

Краткий конспект
Лекция 4. Суффиксные деревья
версия 0.1([draft](#))

Д. Ищенко* Б. Коварский* И. Алтухов* Д. Алексеев*

7 марта, 2016

*МФТИ

1 Задача поиска k мотивов в геноме

Вернемся к задаче поиска мотива m в геноме g . В лекции о Z-алгоритме мы выяснили, что ее можно решить за линейное время $O(|m| + |g|)$, где $|m|$ - длина мотива, $|g|$ - длина генома. Представим, что необходимо найти k различных мотивов в геноме g . Используя предыдущий метод, мы будем конструировать «Z-ящики» для каждой объединенной строки $m\$g$ и производить поиск. Соответственно временная сложность алгоритма вырастет до $O(k(|m| + |g|))$, в виду того, что $|m| \ll |g|$, сложность - $O(k|g|)$.

В реальных задачах k может иметь достаточно большое значение, например, при картировании ридов (коротких последовательностей) на референсный геном, значения k могут быть порядка $10^6 - 10^8$. Возникает необходимость оптимизации алгоритма. В этой лекции мы рассмотрим такую конструкцию, как суффиксное дерево и покажем, как с помощью него решить задачу поиска мотива и несколько других задач, существенно уменьшив временную сложность вычислений относительно Z-алгоритма.

2 Суффиксное дерево

По аналогии с ранее введенными префиксами строки S длиной n , назовем k -ым суффиксом строки S ее подстроку, начинающуюся с k -го и заканчивающуюся последним символом: $S[k..n]$. Например, для строки TATATG все множество суффиксов будет представлять из себя:

- 1: TATATG
- 2: ATATG
- 3: TATG
- 4: ATG
- 5: TG
- 6: G

Отметим, что количество всех суффиксов строки S равно ее длине, а также тот факт, что полная строка также является суффиксом самой себя (первый суффикс).

Введем понятие *суффиксного дерева*. В общем виде – это древовидный граф с набором вершин и ребер, создающийся из всех суффиксов последовательности S . В графе к каждому ребру приписана «метка», соответствующая некоторой подстроке из S . Двигаясь по ребрам от выделенной вершины суффиксного дерева, называемой корнем, к одному из листьев дерева (вершине, к которой ведет только

одно ребро) и соединяя последовательно «метки» ребер в одну последовательность, в конечном итоге мы получаем один из суффиксов исходной последовательности S . Неформально процесс создания суффиксного дерева можно описать следующим образом:

- (i) допишем в конец последовательности S символ, не встречающийся в самой последовательности, например: $\$$, тем самым получим последовательность $S^* = S\$$
- (ii) возьмем все суффиксы последовательности S^* , «закрепим» их начала в одной вершине, называемой корнем дерева (Рис. 1), в «концы» суффиксов поместим вершины и соединим их ребрами с корнем дерева. Вершины пронумеруем от 1 до n , ребрам припишем «метки» равные соответствующим суффиксам.
- (iii) «склеим» ребра с совпадающими началами, объединив их общую часть в одно ребро и «ответвив» от новой внутренней вершины несовпадающие части. Образованный граф представляет из себя суффиксное дерево для последовательности S^* .

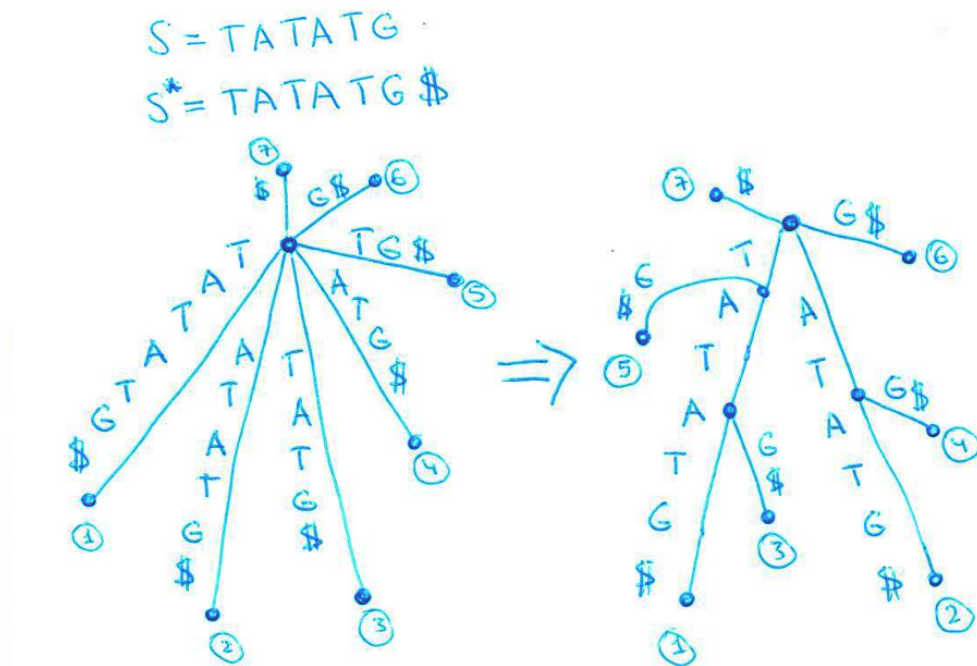


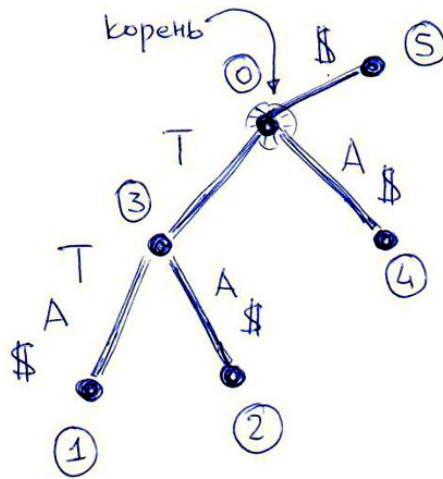
Рис. 1: Создание суффиксного дерева для последовательности S .

Перечислим некоторые свойства суффиксного дерева:

- (i) Количество листьев дерева равно количеству суффиксов последовательности S^* .
- (ii) Каждый узел в дереве, кроме корневого, имеет ровно один родительский узел.
- (iii) Двигаясь по ребрам от корня дерева к одному из *листьев* и объединяя метки соответствующих ребер в одну последовательность, последняя будет соответствовать одному из *суффиксов* последовательности S^* .
- (iv) Двигаясь по ребрам от корня дерева к одной из *внутренних вершин* дерева и объединяя метки соответствующих ребер в одну последовательность, последняя будет соответствовать некоторой *подстроке* из S^* .
- (v) Для любой подстроки S^* можно найти соответствующий ей путь от корня дерева, причем, путь не обязательно завершается в вершине дерева, а может быть завершён в середине одного из ребер.
- (vi) Метки ребер выходящих из корня или любой внутренней вершины отличаются первыми символами (иначе они были бы склеены в одно ребро). Следовательно из любой вершины не может выходить больше $\sigma + 1$ ребер, где σ – размер алфавита (количество разных символов, для нуклеотидной последовательности $\sigma = 4$).
- (vii) Если общее число вершин в дереве равно N , то число ребер: $(N - 1)$.

Предположим, что мы построили для последовательности S суффиксное дерево, как с помощью него определить количество вхождений мотива p в строку S ?

3 Алгоритм построения $O(n^2)$



```

nodes[0] = [3, 4, 5]
nodes[1] = [-1]
nodes[2] = [-1]
nodes[3] = [1, 2]
nodes[4] = [-1]
nodes[5] = [-1]

edges[0] = ""
edges[1] = "TA$"
edges[2] = "A$"
edges[3] = "T"
edges[4] = "A$"
edges[5] = "$"

```

Рис. 2: Пример массивов *nodes* и *edges* для дерева последовательности TTA\$.

```

1: procedure TREECREATION( $S$ )
2:    $S = S + "\$"$ 
3:    $nodes \leftarrow [[1], [-1]]$ 
4:    $edges \leftarrow ["", S]$ 

5:   for  $i \leftarrow 1, len(S)$  do
6:      $suf \leftarrow S[i..len(S)]$ 
7:      $j \leftarrow 0$ 
8:      $cur\_node \leftarrow 0$ 
9:      $to\_node \leftarrow -1$ 
10:     $isAdded \leftarrow False$ 

11:    while not isAdded do
12:      for  $k$  in  $nodes[cur\_node]$  do
13:        if  $suf[j] = edges[k][0]$  then
14:           $to\_node \leftarrow k$ 
15:        end if
16:      end for

17:      if  $to\_node = -1$  then
18:        ... ▷ Добавляем новый лист и ребро
19:         $isAdded \leftarrow True$ 
20:      else
21:        for  $p \leftarrow 1, len(edges[to\_node])$  do
22:          if  $suf[j + p] \neq edges[to\_node][p]$  then
23:            ... ▷ Добавляем новый лист и внутреннюю вершину
24:             $isAdded \leftarrow True$ 
25:            break
26:          end if
27:        end for
28:         $j \leftarrow j + len(edges[to\_node])$ 
29:         $cur\_node = to\_node$ 
30:      end if
31:    end while
32:  end for

33:  return  $[nodes, edges]$ 
34: end procedure

```

4 Поиск в глубину

Решаем задачу поиска мотива, как посчитать количество вхождений?

```
1: procedure LEAVESCOUNT( $i, nodes$ )
2:   if  $nodes[i][0] = -1$  then
3:     return 1
4:   end if

5:    $lCount \leftarrow 0$ 
6:   for  $k$  in  $nodes[i]$  do
7:      $lCount = lCount + LeavesCount(k, nodes)$ 
8:   end for

9:   return  $lCount$ 
10: end procedure
```

5 Какие еще задачи можно решить с помощью дерева

Поиск повтора, поиск максимальной общей подстроки, нечеткий поиск.

6 Ссылки