МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

«Отдельные вопросы алгоритмизации»

по дисциплине «Структуры и алгоритмы обработки данных (часть 2/2)»

Выполнил студент группы ИКБО-41-23 Попов А.В.

Принял Рысин М.Л.
Ассистент

Практические работы выполнены «___»_____2024 г. (подпись студента)

«Зачтено» «___»_____2024 г. (подпись преподавателя)

Москва 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Получение навыков применения метода динамического программирования для решения задач, сокращающих количество переборов, а также сравнение эффективности полного перебора и оптимизированных решений.

2 ЗАДАНИЕ

2.1 Формулировка задачи

Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода.

Персональный вариант №1:

Посчитать число последовательностей нулей и единиц длины n, в которых не встречаются две идущие подряд единицы. Использовать метод динамического программирования.

2.2 Решение

Полный перебор

Метод полного перебора заключается в генерации всех возможных последовательностей длины и проверке каждой из них на соответствие условию (отсутствие двух единиц подряд). Этот подход имеет экспоненциальную сложность, что делает его неэффективным для больших значений .Сложность алгоритма: n^2

Динамическое программирование

Динамическое программирование — это метод решения задач, который позволяет избегать избыточных вычислений путём хранения промежуточных результатов. Для данной задачи будем использовать два состояния:

- количество последовательностей длины , заканчивающихся на 0.
- количество последовательностей длины , заканчивающихся на 1.

Сложность алгоритма: 2 * (n – 1).

Алгоритм полного перебора

1. Сгенерировать все возможные последовательности длины .
2. Проверить каждую последовательность на соответствие условию.
3. Подсчитать количество допустимых последовательностей.

Алгоритм динамического программирования

1. Создать переменные для хранения текущего количества последовательностей, заканчивающихся на 0 и 1.
2. Использовать рекуррентные соотношения для вычисления количества последовательностей длины .
3. Вывести результат как сумму двух состояний.

2.3 Реализация задачи

```cpp
bool isValid(const vector<int>& sequence) {
    for (size_t i = 1; i < sequence.size(); i++) {
        if (sequence[i] == 1 && sequence[i - 1] == 1) {
            return false;
        }
    }
    return true;
}

int bfCount(int n, int& bfOperations) {
    int total = 0;
    int combinations = pow(2, n);

    for (int i = 0; i < combinations; ++i) {
        vector<int> sequence;
        for (int j = 0; j < n; ++j) {
            sequence.push_back((i >> j) & 1);
        }
        bfOperations++;
        if (isValid(sequence)) {
            total++;
        }
    }
    return total;
}
```

Рисунок 1 — Функция перебора последовательностей в лоб и вспомогательная функция проверки последовательности

```cpp
int dpCount(int n, int& dpOperations) {
    if (n == 1) return 2;
    if (n == 2) return 3;

    int dp_0 = 1, dp_1 = 1;
    int new_dp_0, new_dp_1;

    for (int i = 2; i <= n; i++) {
        new_dp_0 = dp_0 + dp_1;
        new_dp_1 = dp_0;

        dp_0 = new_dp_0;
        dp_1 = new_dp_1;

        dpOperations += 2;
    }

    return dp_0 + dp_1;
}
```

Рисунок 2 — Функция подсчета последовательностей методом динамического программирования

```cpp
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    int n;
    cout << "Введите длину последовательности n: ";
    cin >> n;

    int bfOperations = 0;
    int dpOperations = 0;

    int resultDP = dpCount(n, dpOperations);
    cout << "Динамическое программирование:\n";
    cout << "Количество последовательностей: " << resultDP << endl;
    cout << "Количество операций: " << dpOperations << endl;

    int resultBF = bfCount(n, bfOperations);
    cout << "\nПолный перебор:\n";
    cout << "Количество последовательностей: " << resultBF << endl;
    cout << "Количество операций: " << bfOperations << endl;

    return 0;
}
```

Рисунок 3 — Основная функция программы

2.4 Тестирование

Длина последовательности (n)	Результат (полный перебор): кол-во последовательностей и кол-во операций	Результат (динамическое программирование): кол-во последовательностей и кол-во операций
5	13: 32 операции	13: 8 операций
10	144: 1024 операции	144: 18 операций
15	1597: 32768 операций	1597: 28 операций

Таблица 1 — Результаты тестирования

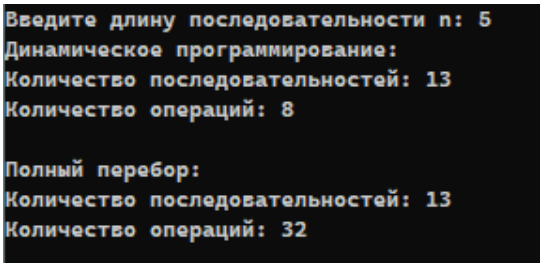

Рисунок 4 — Длина n = 5

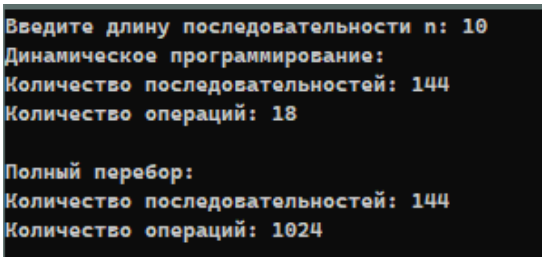

Рисунок 5 — Длина n = 10

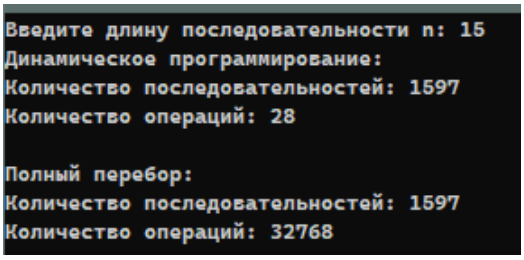

Рисунок 6 — Длина n = 15

3 ВЫВОД

В ходе работы были реализованы два метода решения задачи подсчета количества бинарных последовательностей заданной длины, где отсутствуют две единицы подряд: метод полного перебора и метод динамического программирования. Тестирование подтвердило корректность работы обоих подходов, кроме того было доказано, что метод динамического программирования значительно превосходит метод полного перебора по эффективности, особенно при увеличении длины последовательностей. Это связано с тем, что динамическое программирование сокращает сложность задачи с экспоненциальной до линейной.

В результате выполнения работы приобретены навыки использования динамического программирования для оптимизации вычислений и проведен анализ эффективности различных подходов к решению задачи.