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. А. Литвина
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №7

Задача: При помощи метода динамического программирования разработать алгоритм решения задачи, определяемойсвоим вариантом.
6. Палиндромы
Задана строка S состоящая из n прописных букв латинского алфавита. Вычеркиванием из этой строки некоторых символов можно получить другую строку, которая будет являться палиндромом. Требуется найти количество способов вычеркивания из данного слова некоторого (возможно, пустого) набора таких символов,что полученная в результате строка будет являться палиндромом. Способы, отличающиееся только порядком вычеркивания символов, считаются одинаковыми.

Входные данные: Задана одна строка S ($|S| \leq 100$).
Выходные данные: Необходимо вывести одно число – ответ на задачу. Гарантируется, что он $\leq 2^{63} - 1$.

1 Описание

Динамическое программирование — 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.

В моем случае, сложность алгоритма динамического программирования $O(n^2)$, в то время как сложность "наивного"алгоритма - $O(n2^n)$.

2 Исходный код

Создаем квадратную матрицу размером длины слова. Заполняем ее ячейки нулями, кроме элементов на диагонали, их заполняем единицами. Далее проходимся по нижней половине матрицы и, если соответствующие буквы равны, то в текущую ячейку записываем значение, равное сумме двух ячеек, находящихся сверху и сбоку от нее (которые мы уже посетили), + 1, иначе равное сумме двух ячеек, находящихся сверху и сбоку от нее, минус значение ячейки по диагонали от нее. Выводим значение угловой ячейки (нижней левой). Это и есть наш ответ.

```
#include <iostream>
#include <cstdio>
#include <cstring>

const int MAX = 100;
using namespace std;

int main() {
  char s[MAX];
  cin >> s;
  int n = strlen(s);
  long matrix[MAX][MAX];

  for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
      if (i==j) matrix[i][j]=1;
      else matrix[i][j]=0;
    }
  }

  for (int i=0; i<n; i++) {
    for (int j=i-1; j>=0; j--) {
      if (s[i]==s[j])
        matrix[i][j]=matrix[i-1][j]+matrix[i][j+1]+1;
      else
        matrix[i][j]=matrix[i-1][j]+matrix[i][j+1]-matrix[i-1][j+1];
    }
  }

  cout << matrix[n-1][0] << endl;
}
```

3 Консоль

```
anast@anast-Lenovo-B590:~$ g++ da7.cpp
anast@anast-Lenovo-B590:~$ ./a.out
baobab
22
anast@anast-Lenovo-B590:~$ ./a.out
abrakadabra
131
anast@anast-Lenovo-B590:~$
```

4 Тест производительности

Тест производительности представляет из себя слово, длиной в 25 символов. Сравниваю с "наивным"алгоритмом.

```
anast@anast-Lenovo-B590:~$ ./a.out | grep "time"
Dinamic time: 0,000625
Standart time: 838,8608
anast@anast-Lenovo-B590:~$
```

Как видно, разница существует огромная, что в очередной раз показывает эффективность динамического программирования.

5 Выводы

Я познакомилась с таким замечательным разделом программирования, как динамическое программирование. Оно мне понравилось простотой своих алгоритмов, наглядностью задач, а так же своей эффективностью. Конечно, динамическое программирование подходит далеко не для всех задач, но для того спектра задач, для которого подходит, оно является настоящей находкой и прорывом в области программирования.

Список литературы

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.))

Динамическое программирование — Википедия.
URL: `https://ru.wikipedia.org/wiki/Динамическое_программирование`.