Лабораторная работа № 7 по курсу дискретного анализа: Динамическое программирование

Выполнила студентка группы 08-308 МАИ *Шевлякова София.*

Условие

Задано целое число n. Необходимо найти количество натуральных (без нуля) чисел, которые меньше n по значению и меньше n лексикографически (если сравнивать два числа как строки), а так же делятся на m без остатка.
Формат ввода: В первой строке строке задано $1 \leq n \leq 10^{18}$ и $1 \leq m \leq 10^5$.
Формат вывода: необходимо вывести количество искомых чисел.

Метод решения

Динамическое программирование - метод решения задач, при котором сложная задача разбивается на более простые. Решение сложной задачи составляется из решений более простых подзадач. Попытаемся применить этот метод к нашей задаче.
Очевидно, что операция лексикографического сравнения довольно затратна, поэтому попытаемся избавиться от нее. Число меньшее по значению может оказаться лексикографически больше, только когда оно меньше по длине, и, например, его первая цифра больше, чем первая цифра исходного числа, или первые цифры равны, но больше вторая цифры и другие вариации. Давайте рассмотрим сначала все числа длины 1, которые меньше первой цифры числа n, потом числа длины 2, которые меньше числа, образуемого первыми двумя цифрами числа n, и так пока мы не дойдем до самого числа n. То есть мы разбиваем нашу задачу на подзадачи, при этом все числа, являющиеся ответом для первого этапа, гарантированно будут меньше по значению и лексикографически, чем число длины 2. Поэтому ответом для второго этапа будет решение предыдущего шага и результат, полученный для чисел длины 2. Именно в этом и заключается идея динамического программирования в решении этой задачи.
Также мы можем оптимизировать нахождение чисел, делящихся на m на определенном промежутке. Для этого можем воспользоваться формулой $count = B/m - A/m$, где B - правая граница, равная исходному числу, состоящему из k-первых символов, а A - левая граница, равна минимальному числу из k символов, то есть 10^{k-1}.

Описание программы

В данной программе содержится один файл main.cpp, в котором реализован алгоритм решения задачи методом динамического программирования.
Основные этапы работы программы:

- Вспомогательные функции: log и power, это было необходимо, чтобы упростить обращение с 10, находить длину числа и возводить 10 в нужную степень. Ведь

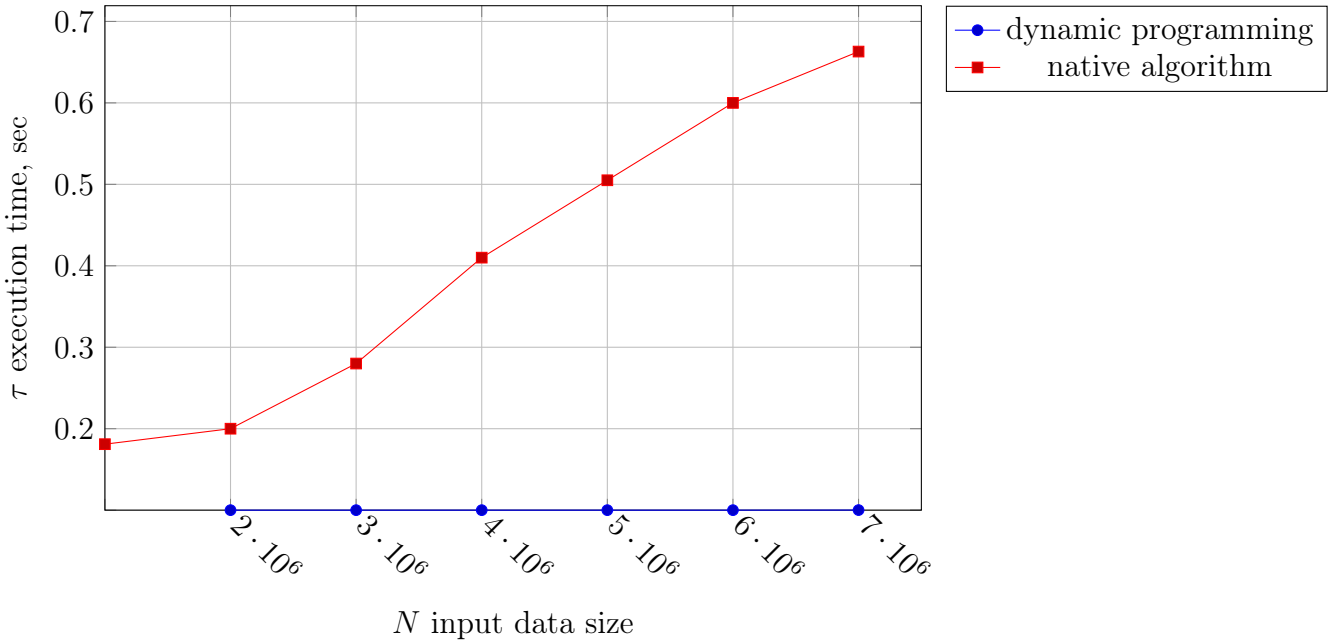
стандартные функции из библиотеки <cmath> работают с дробными числами, при больших входных данных могло возникнуть переполнение.

- В функции main создадим массив dp, где в dp[i] будет храниться результат для числа длины i+1. Получается, что итоговый ответ хранится в ячейке dp[length_n - 1], где length_n - длина исходного числа n. Все значения этого массива заполняются в цикле, за исключением первой ячейки, так как на каждом шаге мы должны будем использовать значение предыдущего вычисления.

Дневник отладки

№	Описание ошибки	Способ устранения
1	WA4: на тесте, когда n = 1 и m = 1, последнее условие, необходимое для того, чтобы если само число n, которое делится без остатка на m, не вошло в конечный ответ, заносило в ячейку значение -1.	Прописала дополнительное условие, что при этом в ячейки должно быть значение большее 0.
2	WA8: формула для нахождения первой цифры, выглядела так: $n/(10 * length\_n - 1)$, видимо, она работала на первых тестах.	Исправила формулу для нахождения первой цифры числа на правильную $n/power(length\_n - 1)$.
3	WA27: формула работает правильно только если левая граница не делится без остатка на m, потому что в этом случае при целочисленном делении мы не потеряем еще один ответ.	Необходимо увеличить количество чисел, делящихся без остатка на m, если левая граница делится без остатка на m.

Тест производительности



По результатам тестирования не особо видно, что сложность выполнения алгоритма является логарифмической. Но анализируя алгоритм, мы можем сказать, что количество итераций в цикле равно длине входного числа n, длину десятичного числа можно определить через логарифм. Решение этой задачи динамическим программированием соответствует сложности O($log_{10}n$), где n — входное число.
Также я написала наивный алгоритм решения этой задачи, который рассматривает каждое число от 1 до n, проверяет его делимость, а потом проверяет, что оно лексикографически меньше, сравнивая строчки, сложность такого алгоритма составляет O($n + n * log_{10}n$) = O($n * log_{10}n$).

Выводы

Проделав лабораторную работу, я познакомилась с новым подходом решения алгоритмических задач — динамическим программированием, а также написала интересный алгоритм, который работает существенно быстрее, чем наивный для данной задачи.
Динамическое программирование позволяет разработать точные и относительно быстрые алгоритмы для решения сложных задач в то время, как решение перебором слишком медленное, а жадный алгоритм не всегда даёт правильный результат.