Suffix Links

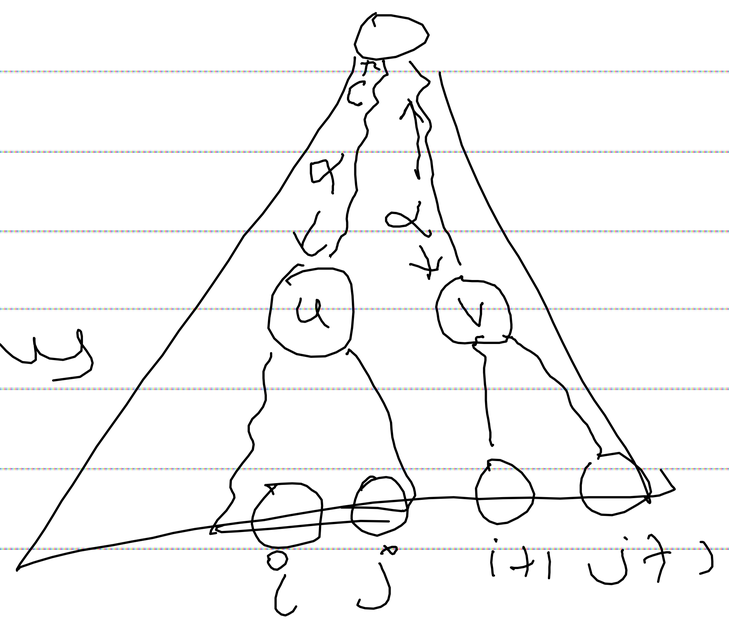
Let u be an internal node or leaf s.t
path-label$(u) = c\alpha$ where $c \in \Sigma$, $\alpha \in \Sigma^*$
(c → character, α → string)

Definition:
The suffix link of node (u) is a pointer to another node (v) s.t.
path-label$(v) = \alpha$

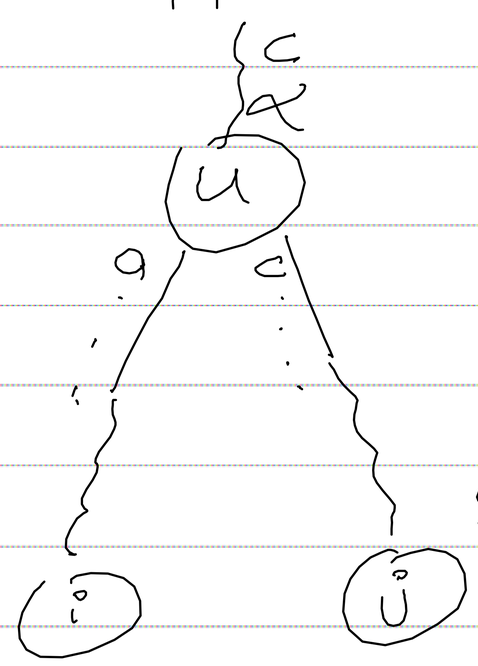
Lemma 1: For each node u, there exists a corresponding node (v) s.t., suffixlink$(u) = v$

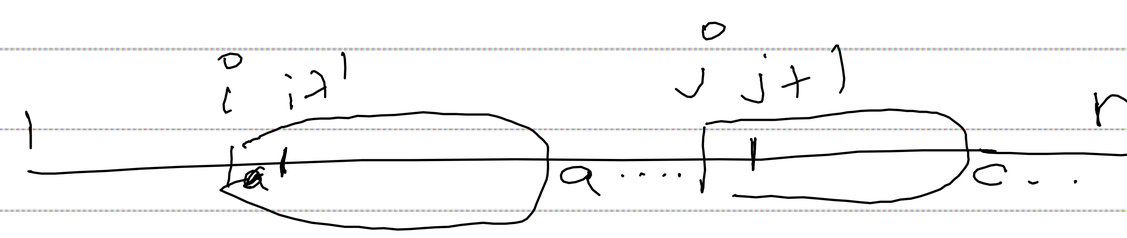
Proof:

Since u is an internal node,
$\Rightarrow$ If two suffixes i & j, under its subtree in two different branches

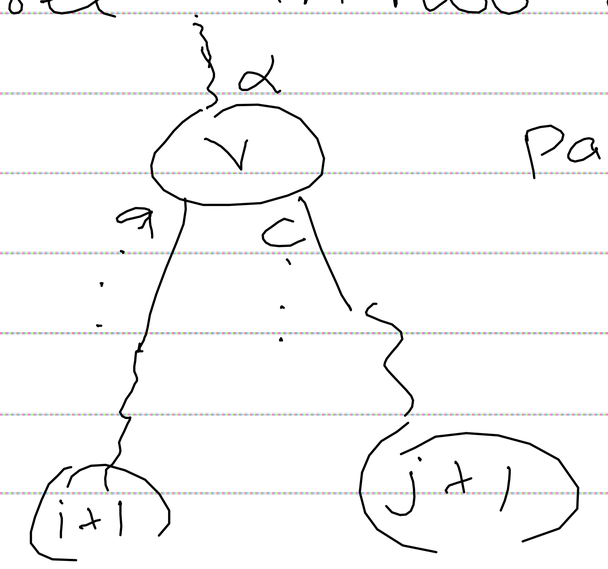


e.g.,





$\Rightarrow$ There has to exist an internal node v which has suffixes $i+1$ & $j+1$ under its subtree in two different branches



path-label$(v) = \alpha$

□.

Corollary:

By definition of suffix tree & suffix link, each node u has exactly one suffix link. □

SUFFIX TREE CONSTRUCTION ALGO

McCreight's algorithm: (Input: $S[1..n]$)

1) Initialize an empty tree $\Rightarrow T_0$

2) For $i = 1$ to n do {

$T_i \leftarrow$ "insert" suffix i into T_{i-1}

}

3) Output T_n (same as $ST(S)$).

$T_0 \Rightarrow T_1 \Rightarrow \ldots \boxed{T_{i-1} \Rightarrow T_i \Rightarrow} \ldots \Rightarrow T_n$

(Q) How to insert $suff_i$ into T_{i-1}?

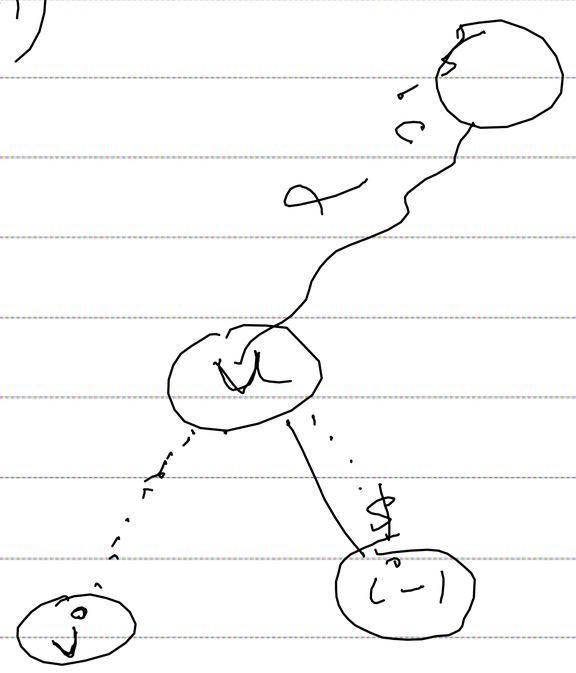
Let at the end of iteration $i-1$, $leaf_{i-1}$ ← correspond to suffix $i-1$

& let u be $leaf_{i-1}$'s parent

(2 possibilities)

u is new (ie., got created in iteration $i-1$)

u already existed before iteration $i-1$



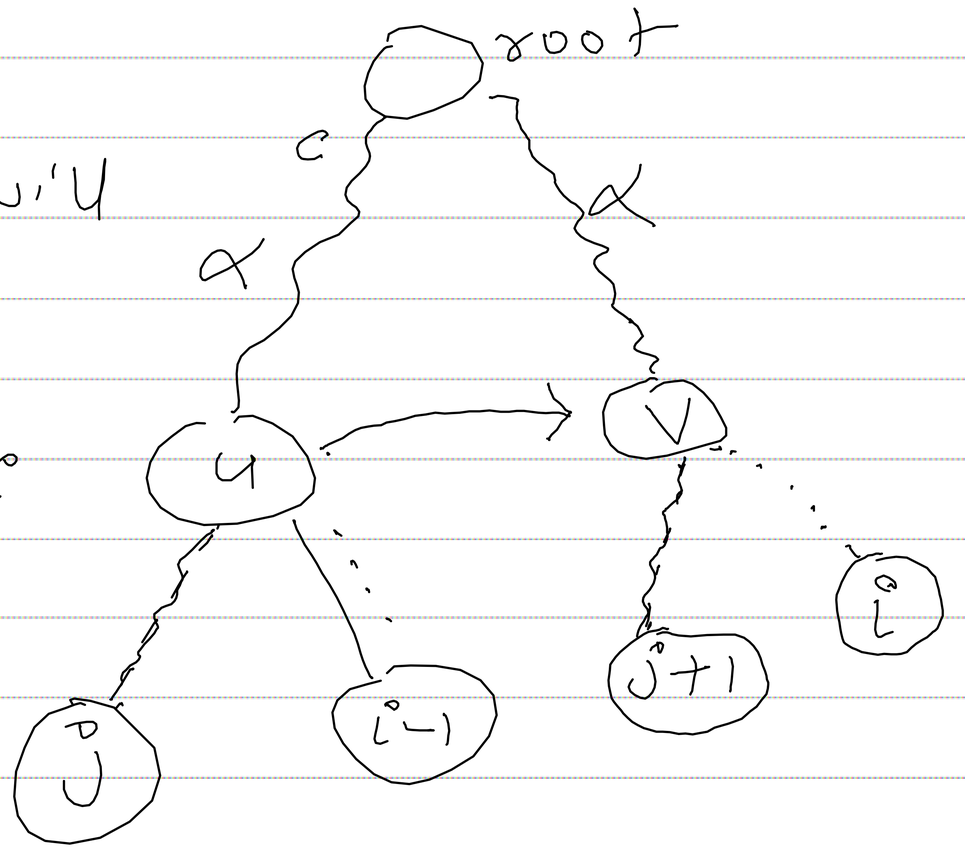
Lemma: If u is an internal node that was newly created in iteration $i-1$, then its suffix link will be definitely established by the end of iteration i.

Proof:

At the end of iteration i, $suff_i$ will be inserted

$\Rightarrow$ longest common prefix (or) "lcp" of $suff_{j+1}$ & $suff_i$ will be α

i.e., $lcp(j+1, i) = \alpha$



$\Rightarrow$ There has to exist an internal node at the end of α's path from root

$\Rightarrow$ $SL(u)$ will be established at the end of iteration i. □

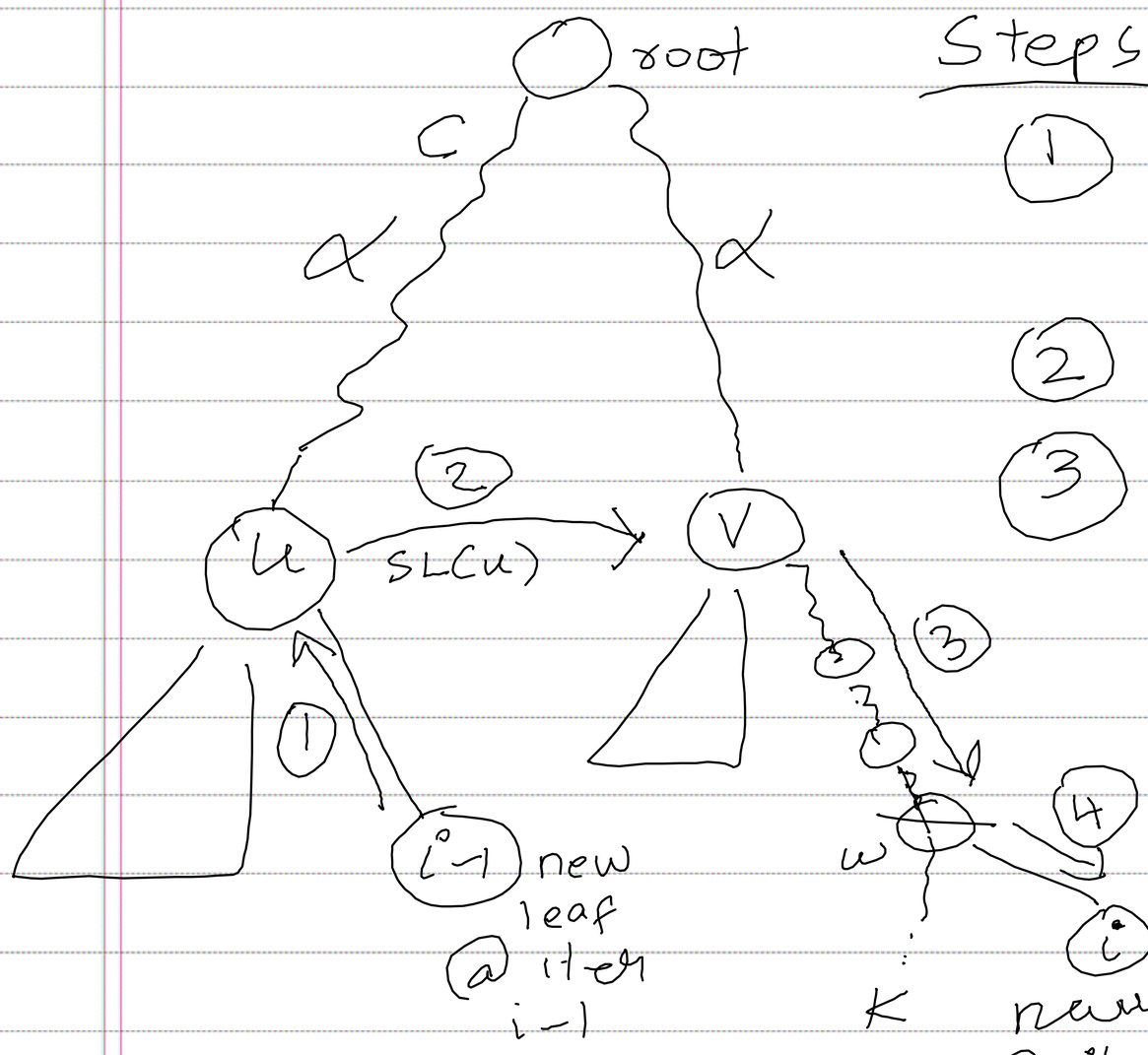
Using the above lemma, we can construct T_i from T_{i-1} as follows:

CASE A) u was created new @ iteration $i-1$

CASE B) u was already there before iteration $i-1$.

For CASE B) If u already existed before iter. i-

⇒ SL(u) is already defined



Steps:

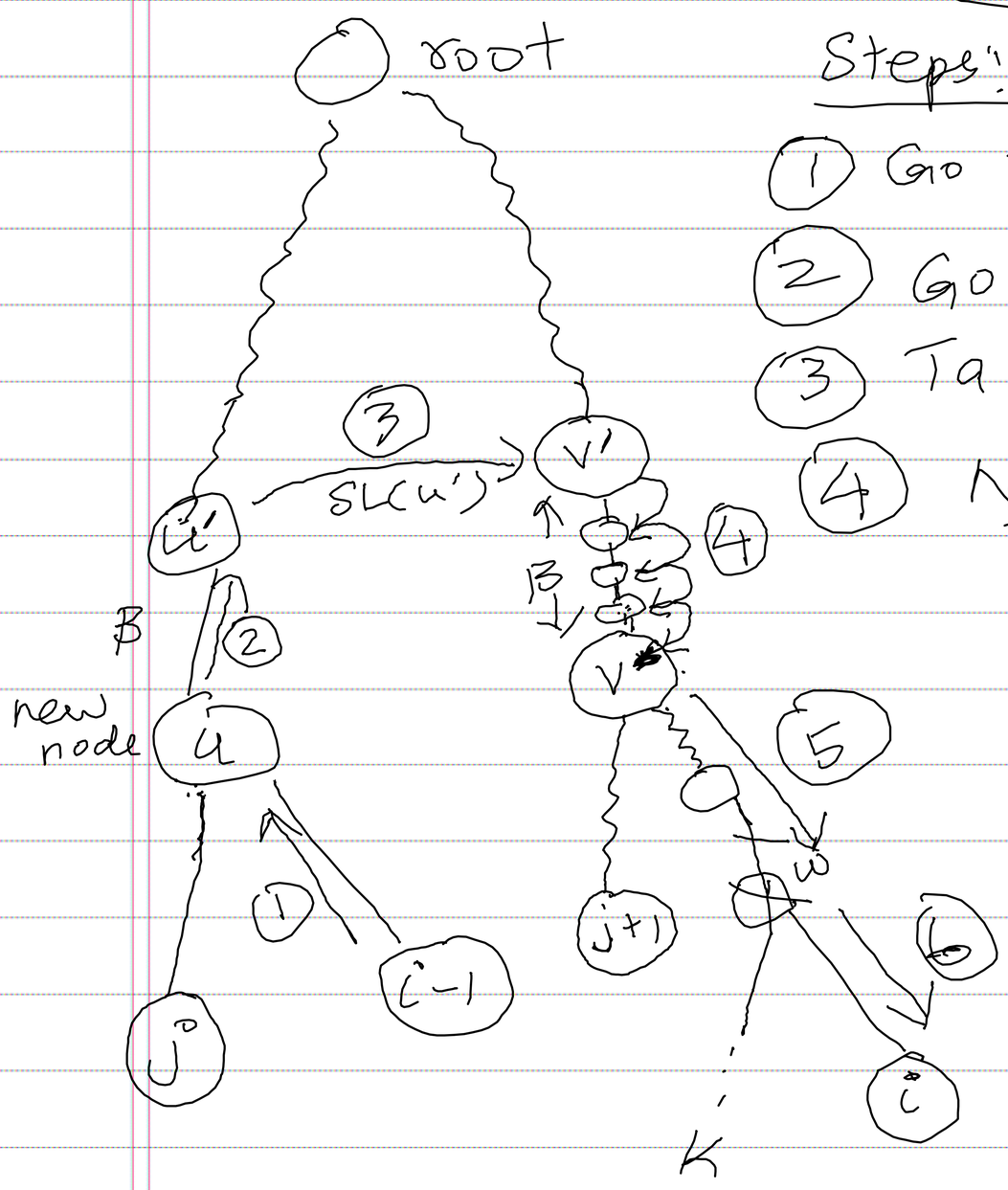
(1) from leaf i-1, go up to one level to u

(2) Take SL(u) pointer to v

(3) From v, do character comparisons for the remaining part of suffix i. (minus α) ie:,

(4) When matching stops, create a new internal node w (if it already does not exist) and spawn a new branch to complete leaf i

For CASE A)



Steps:

(1) Go to leaf i-1's parent u' (but u just got created; so no SL(u) yet)

(2) Go to u's parent: u'

(3) Take SL(u') to v'

(4) Node Hops:

Let β ← edge-label from u' to u

From v', hop node to node until you exhaust β

(or)

land in an int. node (v)

land in the middle of an edge

In this case, create node v

establish SL(u) ⇒ v

— Steps (5) & (6) are same as steps (3) & (4) for case A

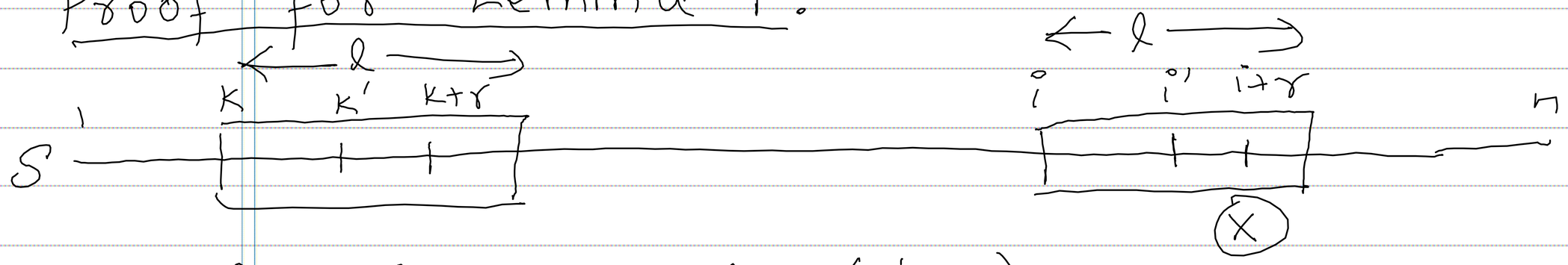
Analysis: (amortized analysis)

Lemma 1: The # character comparisons (total amortized) over all iterations in steps (3) of case B and step (5) of case A is $O(n)$

Lemma 2: The # node hops (total amortized) over all iterations in step (4) of case A is $O(n)$

Corollary: McCreight Algo's total time complexity = $O(n)$

Proof for Lemma 1:



$l \leftarrow$ length of lcp (i, k)

(X) charge comparison of $s[i+r]$ vs. $s[k+r]$ to position $i+r$

To show, $i+r$ will not be charged again,

$\Rightarrow$ consider i' s.t $i < i' < i+r$

charging $s[i+r]$ may be possible only when inserting another suffix like i'

But could it happen?

NO. Because the comparison betn $suff_{i'}$ & $suff_{k'}$ will start at least as $s[k+l]$ vs. $s[i+l]$ (due to suffix links).

□

Proof for Lemma 2:

In case A), from u we go to u' and then to v'

Define "node-depth" of a node to the number of edges from the root to that node in the tree.

$$\text{node-depth}(u') = \text{node-depth}(u) - 1$$

Also, $\text{node-depth}(v') \geq \text{node-depth}(v) - 1$ (due to suffix link property)

$\Rightarrow$ $\text{node-depth}(v') \geq \text{node-depth}(u) - 2$

$\Rightarrow$ When going from u to v', we are ascending atmost 2 levels up (relative to u)

$\Rightarrow$ Since there could be only a total of n levels atmost in the tree,
over all iterations, the # total number of levels to ascend $\leq 2 \times n$

$\Rightarrow$ Amortized analysis for # node hops over all iterations $\leq 2n$

$= O(n)$