CSE 5095: Research Topics in Big Data Analytics **Professor Rajasekaran**
Lecture 16: 3/25/14 **Notes by Christopher Kuhn**

In this lecture we'll present the linear time algorithm of (Kärkäinen and Sanders 2003) for the construction of the suffix array for any given input string.

Let $T = t_0\ t_1\ \ldots\ t_{m-1}$ be the given input string.
For $k = 0, 1$, and 2 define $B_k = \{i \in [\,0, m] : i \bmod 3 = k\}$
Let $B = B_1 \cup B_2$.
Let S_i stand for the suffix of T starting at position i, for $0 \leq i \leq m$-1.
Let S_c denote the collection of suffixes S_j for each $j \in C$, where $C \subseteq [0, m\text{-}1]$.

Algorithm:
1. Sort the suffixes S_B; Let this sorted sequence be Q;
2. Using the order obtained in step 1, sort the suffixes S_{B_0} to get Q';
3. Merge Q with Q';

Note: It suffices to assume that $\sum = [1, m]$

For $k = 1$ or 2 define:

$$R_k = [\ t_k\ t_{k+1}\ t_{k+2}\ ]\ [\ t_{k+3}\ t_{k+4}\ t_{k+5}\ ]\ \ldots\ [\ t_{maxBk}\ t_{maxBk+1}\ t_{maxBk+2}\ ]$$

In this string, each substring of length 3 enclosed within square brackets is thought of as a single super character. Any such super character is an integer in the range $[1, m^3]$.

Example:

Position	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
T =	5	2	1	4	3	3	1	5	3	4	4	1

1. R_1 = [2 1 4] [3 3 1] [5 3 4] [4 1 0]
   R_2 = [1 4 3] [3 1 5] [3 4 4] [1 0 0]

Construct the string R = R_1 R_2. In the above example,

R = [2 1 4] [3 3 1] [5 3 4] [4 1 0] [1 4 3] [3 1 5] [3 4 4] [1 0 0]

Observation: The relative ordering of the suffixes in R is the same as the relative ordering of the suffixes in S_B.

1a. Sort the super characters in R using radix sort in linear time and replace each super character with its sorted rank. As a result, each super character is replaced with an integer in the range [1, $|R|$]. If the characters in R are now distinct, we are done with sorting S_B.

Rank	3	5	8	7	2	4	6	1
R =	2 1 4	3 3 1	5 3 4	4 1 0	1 4 3	3 1 5	3 4 4	1 0 0

1b. If the characters in R are not distinct, then recursively sort the suffixes in the resultant string (where each character is an integer in the range [1, $|R|$]).

2. To sort S_{B_0}:
Let rank(S_i) be the rank (among the suffixes in S_B) of the suffix S_i where $i \in B$.
Note: $S_j \leq S_k$ where $j, k \in B_0$ if and only if (t_j, rank(S_{j+1})) $\leq$ (t_k, rank(S_{k+1}))

Example1: $S_3 \leq S_0$ since (4, 5) $\leq$ (5, 3)

Example2: $S_3 \leq S_9$ since (4, 5) $\leq$ (4, 7)

To sort S_{B_0}, sort pairs of the form (t_j, rank(S_{j+1})) for $j \in B_0$ using integer sort. This takes O(m) time.

3. Merging Q and Q':
Let S_i and S_j be two suffixes such that $S_i \in B_0$ and $S_j \in B_1$ or B_2

Case 1: $S_j \in B_1$

$S_i \leq S_j$ if and only if (t_i, rank(S_{i+1})) $\leq$ (t_j, rank(S_{j+1}))

Case 2: $S_j \in B_2$

$S_i \leq S_j$ if and only if (t_i, t_{i+1}, rank(S_{i+2})) $\leq$ (t_j, t_{j+1}, rank(S_{j+2}))

Let $T(m)$ be the RUN TIME of this algorithm on any string of length m.

Then, $T(m) = T(\frac{2}{3} m) + O(m) = O(m)$.

Note: This algorithm is known as the skew algorithm (since the split is not ½, ½).

DATA MINING

Association Rules Mining:
Input: A set of transactions $= \{t_1, t_2, ..., t_n\}$
Let I be a set of possible items. Let $I = \{i_1, i_2, ..., i_d\}$.
Each $t_i \subseteq I$, $1 \leq i \leq n$.

An association rule is an implication $X \rightarrow Y$ where $X \neq \phi$; $Y \neq \phi$; $X \cap Y = \phi$; $X \subseteq I$; and $Y \subseteq I$.

Definition: An itemset is a set of items. A k-itemset is an itemset with k items.
Let X be and itemset. Then the support for X, denoted as $\sigma(X)$, is defined as the number of transactions that contain X.

Support for a rule $X \rightarrow Y$ is $\sigma(X \cup Y)/n$.

Confidence for the rule $X \rightarrow Y$ is $\sigma(X \cup Y)/\sigma(X)$.

Problem:
Given a database *DB* of transactions, and two real numbers *minSupport* and *minConfidence,* find all the rules $X \rightarrow Y$ whose support is ≥ *minSupport* and whose confidence is ≥ *minConfidence*.

Definition: An itemset is frequent if its support is ≥ *n. minSupport.*

The problem of finding association rules is normally done in two steps.

1. Find all the frequent itemsets; and
2. Using the frequent itemsets generate all the relevant association rules.

Example: If X is frequent and $X = X_1 \cup X_2$, then check if $X_1 \rightarrow X_2$ has enough confidence, for every nonempty and proper subset X_1 of X.