Домашнее задание №12
Алгоритмы. 5 курс. Весенний семестр.

Горбунов Егор Алексеевич

11 мая 2016 г.

Задание №1 Найти подстроку в тексте. При сравнении строк можно делать циклический сдвиг алфавита в одной из них. $\mathcal{O}(n+m)$, алфавит — не константа.

Решение: У нас алфавит конечный, пусть всем буквам сопоставлено целое число в соответствие. Теперь заметим, что если строка $c_1 c_2 \ldots c_l$ равна с точностью до сдвига алфавита строке $c_1' c_2' \ldots c_l'$, то это значит, что равны следующие массивы (строки):

$$(c_2 - c_1), (c_3 - c_2), \ldots, (c_k - c_{k-1}) \text{ и } (c_2' - c_1'), (c_3' - c_2'), \ldots, (c_k' - c_{k-1}')$$

Разности соседних элементов не меняются при сдвиге алфавита. Такие массивы по сути тоже строки, поэтому мы можем применять к ним тот же аппарат, что применяем к строкам.

Итак. Данные на вход строку и текст мы переделаем в массивы разностей, после чего будем использовать, например, алгоритм Кнута-Морриса-Пратта, что даст нам асимптотику в $\mathcal{O}(n+m)$. ∎

Задание №2 Для каждого префикса строки найти количество его префиксов равных его суффиксу. $\mathcal{O}(n)$

Решение: Решается префикс-функцией. Рассмотрим какой-то префикс строки:

$$\overbrace{s_0, s_1, s_2}, \ldots, \overbrace{s_{k-3}, s_{k-2}, s_{k-1}}$$

Выше обозначены максимальный префикс префикса равный его суффиксу, который мы получаем посчитав префикс-функцию. Заметим, что все остальные длины префиксов префикса, равные его суффиксу, можно легко перебрать: $next = p[p[k-1]]$. Соответственно для k-го префикса ответ считается так: $i = k - 1$, пока $p[i] > 0$ прибавить к ответу 1 и $i := p[i]$, а иначе закончить. ∎

Задание №3 Преобразовать Z-функцию в префикс-функцию без промежуточного восстановления строки за $\mathcal{O}(n)$

Решение: $z[i]$ — размер максимального префикса подстроки $s[i..n-1]$ равный префиксу строки $s[0..n-1]$ (Z-функция).

$p[i]$ — размер максимального суффикса подстроки $s[0..i]$ равный собственному префиксу этой подстроки.

Пускай у нас $z[i] > 0$, что обозначает, что $s[0..z[i]-1] = s[i..(i+z[i]-1)]$. Заметим тогда, что для подстроки $s[0..(i+j)]$, где $j \in [0, z[i])$ точно верно, что её суффикс длины $j+1$ равен префиксу.

Будем идти по массиву префикс-функции слева направо, т.е. в порядке убывания длины рассматриваемого суффикса строки. Для каждого рассматриваемого $z[i]$ будем перебирать j в порядке убывания от $z[i]-1$ до 0 и если $p[i+j]$ ещё не проставлено, то присваивать $p[i+j] = j+1$, а иначе переходить к следующему i.

Видим, что каждое значение $p[k]$ присваивается максимум один раз. Значит это работает за линейное время.

Но почему это вообще работает? Пускай в какой-то момент мы присвоили $p[i+j] = j+1$. Пускай теперь по ходу алгоритма мы наткнулись на i' и j', что $i' > i$ (подругому и никак) и $i'+j' = i+j$. Но тогда очевидно, т.к. $i' > i$, то $j' < j$, а значит и $j'+1 < j+1$. Т.е. один раз проставив $p[i+j]$ его уже проставлять не нужно. Теперь заметим так же, что если в какой-то момент алгоритм проставил $p[i+j]$, то $p[i], p[i+1], \ldots p[i+j-1]$ тоже будут (были) проставлены (либо уже, либо будут на текущей итерации) в силу того, что i перебирается в порядке возрастания. Оно работает. ■

Задание №5 Даны бор A и строка s. Нужно вернуть вершину бора v, от которой строку s можно отложить вниз. Размер алфавита $\mathcal{O}(1)$. Время $\mathcal{O}(|A| + |s|)$.

Решение: Давайте построим эйлеров обход бора по рёбрам. Будем хранить два таких эйлерова обхода X и Y, но в X будем держать буквы: если в обходе на позиции i стоит ребро (u, v), причём u — это родитель v в боре, то в X на позиции i будет буква, соответствующая ребру (u, v), а если же (u, v) — это возвратное ребро, то поставим на этой позиции в X символ \$, а в Y будем держать ссылки на вершины бора, соответствующие первой вершине ребра (т.е. если в обходе на позиции i стоит ребро (u, v), то в Y на позиции i будет ссылка на u). Идея в том, чтобы теперь искать s в строке X используя КМП, после чего, если мы нашли в X такую позицию u начиная с которой в ней лежит s, то ответом на запрос будет $Y[i]$. ■