Лабораторная работа № 6 по курсу: Дискретный анализ

Выполнил студент группы М8О-208Б-17 МАИ *Милько Павел.*

Задача

При помощи метода динамического программирования разработать алгоритм решения задачи, определяемой своим вариантом; оценить время выполнения алгоритма и объём затрачиваемой оперативной памяти. Перед выполнением задания необходимо обосновать применимость метода динамического программирования. Разработать программу на языке C или C++, реализующую построенный алгоритм. Формат входных и выходных данных описан в варианте задания.

Вариант 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.

Информация

Динамическое программирование – это способ решения сложных задач, путём разбиения их на подзадачи.Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с “наивными” методами, можно значительно сократить.

Основа способа состоит в том, чтобы не решать меньше подзадачи несколько раз, а затем объединить решения.

Этапы построения алгоритма решения подзадач:

- Описать структуру оптимального решения.
- Составить рекурсивное решение для нахождения оптимального решения.
- Вычисление значения, соответствующего оптимальному решению, методом восходящего анализа.
- Непосредственное нахождение оптимального решения из полученной на предыдущих этапах информации.

Метод решения

Для решения моей задачи можно применить динамическое программирование, выделив следующие подзадачи:

Для задачи n, m, где len_n – количество разрядов числа n подзадача будет состоять из количества чисел, удовлетворяющих условию и по длине равных len_n :

$$T(n, m) = n \ div \ m - 10^{len_n - 1} \ div \ m$$

Важно не забыть про границы отрезка, если число $10^{len_n - 1}$ делится на m нацело, то его нужно добавить к результату. Следующая подзадача будет решаться для чисел $n \ div \ 10$ и m.

Исходный код

main.cpp

```
#include <cmath>
#include <iostream>

inline long long len(long long num)
{
    long long res = 0;
    while (num /= 10, res++, num)
        ;
    return res;
}

int main()
{
    long long n, m, res;
    std::cin >> n >> m;
    res = n % m ? 0 : -1;
    while (n) {
        if (len(n) != len(n - 1))
            break;
        long long temp = pow(10, len(n) - 1);
        temp = temp % m || !temp ? temp : temp - 1;
        res += n / m - temp / m;
        n /= 10;
    }
    std::cout << (res < 0 ? 0 : res) << std::endl;
    return 0;
}
```

Генератор тестов

```
#!/usr/bin/python
from sys import exit,argv
```

```
def get_res(n, m):
    count = 0
    for i in range(1, n):
        if str(i) < str(n) and i % m == 0:
            count += 1
    return count


def main():
    if len(argv) < 3:
        print("Usage: {} n m".format(argv[0]))
        return 1
    i = int(argv[1])
    j = int(argv[2])
    test = open("./tests/{}.t".format(argv[1]+"_"+argv[2]), 'w')
    ans = open("./tests/{}.a".format(argv[1]+"_"+argv[2]), 'w')
    print(i, j, file=test)
    print(get_res(i, j), file=ans)
    test.close()
    ans.close()
    return 0

if __name__ == "__main__":
    exit(main())
```

Выводы

Динамическое программирование довольно распространено, в различных олимпиадах около половины задач решаются с помощью ДП. Всюду, где встречаются подзадачи, и где их можно легко выделить, динамическое программирование позволяет существенно ускорить работу программы. Метод является достаточно гибким, так как это не алгоритм, а метод построения алгоритмов.