CSE 4502/5717 - Big Data Analytics

Lecture 12 - Note by Arun George

March 5th, 2018

Suffix Trees: Applications

Problem 4: Longest Common Substring Problem
Input : Two strings S_1 and S_2 with $s_1 = |S_1|$ and $s_2 = |S_2|$.
Output : The longest common substring between S_1 and S_2.

An Example: If $S_1 = aa\underline{babba}b$, and $S_2 = b\underline{babba}a$, then the longest common substring between S_1 and S_2 is *babba*.

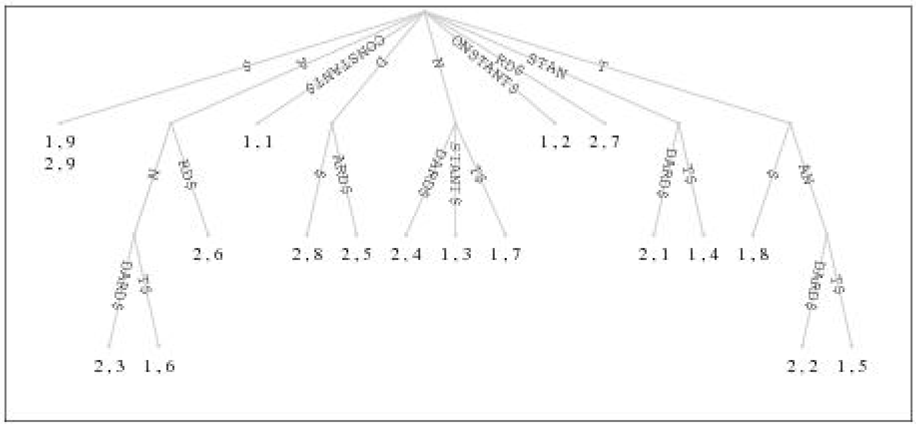
Fact : We can solve this problem in $O(M)$ time, where M = $s_1 + s_2$.

Here is an Algorithm :

1) Construct a generalized suffix tree (GST) Q for S_1 & S_2;
2) Do a traversal on Q and label any node u with 1 if there exists a leaf in the subtree rooted at u, corresponding to a suffix of S_1, and label any node u with 2 if there exists a leaf in the subtree rooted at u, corresponding to a suffix of S_2;
3) Do a traversal on Q, to identify the node u that is labeled with 1 & 2, and whose string depth is the largest. The output will be the path label of u.

An Example
If $S_1 = constant$ and $S_2 = standard$, then the longest common substring between S_1 and S_2 is *stan*.

	1	2	3	4	5	6	7	8	9
T_1	C	O	N	S	T	A	N	T	\$
T_2	S	T	A	N	D	A	R	D	\$



Problem 5

Input : Two strings S_1 and S_2, and an integer l;

Output : All substrings of S_2 of length $\geq l$, that occur in S_1

Fact : We can solve this in $O(M)$ time, where $M = |S_1| + |S_2|$.

Here is an Algorithm :

1) Construct a GST Q on S_1 & S_2. Label the nodes of Q with 1 & 2 as before;

2) Look for all the nodes u, that are labeled with 1 & 2 and whose string depths are $\geq l$; The path label of any such u is a correct answer.

Problem 6

Input : Strings S_1, $C_1, C_2, \ldots, C_k$, and an integer l;

Output : All substrings of $C_1, C_2, \ldots, C_k$ of length $\geq l$, that occur in S_1;

Fact : We can solve this problem in $O(M)$ time, where M = $|S_1| + \sum_{i=1}^{k} |C_i|$

Here is an Algorithm :

1) Construct a GST Q on S_1, & $C_1, C_2, \ldots, C_k$;

2) Mark the nodes such that a node u is marked if the subtree rooted at u has a leaf corresponding to a suffix of S_1 and a leaf corresponding to a suffix of at least one of $C_1, C_2, \ldots, C_k$;

3) If a node is marked and if its string depth is $\geq l$, then the path label of u is a correct answer.

Problem 7

Input : Strings $S_1,\ S_2, \ldots, S_k$;

Output : $l[2:n]$, such that $l[i]$ is the length of the longest common substring in $\geq i$ strings;

An Example:

$S_1 = abaabba$
$S_2 = aaababa$
$S_3 = aabaabbb$
$S_4 = aaaabbaa$
$S_5 = bbaaabab$

In this case,

$l[5] = 3$ and the corresponding longest common substring is aab;
$l[4] = 3$ and the corresponding longest common substring is aab;
$l[3] = 4$ and the corresponding longest common substring is $aabb$;
$l[2] = 6$ and the corresponding longest common substring is $aaabab$.

Claim : We can solve this in $O(Mn)$ time, where $M = \sum_{i=1}^{k} |S_i|$.

Here is an Algorithm:

1) Construct a GST Q on $S_1, S_2, \ldots, S_k$. This will take $O(M)$ time;
2) For every node v in Q compute $c[v]$ as the number of distinct strings represented in the leaves of the subtree rooted at v; An algorithm for computing these value is given below.
3) **for** $2 \leq i \leq n$ **do**
   Do a traversal on Q to identify the node v such that $c[v] = i$ and the string depth of v is the largest. Let $q[i] =$ the string depth of v.
   This step will take $O(Mn)$ time;
4) Output $l[n], l[n-1], \ldots, l[2]$ as the prefix maxima of $q[n], q[n-1], \ldots, q[2]$. This step will take $O(n)$ time.

Computing $c[]$ values:

To compute $c[]$ values use a bit array $u^v[1:n]$, for every node v in Q.

for every node v in Q **do**

If v is a leaf, then set $u^v[i] = 1$ if v has a label corresponding to a suffix of S_i; otherwise set $u^v[i] = 0$;

If v is an internal node, then, compute $u^v[1:n]$ as the boolean OR of the bit arrays of its children.

For any node $v \in Q$, we can get $c[v]$ as the number of ones in $u^v[1:n]$. The total time taken is $O(Mn)$.

Problem 8: All pairs suffix-prefix problem

Input : Strings $S_1, S_2, \ldots, S_k$;

Output : $\forall i, j$ output the length of the largest suffix of S_i that is a prefix of S_j.

An Example:

$S_1 = \underline{\text{aaba}}\text{bbaabb}\mathbf{b}$

$S_2 = \mathbf{b}\text{abaa}\underline{\text{aaba}}$

In this example, the longest suffix of S_2 that is a prefix of S_1 is $aaba$. The longest suffix of S_1 that is a prefix of S_2 is b.

An interesting application for this problem is in *de novo* sequence assembly.

Claim : We can solve this problem in $O(M + n^2)$ time, where M $= \sum_{i=1}^{n} |S_i|$.

Proof: We offer an algorithm. Construct a GST Q on $S_1, S_2, \ldots, S_k$.

Definition : Let an edge be terminal if it is labeled with \$ with one end point being a leaf.

For every S_j there is a leaf in Q labeled (j,1), the path from the root to this leaf corresponding to $S_j, 1 \le j \le n$.

(To be continued)