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: Ф. А. Иванов
Преподаватель: И. Н. Симахин
Группа: М8О-208Б-19

Дата: 14.04.2022
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №7

Задача: При помощи метода динамического программирования разработать алгоритм решения задачи, определяемой своим вариантом; оценить время выполнения алгоритма и объем затрачиваемой оперативной памяти. Перед выполнением задания необходимо обосновать применимость метода динамического программирования.

Разработать программу на языке C или C++, реализующую построенный алгоритм. 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, отличающиеся только порядком вычеркивания символов, считаются одинаковыми.

Формат входных данных: Задана одна строка S ($|S|100$).

Формат результата: Необходимо вывести одно число – ответ на задачу. Гарантируется, что он меньше $2^{63} - 1$.

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.

Нам требуется найти количество вариантов вычеркиваний приводящих строку к полиндрому.

Поэтому, нужно выделить в нашей задаче меньшие подзадачи. Учитывая независимость от порядка вычеркиваний можно выделить три подзадачи: количество вариантов без первого символа, количество вариантов без последнего символа, количество вариантов без первого и последнего символов.

Рассмотрим вариант задачи, где первый и последний символ отличаются. В таком случае количество вариантов вычеркиваний равно: количеству вариантов без первого символа плюс количество вариантов без последнего символа и минус количество-вариантов без первого и последнего символов. Вычитание нужно, т.к. два других слогаемых содержат его в себе, и из за этого мы считали бы его 2 раза.

$$Var(str(i+1,j)) + Var(str(i,j-1)) - Var(str(i+1,j-1))$$

Теперь рассмотрим вариант, где первый и последний символ равны. Этот случай похож на предыдущий. Во-первых, если мы вычтем всю середину, то получим еще один полиндром. Во-вторых, если мы к каждому полиндрому середины прибавим по одному символу слева и справа, то тоже получим полиндром. Т.е. мы должны прибавить еще количество вариантов без первого и последнего символа. Слогаемые сокращаются и получаем: количество вариантов без последнего символа плюс полиличество вариантов без первого символа плюс единица.

$$Var(str(i+1,j)) + Var(str(i,j-1)) + 1$$

2 Исходный код

```
#include <iostream>
#include <string>
#include <cstring>

long countingVariants(std::string &str,long cache[100][100], int i, int j){
    if(j < i){
        return 0;
    } else {
        if(cache[i][j] == 0){
            if(str[i] == str[j]){
                cache[i][j] = countingVariants(str,cache,i + 1,j) + countingVariants(str
                    ,cache,i,j - 1) + 1;
            } else {
                cache[i][j] = countingVariants(str,cache,i + 1,j) + countingVariants(str
                    ,cache,i,j - 1) - countingVariants(str,cache,i + 1,j - 1);
            }
        }
        return cache[i][j];
    }
}

int main(){
    long cache[100][100];
    std::memset(cache,0,sizeof(long)*10000);
    std::string str;
    std::cin >> str;
    std::cout << countingVariants(str,cache,0,str.size() - 1) << std::endl;
    return 0;
}
```

3 Консоль

```
orion@orion-laptop:~/University/DA/Lab_07$ ./main
QWERTY
6
orion@orion-laptop:~/University/DA/Lab_07$ ./main
AAA
7
orion@orion-laptop:~/University/DA/Lab_07$ ./main
ALLAKOALLAOK
173
```

4 Тест производительности

Прведем 3 теста.

Для 10.

```
orion@orion-laptop:~/University/DA/Lab_07$ ./main <test.txt
2.1765e-06
orion@orion-laptop:~/University/DA/Lab_07$ ./naiv <test.txt
9.1075e-05
```

Для 25.

```
orion@orion-laptop:~/University/DA/Lab_07$ ./main <test.txt
8.9935e-06
orion@orion-laptop:~/University/DA/Lab_07$ ./naiv <test.txt
4.87439
```

Для 35.

```
orion@orion-laptop:~/University/DA/Lab_07$ ./main <test.txt
2.1032e-05
orion@orion-laptop:~/University/DA/Lab_07$ ./naiv <test.txt
212.90438
```

Из тестов видно, что наивный алгоритм значительно проигрывает динамическому. Сложность наивного алгоритма выше, чем динамического.

5 Выводы

Выполнив данную лабораторную работу, я познакомился с методом динамического программирования. С помощью этого метода можно решать мнрожество задач оптимизации. Иногда ДП требует много ресурсов, тогда предпочтительней использовать метод жадных алгоритмов.

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.))