Лекция про строки

Филипп Рухович, Александр Голованов

3 марта 2015

1 Обозначения

В этом конспекте будем индексировать символы строки s, начиная с 1; длину строки будем обычно обозначать через n; подстроку строки s, начинающуюся с l-го символа и заканчивающегося r-ым, будем обозначать как $s[l..r]$.

Будем говорить, что строка s имеет **период** t, если s есть конкатенация d одинаковых строк t, где d — некоторое натуральное число. **Степенью** строки s назовем максимальное такое d; соответствующая ему строка t есть минимально возможный период s.

Строку s будем называть k-периодической, если $k \leq n$ и $s[1] = s[k+1], s[2] = s[k+2], ..., s[n-k] = s[n]$.

2 Необходимая теория

Описание z-функции и префикс-функции, описание полиномиального хэширования строк и их подстрок, а также способ поиска заданной подстроки t в строке s с помощью вышеописанных трех алгоритмов можно найти на сайте e-maxx.ru/algo .

3 Циклический сдвиг

Задача. Даны две строки s и t, одинаковой длины n. Верно ли, что строка t является циклическим ствигом строки s, т.е. $t = s_{k+1}s_{k+2}...s_n s_1 s_2 ... s_k$, где k, $0 \leq k \leq n-1$ — величина сдвига?

Решение. Рассмотрим строку T, tt. Тогда t — циклический сдвиг s величины k т.и т.т., когда $s = T[k..k+n-1]$. Таким образом, задача сводится к поиску подстроки s в строке T. Время работы алгоритма $O(n)$.

Задача. Даны два многоугольника A и B без самопересечений и самокасаний, никакие три вершины в которых не расположены на одной прямой. Верно ли, что один из этих многоугольников можно получить из другого с помощью параллельного переноса?

Решение. Заметим, что с точностью до параллельного переноса многоугольник $A = A_1 A_2 A_n$ можно задать последовательностью векторов A_1A_2, A_2A_3, ..., A_nA_1. Отсюда, казалось бы, следует, что задача сводится к предыдущей (только в строке вместо символов — вектора), но это не совсем так, ибо последовательности векторов могут «обходить» одинаковые многоугольники в разные стороны и являться при это разными по составу; поэтому проверку на цилкический сдвиг нужно осуществить два раза — для двух способов обхода многоугольника B. Время работы по-прежнему $O(n)$.

4 Периодичность строк

Задача. Найти степень строки s. **Решение.** Задача эквивалентна поиску минимального перода строки s. Заметим, что строка имеет период длины $k <=> n$ делится на k и $s[1..n-k] = s[k+1..n] <=> z[k+1] = n-k$ ($z[i]$ есть z-функция строки s в i-ом символе). Таким образом, достаточно вычислить z-функцию строки s, перебрать все возможные k и понять для каждого из них, является ли $s[1..k]$ периодом s. Время работы продолжает быть линейным)))

Задача. Найти степени всех префиксов строки s.

Решение. Самый простой (идейно) способ — запустить предыдущее решение для всех префиксов строки s, за время $O(n^2)$. Хорошей идеей является заполнить массив $maxp[1..n]$, в который в итоге запишем минимальные периоды префиксов s, числами с 1 по n, после чего для каждого k перебрать префиксы всех делящихся на k длин; для каждого такого префикса с помощью z-функции за O(1) поймем, является ли строка k-периодичной; если да, то обновим текущее значение в $maxp$ для такого префикса. Т.к. для каждого k перебирается $O(n/k)$ префиксов, то согласно известному факту о сумме гармонического ряда общее время работы алгоритма не будет превосходить $O(nlogn)$.

Можно ли решать задачу за линейное время? Оказывается, можно! В этом нам помогут следующие леммы:

Лемма 1: Если строка s длины n k-периодична и l-периодична, $k \leq l$, а $k + l \leq n$, то s $l - k$-периодична.

Доказательство: Если $1 \leq i \leq n - l$, то $s[i] = s[i + l] = s[i + l - k]$; в противном случае $s[i] = s[i - k] = s[i - k + l]$.

Лемма 2: Если строка s длины n k-периодична и l-периодична, $k \leq l$, а $k + l \leq n$, то s $gcd(k, l)$-периодична ($gcd(x, y)$ есть наибольший общий делитель чисел x и y).

Доказательство: Прямое следствие Леммы 1 и алгоритма Евклида.

Лемма 3: Если строка s длины n имеет период длины k, меньшей n, l-периодична, $k < l$, то строка имеет период длины, меньшей l.

Доказательство: Очевидно, что $l \leq n/2$, откуда $k + l < l + l \leq n$. Тогда по лемме 2 s g-периодична, где $g = gcd(k, l)$. Однако g — делитель l — делителя n; следовательно, s имеет период длины g, QED.

Вернемся к исходной задаче, которую будем решать за линейное время. Переберем все возможные длины периодов, и для каждого попытаемся найти все префиксы с такими периодами. Более того, пусть после $i-1$-ой итерации мы храним число r — длину максимального префикса s, имеющего период меньшей i длины. Будем, как и в решении за $O(nlogn)$, перебирать префиксы длин $i \cdot j$ для натуральных j и для каждой из них пытаться обновить ответ, причем 1) как только для очередной итерации i-периодичность нарушилась, обрываем процесс; 2) начинаем перебор с такого минимального j, что $i \cdot j > r$. Первое отсечение, очевидно, ответ не портит; корректность же второго отсечения есть, для $j > 1$, прямое следствие Леммы 3 (ибо префикс длины $i \cdot j$ l-периодичен, где $l < i$; если он (префикс) имеет период i, то он по Лемме 3 имеет и меньший период, что уже отмечено в $maxp[i \cdot j]$;

следовательно, рассмотрение такого случая бессмысленно). После перебора остается лишь обновить r, если нужно. Так как каждый перебранный $i \cdot j$-префикс, кроме последнего, увеличивает r, то время работы алгоритма, аналогично доказательству времени работы в z-функции, есть $O(n)$.

5 Префиксный автомат и динамическое программирование

Задача. Сколько существует строк s длины n, состоящих из символов алфавита A размера k, **НЕ** содержащих заданную строку t длины m в качестве подстроки? Требуется найти ответ по модулю $10^9 + 7$.

Решение. Воспользуемся методом динамического программмирования. Пусть $d[i][j], 1 \leq i \leq n, 0 \leq j < m$, есть количество строк u длины i, удовлетворяющих условию задачи, также равенству $prefix(t+'?'+u) = j$, где $'?'$ есть не лежащий в алфавите A символ (другими словами, максимальная длина суффикса u, равного префиксу t, должна быть равна j). Воспользуемся динамикой вперед, то есть будем увеличивать $d[i+1][l]$, зная $d[i][j]$. Итак, пусть мы знаем $d[i][j]$. Если приписать к такой строке символ произвольный c из алфавита A, то j превратится в значение некоторой функции $go = go(j, c)$, которую можно предподсчитать за время $O(nk)$ (действительно, если $s[j+1] = c$, то $go(j, c) = j + 1$; иначе $go(j, c) = go(pr[j], c)$, где $pr[1..m]$ есть префикс-функция t); сделаем это предварительно. Теперь, пусть мы знаем $d[i][j]$. Переберем все символы c алфавита A; если $go(j, c) < m$, то увеличим $d[i+1][go(j, c)]$ на $d[i][j]$. Проделав это для всех i, j в порядке возрастания сначала i, а потом j, получим $d[n][m]$ — ответ на задачу. Время работы такого алгоритма $O(nmk)$. Обычно k считается константой; в таком случае время работы оказывается $O(nm)$.

6 Хэши и декартовы деревья

Задача. Дана строка s. Требуется реализовать структуру данных, которая делает со строкой следующие операции:

- Добавить символ в произвольное место строки;
- Удалить символ в произвольное место строки;
- Узнать, верно ли, что две заданные в запросе подстроки равны.

Решение. Будем хранить строку в декартовом дереве по неявному ключу, где каждое поддерево с корнем в вершине v хранит символ $c(v)$ и соответствует строке $s(v)$, равной $s(v.left) + c(v) + s(v.right)$ ($v.left$ и $v.right$ есть левый и правый сыновья v; если $v.left$ не существует, то $s(v.left)$ — пустая строка; аналогично $s(v.right)$). В вершине v декартова дерева будем хранить: а) $c(v)$; б) количество вершин в поддереве v ; в) полиномиальный хэш строки $s(v)$. Тогда добавление делается за один split и два merge-а (соответствует разделению строки на две, созданию строки из нового символа и двум конкатенациям); удаление — за два split-а и один merge (аналогично). Сравнивать же две подстроки будем по их хэшам; хэш же подстроки вычисляется путем разбиения строки (в лице дерева) на три части (подстрока, все, что слева подстроки, и все, что справа), запоминания хэша подстроки (записано в корне дерева, соответствующего подстроке) и объединения строки обратно (всего два split-а и два merge-а). Таким образом, все запросы мы научились делать за не более чем 4 операции split/merge, что дает нам асимптотику ответ на запрос $O(log n)$. Детали пересчета статистик в вершине при split/merge оставляем читателям.