# McCreight's Suffix Tree Construction Algorithm: Linear time $O(n)$ ①

Input : $S[1..n]$

Algo:

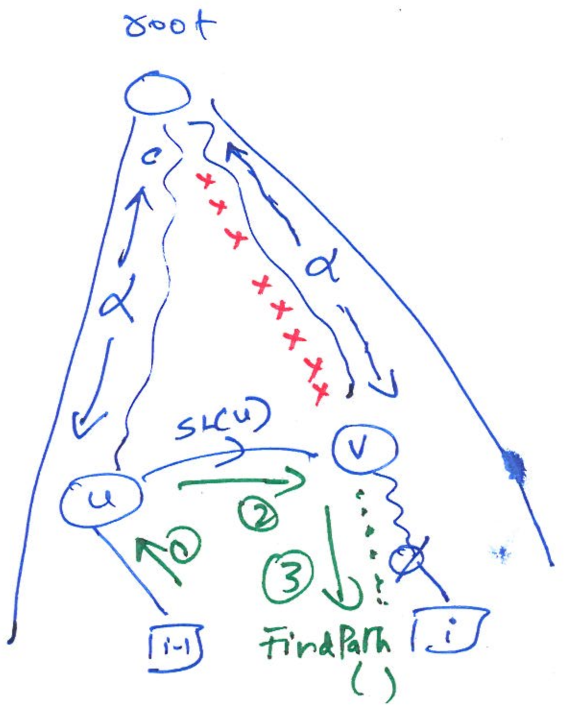
```
Init: $T_\phi \leftarrow$ root node
For i=1 to n do:
{
    $T_i \leftarrow$ "insert" suffix i into $T_{i-1}$
}
$ST(S) \leftarrow T_n$
```

$$T_0 \Rightarrow T_1 \Rightarrow \dots \; T_{i-1} \xrightarrow[\text{ins}(\text{suff}_i)]{} T_i \Rightarrow \dots \; \underset{\substack{\uparrow \\ ST(S)}}{T_n}$$

Main Idea: For inserting $\text{suff}_i$ into $T_{i-1}$ use suffix links

Illustration of the main Idea:

$T_{i-1}$ :



• characters compared to insert $\text{suff}_i$'s leaf

× corresponds to characters (which constitute α) that are not compared
⇒ savings.

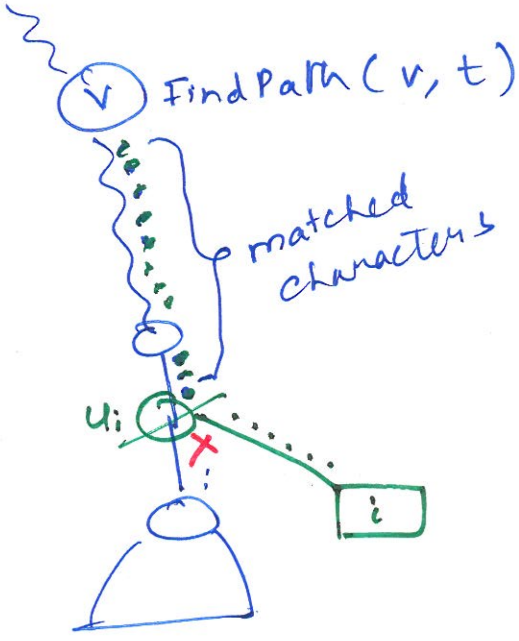
Find Path (node *v, string t)

Goal: Finds the path ~~that~~ ~~spe~~ to insert suffix i

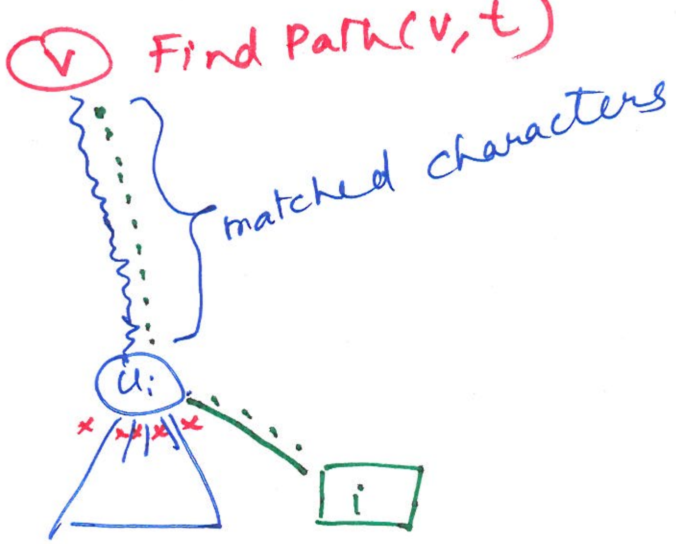
```
{
  1) Find the path that spells out the longest possible prefix of
     string t, under v's subtree

  2) If the mismatch's location is along an edge (i.e, before
case i)  the edge label is exhausted), then break that edge,
     introduce a new internal node, "$u_i$", under which
     leaf i (for $suff_i$) is created.
  (or) If the mismatch's location ends at the end of ~~an~~ edge
     => that means there must already be an ~~internode~~ internal node
case ii)  there. ~~So~~ That is $u_i$. Insert $suff_i$'s leaf under $u_i$.
}
```

case i)

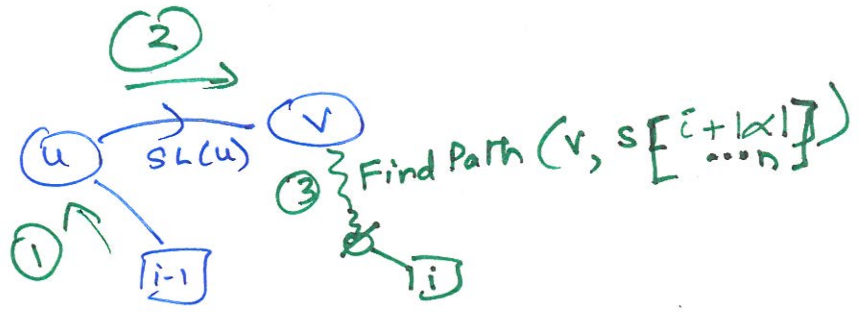


case ii)

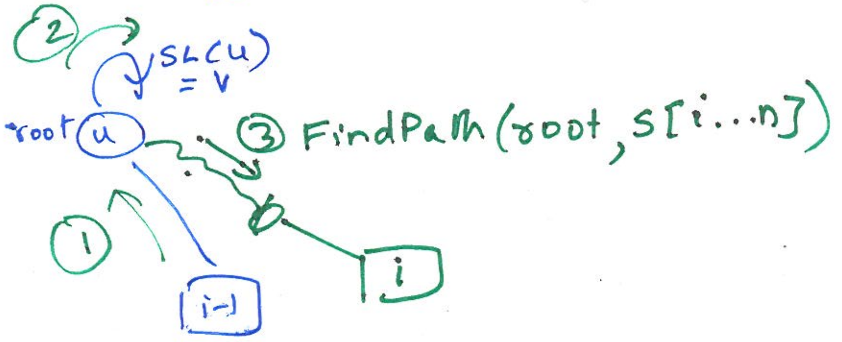


Four Cases: Note: pathlabel(u) = c$\alpha$ $\Rightarrow$ pathlabel(v) = $\alpha$
(for case IA)

Case I A) SL(u) is known & u ≠ root



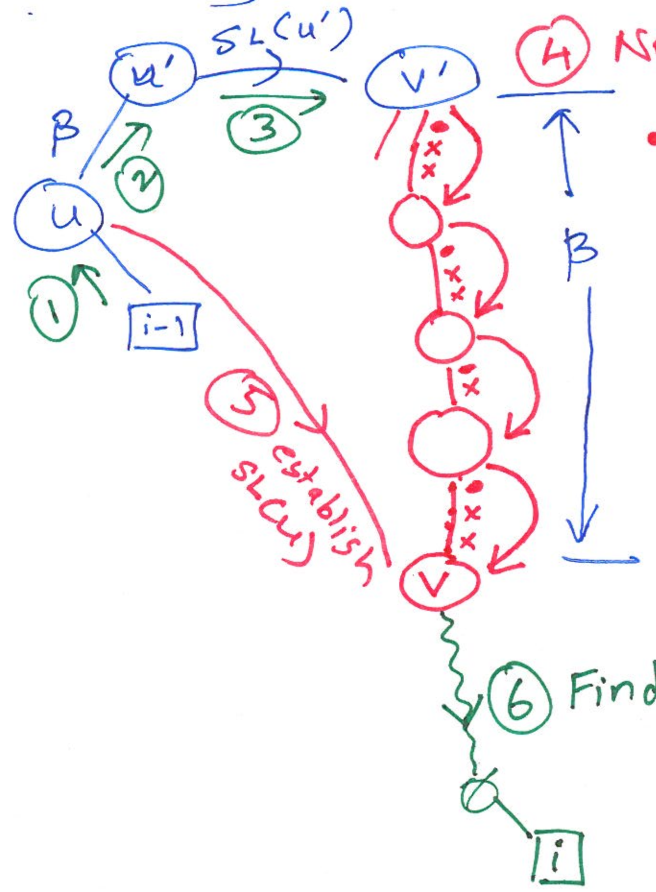
Case I B) SL(u) is known & u = root



Case II A) SL(u) is unknown & u ≠ root

Case II B) SL(u) is known & u = root

Case II A) SL(u) is unknown & $u' \neq$ root (Note: $u' = \text{parent}(u)$)



4 Node Hops($v'$, $\beta$):

- only first character of an edge is compared against the relative character of $\beta$

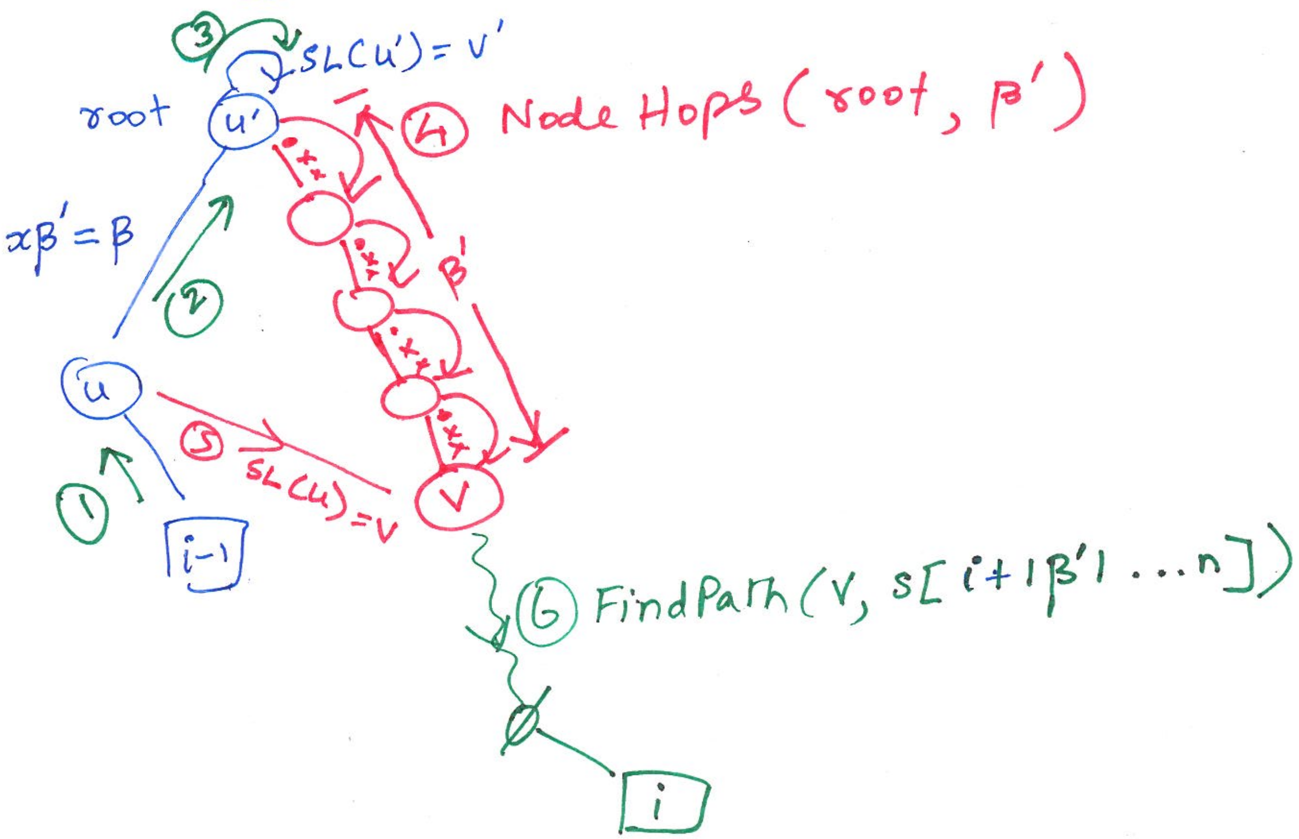
x ~~r~~ remaining characters of ~~β~~ edge are not compared (i.e., skipped)

Note: you may have to break an edge to create node v (at the end of $\beta$) if no such internal node already exists

6 Find Path ($v$, $S[i+|\alpha| \ldots n]$)

Case II B) $SL(u)$ is unknown & $u' = root$ (we split $\beta = x\beta'$, where $x \in \Sigma$, $\beta' \in \Sigma^*$)

Note: for this case $\alpha$ will be same as $\beta$.



③ SL(u') = v'
root
u'
④ Node Hops (root, β')
xβ' = β
②
β'
u
⑤ SL(u) = v
①
v
i-1
⑥ FindPath(v, s[i+|β'|...n])
i