Лабораторная работа № 7 по курсу дискретного анализа: Динамическое программирование

Выполнил студент группы М8О-308Б-21 *Армишев Кирилл.*

Вариант 2: Количество чисел

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.

Метод решения

Динамическое программирование (DP) - это метод программирования, используемый для решения различных задач, которые можно разбить на подзадачи. Основная идея состоит в том, чтобы решать каждую подзадачу только один раз и сохранять ее решение для будущего использования, чтобы избежать избыточных вычислений. DP широко применяется в задачах оптимизации, поиске наиболее эффективного пути и многих других областях.

Этапы построения алгоритма с помощью динамического программирования обычно включают в себя следующие шаги:

- Определение оптимизируемой задачи
- Разбиение задачи на подзадачи
- Определение функции подсчета для каждой подзадачи
- Создание таблицы (массива): Для хранения решений подзадач создается таблица (обычно двумерный массив), в которой значения будут вычисляться и заполняться по мере необходимости.
- Итеративное заполнение таблицы(массива): Начиная с наименьших подзадач и двигаясь к исходной задаче, выполняется итеративное заполнение таблицы(массива), используя рекуррентные формулы. Решения более крупных подзадач строятся на основе решений меньших подзадач.
- Получение решения исходной задачи: После того как таблица(массив) полностью заполнена, решение исходной задачи может быть получено из таблицы(массива), как значение, которое соответствует исходным параметрам задачи.
- Оптимизация : В некоторых случаях, чтобы улучшить эффективность, можно провести оптимизации, такие как использование мемоизации (хранение решений подзадач в кеше для избежания повторных вычислений) или применение более эффективных структур данных.

Динамическое программирование применяется в различных задачах, включая задачи нахождения наибольшей общей подпоследовательности, нахождения наименьшей стоимости пути в графе, оптимизации рюкзака и многие другие.

Метод решения моего варианта задания

Просто посчитать кол-во делителей, меньших m и кратных n, достаточно просто, по формуле n/m. Но как из этого кол-ва удалить все делители, которые лексикографически больше числа m? Можно заметить, что для однозначных чисел лексикографически больших делителей нет, для 2-значных - они ВСЕ точно лежат на отрезке [1,9], для 3-значных на отрезке [1,99] и т.д. Поэтому по формуле $(10^{|str(n)|-1}-1)/m$ получим кол-во делителей, которые меньше m по значению, но больше лексикографически(для одного старшего разряда). Рассмотрим на примере(для более лучшего понимания, по возрастанию кол-ва разрядов):
Ввод:
1532 3
Решение:

- 1: 1/3-0/3=0 - кол-во делителей на отрезке [1,1].
- 2: 15/3-9/3=2 - кол-во делителей на отрезке [10,15].
- 3: 153/3-99/3=18 - кол-во делителей на отрезке [100,153]
- 4: 1532/3-999/3=510-333=177 - кол-во делителей на отрезке [1000,1532]

То есть для решения задачи необходимо посчитать кол-во делителей, которые лежат только в тех отрезках, в которых все числа лексикографически меньше m. В конце просуммировать их. Сложность алгоритма: $O(lg(n))$

Описание программы

Реализация довольно простая, состоит из следующих шагов:

1. Создание массива count размерностью(length), где length - кол-во разрядов в числе n.
2. Проход по элементам массива и подсчет кол-ва делителей для каждого разряда по формуле: $n/m-(10^{|str(n)|-1}-1)/m$
3. Сумма всех элементов массива - это и есть ответ.

Дневник отладки

1. 4 окт 2023, 19:29:04 WA на 12 тесте.

   Причина: в одном месте использовал long long вместо int. После замены программа прошла все тесты.

Тест производительности

Сравним написанную программу с наивным подсчетом кол-ва делителей: будем проверять все числа на отрезке [1, n), чтобы они подходили под условие(циклом for).
Очевидно, что разница во времени работы алгоритмов будет очень большой.

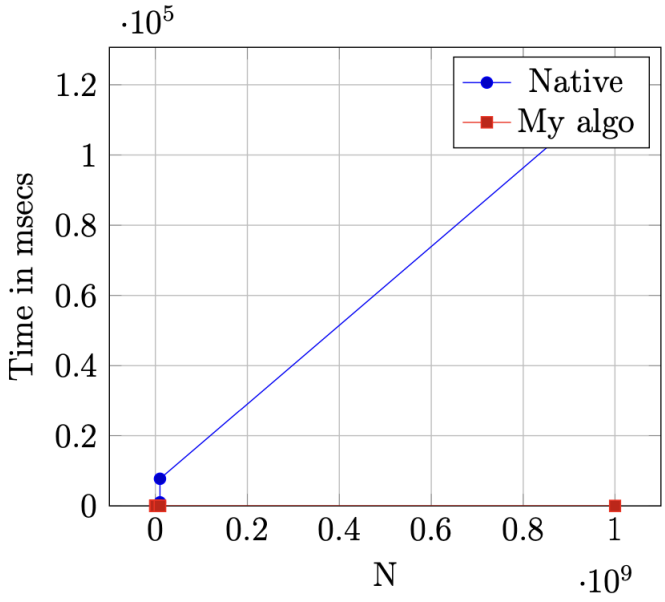

Рис. 1: Сравнение с наивным алгоритмом

Выводы

Выполнив данную лабораторную работу, я изучил классические задачи динамического программирования и их методы решения, реализовал алгоритм для своего варианта задания. Разобрался с данным подходом построения алгоритмов. Динамическое программирование позволяет разработать точные и быстрые алгоритмы для решения сложных задач, в то время, как решение перебором слишком медленное, а жадный алгоритм не всегда даёт оптимальный результат.