# Отчет

Вариант 1

Описание алгоритма:

На вход поступает строка, затем запускается цикл. Цикл идет от последнего элемента строки к первому (вообще может быть и наоборот). На первой итерации цикла берется сначала строка из одного символа, затем из двух и так далее, пока не получим исходную строки (**Пример для строки abc**: c -> bc -> abc). Для полученной строки на каждой итерации строится префикс-функция и вычисляется её максимальное значение. Затем необходимо подсчитать количество подстрок, для этого мы складываем длину строки рассматриваемой на данной итерации со значением количества на предыдущей, а также вычитаем текущее значение максимума префикс функции.

Описание алгоритм подсчета перебором:

На вход поступает строка, заводится большой массив строк. В данный массив будут записываться все уникальные подстроки исходной строки. Затем запускается цикл необходимый для того чтобы пройти по всей строке и вычленить из неё последовательно подстроки определенной длинны (Сначала длинна равна 1, затем 2, затем 3 и тд). Во вложенном цикле мы уже перебираем сами элементы строки и вычленяем значение подстроки. В еще одном цикле мы совершаем проход по элементам массива и проверяем встречалась ли данная подстрока в нем. Если да, то прерываем цикл и берем следующую подстроку, если же нет, то добавляем её в массив и увеличиваем счетчик на единицу.

Затем я сравнил скорости работы двух алгоритмов. Для это воспользовался ctime. В результате получил, что алгоритм основанный на префикс-функции для строки в 128 символов выполнялся всего пару миллисекунд, а вот алгоритм перебора для этой же строки почти 5 секунд. Для строки в 1024 символа алгоритм с использованием префикс функции показал время в 7-8 миллисекунд. В результате видно, что алгоритм использующий префикс-функции по времени значительно быстрее.