Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Дискретный анализ»

Студент: М. В. Леухин
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20
Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №7

Задача:

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.

1 Описание

Для решения задачи воспользуемся подходом динамического программирования. Так как искомые делители должны быть меньше лексикографически, то необходимо просматривать только следующие диапазоны: $[10^k, n], [10^{k-1}, n/10], [10^{k-2}, n/10^2], ..., [1, n/10^k]$. Таким образом мы свели задачу нахождения всех делителей к задаче суммирования делителей, наёденных на каждом из вышеописанных отрезков.

На них искать делители будем следующим образом: поделим левую и правую границы отрезка на заданный делитель (учитывая, что сама левая граница тоже может делиться нацело), после чего искомое количество делителей станет просто длиной отрезка.

```
1
2  #include <iostream>
3  #include <cmath>
4
5  using namespace std;
6
7  size_t length(long long n) {
8      return to_string(n).length();
9  }
10
11 long long numberOfDivisibleInRange(long long leftBound, long long rightBound, int
       divider) {
12
13     long long l = leftBound / divider + (leftBound % divider == 0 ? 0 : 1);
14     long long r = rightBound / divider;
15
16     return r - l + 1;
17
18 }
19
20 long long dp(long long currentNumber, int divider) {
21
22     if (currentNumber < divider) {
23         return 0;
24     }
25
26     return numberOfDivisibleInRange(pow(10, length(currentNumber) - 1), currentNumber,
           divider) +
27            dp(currentNumber / 10, divider);
28
29 }
30
31 int main() {
32
33     long long n;
34     int m;
```

```
35
36      while (cin >> n >> m) {
37          cout << dp(n, m) - (n % m == 0 ? 1 : 0) << "\n";
38      }
39
40  }
```

2 Консоль

```
C:/Users/leyhi/CLionProjects/contest/cmake-build-debug/lab7.exe
>42 3
11
>126 8
3
>4048 78
43
```

3 Тест производительности

Тест производительности представляет из себя следующее: сравнение эффективности реализованного реализованного алгоритма и наивного перебора.

```
C:/Users/leyhi/CLionProjects/contest/cmake-build-debug/lab7.exe
11 1
DP: 2
DP: 757 mcs.
Naive: 2
Naive: 774 mcs.
C:/Users/leyhi/CLionProjects/contest/cmake-build-debug/lab7.exe
111111 11
DP: 1124
DP: 607 mcs.
Naive: 1124
Naive: 853 mcs.
C:/Users/leyhi/CLionProjects/contest/cmake-build-debug/lab7.exe
1111111111 11
DP: 11223348
DP: 487 mcs.
Naive: 11223348
Naive: 4542429 mcs.
```

Из полученных результатов видно, что реализованный алгоритм динамического программирования, имеющий сложность $O(k)$, где k — количество разрядов в n, значительно превосходит по быстродействию наивный алгоритм перебора со сложностью $O(n * k)$.

4 Выводы

Выполнив седьмую лабораторную работу по курсу «Дискретный анализ», я познакомился с динамическим подходом к решению задач. Динамическое программирование хорошо подходит для решения задач, результат которых зависит от решения подзадач. Соответственно, и применим он только в том случае, если разбиение исходной задачи на более маленькие вообще возможно.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Динамическое программирование — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Динамическое-программирование` (дата обращения: 04.11.2022).