Лабораторная работа №7 по курсу дискрeтного анализа:

динамическое программирование.

Выполнил студент группы М80-308Б-20 Морозов Артем Борисович.

# Условие

Вариант алгоритма:

3. 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 также делятся на m без остатка.

Входные данные:

В первой строке строке задано 1 ≤ n ≤ 1018 и 1 ≤ m ≤ 105.

Выходные данные:

Необходимо вывести количество искомых чисел.

# Метод решения Метод решения данной задачи целиком и полностью основан на такой идее, как динамическое программирование. Дело в том, что наивный алгоритм в данной задаче будет работать очень долго – это будет O(n), где n – входное число, относительно которого мы проверяем условие на деление. Как же можно ускорить? И тут нам на помощь приходит вышеупомянутое динамическое программирование. Оказывается, мы можем решить задачу иным способом: мы можем разбить наше решение на сумму решений на интервалах от 10^i до di, где di – это какая-то i-тая подстрока нашего числа m. Чтобы было понятнее, приведу пример: допустим, у нас есть число 621, и мы хотим для него решить нашу задачу. Тогда мы разбиваем нашу задачу на 3 интервала: (0, 6), (10, 62), (100, 621), и считаем на них количество таких чисел, которые лексикографически меньше n и n при делении на них не дает остатка. Преимущество этого алгоритма в том, что мы сразу же получаем неимоверный буст по сложности – вместо прежних O(n) мы получаем O(k), где k – количество цифр в числе n.

Описание программы

Программа состоит из одного файла – лаконичного решения задачи. На вход поступает два числа, при этом мы сразу вводим левые и правые границы. Далее в цикле считаем минимальное число в интервале, большее 10^i и делящееся на m, а также максимальное число в интервале, меньшее di и тоже делящееся на m. Далее считаем ответы на каждом из интервалов.

# Дневник отладки В процессе отладки программы была убрана неточность с поиском на интервале, а также финальное условие проверки n%m == 0. Помимо этого, не был сразу учтен формат входных данных, поэтому int пришлось менять на long long.

# Тест производительности В своем бенчмарк-тесте я решил сравнить алгоритм динамического программирования с вышеупомянутым наивным алгоритмом.

1 тест: 42 3

 - наивный алгоритм

 - алгоритм динамического программирования

2 тест: 960 30

 - наивный алгоритм

 - алгоритм динамического программирования

3 тест: 10000 10

 - наивный алгоритм

 - алгоритм динамического программирования

4 тест: 1000000000 104

 - наивный алгоритм

 - алгоритм динамического программирования

Как мы видим, уже на самом простом тесте время выполнения двух алгоритмов различается более, чем в 2 раза. На больших же данных разница растет еще сильнее.

# Недочёты

Недочетов в программе обнаружено не было, однако стоит упомянуть, что программа работает только при условии корректного ввода, так как была разработана исключительно в учебных целях. Любой неправильный ввод может убить работоспособность моей программы.

# Выводы

Данная лабораторная работа помогла мне лучше осознать такую алгоритмическую идею, как динамическое программирование. Оно очень хорошо тем, что позволяет решать задачи относительно быстро, при этом деля задачу на подзадачи. Я решил задание по варианту и отработал навыки динамического программирования на практике.