

Описание строковых алгоритмов

m – длина строки, n – длина текста, s – размер алфавита,

Алгоритм	Описание	Время на пред. обработку	Среднее время поиска	Худшее время поиска	Память
Алгоритм Кнута-Морриса-Пратта (КМП)	Алгоритм поиска подстроки в строке, который использует таблицу префикс-функций для оптимизации процесса сравнения.	$O(m)$	$O(n+m)$	$O(n+m)$	$O(m)$
Алгоритм Бойера-Мура	Алгоритм поиска подстроки в строке, основанный на 3 идеях: сканирование слева направо, сравнение справа налево эвристика стоп-символа эвристика совпавшего суффикса Наиболее эффективны в обычных ситуациях. Быстродействие повышается при увеличении строки или алфавита.	$O(m+s)$	$O(n+m)$	$O(n*m)$	$O(m+s)$
Алгоритм Рабина-Карпа	Алгоритм поиска подстроки в строке, который использует хеширование для быстрого сравнения подстрок. Сначала вычисляется хеш всех подстрок текста, затем ищется совпадение.	нет	$O(n+m)$	$O(n*m)$	нет
Алгоритм поиска строки в тексте с использованием суффиксных	Используется для поиска подстроки в строке t с помощью суффиксного массива. Для каждого запроса s_i ищется минимальный индекс в суффиксном массиве, где подстрока s_i является префиксом суффикса строки t . Поиск	$-O(n \log n)$ цифровой сортировкой $-O(n)$ Укконеном	$O(s_i + \log t)$ на запрос	$O(s_i + \log t)$ на запрос	$O(n)$

массивов	подстроки в тексте с использованием бинарного поиска по суффиксному массиву. Находит минимальный индекс, где подстрока является префиксом строки.				
Поиск подстроки в строке с помощью Z-функции	Алгоритм поиска подстроки в строке с использованием Z-функции. Строка образуется как <code>pattern + # + text</code> , где <code>#</code> — уникальный разделительный символ. После вычисления Z-функции, позиции с Z-значением, равным длине паттерна, указывают на места, где происходит совпадение.	$O(m)$	$O(n + m)$	$O(n)$	$O(n + m)$
Алгоритм Ахо-Корасик	Предназначен для поиска множества подстрок в тексте. Используется бор и суффиксные ссылки. x — количество всех возможных вхождений всех строк из словаря в тексте.	$O(\sum m)$	$O(\sum m * s + n + x)$	$O(\sum m * s + n + x)$	$O(\sum m * s)$
'Не такой уж наивный' алгоритм	Алгоритм поиска подстроки в тексте. Преимуществом является то, что время на предобработку и память константные.	$O(1)$	$O(n+m)$	$O(n*m)$	$O(1)$
Поиск наибольшей общей подстроки	Алгоритм поиска наибольшей общей подстроки двух строк x и y .	нет	$O(\log(\min(x , y)) \cdot \max(x , y))$	$O(\log(\min(x , y)) \cdot \max(x , y))$	$O(x + y)$
Алгоритм Касаи	Основан на построении суффиксного массива и поиска LCP для суффмассива. Плюс RMQ	$O(n)$	$O(m)$	$O(m)$	$O(n)$
Быстрый поиск	Алгоритм поиска подстроки в тексте. Эффективен на большом алфавите, но с увеличением длины образца эффективность снижается.	$O(m+s)$	$O(n+m)$	$O(n*m)$	$O(m+s)$