Домашнее задание №3

Д.А. Першин

24 апреля 2015 г.

1 Словесное описание алгоритма

Дана строка S длины $n \leq 100\,000$ над алфавитом Σ, являющимся строчными латинскими буквами. Необходимо найти количество ее различных непустых подстрок. Подстроки считаются одинаковыми, если они совпадают, как отдельно взятые строки.

Для нахождения всех различных подстрок строки S будем использовать следующие структуры данных: суффиксный массив (для его построения будем использовать алгоритм Сандерса - время построения $O(n)$, память - $O(n)$) и lcp массив (алгоритм Касаи - время построения $O(n)$, память $O(n)$).

Построим суффиксный массив $suffix$ и lcp массив lcp. Алгоритм основывается на следующем факте: множество всех префиксов всех суффиксов строки образует множество всех подстрок. Остается только удалить из этого множества повторяющиеся подстроки. Далее воспользуемся тем фактом, что в суффиксном массиве все суффиксы отсортированы, а текущий суффикс даст в качестве новых подстрок все свои префиксы, кроме совпадающих с префиксами предыдущего суффикса в суффиксном массиве (т.к. они уже были учтены на предыдущей итерации). Но кол-во префиксов, совпадающих с префиксами предыдущего суффикса можно получить за константное время, используя lcp массив ($O(n)$ - в сумме для всех суффиксов). Данная операция повторяется для всех суффиксов массива $suffix$. Сумма полученных на каждой итерации значений дает искомое кол-во всех уникальных подстрок строки S.Итого, получаем следующую формулу:

$$substrings = \sum_{i=0}^{n} (n - suffix[i] - lcp[i])$$

Алгоритм:

- Пусть S - исходная строка длины n.

Algorithm 1: кол-во различных подстрок строки

```
procedure SubstringsCount(S)
    suffix = SuffixArray(S)            ▷ по строке S построим суффиксный массив
    lcp = LCPArray(S, suffix)          ▷ по строке S и суффиксному массиву suffix построим lcp массив
    substrings = 0
    for i ∈ [0, n − 1] do              ▷ для всех суффиксов массива suffix
        substrings+ = n − suffix[i] − lcp[i]   ▷ добавляем новые уникальные подстроки
    end for
    return substrings
end procedure
```

2 Доказательство корректнсти

Доказательство будем проводить по индукции. Будем последовательно рассматривать все суффиксы строки S. Для суффикса длины 1 строки S доказательство очевидно - кол-во уникальных подстрок равно 1. Далее предположим, что для некоторого суффикса $suff$ строки S рассчитано кол-во различных подстрок. Расширим суффикс $suff_i$ строки S на один символ, добавив предшествующий $suffix_i$ символ c в его начало, получим суффикс $suff_{i+1} = c\, suff_i$. Очевидно, что кол-во подстрок увеличилось на длину нового суффикса (каждый префикс - новая подстрока), т.е. на $len(suff_{i+1})$. При этом некоторое кол-во подстрок (префиксов $suffix_{i+1}$) может совпадать с уже найденными ранее. Далее обратим внимание на тот факт, что если подстрока уже была учтена ранее, то также были учтены и все ее префиксы (следует из выше описанного), т.е. было учтено кол-во подстрок, равное длине самой подстроки. Следовательно, вычтя длину наиболее длинного префикса предыдущих суффиксов, совпадающего с префиксом текущего суффикса из длины текущего суффикса мы получим кол-во новых уникальных подстрок. Решение для суффикса длины, равной длине самой строки, является решением задачи.

3 Асимптотические оценки

Пусть длина строки S равна n, длина алфавита Σ (по условию $\Sigma \leq 26$). В результате получаем сложность по памяти $O(n)$, так как мы используем алгоритм Сандерса для построения суффиксного массива - $O(n)$ и алгоритм Касаи для построения lcp массива - $O(n)$.

Сложность по времени также равна $O(n)$, алгоритм Сандерса - $O(n)$, алгоритм Касаи - $O(n)$, поиск кол-ва различных подстрок - $O(n)$.
Итого, получаем временную сложность $O(n)$, сложность по памяти $O(n)$