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: А. В. Семин
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №7

Задача: Имеется натуральное число n. За один ход с ним можно произвести следующие действия:

- Вычесть единицу.
- Разделить на два.
- Разделить на три.

При этом стоимость каждой операции – текущее значение n. Стоимость преобразования - суммарная стоимость всех операций в преобразовании. Вам необходимо с помощью последовательностей указанных операций преобразовать число n в единицу таким образом, чтобы стоимость преобразования была наименьшей. Делить можно только нацело.

Формат ввода
В первой строке строке задано $2 \leqslant n \leqslant 10^7$.

Формат вывода
Выведите на первой строке искомую наименьшую стоимость. Во второй строке должна содержаться последовательность операций. Если было произведено деление на 2 или на 3, выведите /2 (или /3). Если же было вычитание, выведите -1. Все операции выводите разделяя пробелом.

1 Описание

Описание метода динамического программирования, приведенной из книги [1]:

> Динамическое программирование позволяет решать задачи, комбинируя решения вспомогательных задач. (Термин “программирование” в данном контексте означает табличный метод, а не составление компьютерного кода.) В главе 2 уже было показано, как в алгоритмах разбиения задача делится на несколько независимых подзадач, каждая из которых решается рекурсивно, после чего из решений вспомогательных задач формируется решение исходной задачи. Динамическое программирование, напротив, находит применение тогда, когда вспомогательные задачи не являются независимыми, т.е. когда разные вспомогательные задачи используют решения одних и тех же подзадач. В этом смысле алгоритм разбиения, многократно решая задачи одних и тех же типов, выполняет больше действий, чем было бы необходимо. В алгоритме динамического программирования каждая вспомогательная задача решается только один раз, после чего ответ сохраняется в таблице. Это позволяет избежать одних и тех же повторных вычислений каждый раз, когда встречается данная подзадача.
> Динамическое программирование, как правило, применяется к задачам оптимизации (optimization problems). В таких задачах возможно наличие многих решений. Каждому варианту решения можно сопоставить какое-то значение, и нам нужно найти среди них решение с оптимальным (минимальным или максимальным) значением. Назовем такое решение одним из возможных оптимальных решений. В силу того, что таких решений с оптимальным значением может быть несколько, следует отличать их от единственного оптимального решения. Процесс разработки алгоритмов динамического программирования можно разбить на четыре перечисленных ниже этапа.
>
> 1. Описание структуры оптимального решения.
> 2. Рекурсивное определение значения, соответствующего оптимальному решению.
> 3. Вычисление значения, соответствующего оптимальному решению.
> 4. Составление оптимального решения на основе информации, полученной на предыдущих этапах.

Описание решения:
Для решения нам необходим массив размера n, на каждой позиции i хранящий минимальную сумму операций, необходимых для преобразования числа i к 1 с помощью

доступных операций. Таким образом, по итогам работы алгоритма на позиции n-1 будет лежать необходимый ответ. На каждом шаге происходит самый оптимальный выбор. Ведя префиксный массив, на каждом этапе мы решаем, каким образом выгоднее получить число i из предыдущих: прибавить 1, умножить на два или умножить на 3 - и в соответствии с этим вычисляем минимально необходимую сумму операций для данного числа.

2 Исходный код

Файл main.cpp:

```cpp
#include <bits/stdc++.h>

#define endl '\n'

enum Operation {
  subtractionOfOne, // -1
  divisionByTwo, // /2
  divisionByThree // /3
};

void printOperations(const std::vector<Operation> operations, int num) {
  while (num > 1) {
    switch (operations[num]) {
    case subtractionOfOne:
      std::cout << "-1 ";
      --num;
      break;
    case divisionByTwo:
      std::cout << "/2 ";
      num /= 2;
      break;
    case divisionByThree:
      std::cout << "/3 ";
      num /= 3;
      break;
    default:
      std::cout << "broken" << endl;
      break;
    }
  }
}

int main() {
    std::ios_base::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int n;
  std::cin >> n;
  std::vector<int> prefixSumm(n + 1);
  std::vector<Operation> operations(n + 1, subtractionOfOne);

  for (int i = 2; i < prefixSumm.size(); i++) {
    prefixSumm[i] = prefixSumm[i - 1] + i;

    if (i % 2 == 0 && prefixSumm[i / 2] + i < prefixSumm[i]) {
      prefixSumm[i] = prefixSumm[i / 2] + i;
      operations[i] = divisionByTwo;
```

```
47      }
48      if (i % 3 == 0 && prefixSumm[i / 3] + i < prefixSumm[i]) {
49        prefixSumm[i] = prefixSumm[i / 3] + i;
50        operations[i] = divisionByThree;
51      }
52
53    }
54
55    std::cout << prefixSumm[n] << endl;
56
57    printOperations(operations, n);
58  }
```

3 Консоль

console input:

```
82
```

console output:

```
202
-1 /3 /3 /3 /3
```

4 Тест производительности

Тест производительности представляет собой сравнение реализованного мной метода динамического программирования за O(n) и наивного алгоритма.

Сравнение производится путем запуска запуска теста на двух больших тестах.

В результате работы *benchmark.cpp* видны следующие результаты:

```
a@WIN-THNQL51M105:~/Desktop/DA/lab7$ make
g++ benchmark.o -o solution
a@WIN-THNQL51M105:~/Desktop/DA/lab7$ ./solution <test.txt
1000000
Search time DP = 0.014698
Search time default = 1.03052
```

5 Выводы

Выполнив седьмую лабораторную работу по курсу «Дискретный анализ», мною был изучен алгоритм динамического программирования, его преимущества, случаи и способы его применения на практике для решения задач.

Динамическое программирование позволяет решать задачи оптимизации, максимизации и минимизации наиболее быстрым способом, но при этом не всегда может найти все решения для задачи.

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.))