ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу:

Мавров Артём

Академическая группа №J3112

Принято:

Ассистент (квалификационная категория "ассистент"), Дунаев Максим

Санкт-Петербург

2024

Введение

Цель лабораторной работы — на практике изучить принципы жадных алгоритмов.

Задачи – изучить жадные алгоритмы, применить знания на практике решив hard задачу на leetcode.

**Теоретическая подготовка:**

Для реализации этой лабораторной работы потребовалось:

* Базовые знания языка C++
* Знания жадных алгоритмов

**Реализация:**

* Реализация задачи с https://leetcode.com/problems/super-washing-machines/

Экспериментальная часть

**Реализация решения при помощи жадного алгоритма(Time limit), подсчёт асимптотики:**

class Solution {

public:

int findMinMoves(vector<int>& machines) {

int n = machines.size();

int sm = 0;

for (int i = 0; i < n; i++) {

sm += machines[i];

}

if (sm % n != 0) { //проверка на наличие решения

return -1;

}

int amt\_dresses = sm / n; //количество платьев, которое должно быть в каждой машинке

int sm\_l;

bool flag = false;

int step = 0;

while (!flag) { //в худшем случае O(N + max(machines))

sm\_l = machines[0];

flag = true;

if (machines[0] > amt\_dresses) {

machines[1] += 1;

machines[0] -= 1;

}

for (int i = 1; i < n; i++) { //O(N)

if (machines[i] > 0){

if (sm\_l < i \* amt\_dresses) {

machines[i - 1] += 1;

machines[i] -= 1;

}else if (sm - sm\_l - machines[i] < i \* amt\_dresses && i != n - 1) {

machines[i + 1] += 1;

machines[i] -= 1;

}

}

sm\_l += machines[i];

}

step += 1;

for (int i = 0; i < n; i++) { //O(N)

if (machines[i] != amt\_dresses) {

flag = false;

break;

}

}

}

return step;

}

};

//Итого сложность: O(N \* (2 \* N + max(machines))

//Пространственная сложность: Использовались только незначительные переменные, массив был дан так что O(1)

(приложение 1: решение задачи жадным алгоритмом)

**Пояснение к работе алгоритма:**  
В начале идёт проверка на существование решения. Далее вычисляется сколько должно быть белья в каждой машинке и записывается в переменную amt\_dresses. После алгоритм проводит операции перекладывания белья, учитывая баланс слева и справа от элемента, таким образом гарантируется оптимальность перекладывания белья.

**Почему жадный алгоритм не подходит:**

У данной задачи есть все критерии жадной, так как решение подзадач оптимальным способом влечёт за собой оптимальное решение(к выше сказанному она имеет подпись “Greedly” на leetcode). Она действительно может быть решена жадным алгоритмом, но это не является эффективном решением так как при оптимальном жадном решении необходимо проделать все подзадачи, и как было выше подсчитано сложность при максимальных входных данных будет (O(1e4 \* (2e4 + 1e5)).

Вместо жадного решение можно привести алгоритм, который основывается на жадном решении, но не выполняет все подзадачи, а заранее ищет худший случай для переноса платьев и тем самым давая корректный ответ.

**Пошаговый пример:**

Start 1 0 5 ->

1 0 5 -> 1 0 5 -> 1 1 4 ->

1 1 4 -> 2 0 4 -> 2 1 3 ->

2 1 3 -> 2 1 3 -> 2 2 2 end

Перекладываем если сумма слева или справа меньше чем должна быть для баланса.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание  
(приложение 2: решение задачи на leetcode)

**Реализация более оптимального решения, подсчёт асимптотики:**class Solution {

public:

    int findMinMoves(vector<int>& machines) {

        int n = machines.size();

        int sm = 0;

        for (int i = 0; i < n; i++) {

            sm += machines[i];

        }

        if (sm % n != 0) {

            return -1;

        }

        int amt\_dresses = sm / n;

        int total\_balance = 0;

        int max\_moves = 0;

        for (int i = 0; i < n; i++) { //O(N)

            int diff = machines[i] - amt\_dresses;

            total\_balance += diff;

            max\_moves = max(max\_moves, max(abs(total\_balance), diff));

        }

        return max\_moves;

    }

};

//Итого сложность: O(N )

//Пространственная сложность: Использовались только незначительные переменные, массив был дан так что O(1)

(приложение 3: реализация более оптимального решения)

**Пояснение к работе алгоритма:**

Вместо того чтобы перекладывать бельё каждый раз моделируя задачу, можно предсказать на основе прошлого решения, сколько потребуется оптимальных действий. Алгоритм для каждой стиральной машинке находит максимум между количеством перемещений для выравнивания пред идущих машин и избытка/недостатка в текущей машинке. Таким образом данный подход гарантирует, что мы учитываем как глобальные, так и локальные изменения, минимизируя количество шагов.

**Пошаговый пример:**

Машина 0:

* total\_balance = −1
* diff = -1
* max\_moves = 1

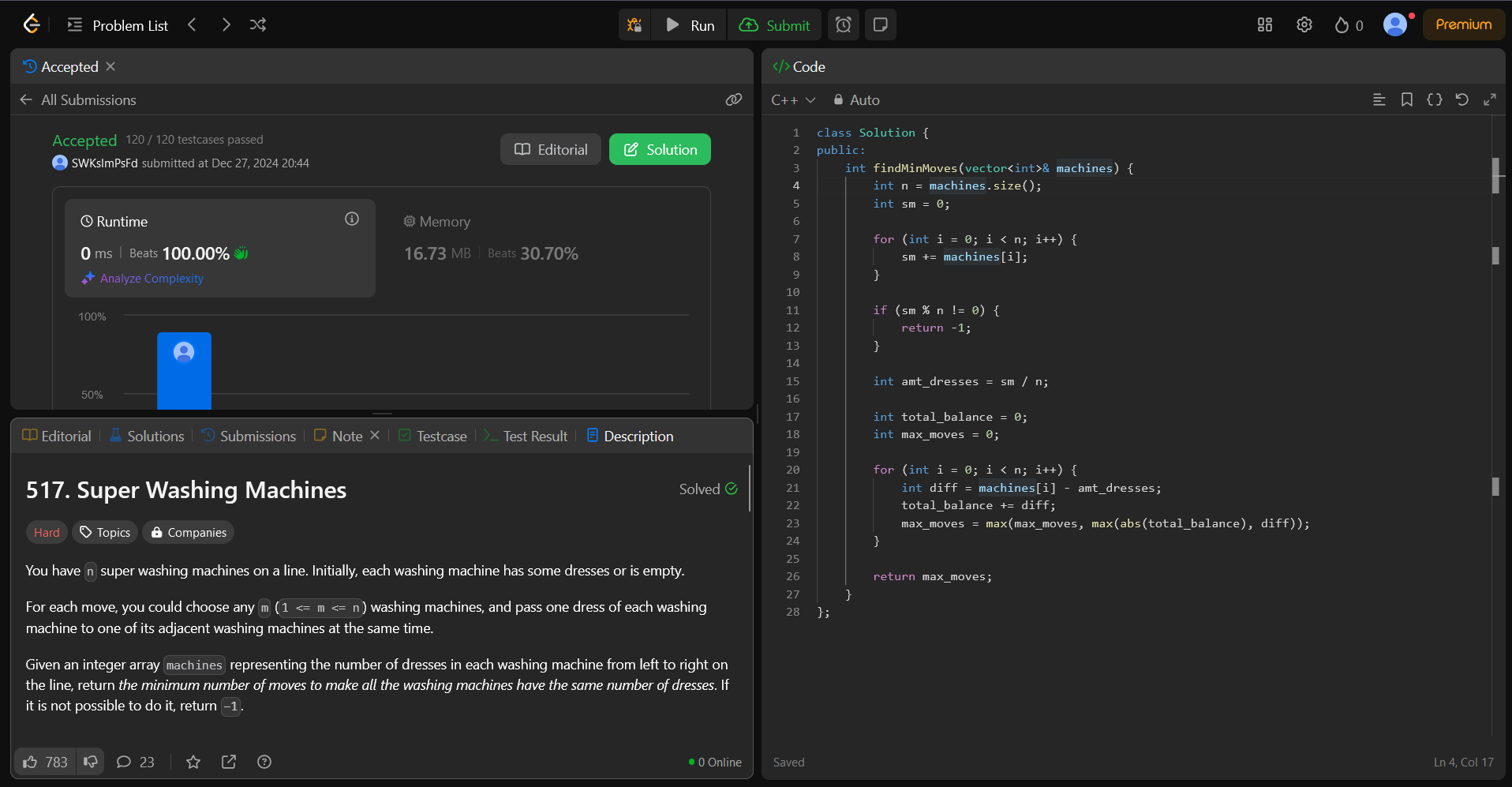
Машина 1:

* total\_balance = −3
* diff = -2
* max\_moves = 3

Машина 2:

* total\_balance = 0
* diff = 3
* max\_moves = 3

Итого mac\_moves = 3

****(приложение 4: решение задачи на leetcode, при помощи оптимального решения)