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: Д. М. Чистяков
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20
Дата:
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №7

1 Описание

Динамическое программирование
в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые 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.

Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Метод динамического программирования сверху — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. Динамическое программирование снизу включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

2 Исходный код

Фрагмент кода, где применяются рекуррентные формулы:

```
#include <iostream>
#include <vector>

const int OP_MINUS_ONE = 0;
const int OP_DIVIDE_TWO = 1;
const int OP_DIVIDE_THREE = 2;

int main() {
 int n;
 std::cin >> n;
 std::vector<int> coast(n + 1);
 std::vector<int> res(n + 1);

 for (int i = 2; i <= n; i++) {
   coast[i] = coast[i - 1] + i;
   res[i] = OP_MINUS_ONE;

   if (i % 2 == 0 && coast[i / 2] + i < coast[i]) {
     coast[i] = coast[i / 2] + i;
     res[i] = OP_DIVIDE_TWO;
   }
   if (i % 3 == 0 && coast[i / 3] + i < coast[i]) {
     coast[i] = coast[i / 3] + i;
     res[i] = OP_DIVIDE_THREE;
   }

 }
 std::cout << coast.back() << '\n';

 for (int i = n; i > 1;) {
   switch (res[i]) {
     case OP_MINUS_ONE:
     std::cout << "-1";
     i--;
     break;
     case OP_DIVIDE_TWO:
     std::cout << "/2";
     i /= 2;
     break;
     case OP_DIVIDE_THREE:
     std::cout << "/3";
     i /= 3;
     break;
   }
```

```
45      if (i != 1) {
46        std::cout << " ";
47      }
48    }
49    std::cout << '\n';
50
51    return 0;
52  }
```

3 Консоль

```
den@vbox:~/Документы/DA/lab7$ ./a.out
82
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:

```
den@vbox:~/Документы/DA/lab7$ ./a.out <test.txt
1000000
Search time DP = 0.014698
Search time default = 1.03052
```

5 Выводы

Выполнив седьмую лабораторную работу по курсу «Дискретный анализ», я научилась решать задачи методом динамического программирования: искать оптимальную подструктуру и составлять рекурсивное решение, когда подпрограмма вызывает сама себя, и нерекурсивное решение.

Динамическое программирование стоит применять для решения задач, которые обладают двумя характеристиками:

1. Можно составить оптимальное решение задачи из оптимального решения ее подзадач.

2. Рекурсивный подход к решению проблемы предполагал бы многократное (не однократное) решение одной и той же подпроблемы, вместо того, чтобы производить в каждом рекурсивном цикле все новые и уникальные подпроблемы.

Так, в решаемой мной задаче методом динамического программирования, можно заметить, что имеется оптимальная подструктура решения - его можно свести к решению подзадач меньшего размера. Действительно, минимальная стоимость p[n] преобразования числа n равна n + min(p[n/2], p[n/3], p[n 1]). Стоимость p[n/2] или p/n/3] считается бесконечной, если n не делится на 2 или 3 соответственно, так как деление происходит только нацело. При построении рекурсивного решения становится ясно, что имеет место перекрытие вспомогательных подзадач, значит оптимальное решение можно построить методом восходящего анализа. Таким образом, решение имеет оценку $O(n)$ для времени в памяти.

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.))