Лабораторная работа № 7 по курсу дискретного анализа: динамическое программирование

Выполнил студент группы 08-208 МАИ *Сизонов Артём.*

Условие

Кратко описывается задача:

1. При помощи метода динамического программирования разработать алгоритм решения задачи, определяемой своим вариантом; оценить время выполнения алгоритма и объем затрачиваемой оперативной памяти. Перед выполнением задания необходимо обосновать применимость метода динамического программирования.

   Разработать программу на языке C или C++, реализующую построенный алгоритм.

2. Вариант задания: **3. 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

Применимость метода динапического программирования. Для решения задачи методом динамического программирования необходимо:
1. Определить подзадачи более мелкой размерности.
2. Найти функцию, связывающую решение искомой задачи с решениями подзадач.
Важно, что искомая задача отличается от всех подзадач только размерностью.
Решение варианта задания: Пусть заданы, соответственно, числа n и m. Тогда, k, равное длине числа n, является размерность искомой задачи. Легко заметить, что для чисел одинаковой длины арифметическое и лексикографическое равенства эквивалентны. Также все числа длиной k больше, чем числа длиной $k-1$. Тогда, для того, чтобы определить решение искомой задачи нужно выяснить, какое количество чисел является решением подзадачи размерности $k-1$. Далее остаётся сложить результат с числом, соответсвующим количеству чисел, меньших n и делящихся на m длиной k. Это можно сделать по формуле: upDiv(curr, m)/m - downDiv(10^{k-1}, m)/m + 1, где curr – текущее число длиной k, upDiv – поиск наименьшего числа, большего или равного curr и делящегося на m, downDiv – поиск наибольшего числа, меньшего или равного curr и делящегося на m. Формально, формула:
$f(k) = f(k-1) + upDiv(curr, m)/m - downDiv(10^{k-1}, m)/m + 1$

Описание программы

Программа написана одним файлом.
uint64_t toInt(string&, int, int) – функция, возвращающая число, соответсвующее числовому промежутку с индекса l до r строки.
uint64_t downDividend(uint64_t a, uint64_t m) – возвращает наибольшее число, меньшее или равное a и делящееся на m.
uint64_t upDividend(uint64_t a, uint64_t m) – возвращает наименьшее число, большее или равное a и делящееся на m.

Дневник отладки

Посылка 1: *Status:* Wrong answer at test 20.t. Проблема: для вычисления промежуточных значений использовался тип *int*, а не *uint*64_*t*.
Посылка 2: *Status:* OK.

Тест производительности

```
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN7$ cat test1
99 4
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN7$ time cat test1 |  ./a.out
24

real    0m0.007s
user    0m0.004s
sys     0m0.000s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN7$ cat test2
9999999999999999 4
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN7$ time cat test2 |  ./a.out
2499999999999999

real    0m0.007s
user    0m0.008s
sys     0m0.000s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN7$
```

Недочёты

Выявленных недочётов нет.

Выводы

Динамическое программирование решает большой спектр задач. Данный метод трудно обобщить, ведь, чаще всего, затруднительно понять, решается ли вообще задача методом динамического программирования. Ведь точных необходимых и достаточных условий возможности решения конкретной задачи с помощью метода динамического программирования не сформулировано. Если задача решается данным методом, то необходимо определить функцию, связывающую решение искомой задачи с решением подзадач. Важное отличие данного метода от перебора это то, что решение каждой

подзадачи происходит только один раз. Также, все подзадачи, любого уровня, должны иметь одну топологию.
Решенная задача имеет вычислительную сложность $O(1)$, так как вычисления зависят от длины числа n, которая ограничена числом 18. Пространственная сложность также $O(1)$.