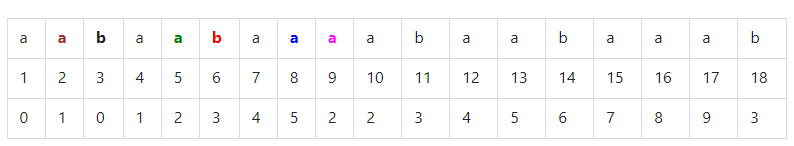
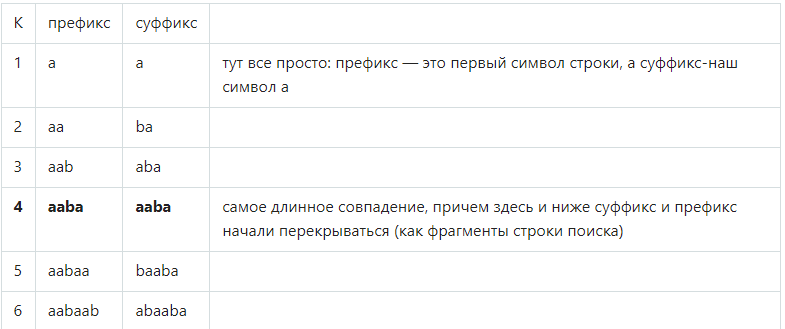
[Алгоритм Кнута-Морриса-Пратта](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%BD%D1%83%D1%82%D0%B0_%E2%80%94_%D0%9C%D0%BE%D1%80%D1%80%D0%B8%D1%81%D0%B0_%E2%80%94_%D0%9F%D1%80%D0%B0%D1%82%D1%82%D0%B0) используется для поиска подстроки (образца) в строке. Кажется, что может быть проще: двигаемся по строке и сравниваем последовательно символы с образцом. Не совпало, перемещаем начало сравнения на один шаг и снова сравниваем. И так до тех пор, пока не найдем образец или не достигнем конца строки. Функция примерно такая:



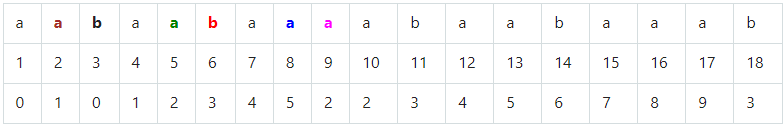
Простой случай поиска'игла' — образец'стогистогстогигстогстогиглстогстогигластогигластог' — строка поиска Сложный случай поискаaabbaabaabaabbaaabaabaabaabaabbaabbФункция работает и довольно шустро, если образцы и строка «хорошие». Хорошие — это когда внутренний цикл прерывается быстро (на 1-3 шаге, скажем, как в простом случае). Но если образец и строка содержат часто повторяющиеся вложенные куски (как сложном случае выше), то внутренний цикл обрывается ближе к концу образца и время поиска оценивается как О(<длина образца>\*<длина строки>). Если длина строки 100тыс, а длина образца 100, то получаем О(10млн). Конечно, реально редко встретишь образец длиной 100, но в олимпиадных задачах «на поиск» это обычное дело, поэтому там простой поиск часто не подходит.А если строка — это длинный текст, а образец-фрагмент слова, и надо найти все вхождения этого фрагмента, причем на лету, по мере набора слова (замечали, как быстро это делают браузеры)? Алгоритм КМП находит все вхождения образца в строку и его скорость О(<длина образца>+<длина строки>), поэтому на больших текстах/образцах или на слабых процессорах (как в низкобюджетных сотовых) он вне конкуренции.А теперь посмотрим на заголовок? Почему «маленькое»? Потому, что изюминка КМП — это префикс-функция, а она действительно маленькая. А почему «чудо»? Потому что, он вроде как решает совсем другую задачу и это решение, после некоторого чудесного трюка, превращается в решение задачи поиска всех вхождений образца в строку.Для того, чтобы понять, что и как делает префиксная функция, посмотрим на сложную строку



Строка под ней — номер (позиция) символа в строке (для удобства описания алгоритма считаем номер с 1), а самая нижняя строка- массив M длин префиксов, ключ к пониманию префикс-функции.Возьмем символ с номером 7 (это a) и для K от 1 до 6 рассмотрим строки-префиксы (подстрока, начинающаяся с первого индекса строки) и суффиксы (подстрока, последний символ которой в строке в позиции 7 ( это «наш» символ a) длины K.

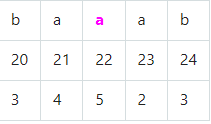
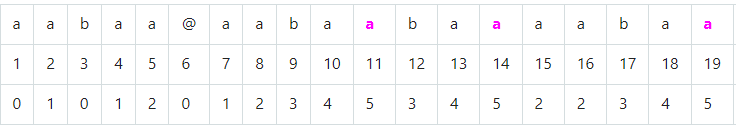


Обратите внимание: для длины 4 суффикс (как последовательность символов) совпадает с префиксом и это максимальное значение K при котором суффикс совпадает с префиксом. Именно оно и заносится в соответствующую позицию (7) массива длин префиксов. Можно заметить, что для K=7 префикс тоже совпадет с суффиксом, поскольку это одна и та же строка. Но этот тривиальный случай нам не подходит, т.к. для работы алгоритма нужны именно подстроки.Обозначим S-исходная строка, S(n) -начало (префикс) строки S длины n, S[n]-символ в позиции n строки S. M[n] значение в массиве, S(M[n])- та самая строка, которая являемся префиксом и суффиксом максимальной длины для позиции n (для краткости обозначим её P(n)). Строка P(n) как бы «виртуальная», она не формируется и никуда не пишется. Это просто начальный фрагмент1 исходной строки S длины M[n]. И этот начальный фрагмент1 совпадает (как последовательность символов) с фрагментом2 длины M[n], последний символ которого в позиции n. Если M[n]=0, то совпаденией нет.Имеем: позиция 7 массива заполнена значением М[7]=4, самая длинная строка P(7)='aaba' длины 4 (естественно), надо перейти к позиции 8 и заполнить M[8]. Можно тупо посчитать все префиксы и суффиксы длиной от 1 до 7, сравнить их и записать максимальную длину в позицию 8. Но мы пойдем другим путем (вслед за КМП). Пусть найдена максимально длинная строка P(8) длины k, которая префикс и суффикс для позиции 8. Строка p7 из первых k-1 символов является префиксом и суффиксом для позиции k-1. Не факт, что для 7й позиции она самая длинная. Однако, если оказалось, что p7=P7, то P8 — это расширение P7 на один символ. Чтобы проверить, можно ли расширить P7 на одну позицию, надо проверить, совпадает ли добавляемый в суффикс символ (это символ S[8]=a) со следующим символом префикса. Следующий символ префикса a находится в позиции М[7]+1=5 (подумайте, почему так). Если совпал (а в нашем случае он совпал), то задача выполнена — М[8]=М[7]+1, а P(8)=P(7)+символ в 8 позиции S[8]=a. Получаем P(8)='aabaa'. При успешном расширении надо всего одно сравнение для вычисления очередного значения массива. Кстати, отсюда следует, что при движении вдоль массива значения могут возрастать максимум на 1.Для удобства повторю сложную строку, чтобы не нужно было перемещаться вверх-вниз.



## Переходим к трюку.

Для нахождения образца ааbаа в строке ааbааbааааbааbаааb склеим образец со строкой вот так ааbаа@ааbааbааааbааbаааb и вызовем для нее префикс-функцию для заполнения массива.



Символ '@' играет роль разделителя, его заведомо нет ни в образце, ни в строке поиска (нужно подобрать именно такой символ). Посмотрим на позиции 11, 14, 19, 22 массива. Значения в массиве равны 5, это означает, что суффикс длиной 5 (фрагмент строки поиска) совпадает с 5 символами префикса. А 5 символов префикса — это и есть образец для поиска! Алгоритм поиска получатся такой — склеиваем с помощью разделителя образец и строку поиска, передаем «склейку» префиксной функции и потом ищем в массиве элементы, равные длине образца, Можно заметить, что значений больше длины образца не будет из-за символа-разделителя, а значения, равные длине образца могут появиться только в позициях, соотвествующих исходной строке поиска. Склеенная строка имеет длину <длина образца>+<длина строки>, поэтому время расчета оценивается как О(<длина образца>+<длина строки>), как и говорилось в начале статьи. Объем необходимого префикс-функции буфера равен <длина образца>+<длина строки>, но можно модифицировать префикс-функцию так, чтобы хватило буфера <длина образца> (пример модификации в дополнении)