algorithm>

int minimumEffort(std::vector<std::vector<int>>& tasks) {

std::sort(tasks.begin(), tasks.end(), [](std::vector<int>& a, std::vector<int>& b) {

return a[0] - a[1] > b[0] - b[1];

});

int answer = 0;

int current\_energy = 0;

for (std::vector<int>& task : tasks) {

if (task[1] > current\_energy) {

answer += task[1] - current\_energy;

current\_energy = task[1];

}

current\_energy -= task[0];

}

// Time Complexity O(n\*log(n))

// Space Complexity O(1)

return answer;

}