CSE 4502/5717: Big Data Analytics
Lecture 15: 3/28/18

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 $0 \le i \le m$-1.
Let S_c denote the collection of suffixes S_j for each $j \in C$, where $C \subseteq [0, m\text{-}1]$.

<u>Algorithm</u>:
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 $\sum = [1, m]$. This is because if the size of the alphabet is larger than m, we can sort the characters in the input and replace each character with its rank in the sorted list.

Let

$R_1 = [\, t_1\, t_2\, t_3\,]\, [\, t_4\, t_5\, t_6\,] \ldots [\, t_{m-2}\, t_{m-1}\, 0\,]$ and
$R_2 = [\, t_2\, t_3\, t_4\,]\, [\, t_5\, t_6\, t_7\,] \ldots [\, t_{m-1}\, 0\, 0\,]$

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]

<u>Observation</u>: 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 rank in the sorted list. 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. <u>To sort S_{B_0}</u>:
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})) ≤ (t_k, rank(S_{k+1}))

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

Example2: $S_3 \leq S_9$ since (4, 5) ≤ (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. <u>Merging Q and Q'</u>:
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})) ≤ (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})) ≤ (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, $T(m) = T(⅔\, m) + O(m) = O(m)$.

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