
Разбор задачи «Язык племени Мотунуи»

Когда в задаче требуется найти множество лексикографически минимальных строк, имеет смысл воспользоваться бором. Разумеется, просто построить бор на всех возможных звучаниях слов не особо реалистично, так как нас интересуют k минимальных, а всего их порядка 2^n , что сильно больше.

Однако можно использовать несколько оптимизаций, чтобы строить k лексикографически минимальных строк быстрее. Для этого необходимо понимать, как строится лексикографически минимальная строка, а также как по любой строке построить следующую в лексикографическом порядке.

Построить минимальную строку не сложно: достаточно заметить, что она состоит из одного единственного минимального звука. Соответственно, эта задача просто эквивалентна поиску минимума в массиве a .

Поиск лексикографически следующей строки устроен следующим образом:

1. найти ближайший к концу текущей строки символ, который можно увеличить;
2. увеличить его на минимальное возможное значение, большее текущего;
3. дополнить минимальным возможным суффиксом.

Итого: в качестве минимальной строки выберем самое левое вхождение минимального звука в a . Далее k раз сделаем следующее:

1. если последняя буква текущего слова — не последняя буква алфавита, можно увеличить слово, дописав к нему в конец минимум из оставшихся букв; хеш строки при этом меняется понятным образом;
2. иначе — предподсчитав для каждого символа строки минимальный, больший его, справа от него (что можно сделать одним проходом со стеком), бинпоиском найдем первый с конца символ, для которого справа от него есть какой-то больший, после чего сделаем замену; минимальный возможный суффикс, который можно дописать — пустой.

Для эффективного выполнения этих операций достаточно иметь дерево отрезков на операцию «минимум» на алфавите (или просто суффиксные минимумы), а также хранить текущую строку в декартовом дереве по неявному ключу (в вершине храним флаг «есть ли в поддереве буква, которую можно увеличить», а также хеш поддерева). Хеши пересчитываются автоматически, а все описанные операции сводятся к операциям `split` и `merge` для ДД по неявному ключу — общее время работы $\mathcal{O}((n + k) \log n)$.