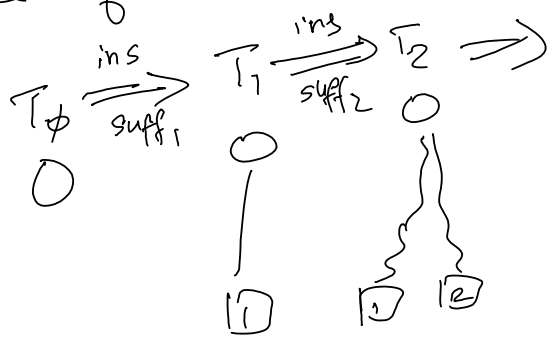
Suffix Tree Construction

Friday, March 5, 2021 10:11 AM

Goal: Given a string $S[1..n]$ over alphabet Σ, construct the suffix tree of S ($ST(S)$).



Naive Algorithm:

Build ST Naive (S [1..n]) {

Init: $T_0 \leftarrow$ root

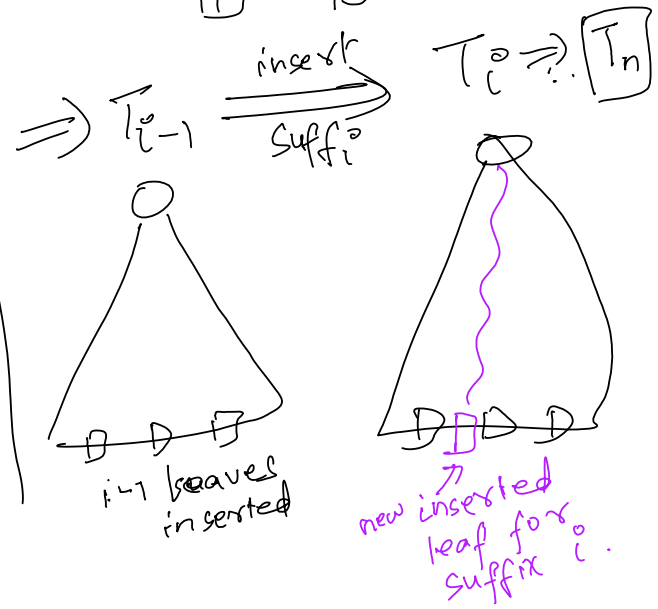
$O(n)$ for (i = 1 to n) do {

$O(n) \rightarrow T_i \leftarrow$ Insert $suff_i$ into T_{i-1}

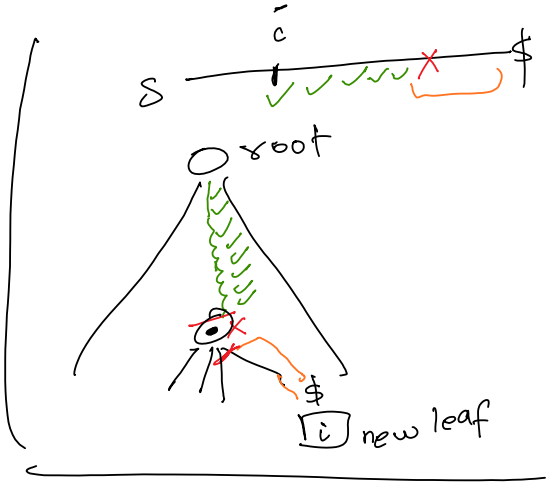
}

$ST(S) \leftarrow T_n$ //return

}

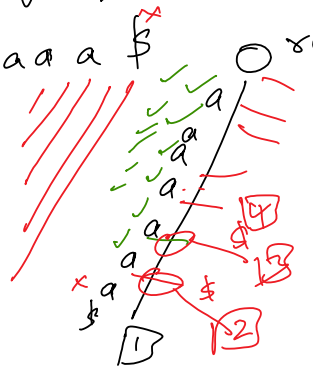


Find Path (root, S[i...n])



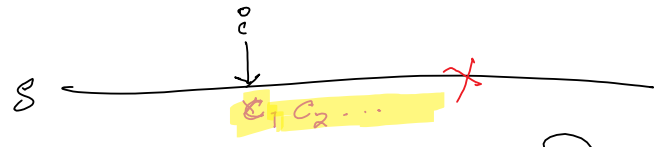
Complexity: worstcase $= O(n^2)$ time

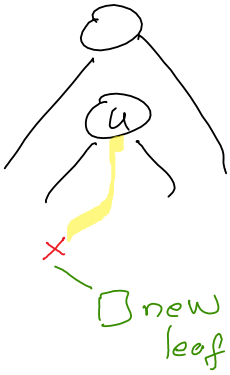
$S = a\ a\ a\ a\ a\ a\ a\ \$$



FindPath function

Monday, March 1, 2021 10:50 AM





FindPath (Tree node * u, <string s, index i>)

//Goal: to insert suffix S[i...] under node u.

{

0) Initialize: $v \leftarrow u$, $x \leftarrow S[i..end]$

1) REPEAT {

a) Find the child branch that starts with x[1]

b) if no such branch exists {

- insert new leaf for s under u
- return.

}

c) Compare characters of the edge label against x[1...] until first mismatch or edge label exhausted

d) if mismatch {

- break edge
- create a new internal node
- create new leaf for s under that node
- return

}

else { $v \leftarrow$ next internal node

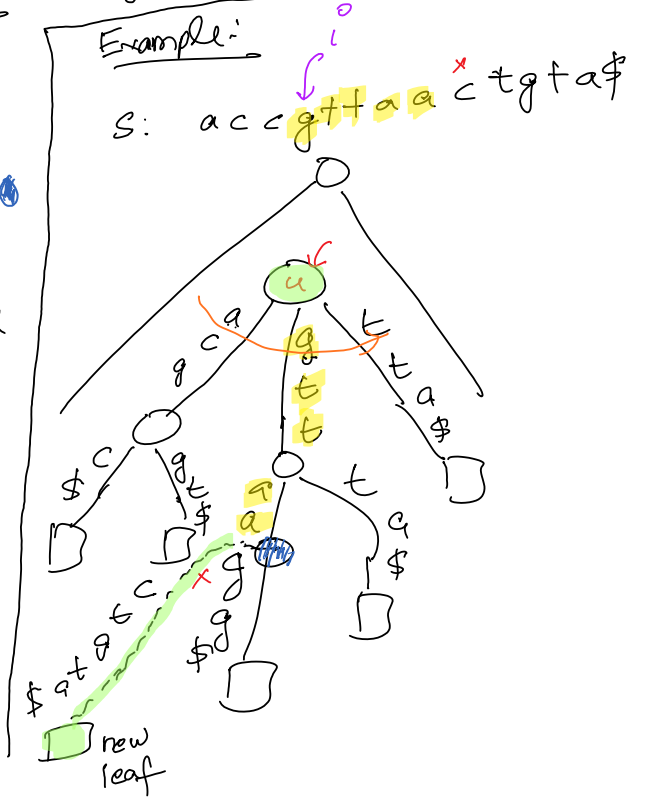
$x \leftarrow x[i+r...]$

}

UNTIL(leaf inserted)

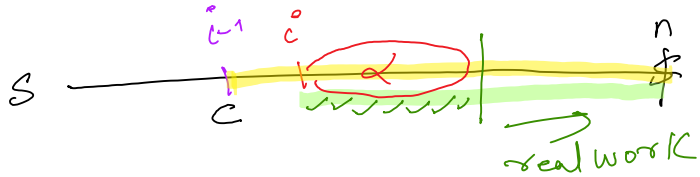
}

Example:



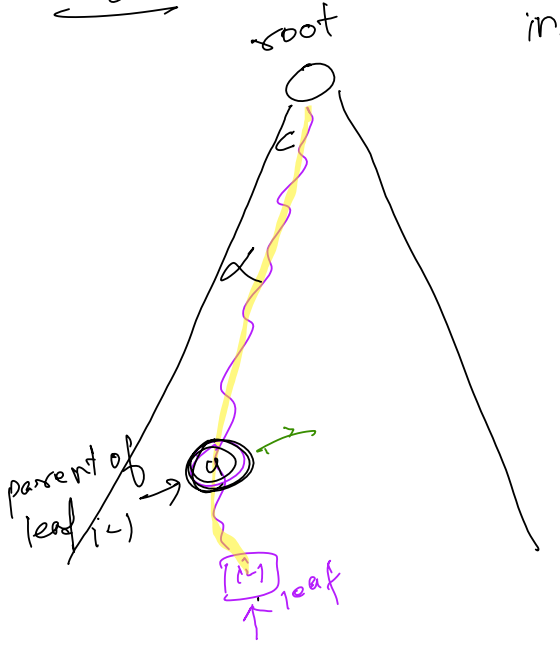
Suffix Links

Friday, March 5, 2021 10:21 AM



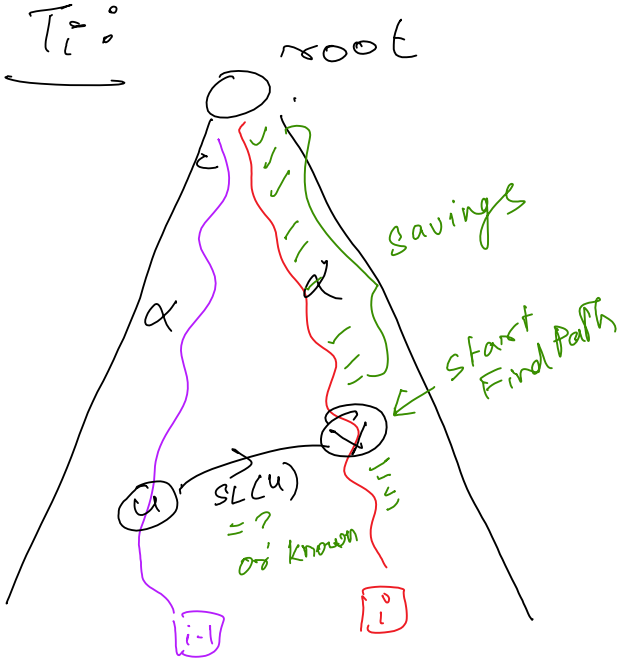
Q] How to improve Insertion of suff$_i$ into T_{i-1} ?

T_{i-1}:



insert suff$_i$

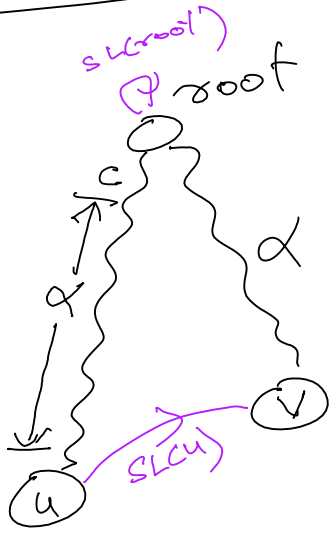
T_i:



Suffix Link: Let u be an internal node with the path label $c\alpha$.

Then, Suffix link of u (SL(u)) is the internal node v with the path label α.

i.e., $SL(u) = v$, for some v, pathlabel(v) $= \alpha$

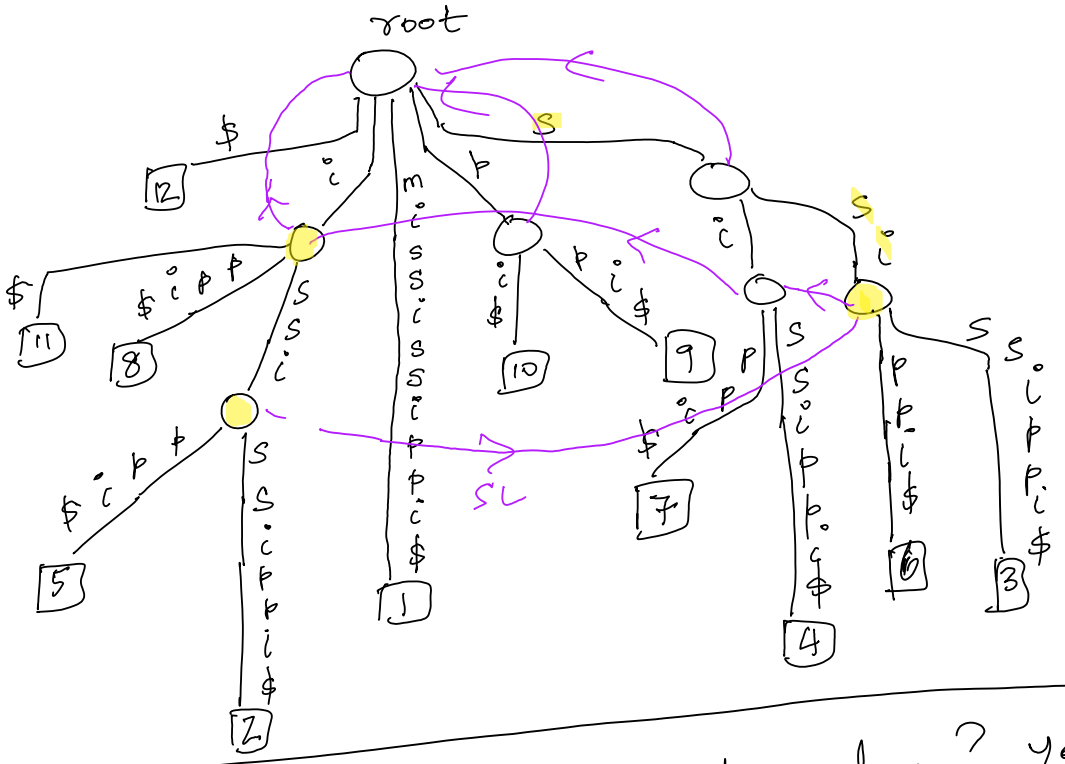


Special case: $SL(root) = root$

Suffix Links: Example

Friday, March 5, 2021 10:21 AM

Example: S = m i s s i s s i p p i $

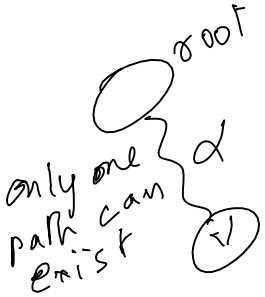


→ Q1) are SLs defined for all internal nodes? yes

Q2) will there be just/exactly one SL(u)?

⇒ could there be > 1 internal node with path label α?

NO because by the defined "trie".



Suffix Links are defined on a suffix tree for all nodes

Friday, March 5, 2021 10:22 AM

Lemma: The suffix link of any internal node u is always defined.

i.e., $SL(u) = v$, where v will always be a valid internal node in $ST(S)$.

Proof:

Since u is an internal node

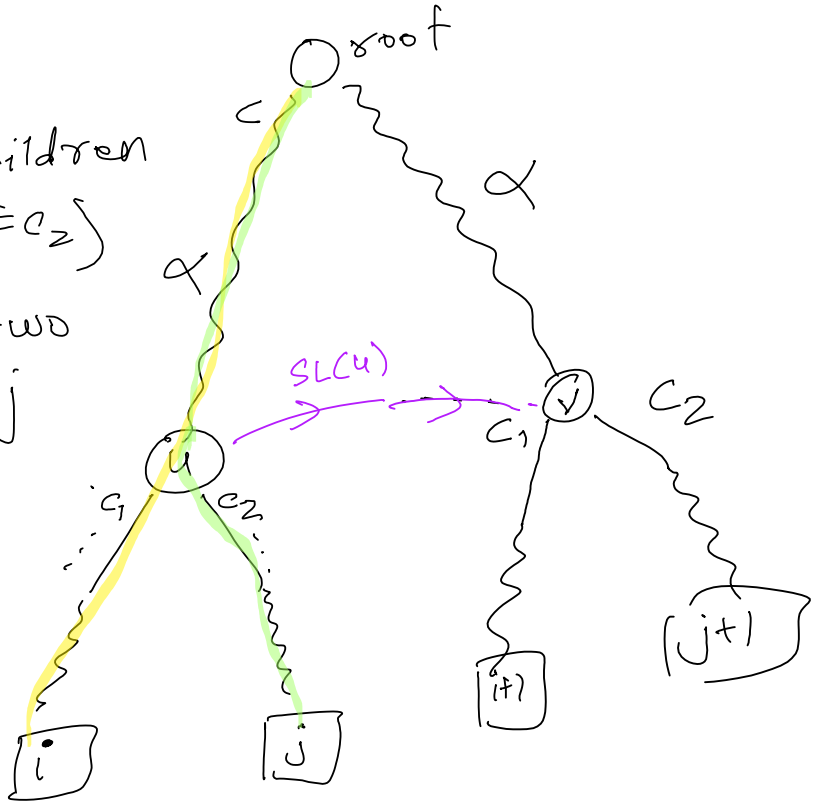
$\Rightarrow$ it has at least two children (with edge labels starting $c_1 \neq c_2$)

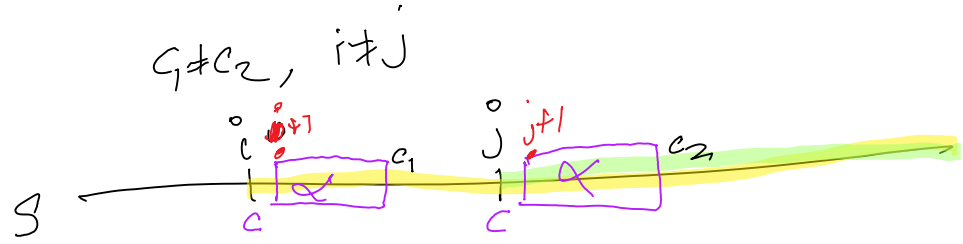
$\Rightarrow$ those two children lead to two different leaves, suffixes i & j $(i \neq j)$

$\Rightarrow lcp(suff_i, suff_j) = c\alpha$

$\Rightarrow lcp(suff_{i+1}, suff_{j+1}) = \alpha$

$\Rightarrow$ there has to exist an internal node with path label α. ☐





(on the timing of the Suffix Link of a node will become available for use)

Suffix Link lemma (during construction)

Monday, March 8, 2021 10:16 AM

- Suffix links provide a shortcut, to insert $suff_i$ into T_{i-1}

Q) However as the tree is being constructed (incrementally), is there a guarantee (of some kind) about the timing by which suffix links will be established for newly created internal nodes? ⇒ Answer: Yes (see below lemma)

Lemma: At the end of iteration $i-1$,

let u be the parent of the leaf corresponding $suff_{i-1}$.

If u was newly created during iteration $\#i-1$, then

$SL(u)$ will be established latest by the end of iteration $\#i$.

Proof: (let $path\ label(u) = c\alpha$)

u was created while inserting $suff_{i-1}$

⇒ $\exists\ suff_j$, where $j < i-1$, already in the tree and under u

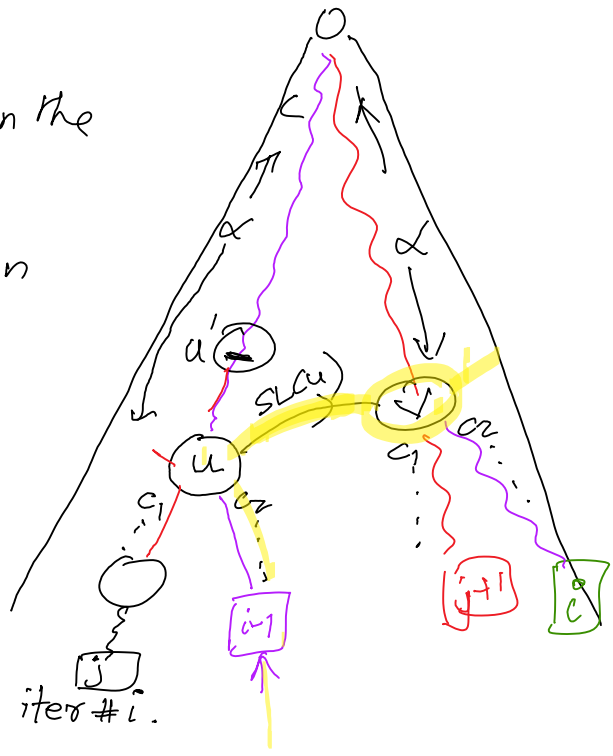
⇒ $suff_{j+1}$ should have also already been inserted in the tree

⇒ while inserting the next suffix, $suff_j$ into T_{i-1}, internal node v with path label α must be created (if it didn't already exist).

⇒ $SL(u) = v$ can be established during iter $\#i$.

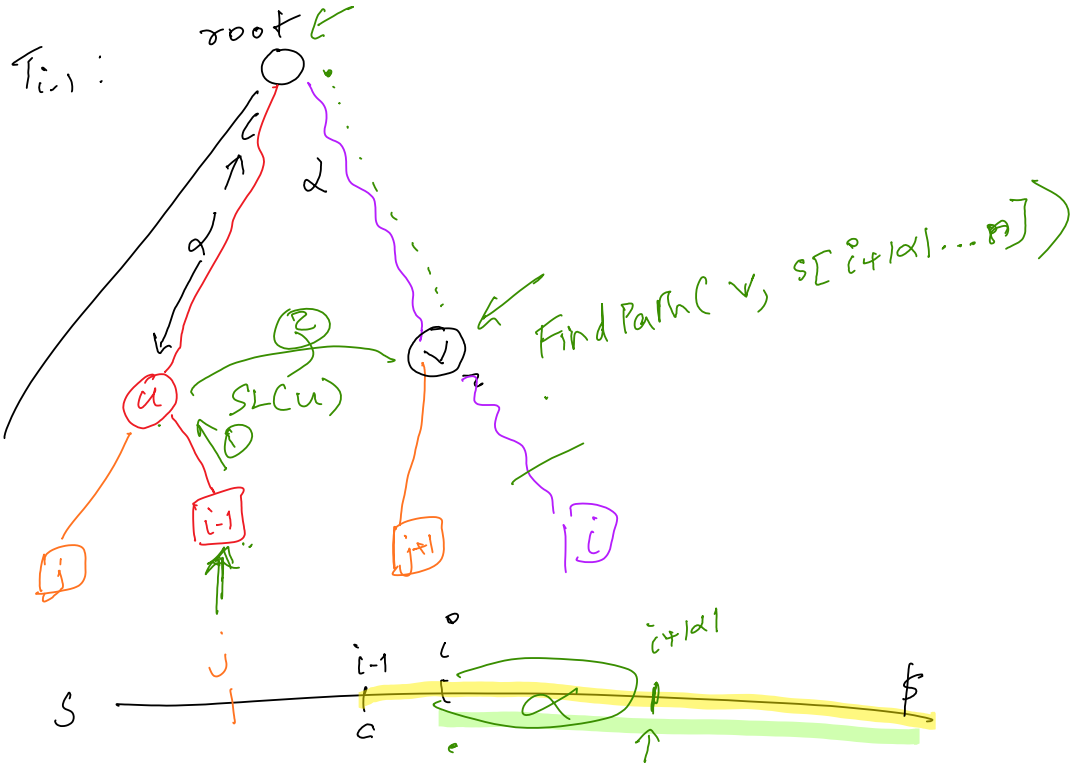
$j < i-1$

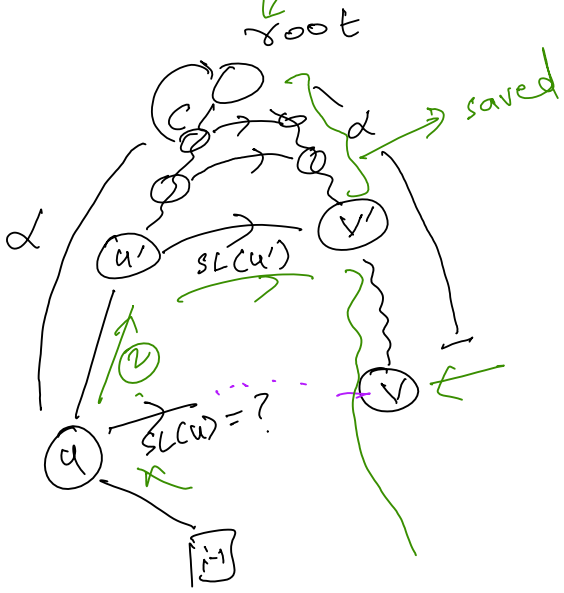
$T_{i-1} \Rightarrow T_i$



Recap

Wednesday, March 10, 2021 10:57 AM





McCreight's algorithm for linear time suffix tree construction (1976)

Monday, March 8, 2021 10:22 AM

Algorithmic Pseudocode:

Build ST (S[1..n], Σ) {

$T_\phi \leftarrow$ root

for (i = 1 to n) { // for each suffix

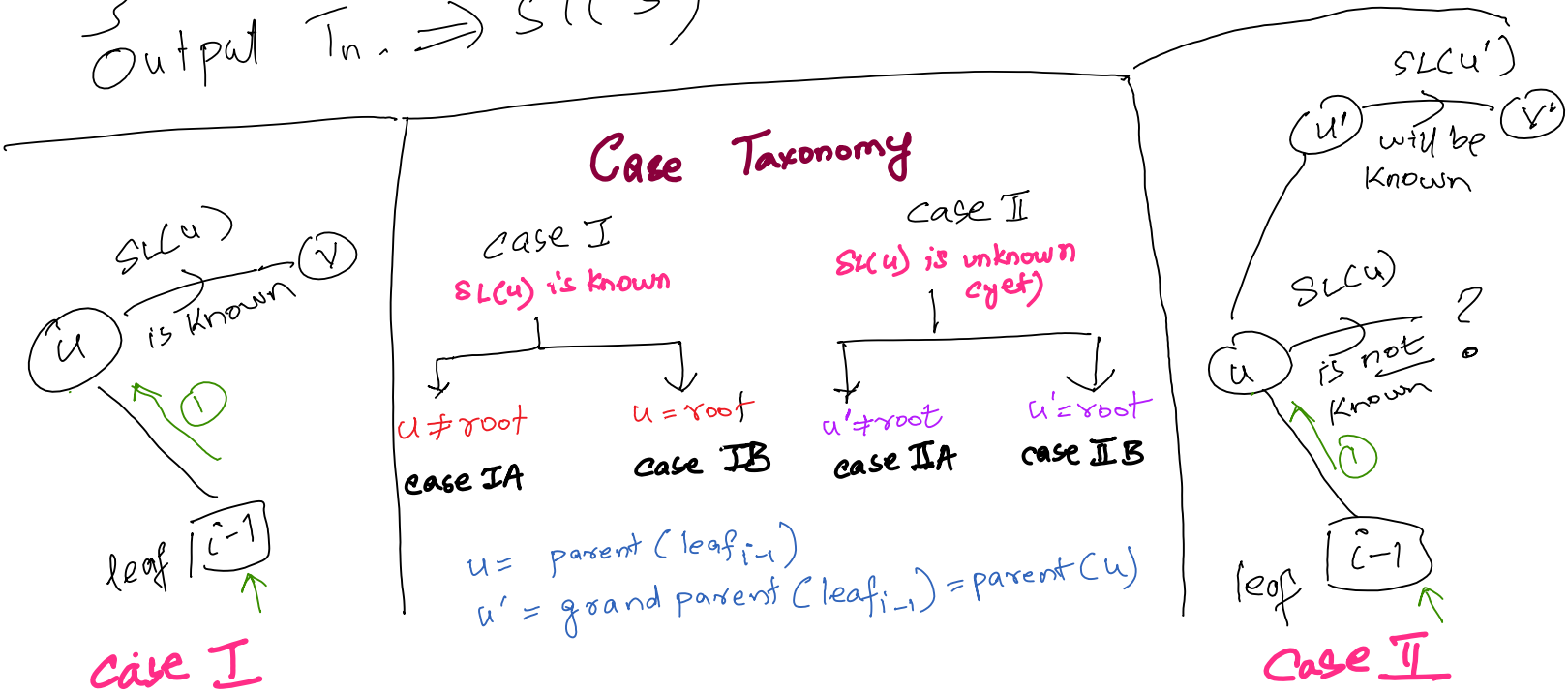
Let u ← parent of leaf$_{i-1}$

Cases:

- u already exists (SL(u) = known)
- u is new (i.e. SL(u) = unknown)

} ⇒ insert suffix$_i$ into T_{i-1} ⇒ T_i

} Output T_n ⇒ ST(S)



McCreight's algorithm: Cases

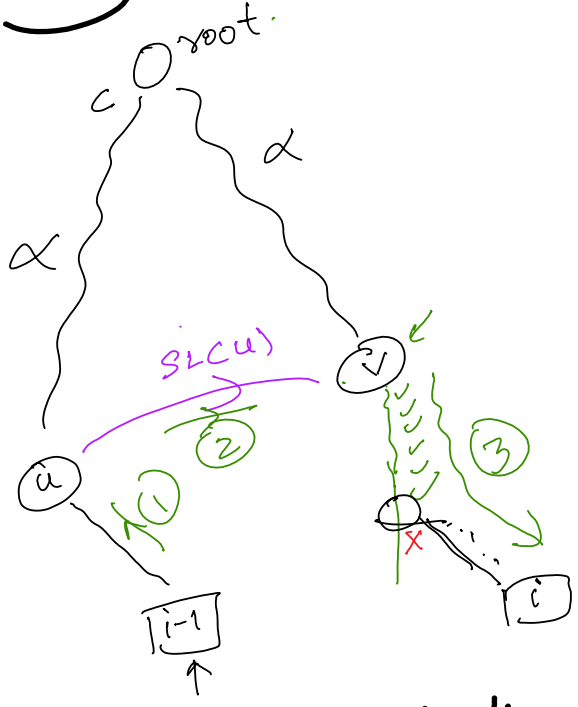
Monday, March 8, 2021 10:23 AM

$|\alpha| = v.\text{stringdepth}$.

Let $u \leftarrow$ parent of leaf $i-1$.

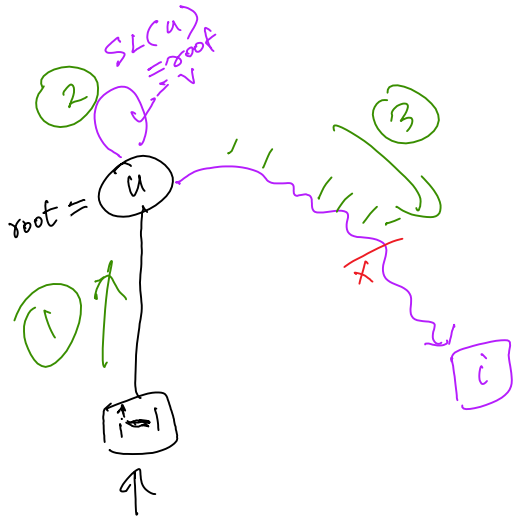
Case IA) SL(u) is known & u is not the root

1. go to parent u
2. Take SL(u) to v
3. FindPath(v, $S[i+|\alpha| \ldots n]$)



Case IB) SL(u) is known & u is the root.

1. Go to parent u (=root)
2. Take SL(root) ⇒ (v = root)
3. FindPath(v = root, $S[i \ldots n]$)

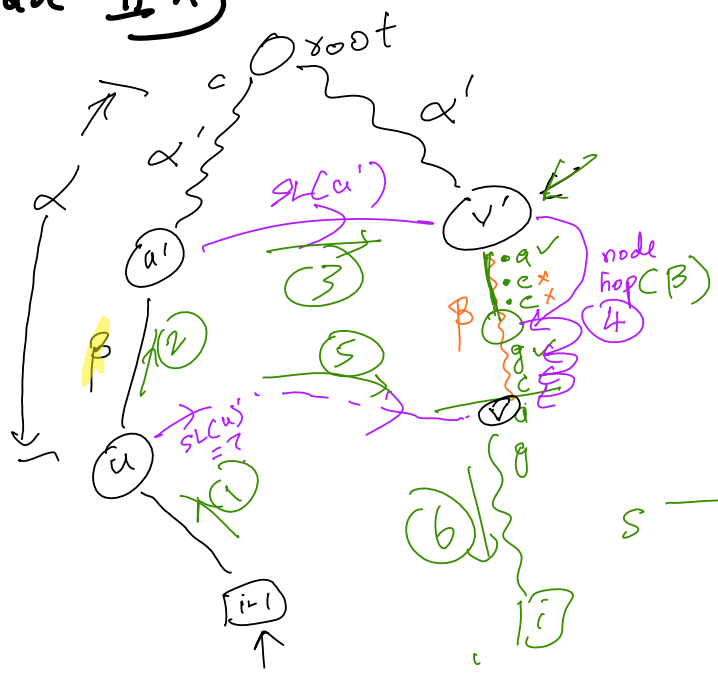


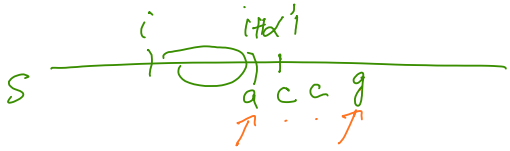
$c\alpha = c\alpha'\beta$

McCreight's algorithm: cases

Monday, March 8, 2021 10:26 AM

Case II A) SL(u) is unknown and u' is not the root.





(1) Go to parent u

(2) Go to grandparent u'
β ← edgelabel(u' to u)

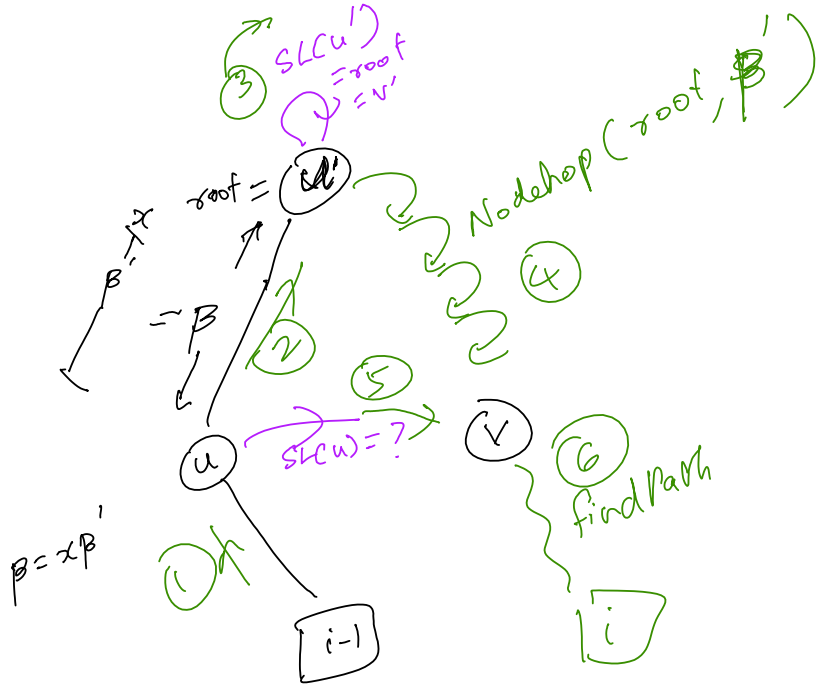
(3) Take SL(u') to v'

(4) NodeHops(v', s[i+|α'|...]) to v
Create v if necess-ary

(5) Establish SL(u) = v

(6) FindPath(v, s[i+|α|...n])

Case II B) SL(u) is unknown and u' is the root.



(1) go to parent u

(2) go to grandparent u'
- let edgelabel(u' to u) be β = xβ'

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

(4) NodeHops(root, β') to v
- create v if necessary

(5) Establish SL(u) = v

(6) FindPath(v, s[i+|β'|...n])

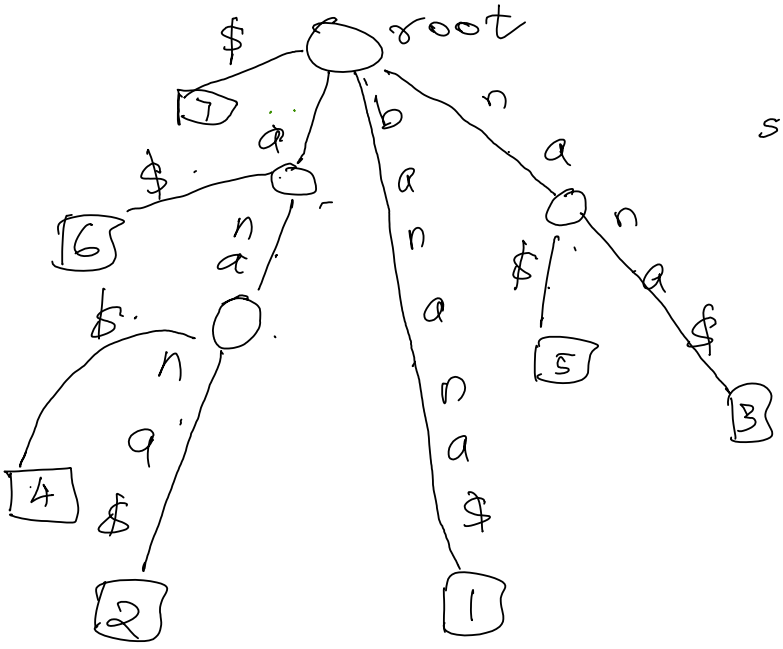
Project 2: Testing your ST construction code

Monday, March 8, 2021 2:18 PM

Use small examples:

$s = banana\$$

$\$ < a < c < g < t$



1	2	3	4	5	6	7
b	a	n	a	n	a	$

(1) Basic Statistics

#nodes

- # internal nodes (4)
- # leaves (7)

#edges = 10

some other examples to try:

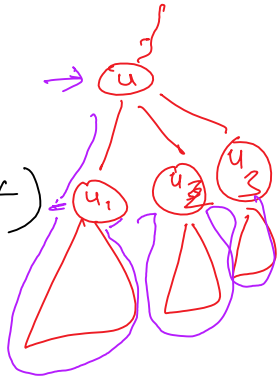
missisippi$

apple maple apply $

anantharaman kalyanaraman$

(2) DFS disp (node *T) **Display** function

```
{
    // print contents of curr.node (T)
    For every child of currnode (from left to right)
        DFSdisp(child)
}
```



Project 2: Testing your ST construction code

Monday, March 8, 2021 2:18 PM

③ Build ST Naive (s[1..n]) // i.e, implement a no-suffix link version for testing

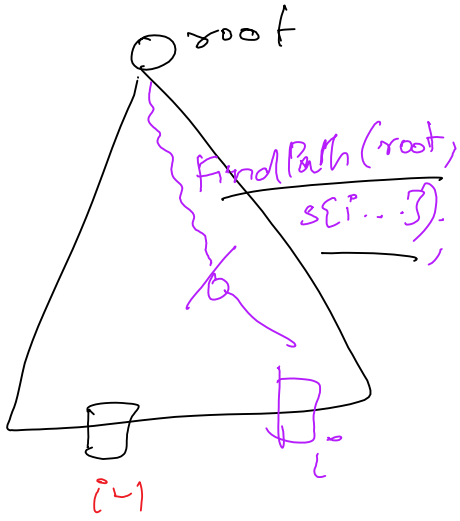
$T_0 \leftarrow$ root

For (i = 1 to n)

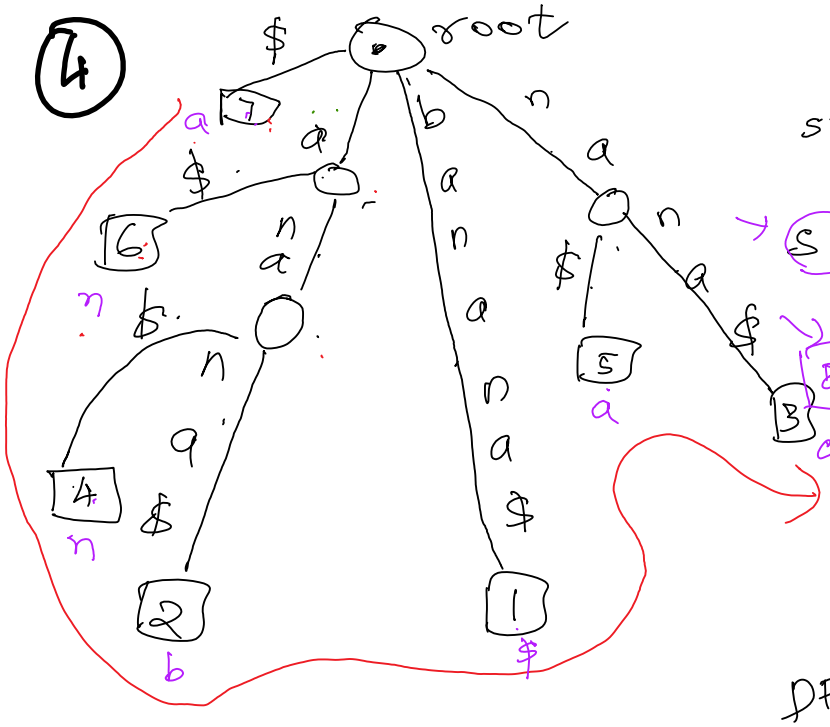
Naive $T_i \leftarrow$ "Insert" $suff_i$ into T_{i-1}

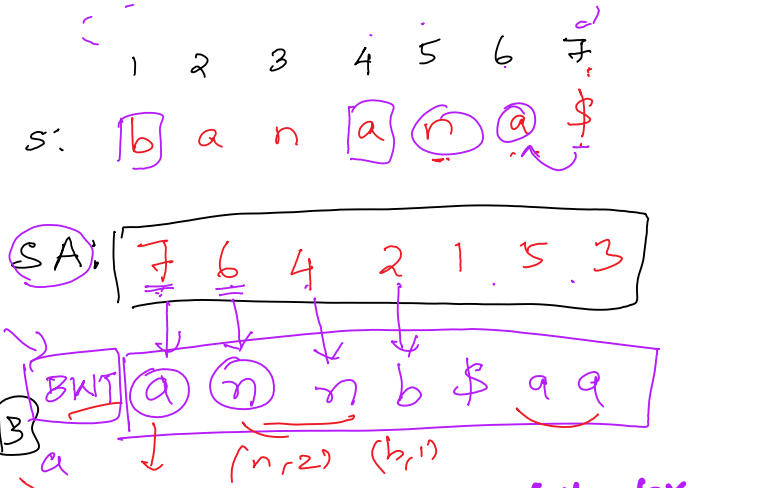
↓

FindPath (root, s[i..$])



④





BWT ← Burrows Wheeler Transform

DFS_BWT

a n n b $ a

If BWT matches, then code most likely right.