

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

Отчёт по лабораторной работе № 6
«Динамическое программирование»

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 динамического программирования и проанализировать его решение

Задачи:

Изучение теоретических основ дин. программирования

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 динамическое программирование, когда оно используется и зачем

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

В целом, так как изначально задача выбиралась с тэгом “Dynamic Programming”, то после первого прочитывания условия было примерно понятно, что он меня хотят, оставалось лишь набросать идею в голове и протестировать ее на маленьком массиве.

3. Реализация решения на c++

После выбора и проверки на работоспособность на маленьком массиве моего решения, осталось лишь переписать его на c++, в ее решении были использованы стандартные типы данных и `std::vector`, а также циклы.

4. Accepted и оптимизация

После того, как я увидел зеленую кнопку “Accepted” я попробовал немного оптимизировать код, поубирать лишние переменные, некоторые из них переименовать.

4. Экспериментальная часть

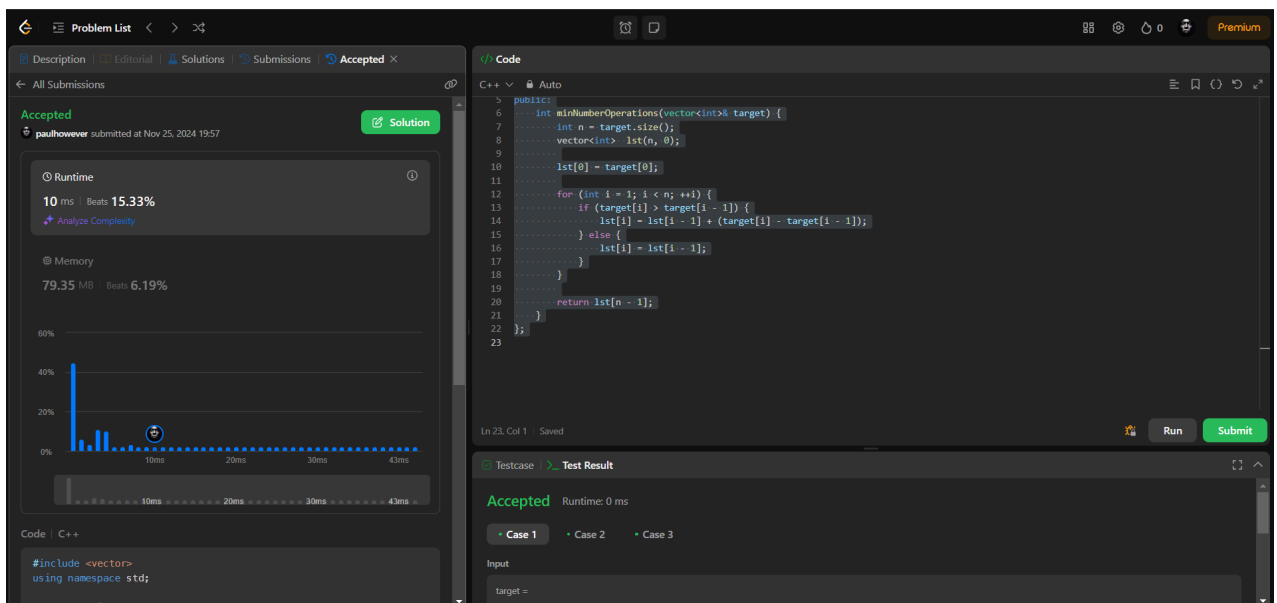
Подсчёт по памяти

1. Переменная `n` — занимает $O(1)$, 8 bite.
2. Массив `lst` — хранит промежуточные результаты для каждого элемента массива `target`, занимает $O(n)$ памяти, $(24+8n)$ bite.
3. Итоговая память: $O(n)$, $(32 + 8n)$ bite.


Подсчёт асимптотики

1. Используется только 1 основной цикл и арифм действия
- Итоговая временная сложность: $O(n)$



Leetcode Hard task DP Рисунок 1.1



1526. Minimum Number of Increments on Subarrays to Form a Target Array

Solved 

Hard

 Topics Companies Hint

Решение подходит для применения ДП, поскольку:

1. Задача сводится к вычислению минимального количества операций для каждого элемента $target[i]$ — т.е., на каждом шаге мы используем результаты предыдущих вычислений для оптимального подсчета текущего значения.
2. Мы используем массив lst для хранения промежуточных результатов, таким образом избегая повторных вычислений. Каждое значение $lst[i]$ строится на основе предыдущего значения $lst[i-1]$.

5. Заключение

В ходе выполнения лабораторной работы были изучены основные принципы динамического программирования. 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 тематику ДП или придумать усложнения для задачи решаемой в ходе данной лабораторной работы.

6. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода файла `leetCodeDp_6lab.cpp`

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

class Solution {
public:
    int minNumberOperations(vector<int>& target) {
```

```
int n = target.size();
vector<int> lst(n, 0);

lst[0] = target[0];

for (int i = 1; i < n; ++i) {
    if (target[i] > target[i - 1]) {
        lst[i] = lst[i - 1] + (target[i] - target[i - 1]);
    } else {
        lst[i] = lst[i - 1];
    }
}

return lst[n - 1];
};
```