кФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу:

Эль-Сибаи Амир

Академическая группа №J3112

Принято:

Ассистент (квалификационная категория "ассистент"), Дунаев Максим

Санкт-Петербург

2024

**Введение**

**Цель** — на практике изучить принципы динамического программирования.

**Задачи:**

* изучить динамическое программирование
* применить знания на практике решив medium задачи на leetcode

**Теоретическая подготовка:**

* Базовые знания языка C++
* Знания динамического программирования

**Реализация (Rotated Digits - https://leetcode.com/problems/rotated-digits/):**

Так как это задание на Динамическое программирование, мне нужно было выделить подзадачи, решить их и собрать в финальный код. Сначала я сделал проверку на присутствие цифр «3, 4, 5» в числе. Если эти цифры присутствовали, я не вносил их в счётчик чисел удовлетворяющих условию разворота на 180°.

**Реализация (Delete and Earn https://leetcode.com/problems/delete-and-earn/):**

Код выбирает число и добавляет его в «копилку», после чего удаляет все числа с индексами [i – 1] и [i + 1]. После этого делается следующий ход по тому же алгоритму. После того, как код прошёл 1 раз, он ищет способы набрать большее количество очков по тому же алгоритму.

**Экспериментальная часть**

**Rotated Digits**

**Приложение 1:**

class Solution {

public:

int rotatedDigits(int n) {

int count = 0;

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

int num = i;

bool hasGood = false;

while (num > 0) {

int digit = num % 10;

if (digit == 3 || digit == 4 || digit == 7) {

hasGood = false;

break;

}

if (digit == 2 || digit == 5 || digit == 6 || digit == 9) {

hasGood = true;

}

num /= 10;

}

count += hasGood;

}

return count;

}

};

Асимптотика и пространственная сложность:

Временная сложность - O(n \* log n), пространственная сложность - O(1) .

**Delete and Earn**

**Приложение 2:**

class Solution {

public:

int deleteAndEarn(vector<int>& nums) {

if (nums.empty()) return 0;

int maxVal = \*max\_element(nums.begin(), nums.end());

vector<int> points(maxVal + 1, 0);

for (int num : nums) {

points[num] += num;

}

int prev2 = 0, prev1 = points[0];

for (int i = 1; i <= maxVal; ++i) {

int current = max(prev1, prev2 + points[i]);

prev2 = prev1;

prev1 = current;

}

return prev1;

}

};

Асимптотика и пространственная сложность:

Временная сложность - O(n + maxVal), пространственная сложность - O(maxVal).