Fast sequence clustering using a suffx array algorithm

Stepachev Maxim

March 9, 2013

Основные проблемы суффиксного дерева:

1. Проблема дополнительной памяти.
2. Аллокатоция памяти.
3. Достаточно высокая сложность реализации.

Пусть у нас есть некоторый образец P и текст S, требуется найти все вхождения образца в тексте.

S: ATGTGTGC

P: TGT

Суффиксный массив — это массив лексикографически отсортированных суффиксов.
Ранг суффикса — то место, которое займет суффикс при сортировке, то есть его позиция в суффиксном массиве.
S - Текст в котором происходит поиск, P - шаблон который мы ищем.
lcp — longest common prefix

	rank	suffix array	
rosalind	7	4	alind
osalind	6	8	d
salind	8	6	ind
alind	1	5	lind
lind	4	7	nd
ind	3	2	osalind
nd	5	1	rosalind
d	2	3	salind

Сколько памяти требует суффиксный массив?

Простейший способ узнать, встречается ли образец в тексте, используя суффиксный массив, — взять первый символ образца и бинарным поиском по суффиксному массиву найти диапазон с суффиксами,начинающимися на такую же букву.

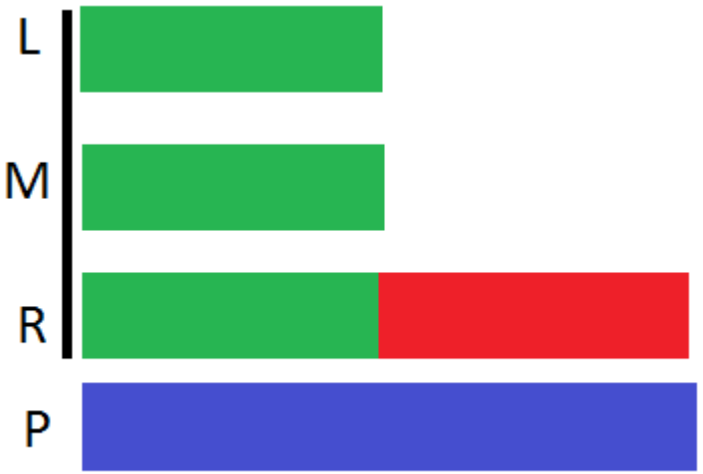
sip	**si**p	**sip**
I	i	i
Ippi	ippi	ippi
Issippi	issippi	issippi
Ississippi	ississippi	ississippi
Mississippi	mississippi	mississippi
Pi	pi	pi
Ppi	ppi	ppi
Sippi	**si**ppi	**sip**pi
Sissippi	**si**ssippi	sissippi
Ssippi	ssippi	ssippi
Ssissippi	ssissippi	ssissippi

Какова сложность простейшего поиска?

Какова сложность простейшего поиска?

Бинарный поиск работает за время равное O(log|S|), а сравнение суффикса с образцом не может превышать длины образца.
Таким образом время работы алгоритмы O(|P|log|S|), где S— текст, P— образец.

Основная идея заключается в вычислении общего префикса на краях диапазона.



Обозначения:
L=lcp(suffix[l],P)
R=lcp(suffix[r],P)
Пример:

L: ACCTCCCC
....
R: ACCTGGGG
P: ACCTGGAA

Утверждение первое: Префикс для любой строки внутри диапазона lcp не меньше, чем минимум этих двух чисел.
Утверждение второе: Если общий префикс образца и любой строки внутри диапазона не меньше m=min(L,R), то m символов можно пропускать сразу.

LCP-массив — это массив значений lcp между лексикографически-соседними суффиксами.

Построить массив lcp достаточно просто. Решение в лоб, это идти по суффиксному массиву и считать lcp соседних — это O(N^2).

Но если идти по массиву в правильном порядке, то можно использовать информацию с предыдущего шага для расчетов.

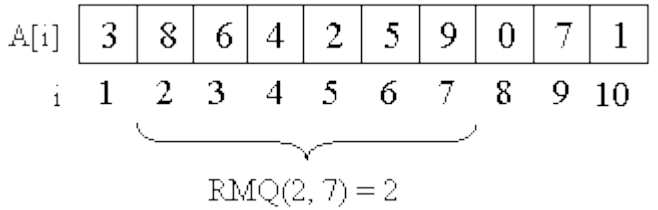
А именно, если перебирать суффиксы в том порядке, в котором они встречаются в исходной строке. Мы получим lcp-массив за линейное время O(N).

#	суффикс	шаг	lcp
1	i	10	1
2	ippi	7	1
3	issippi	4	4
4	ississippi	2	0
5	mississippi	1	0
6	pi	9	1
7	ppi	8	0
8	sippi	6	2
9	sissippi	3	1
10	ssippi	5	3
11	ssissippi	skip	

Range Minimum (Maximum) Query – запрос минимума (максимума) на отрезке в массиве.

Постановка задачи:

Пусть дан массив A[1..n]. Нам необходимо уметь отвечать на запрос вида «найти минимум на отрезке с i-ого элемента по j-ый».



Более-менее простые алгоритмы позволяют отвечать на запросы за <O(nlogn),O(1)>. Нас может устроить и такое решение, если мы строили массив за O(nlogn), но можно быстрее...

Алгоритм Фарака-Колтона и Бендера:

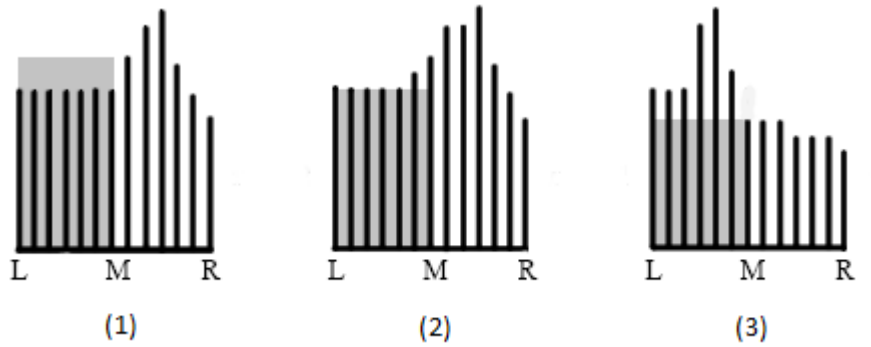
Алгоритм применяется для решения за <O(n),O(1)> времени специального случая задачи RMQ (поиск минимума на отрезке),
в котором соседние элементы входной последовательности различаются на ±1.

Применение в нашей задаче:

Построим так называемый lcp-массив — это массив значений lcp между лексикографически-соседними суффиксами.

Условные обозначения:

Lp и Rp - левая и правая границы диапазона ответов в суффиксном массиве.
L - левая граница диапазона поиска (изначально равна 0).
R -правая граница диапазона поиска (изначально равна |S|-1).
M=(R+L)/2
l=lcp(suffix[L],p)
r=lcp(suffix[R],p)
ml=lcp(suffix[L],suffix[M])
mr=lcp(suffix[M],suffix[R])
Подсчет и можно производить за O(1), если применять алгоритм Фарака-Колтона и Бендера. Любая пара суффиксов suffix[] из диапазона [L,M] имеет хотя бы ml совпадений в префиксах. Аналогично любая пара суффиксов suffix[] из диапазона [M,R] имеет хотя бы mr совпадений в префиксах.



1. ml>l,L=M, l не меняем.
2. ml=l, (R=M, r=l+k+1) или (L=M,l=l+k+1)
3. ml<l,R=M,r=ml

Операция	Suffix tree	Suffix array
Построение:	O(|s|)	O(|s|)
Поиск:	O(|p|)	O(log|s|+|p|)
Память:	O(|s|)	O(|s|)

Литература и источники

http://neerc.ifmo.ru
http://habrahabr.ru/post/115346/
http://software.intel.com/ru-ru/blogs/2012/05/22/accelerate-2012-3
http://en.wikipedia.org/wiki/Suffix_array
http://en.wikipedia.org/wiki/Substring
http://e-maxx.ru/algo/
Simple Linear Work Suffix Array Construction (http://www.cs.helsinki.fi/u/tpkarkka/publications/icalp03.pdf)
Suffix arrays:A new method for on-line string searches (http://webglimpse.net/pubs/suffix.pdf)
Fast sequence clustering using a suffix array algorithm
Handmade ;)