Induced Sorting

Define three type of suffixes $-$, $+$ and $*$ as follows:

$$C^- = \{i \in [0..n) \mid T_i > T_{i+1}\}$$
$$C^+ = \{i \in [0..n) \mid T_i < T_{i+1}\}$$
$$C^* = \{i \in C^+ \mid i-1 \in C^-\}$$

Example 5.23:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$T[i]$	m	m	i	s	s	i	s	s	i	i	p	p	i	i	$
type of T_i	$-$	$-$	$*$	$-$	$-$	$*$	$-$	$-$	$*$	$+$	$-$	$-$	$-$	$-$	

For every $a \in \Sigma$ and $x \in \{-,+.*\}$ define

$$C_a = \{i \in [0..n] \mid T[i] = a\}$$
$$C_a^x = C_a \cap C^x$$

Then

$$C_a^- = \{i \in C_a \mid T_i < a^{n+1}\}$$
$$C_a^+ = \{i \in C_a \mid T_i > a^{n+1}\}$$

and thus the suffix array is $C_0 C_1^- C_1^+ C_2^- C_2^+ \ldots C_{\sigma-1}^- C_{\sigma-1}^+$.

The basic idea of induced sorting is to use information about the order of T_i to **induce** the order of the suffix $T_{i-1} = T[i-1]T_i$. The main steps are:

1. Sort the sets C_a^*, $a \in [1..\sigma)$.
2. Use C_a^*, $a \in [1..\sigma)$, to induce the order of the sets C_a^-, $a \in [1..\sigma)$.
3. Use C_a^-, $a \in [1..\sigma)$, to induce the order of the sets C_a^+, $a \in [1..\sigma)$.

The suffixes involved in the induction steps can be indentified using the following rules (proof is left as an exercise).

Lemma 5.24: For all $a \in [1..\sigma)$

(a) $i-1 \in C_a^-$ iff $i > 0$ and $T[i-1] = a$ and one of the following holds
   1. $i = n$
   2. $i \in C^*$
   3. $i \in C^-$ and $T[i-1] \geq T[i]$.

(b) $i-1 \in C_a^+$ iff $i > 0$ and $T[i-1] = a$ and one of the following holds
   1. $i \in C^-$ and $T[i-1] < T[i]$
   2. $i \in C^+$ and $T[i-1] \leq T[i]$.

To induce $-$-type suffixes:
1. Set C_a^- empty for all $a \in [1..\sigma)$.
2. For all suffixes T_i such that $i-1 \in C^-$ **in lexicographical order**, append $i-1$ into $C_{T[i-1]}^-$.

Note that since $T_{i-1} > T_i$ by definition of C^-, we always have i inserted before $i-1$.

Algorithm 5.25: InduceMinusSuffixes
Input: Lexicographically sorted lists C_a^*, $a \in \Sigma$
Output: Lexicographically sorted lists C_a^-, $a \in \Sigma$

for $a \in \Sigma$ *do* $C_a^- \leftarrow \emptyset$
pushback$(n-1, C_{T[n-1]}^-)$
for $a \leftarrow 1$ *to* $\sigma - 1$ *do*
    $C \leftarrow \emptyset$
    while $C_a^- \neq \emptyset$ *do*
        $i \leftarrow$ *popfront*(C_a^-)
        pushback(i, C)
        if $i > 0$ *and* $T[i-1] \geq a$ *then pushback*$(i-1, C_{T[i-1]}^-)$
    $C_a^- \leftarrow C$
    for $i \in C_a^*$ *do pushback*$(i-1, C_{T[i-1]}^-)$

Inducing $+$-type suffixes goes similarly but in reverse order so that again i is always inserted before $i-1$:
1. Set C_a^+ empty for all $a \in [1..\sigma)$.
2. For all suffixes T_i such that $i-1 \in C^+$ in **descending** lexicographical order, append $i-1$ into $C_{T[i-1]}^+$.

Algorithm 5.26: InducePlusSuffixes
Input: Lexicographically sorted lists C_a^-, $a \in \Sigma$
Output: Lexicographically sorted lists C_a^+, $a \in \Sigma$

for $a \in \Sigma$ *do* $C_a^+ \leftarrow \emptyset$
for $a \leftarrow \sigma - 1$ *downto* 1 *do*
    $C \leftarrow \emptyset$
    while $C_a^+ \neq \emptyset$ *do*
        $i \leftarrow$ *popback*(C_a^+)
        pushfront(i, C)
        if $i > 0$ *and* $T[i-1] \geq a$ *then pushfront*$(i-1, C_{T[i-1]}^+)$
    $C_a^+ \leftarrow C$
    for $i \in C_a^-$ in reverse order *do*
        if $i > 0$ *and* $T[i-1] < a$ *then pushfront*$(i-1, C_{T[i-1]}^+)$

We still need to explain how to sort the $*$-type suffixes. Define

$$F[i] = \min\{k \in [i+1..n] \mid k \in C^* \text{ or } k = n\}$$
$$S_i = T[i..F[i]]$$
$$S'_i = S_i\sigma$$

where σ is a special symbol larger than any other symbol.

Lemma 5.27: For any $i, j \in [0..n)$, $T_i < T_j$ iff $S'_i < S'_j$ or $S'_i = S'_j$ and $T_{F[i]} < T_{F[j]}$.

Proof. The claim is trivially true except in the case that S_j is a proper prefix of S_i (or vice versa). In that case, $S_i > S_j$ but $S'_i < S'_j$ and thus $T_i < T_j$ by the claim. We will show that this is correct.

Let $\ell = j + |S_j| - 1$ and $k = i + \ell - j$. Then

- $\ell \in C^*$ and thus $\ell - 1 \in C^-$. By Lemma 5.24, $T[\ell] < T[\ell - 1]$.
- $T[k-1..k] = T[\ell-1..\ell]$ and thus $T[k] < T[k-1]$. If we had $k \in C^+$, we would have $k \in C^*$. Since this is not the case, we must have $k \in C^-$.
- Let $a = T[\ell]$. Since $\ell \in C_a^+$ and $k \in C_a^-$, we must have $T_k < a^{n+1} < T_\ell$.
- Since $T[i..k) = T[j..\ell)$ and $T_k < T_\ell$, we have $T_i < T_j$.

□

Algorithm 5.28: SAIS

Step 0: Choose C.

- Compute the types of suffixes. This can be done in $\mathcal{O}(n)$ time based on Lemma 5.24.
- Set $C = \cup_{a \in [1..\sigma)} C_a^* \cup \{n\}$. Note that $|C| \le n/2$, since for all $i \in C$, $i - 1 \in C^- \subseteq \bar{C}$.

Example 5.29:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$T[i]$	m	m	i	s	s	i	s	s	i	i	p	p	i	i	$
type of T_i	$-$	$-$	$*$	$-$	$-$	$*$	$-$	$-$	$*$	$+$	$-$	$-$	$-$	$-$	

$C_i^* = \{2, 5, 8\}$, $C_m^* = C_p^* = C_s^* = \emptyset$, $C = \{2, 5, 8, 14\}$.

Step 1: Sort T_C.

- Sort the strings S'_i, $i \in C^*$. Since the total length of the strings S'_i is $\mathcal{O}(n)$, the sorting can be done in $\mathcal{O}(n)$ time using LSD radix sort.
- Assign lexicographic names $N_i \in [1..|C|-1]$ to the string S'_i so that $N_i \le N_j$ iff $S'_i \le S'_j$.
- Construct the sequence $R = N_{i_1} N_{i_2} \dots N_k 0$, where $i_1 < i_3 < \cdots < i_k$ are the $*$-type positions.
- Construct the suffix array SA_R of R. This is done recursively unless all symbols in R are unique, in which case a simple counting sort is sufficient.
- The order of the suffixes of R corresponds to the order of $*$-type suffixes of T. Thus we can construct the lexicographically ordered lists C_a^*, $a \in [1..\sigma)$.

Example 5.30:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$T[i]$	m	m	i	s	s	i	s	s	i	i	p	p	i	i	$
N_i			2			2			1						0

$R = [\texttt{issiz}][\texttt{issiz}][\texttt{iippii\$z}]\$ = 2210$, $SA_R = (3, 2, 1, 0)$, $C_i^* = (8, 5, 2)$

Step 2 Sort $T_{[0..n]}$.

- Run InduceMinusSuffixes to construct the sorted lists C_a^-, $a \in [1..\sigma)$.
- Run InducePlusSuffixes to construct the sorted lists C_a^+, $a \in [1..\sigma)$.
- The suffix array is $SA = nC_1^- C_1^+ C_2^- C_2^+ \dots C_{\sigma-1}^- C_{\sigma-1}^+$.

Example 5.31:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$T[i]$	m	m	i	s	s	i	s	s	i	i	p	p	i	i	$
type of T_i	$-$	$-$	$*$	$-$	$-$	$*$	$-$	$-$	$*$	$+$	$-$	$-$	$-$	$-$	

$C_\$ = (14) \Rightarrow C_i^- = (13, 12)$
$C_i^- C_i^* = (13, 12, 8, 5, 2) \Rightarrow C_m^- = (1, 0),\ C_p^- = (11, 10),\ C_s^- = (7, 4, 6, 3)$
$\Rightarrow C_i^+ = (8, 9, 5, 2)$
$\Rightarrow SA = C_\$ C_i^- C_i^+ C_m^- C_p^- C_s^- = (14, 13, 12, 8, 9, 5, 2, 1, 0, 11, 10, 7, 4, 6, 3)$

Theorem 5.32: Algorithm SAIS constructs the suffix array of a string $T[0..n)$ in $\mathcal{O}(n)$ time plus the time needed to sort the characters of T.

- In Step 1, to sort the strings S'_i, $i \in C^*$, we can replace LSD radix sort with the following procedure (proof omitted):
  1. Construct the sets C^*_a, $a \in [1..\sigma)$ **in arbitrary order**.
  2. Run InduceMinusSuffixes to construct the lists C^-_a, $a \in [1..\sigma)$.
  3. Run InducePlusSuffixes to construct the lists C^-_a, $a \in [1..\sigma)$.
  4. Remove non-*-type positions from $C^+_1 C^+_2 \ldots C^+_{\sigma-1}$.

- With this change, most of the work is done in the induction procedures. This is very fast in practice, because all the lists C^x_a are accessed **sequentially** during the procedures.