Суффиксное дерево. Суффиксным деревом мы будем называть сжатый бор, который будет хранить все суффиксы строки s, и только их. Сжатый бор — это бор, на ребрах которого написаны не буквы, а строки. Вместо строк мы будем хранить подотрезок $[l, l + len)$. Соответственно, если при добавлении строки оказывается так, что строка вдоль ребра не читается, то ребро нужно будет разделить на две части — общую и разную, и поставить между ними промежуточную вершину.

Добавлять в наш сжатый бор мы будем все суффиксы строки.

Квадратичный алгоритм. Будем считывать строку посимвольно. На ребрах будут написаны подотрезки. Поскольку мы не знаем число n, мы введем подотрезки $[l, \infty)$.

Алгоритм будет хранить все терминальные вершины в боре, и при добавлении нового символа будет продлевать все суффиксы строки на этот символ. Понятно, что все терминалы, в которые вели ребра $[l, \infty)$, можно никак не продлевать. Те суффиксы, которые лежали не на бесконечных ребрах, надо продлить на новый символ c. Если перехода нет, надо создать новую вершину, и сделать в нее ребро $[l, \infty)$.

Разделим суффиксы на определившиеся, неопределившиеся и неизвестные. Неизвестные — это суффикс, который еще не добавлен в бор. Определившийся суффикс — это такой суффикс, которому соответствует ребро $[l, \infty)$. Неопределившиеся — это такие суффиксы, которые являются префиксом определившегося суффикса. Будем называть их суффиксами 2, 1, и 0 типа (от определившихся к неизвестным).

Понятно, что суффиксы типа 2 не перестают быть типом 2 — это самый префикс такого типа. Тип 0 переходит в 1 или 2. 1 может переходить в 2. При этом важно понимать, что суффиксы всегда будут иметь вид 222221111110000. Почему? Потому что после какой-то единицы всегда будут идти единицы — если для позиции i суффикс s_i встречался раньше (на позиции $j < i$). то для s_{i+1} можно увидеть, что s_{j+1} соответствовал этому суффиксу.

Алгоритм Укконена. В терминологии 0-1-2 суффиксов мы хотим следующее: поддерживать в боре все 1 и 2-суффиксы, при этом каждый раз рассматриваать только первый по счету 1-суффикс. При переходе по новому символу самый левый 1-суффикс может поменяться. Еслии он поменялся, то нам надо заменить подотрезок из 1-суффиксы на 2-суффиксы. Так мы получим амортизированную линейную сложность.

Тогда мы будем поддерживать указатель на j — первый 1-суффикс в дереве. Если мы видим переход, которого сейчас нет в боре, надо разделить ребро на два (если мы сейчас находимся в середине ребра). После этого из вершины делаем переход по новому симполу. Теперь суффикс имеет тип 2. Сейчас осталось проделать аналогичную операцию для $j + 1$. Проблема в том, что непонятно, где находится соответствующий суффикс.

Введем суффиксные ссылки. Суффиксная ссылка из вершины v указывает на *позицию* в боре, которая соответствует суффиксу v, который на 1 короче текущего. То есть, для перехода мы сможем пользоваться суффиксными ссылками. Осталось понять, как их поддерживать.

Суффиксная ссылка из вершины всегда ведет в вершину. Это верно, потому что вершина соответствовала разветвлению, но тогда и для строки на 1 короче это тоже будет верно.

Пусть мы сейчас хотели сделать переход по суффссылке из вершины v. Посмотрим на $u = link(parent(v))$. Если мы прочитаем из вершины u ребро $parent(v) \to v$, то окажемся в позиции $link(v)$. Это либо будет вершиной, либо в ней сейчас произойдет ветвление. Причем чтение ребра надо реализовать за число вершин на пути, не за число символов.

За сколько это будет работать? Амортизированно за линию. Понятно, что нас интересует сейчас только суммарное время на чтение по ребрам во время поиска суффссылок. Тут можно ввести потенциал

$p(j)$ — число символов на ребре до предка-вершины j. Тогда, когда мы при чтении ребра проходим каждую вершину, мы уменьшаем свой потенциал.