Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет цифровых трансформаций

Дисциплина:

«Алгоритмы и структуры данных»

ПРАКТИЧЕСКАЯ РАБОТА №7

«Жадные алгоритмы»

Выполнил:

Ступичев Алексей, студент группы J3110

_____
(подпись)

Проверил(а):

ФИО преподавателя, должность преподавателя

_____
(отметка о выполнении)

_____
(подпись)

СОДЕРЖАНИЕ

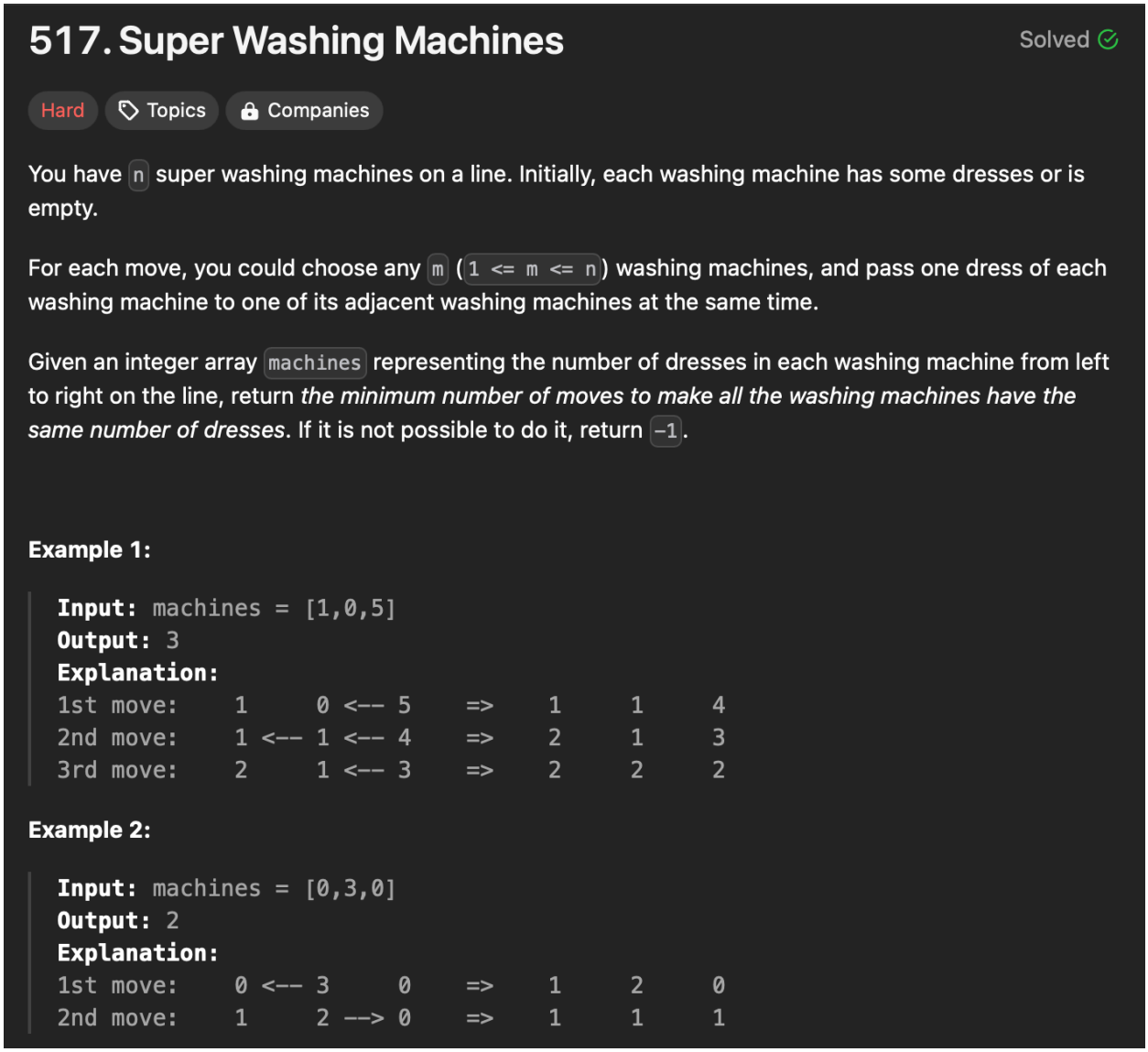
Введение

Цель работы – научится решать задачи на жадные алгоритмы на примере задачи: 517.Super Washing Machines from LeetCode.

Задачи:

1. Придумать алгоритм для решения задачи.
2. Реализовать алгоритм на языке c++.
3. Определить асимптотику алгоритма и затрачиваемую на него память.
4. Составить отчет по работе.

Теоретическая подготовка и Реализация



517. Super Washing Machines

Solved

Hard | Topics | Companies

You have `n` super washing machines on a line. Initially, each washing machine has some dresses or is empty.

For each move, you could choose any `m` (`1 <= m <= n`) washing machines, and pass one dress of each washing machine to one of its adjacent washing machines at the same time.

Given an integer array `machines` representing the number of dresses in each washing machine from left to right on the line, return *the minimum number of moves to make all the washing machines have the same number of dresses*. If it is not possible to do it, return `-1`.

Example 1:

```
Input: machines = [1,0,5]
Output: 3
Explanation:
1st move:    1     0 <-- 5    =>    1     1     4
2nd move:    1 <-- 1 <-- 4    =>    2     1     3
3rd move:    2     1 <-- 3    =>    2     2     2
```

Example 2:

```
Input: machines = [0,3,0]
Output: 2
Explanation:
1st move:    0 <-- 3     0    =>    1     2     0
2nd move:    1     2 --> 0    =>    1     1     1
```

Сначала найдем среднее в machines, таким образом мы сразу узнаем, существует ли решение вообще. Если среднее не целое, то не существует, в ином случае перейдем к решению задачи. Сразу заметим, что довольно удобно рассмотреть массив, где все элементы - разность элементов из machines и среднего. Найдем максимальный положительный элемент в этом массиве, меньше него точно не получится, так как из машины нельзя достать

больше одной вещи за раз. Но если запустить такой алгоритм, он не пройдет, например на массиве {0, 11, 11, 2}: максимальное в нем 5, а ходов понадобится 6. Проблема понятна - числа больше нуля не всегда смогут каждый ход отдавать другим, если их зажали левая или правая границы массива и одно из чисел больше чем они сами. Избежать этого можно тем, чтобы рассматривать не конкретные числа, а подряд идущие отрезки из этих чисел, а точнее их сумму. Тогда мы так же учтем отрицательные числа массива. Очень важно, что сумму будем рассматривать именно с левого края (можно и с правого, но важен факт, что край ограничивает передачу вещей из положительных чисел в одну из сторон) таким образом при каждой итерации текущая сумма будет ограниченна для передачу влево, что делает верным. Не менее важно, что мы будем рассматривать модуль суммы (это на самом деле последствие того, что мы идем с слева направо, а не справа налево). Идея этой сумму в том, чтобы накопить избыток зажатых между собой положительных чисел, максимальное из которых определяет минимальное количество ходов. Но важно, что этот избыток может и не появится, поэтому не стоит забывать и про найденное нами наибольшее положительное число.

Получается, что для поиска ответа нужно пройти один раз по массиву с разницей со средним, для каждого элемента выбирая максимум из предыдущего максимума, из положительного и из модуля суммы всех предыдущих. Код цикла поиска:

```
for (int elem : machines) {
    sum += elem - avr;
    minSteps = max(minSteps, abs(sum), elem - avr);
}
```

Експериментальна часть

Пусть n = machines.size().

Оценка памяти:

В коде создаются только вспомогательные int.

Итог: O(1)

Подсчёт асимптотики:

Один проход по machines для нахождения среднего: O(n)

Еще один проход для нахождения конечного ответа: O(n)

Итого: O(2n) = O(n)

Заключение

В ходе выполнения работы мною был реализован алгоритм поиска минимального числа перекладывания вещей из стиральных машин в соседние по одной штуке, для равномерного заполнения этих машин. Цель работы была достигнута путём успешной сдачи задания на LeetCode.

В качестве дальнейших исследований можно предложить решения других задач на жадные алгоритмы.

Приложение

Листинг кода файла main.cpp:

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

class Solution {
public:
    int average(const vector<int> machines) {
        int sum = 0;
        int size = machines.size();
        for (int i = 0; i < size; i++) {
            sum += machines[i];
        }
        if (sum % size == 0) {
            return sum / size;
        }
        return -1;
    }

    int findMinMoves(vector<int> machines) {
        int avr = average(machines);
        if (avr == -1) {
            return -1;
        }
        int sum = 0;
        int minSteps = 0;
        for (int elem : machines) {
            sum += elem - avr;
            minSteps = max({minSteps, abs(sum), elem - avr});
        }
        return minSteps;
    }
};
```

Подтверждение сдачи: