Домашаня робота №3 з курсу «Алгоритми та структури даних»

Сакевич Руслан

12 лютого 2016 р.

1 Поняття та позначення

Будемо вважати, що строка s з умови задає певний шаблон. Зрозуміло, що будь-яка підсторка $s[l, r]$ строки s також задає певний шаблон.

Позначення 1. ПДП - правильна дужкова послідовність.

Означення 1. Строка s задає ПДП T, якщо існує певна підстановка дужок, замість знаків питання, яка переводить s в T.

Означення 2. ПДП називається **простою**, якщо вона не містить правильну дужкову **підпослідовність**.

Позначення 2. $f(l, r)$ – число ПДП, що підпадають під шаблон $s[l, r]$.

Позначення 3. $g(l, r)$ – число простих ПДП, що підпадають під шаблон $s[l, r]$.

2 Алгоритм

2.1 Про розбиття ПДП

Лема 1. *Будь-яку ПДП T можна розбити на просту ПДП L та ПДП R меншого розміру. Причому $T = L + R$.*

Доведення. Нехай T – ПДП, тоді будемо виконувати наступну ітерацію для чергової дужки з T:

1. Якщо дужка відкриваюча, то додаємо її до стеку S.
2. Якщо дужка закриваюча, то вилучаємо з стеку відповідну їй відкриваючу дужку. Це завжди можна зробити, оскільки T – ПДП.

Зрозуміло, що алгоритм **скінчений**. Розглянемо першу ітерацію, на якій стек S – порожній. На цій ітерації ми вже опрацювали певну підпослідовність дужок L. Тоді L – також ПДП. Більше того L – проста ПДП. Неважко бачити, що $T \backslash L = R$, також ПДП, та оскільки L – містить не менше однієї дужки, то довжина R **менша** довжини T. Особливий випадок виникає тоді, коли T – проста ПДП, тоді $T = L$, а R – порожня, але цей випадок задовольняє умову леми. Отже ми навели алгоритм розбиття, що завершує доведення. $\square$

Зауваження. Легко трансформувати описаний алгоритм побудови розбиття, так щоб він розбивав ПДП T на ПДП L та просту ПДП R, так щоб $T = L + R$.

2.2 Підрахунок $f(l, r)$

Спочатку покажемо як рахувати $f(l, r)$. За лемою 1, будь-яку ПДП можна розбити на просту ПДП та ПДП меншого розміру. В часності будь-яку ПДП породжену шаблоном $s[l, r]$ можна розбити на просту ПДП та ПДП меншого розміру. Нехай $h(l, m, r)$ – число ПДП породжених шаблоном $s[l, r]$, таких, що $s[l, m]$ – проста ПДП, а $s[m + 1, r]$ – ПДП. За правилом добутку $h(l, m, r) = g(l, m) \cdot f(m + 1, r)$. В цьому місці видно, що функцію $f(l, r)$ слід доозначити нулем у випадку $l > r$.

Лема 2. $f(l,r) = \sum_{m=l}^{r} h(l,m,r)$

Доведення. Для доведення леми достатньо показати:

1. Будь-яка ПДП породжена $s[l,r]$ буде врахована.

2. Тільки ПДП породжені $s[l,r]$ будуть враховані, причому тільки один раз.

Для доведення першого пункту достатньо застосувати алгоритм наведений в лемі 1. Також очевидно, що не може бути врахована ПДП, що не підпадає під шаблон $s[l,r]$. Покажемо, що жодної ПДП породженої шаблоном $s[l,r]$ ми не врахуємо двічі. Припустимо, що це не так, і існує ПДП T що підпадає під шаблон $s[l,r]$, і має два розбиття $L_1 + R_1$ та $L_2 + R_2$. Виберемо найкоротшу просту ПДП з L_1 та L_2. Не порушуючи загальності будемо вважати, що це L_1. Тоді L_2 містить в собі L_1. Але оскільки L_1 також ПДП, то L_2 - не може бути ПДП(за визначенням). Отримане протиріччя завершує доведення. □

2.3 Підрахунок $g(l,r)$

Лема 3. *Будь-яка проста ПДП має вигляд*

- *"(" + ПДП + ")"*

- *"[" + ПДП + "]"*

- *"{" + ПДП + "}"*

Доведення. Дійсно, якщо припустити, що існує проста ПДП що підпадає під варіанти умови, то алгоритм розбиття описаний в лемі 1 завершиться раніше, ніж розгляне останню дужку. Що суперечить визначенню простої ПДП. □

Тепер зрозуміло, як рахувати $g(l,r)$. Відкинувши зовнішні дужки перейдемо до шаблону $s[l+1, r-1]$. З леми 3 слідує, що

$$g(l,r) = \lambda \cdot f(l+1, r-1), \text{ де}$$
$$\lambda = \begin{cases} 1, & \text{якщо дужки виду "()", "[]", "\{\}", "?)", "?]", "?\}", "(?", "[?", "\{?"} \\ 3, & \text{якщо дужки виду "??"} \\ 0, & \text{інакше} \end{cases}$$

2.4 Зауваження

Слід зазначити, що при підрахунку $f(l,r)$ або $g(l,r)$ ми завжди зводимо задачу до задач, з **меншими границями**. Тому достатньо вказати початкові умови, і можна буде обрахувати значення функцій, для будь-яких $1 \leq l \leq r \leq n$. Неважко бачити, що відповідь на задачу $f(1,n)$

3 Складність алгоритму

Надалі йдеться лише про реалізації побудовані, на **мемоїзації**, або на **динамічному програмуванні**. Оскільки якщо обраховувати раніше знайдені величини повторно, то складність алгоритму одразу стає **експоненціальною**!

3.1 Часова оцінка

Точну оцінку для алгоритму обрахувати досить важко. Покажемо, що верхня часова оцінка складає $O(N^3)$. Дійсно для обрахунку $f(1,n)$ може знадобитися обрахувати $f(i,j)$, для будь-яких $1 \leq i \leq j < n$. Причому для обрахунку $f(i,j)$ виконується не більше $j - i$ рекурсивних викликів. Маємо:

$$\sum_{1 \leq i \leq j < n} j - i \leq n^3$$

3.2 Оцінка використаної пам'яті

Ця величина дуже сильно залежить від реалізації алгоритму. Найменша оцінка, яку можна досягти: $O(n^2)$ пам'яті, для збереження 32-бітових чисел.

3.3 Хитрощі реалізації

Зрозуміло, що усі операції додавання та множення слід виконувати за модулем 10^5, оскільки нам важливі лише останні 5 цифр числа. Але, слід звернути увагу, на вимогу з умови:

> "Это число может быть очень большим, поэтому выведите только его **последние** 5 **цифр**"

Саме тому потрібно додатково зберігати інформацію, чи відбувалося переповнення при обрахунках шуканої величини.