ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Динамическое программирование»

Выполнил работу

Игнатенок Филипп

Академическая группа №J3111

Принято

Вершинин Владислав

Санкт-Петербург

2024

**Введение**

Цель работы: Цель задачи — найти количество чисел в диапазоне от 1 до n, которые имеют хотя бы одну повторяющуюся цифру. Эта задача направлена на изучение свойств чисел, понимание работы с цифрами внутри числа и использование алгоритмических подходов для проверки условий.

**Теоретическая подготовка**

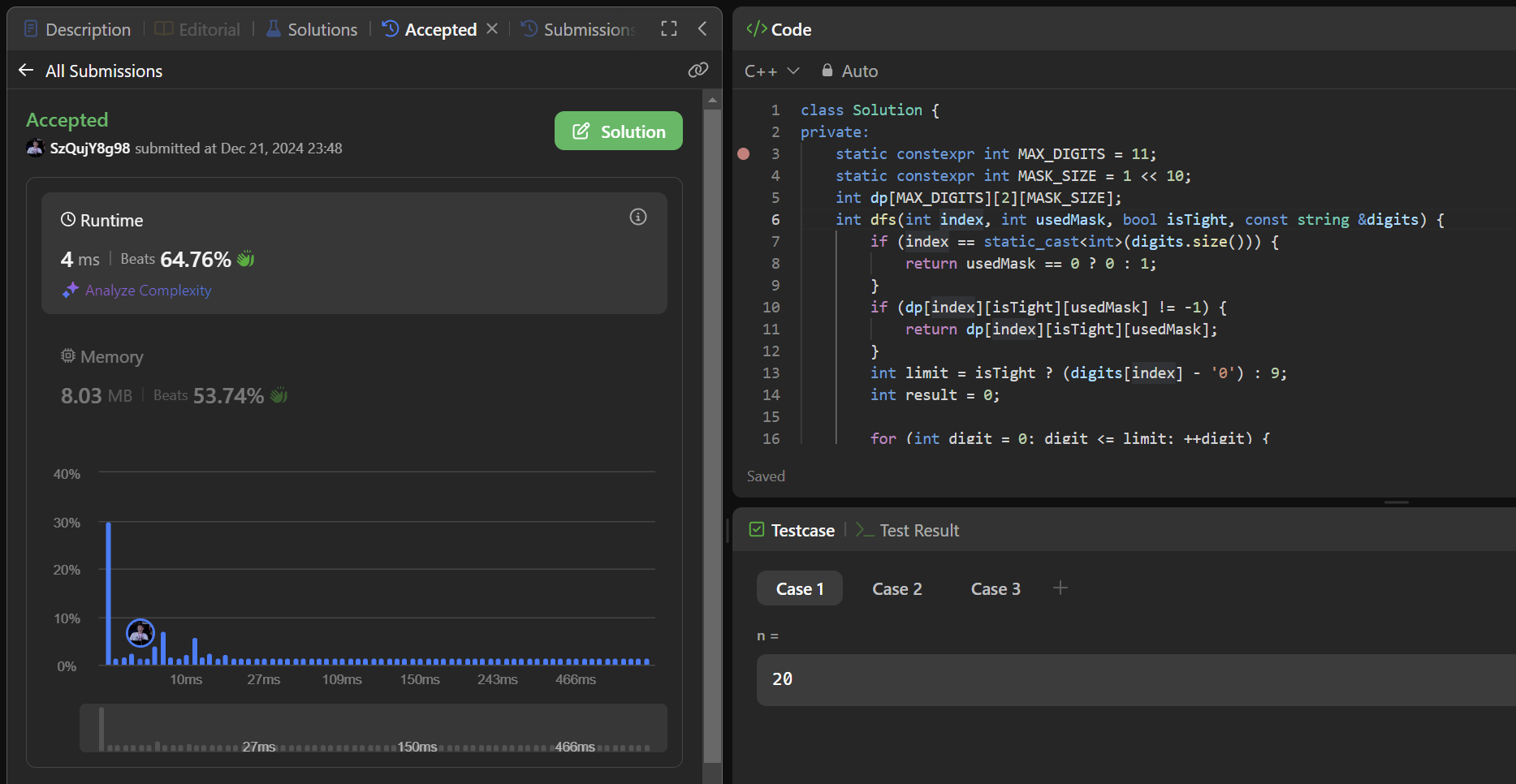
Динамическое программирование используется, потому что:

1. Разделение задачи на подзадачи: ДП помогает анализировать числа по разрядам и решать задачу для меньших диапазонов.
2. Избежание повторных вычислений: Промежуточные результаты сохраняются для повторного использования.
3. Хранение результатов: Используется мемоизация для сокращения времени вычислений.
4. Оптимизация перебора: Сокращается перебор всех чисел благодаря работе с состояниями (например, комбинациями цифр).

**Реализация**

Реализация заключается в использовании цифрового динамического программирования (Digit DP), где создаётся функция для перебора цифр числа по разрядам, сохраняя состояния (например, какие цифры уже встречались). Основная идея — подсчитать все числа до nnn, исключая те, где цифры уникальны, используя мемоизацию для промежуточных результатов.

**Тестовая часть**



**Заключение**

Данная сложность O(d\*2^d\*10) позволяет эффективно решать задачу даже для больших значений n (до 10^18), так как она зависит от количества разрядов числа, а не от самого значения n. Это делает возможным перебор всех состояний цифр с учётом их уникальности или повторения без полного перебора чисел, значительно ускоряя процесс по сравнению с наивным подходом.

**Приложения**

class Solution {

private:

static constexpr int MAX\_DIGITS = 11;

static constexpr int MASK\_SIZE = 1 << 10;

int dp[MAX\_DIGITS][2][MASK\_SIZE];

int dfs(int index, int usedMask, bool isTight, const string &digits) {

if (index == static\_cast<int>(digits.size())) {

return usedMask == 0 ? 0 : 1;

}

if (dp[index][isTight][usedMask] != -1) {

return dp[index][isTight][usedMask];

}

int limit = isTight ? (digits[index] - '0') : 9;

int result = 0;

for (int digit = 0; digit <= limit; ++digit) {

if (usedMask == 0 && digit == 0) {

result += dfs(index + 1, usedMask, isTight && (digit == limit), digits);

} else {

if ((usedMask & (1 << digit)) == 0) {

result += dfs(index + 1, usedMask | (1 << digit), isTight && (digit == limit), digits);

}

}

}

dp[index][isTight][usedMask] = result;

return result;

}

public:

int numDupDigitsAtMostN(int n) {

memset(dp, -1, sizeof(dp));

string digits = to\_string(n);

return n - dfs(0, 0, true, digits);

}

};