





On-Line Construction of Suffix Trees¹

Zichuan Wang (Jack)

Nankai University, China

¹Ukkonen, E. (1995). On-Line Construction of Suffix Trees. *Algorithmica*, 14(3), 249-260.

Content

- Introduction 
- More Details on Constructing Suffix Trees 
- Algorithms 
- Execution on an Example 

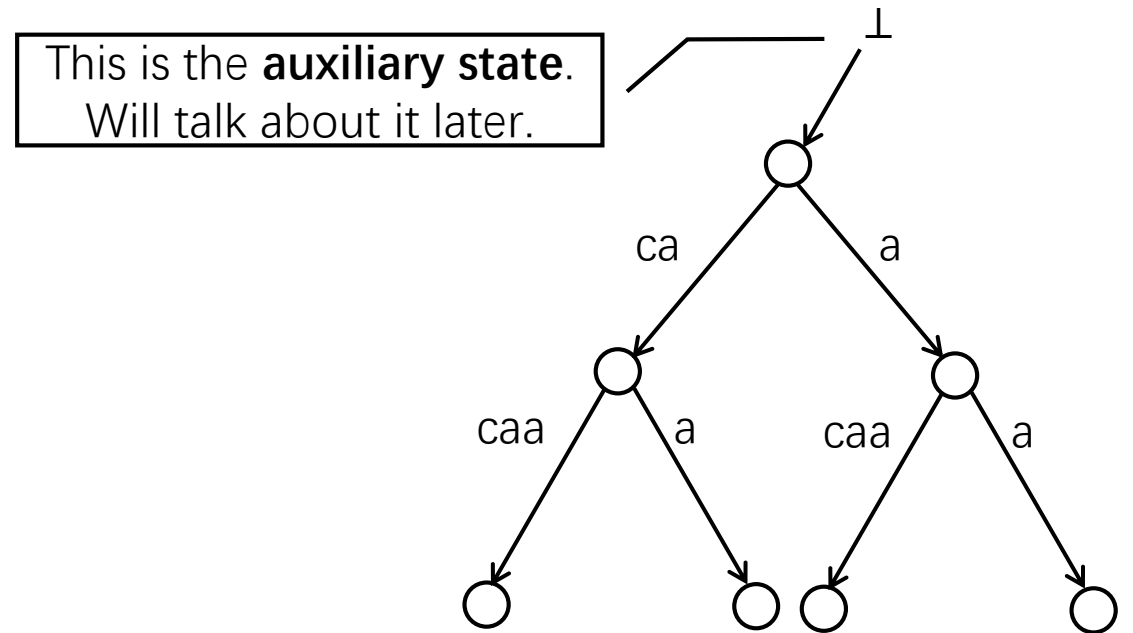
Introduction to Suffix Trees

Given a string T , The suffix tree of T is a tree structure representing $STrie(T)$ in **space linear** in the length $|T|$.

Example:

$T = caca$

suffixes = {caca, aca, ca, a, ϵ }







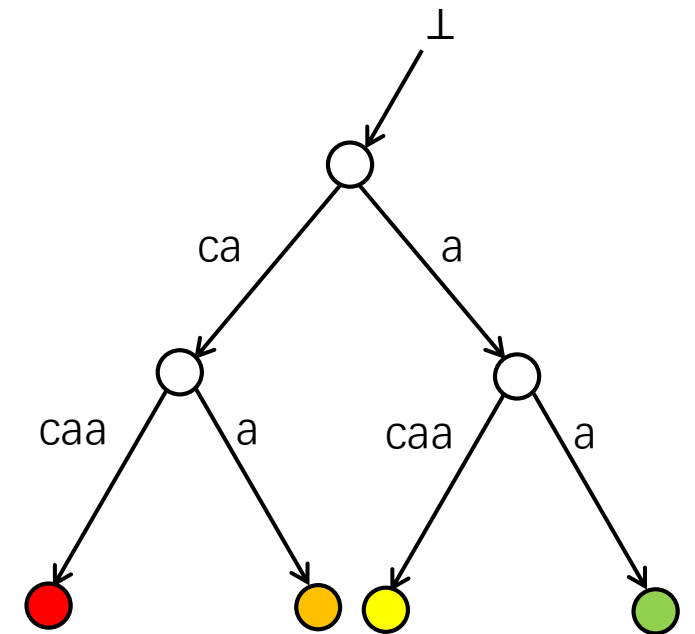
The suffix tree of string *caca*

Example:

$T = \text{cacaa}$

suffixes = {cacaa, acaa, caa, aa, a, ϵ }

Leaf Node	Corresponding Suffix
	cacaa
	caa
	acaa
	aa



The suffix tree of string *cacaa*

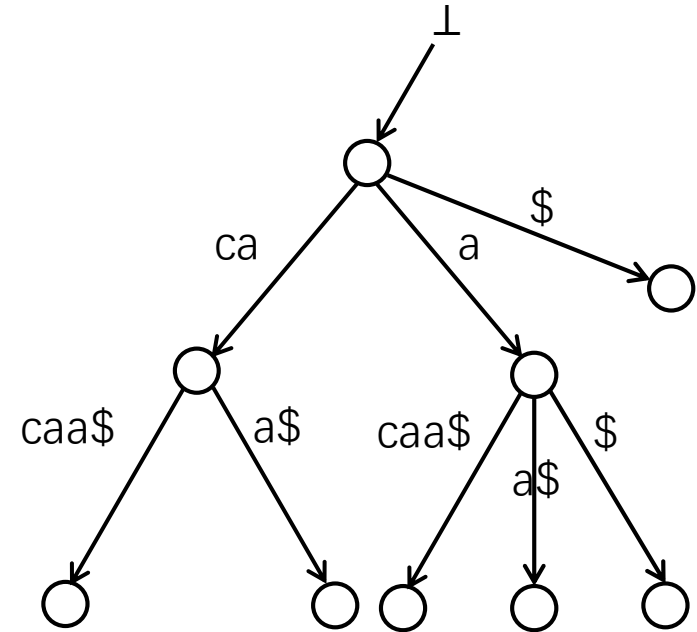
Example

$T = \text{cacaa}\$$

$\text{suffixes} = \{\text{cacaa}\$, \text{acaa}\$, \text{caa}\$, \text{aa}\$, \text{a}\$, \$, \epsilon\}$

$\$$ is a terminal symbol ($\$ \notin \Sigma = \{a, c\}$).

In such suffix trees, the **leaves** are in one-to-one correspondence with the **suffixes** of cacaa .









The suffix tree of string $\text{cacaa}\$$

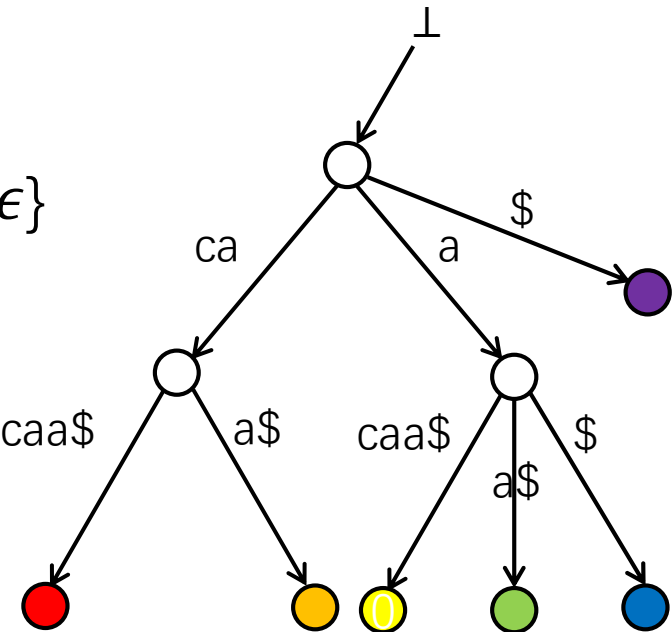
Example

$T = \text{cacaa\$}$

suffixes of $\text{cacaa\$}$ = { $\text{cacaa\$}$, $\text{acaa\$}$, $\text{caa\$}$, $\text{aa\$}$, $\text{a\$}$, $\text{\$}$, ϵ }

suffixes of cacaa = { cacaa , acaa , caa , aa , a , ϵ }

Leaf Node	Corresponding Suffix of <i>cacaa</i>
	<i>cacaa</i>
	<i>caa</i>
	<i>acaa</i>
	<i>aa</i>
	<i>a</i>
	ϵ



The suffix tree of string *cacaa\$*

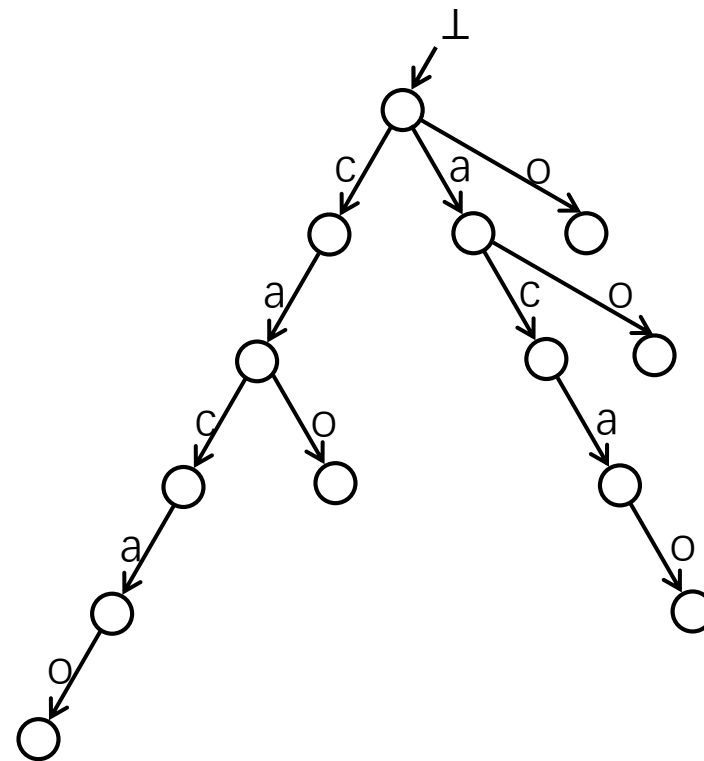
The algorithm that constructs suffix trees in linear time is based on the algorithms that constructs suffix tries in quadratic time. Before we dive into suffix trees, let's first take a look at suffix tries.

Introduction to Suffix Tries

Given a string T , The suffix trie of T is a trie representing all **suffixes** of T .
Different from suffix trees, each edge of a trie only represents **one letter**.

Example:

$T = \text{cacao}$

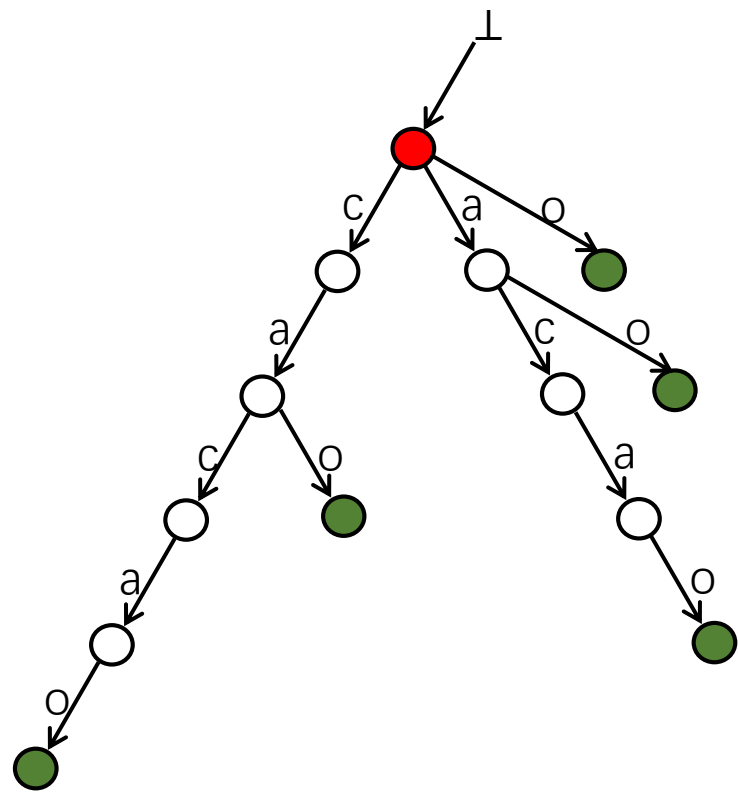
$$\text{suffixes} = \{\text{cacao}, \text{acao}, \text{cao}, \text{ao}, \text{o}, \epsilon\}$$


The suffix trie of string *cacao*

Formal Definition

Definitions Related to Suffixes		
Symbol	Definition	Example
Σ	alphabet Σ is a set of letters	{a, c, o}
T	$T = t_1 t_2 \cdots t_n$ is a string over an alphabet Σ	cacao
x	each x s.t. $T = uxv$ is a substring of T (u and v are possibly empty)	aca
T_i	each $T_i = t_i \cdots t_n$ ($1 \leq i \leq n$) is a suffix of T	$T_4 = \text{ao}$
ϵ	empty string (also denoted as T_{n+1})	
suffix	each string T_i ($1 \leq i \leq n + 1$) is a suffix of T	ϵ , cacao, o, <i>etc.</i>
$\sigma(T)$	the set of all suffixes of T	{cacao, acao, cao, ao, o, ϵ }

Definitions Related to Suffix Tries		
Symbol	Definition	Example
$STrie(T)$	$STrie(T) = (\Sigma, Q \cup \{\perp\}, root, F, g, f)$ is the suffix trie of T (Note: $STrie(T)$ is augmented with \perp and f)	
Σ	an alphabet	{a, c, o}
x	a substring of T	aca
\bar{x}	the state that corresponds to x	each circle
Q	the set of states (Note: there exists a one-to-one correspondence between Q and all substrings)	
\perp	auxiliary state	
$root$	$root$ corresponds to the empty string ϵ	the red state
F	the set of accepting states , namely the set of suffixes of T (denoted by $\sigma(T)$)	all green and red states
g	$g: Q \cup \{\perp\} * \Sigma \rightarrow Q$ is the transition function	$g(\overline{aca}, o) = \overline{aca}o$
f	$f: Q \rightarrow Q \cup \{\perp\}$ is the suffix function	$f(\overline{aca}o) = \overline{ca}o$



The suffix trie of *cacao*

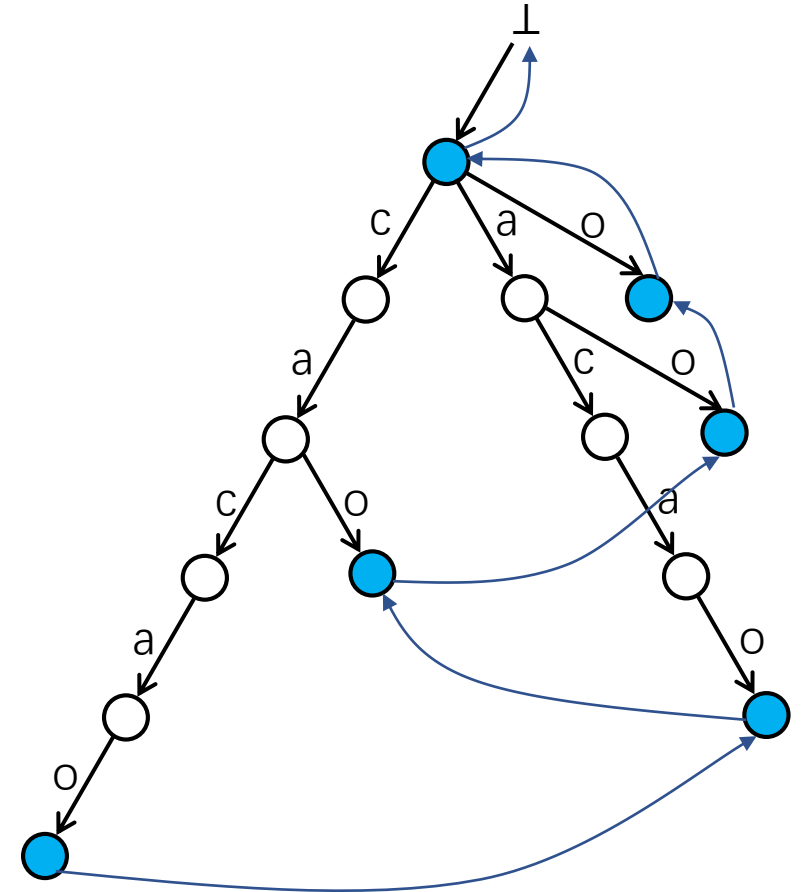
Why are f and the auxiliary state \perp important?

- We can find all states corresponding to $\sigma(T)$ by

$$\bar{T}_i = f^{i-1}(\bar{T}) \quad (1 \leq i \leq n+1). \quad (1)$$

Then, we can find a path that goes through all suffixes and ends at the auxiliary state \perp . We call it the **boundary path**.

- Without the auxiliary state \perp , f is not well-defined on the set of states Q . (Since $root \in Q$, we need to define $f(root)$.) Also, we need $g(\perp, l) = root$ later.



The boundary path of $STrie(cacao)$

On-Line Construction: An Observation

- We have defined the suffix $t_i \cdots t_n$ of a string $T = t_1 t_2 \cdots t_n$ by T_i . Now we define the **prefix** $t_1 \cdots t_i$ by T^i ($0 \leq i \leq n$).
- To construct $STrie(T^i)$ based on $STrie(T^{i-1})$, we only need to create states that correspond to **suffixes of T^i** .
- $\sigma(T^i) = \sigma(T^{i-1})t_i \cup \{\epsilon\}$. This equation means to get suffixes of T^i , we simply concatenate suffixes of T^{i-1} and t_i , then add ϵ .

suffixes of ac	suffixes of aca
ϵ	a
c	ca
ac	aca
	ϵ

Execution on an Example

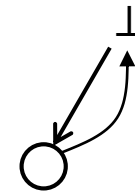
$T = cacao$

Initialization:

$T^0 = \epsilon$

g	
input	output
(\perp, l)	$root$

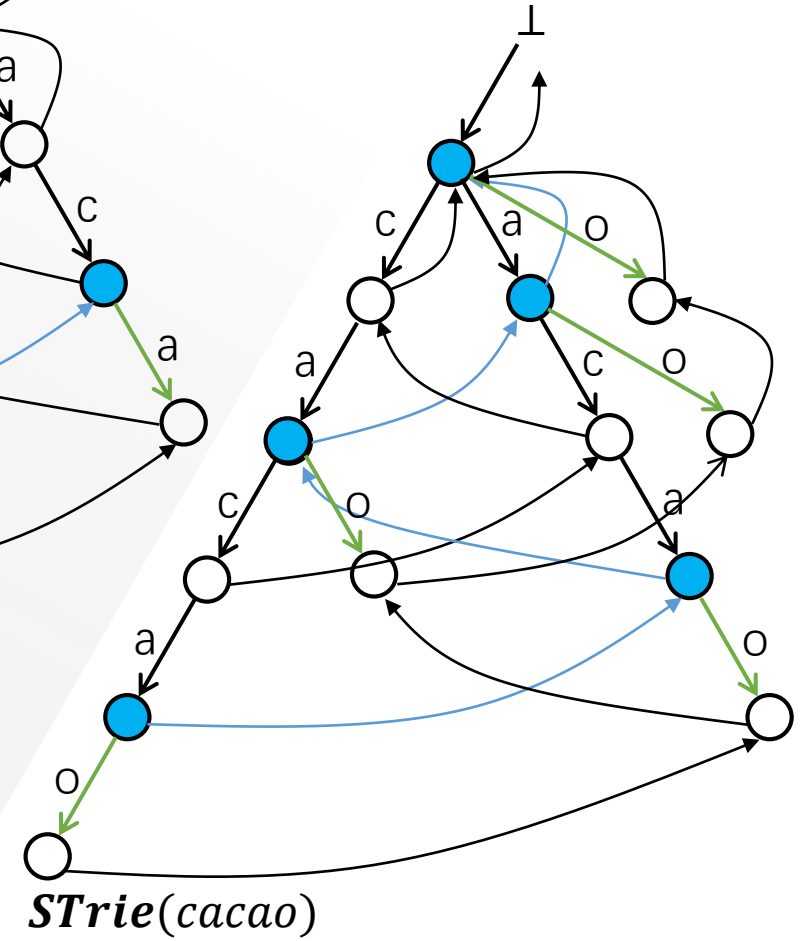
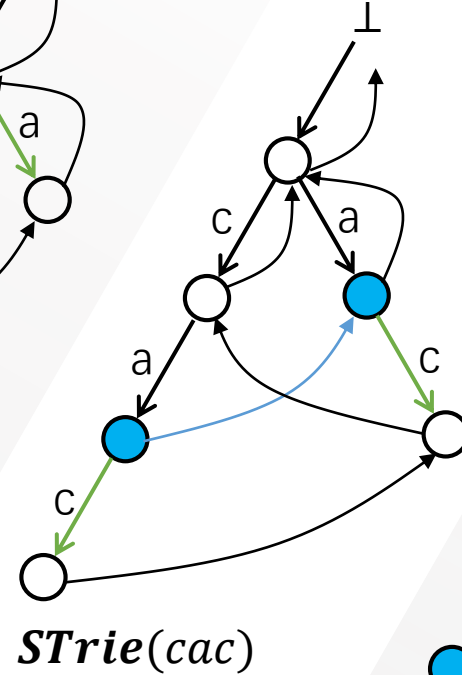
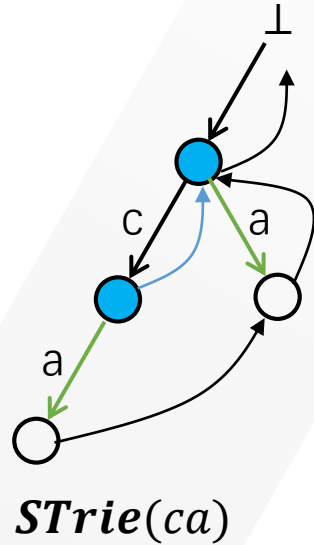
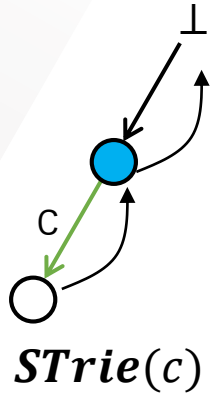
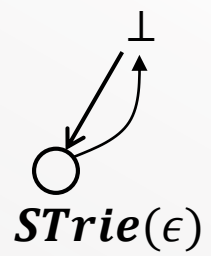
f	
input	output
$root$	\perp



In each iteration,

follow the **boundary path** of $\mathbf{STrie}(T^{i-1})$, and

create t_i -**transitions** that do not exist in $\mathbf{STrie}(T^{i-1})$.



In each iteration,
follow the **boundary path** of $STrie(T^{i-1})$, and
create **t_i -transitions** that do not exist in $STrie(T^{i-1})$.

symbol	explanation
	suffix function
	transition function
blue path	boundary path of $STrie(T^{i-1})$
green transition	new t_i -transition

Time Complexity Analysis

Theorem 1

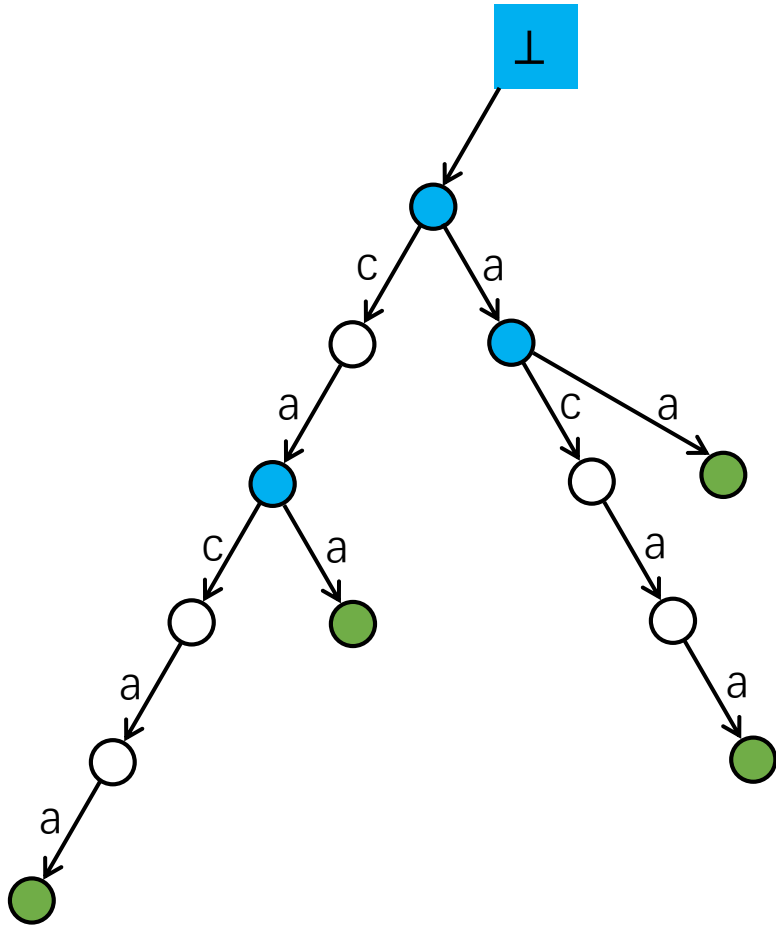
Suffix Trie $STrie(T)$ can be constructed in time proportional to $|Q|$, where Q is the set of states. $|Q|$, in the worst case, is $O(|T|^2)$.

Sketch of Proof

Each state corresponds to a distinct substring of T . The number of different substrings of T is at most $O(|T|^2)$. E.g., $T = a^n b^n$. Therefore, this algorithm has quadratic time complexity.

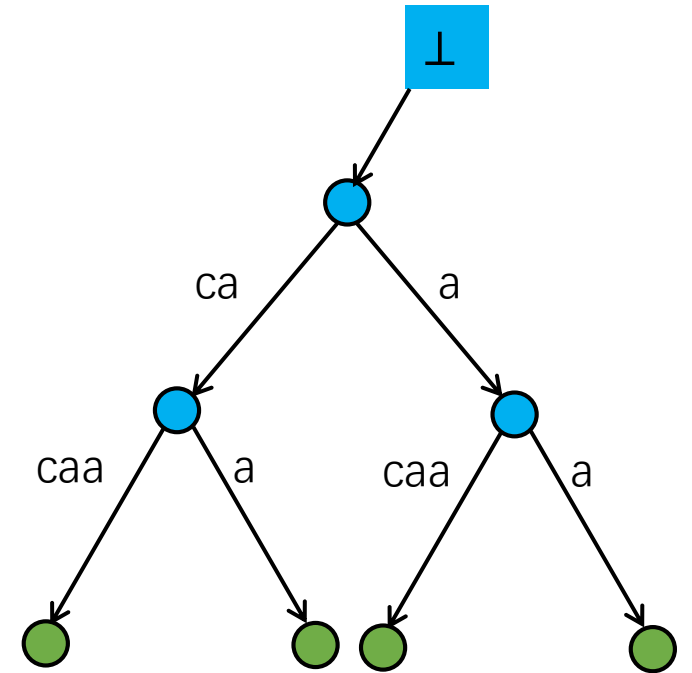
■ Suffix Trees Construction: An Overview

- Similar to the construction of suffix tries, to construct suffix trees, we create states that correspond to suffixes of T^i .
- The reason why the construction of suffix trees runs in linear time is that we discard states that only have one transition and only keep **branching states and leaves**. In this way, we compress suffix tries and there are $O(|T|)$ states in a suffix tree.



the suffix trie of string *cacao*
totally 13 states

- branching states
- leaves
- implicit states



the suffix tree of string *cacao*
totally 8 states

Execution on an Example

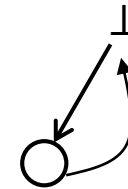
$T = caciaa$

Initialization:

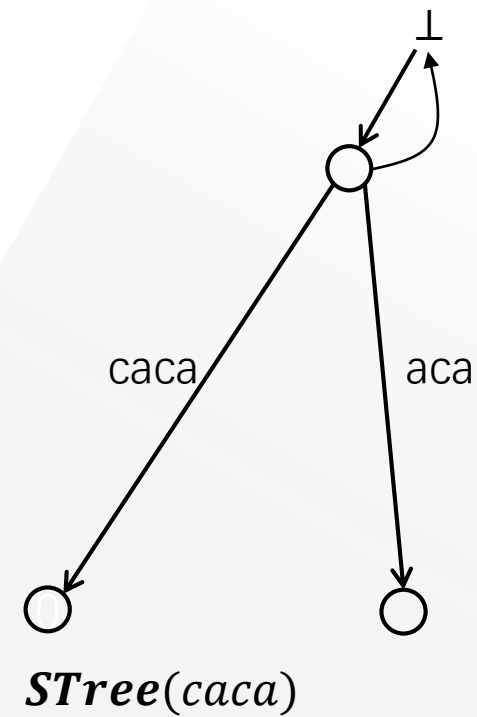
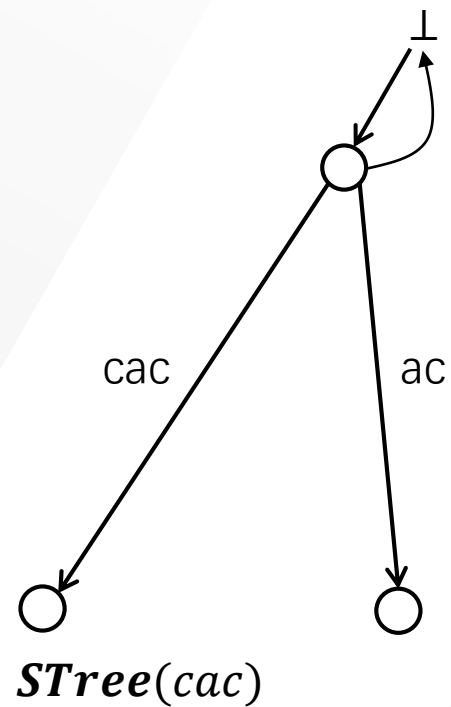
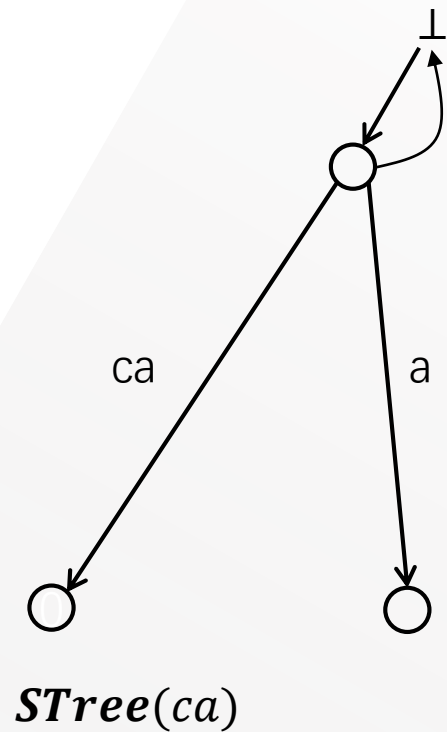
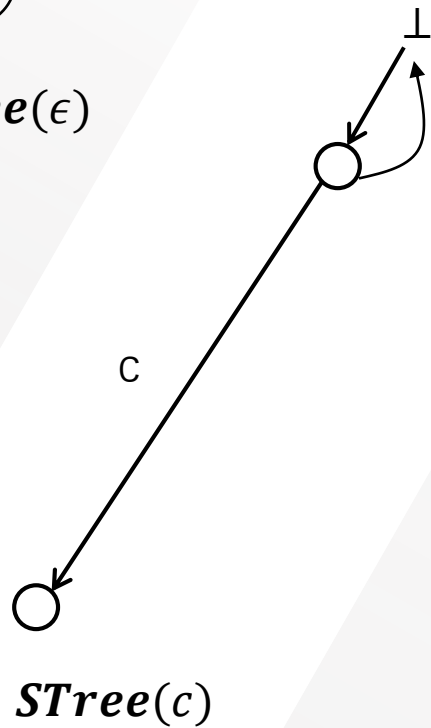
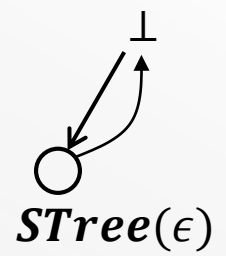
$T^0 = \epsilon$

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>

f'	
input	output
<i>root</i>	\perp



In each iteration, create a new internal state only when it's a **branching state** (states with more than two transitions).

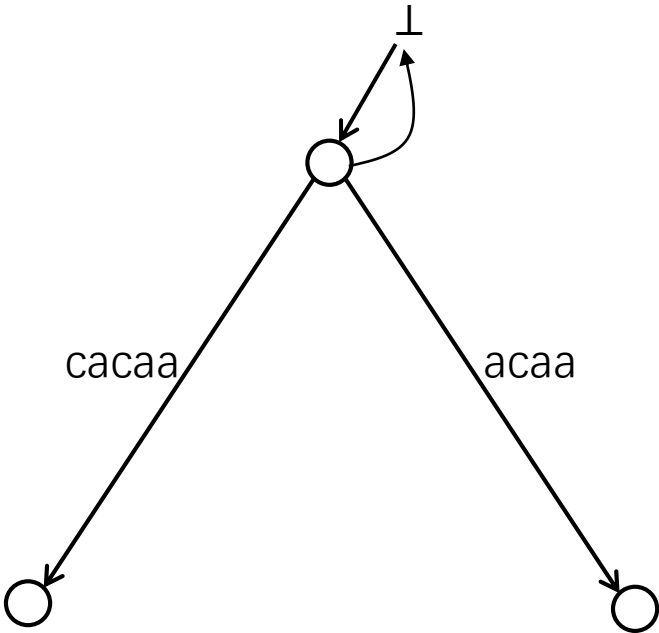


Until ***STree(caca)***, there is no branching state besides *root*.

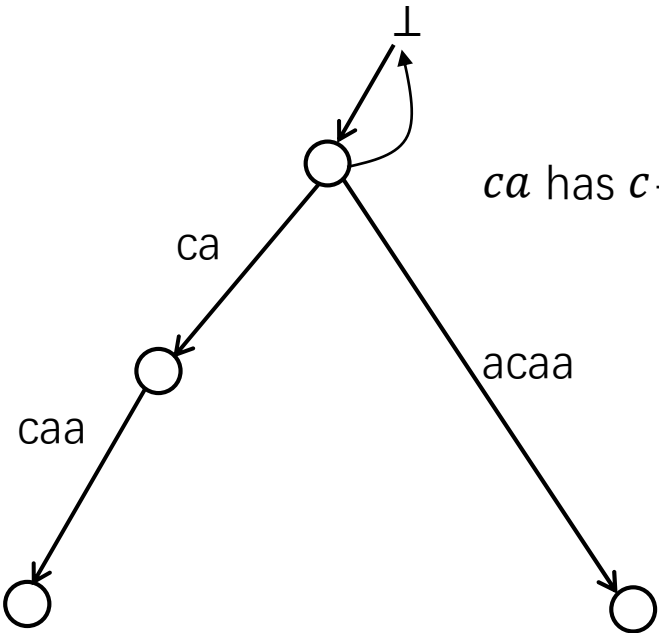
***STree*(T^4) \rightarrow *STree*(T^5)**

$T^4 = caca$

$T^5 = cacaa$

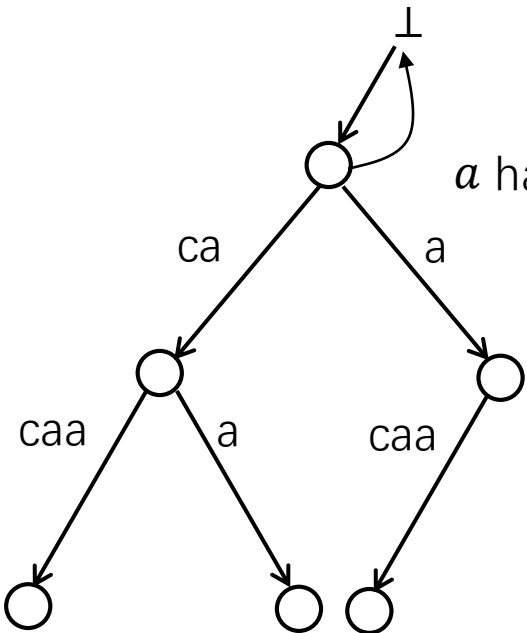
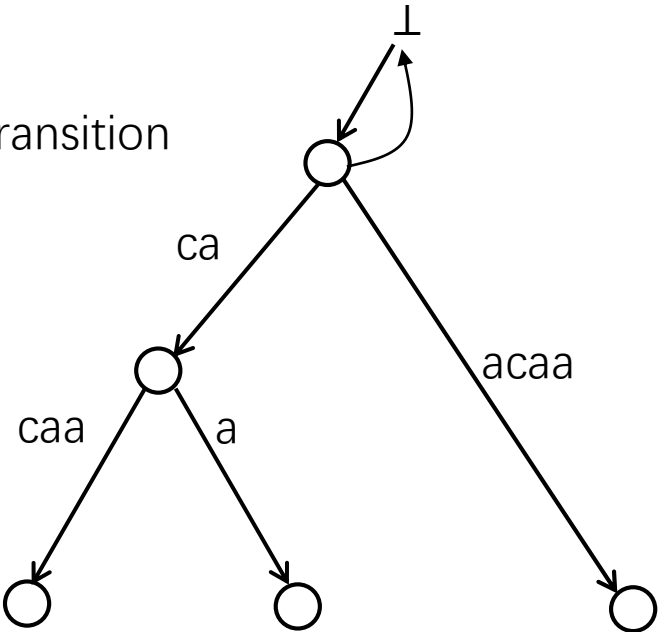


1

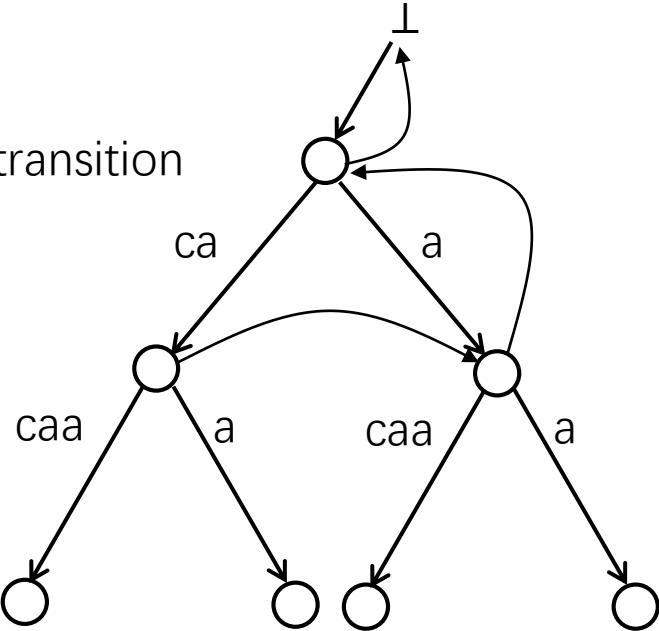


ca has *c*- and *a*-transition
cacaa
cacaa

2	3
4	5



a has *c*- and *a*-transition
cacaa
cacaa



Time Complexity Analysis

Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Sketch of Proof

The internal nodes of $STree(T)$ have more than one children. In such trees, the number of nodes is linear to the number of leaves, which is $O(n)$ because each leaf corresponds to a suffix of T .

Therefore, $STree(T)$ can be constructed on-line in time $O(n)$.

Q.E.D.

More Details on Constructing Suffix Trees

In this section, we will provide more details on how to construct suffix trees in linear time.

Formal Definition

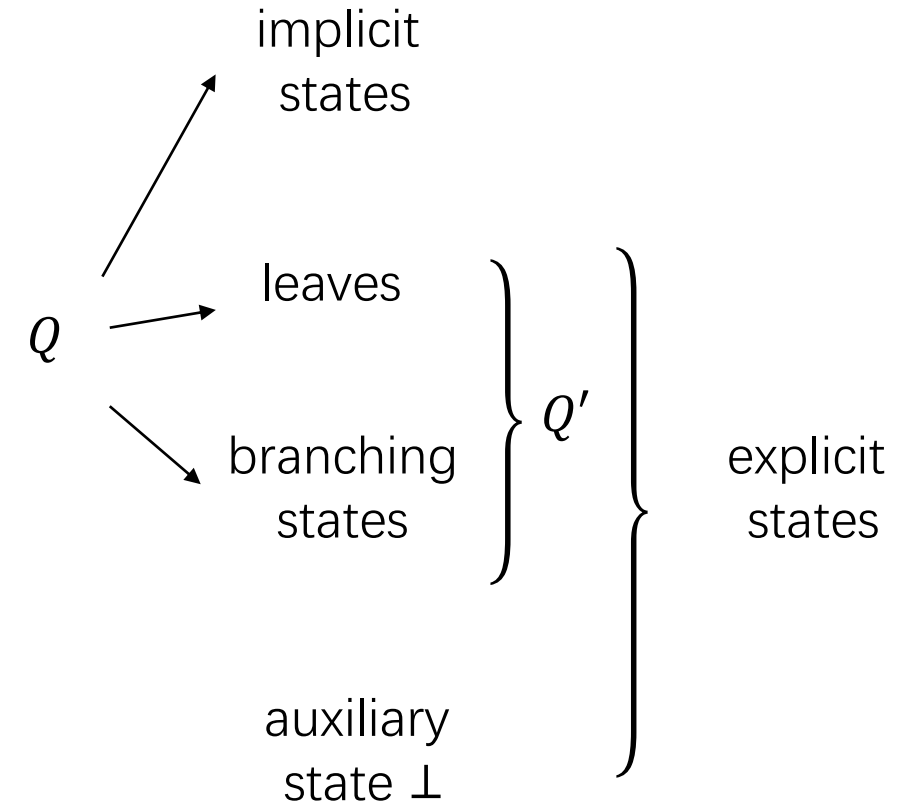
Definitions Related to Suffix Trees	
Symbol	Definition
T	$T = t_1 t_2 \cdots t_n$ is a string over an alphabet Σ
$STrie(T)$	$STrie(T) = (\Sigma, Q \cup \{\perp\}, root, F, g, f)$ is the suffix trie of T
$STree(T)$	$STree(T) = (\Sigma, Q' \cup \{\perp\}, root, g', f')$ is the suffix tree of T

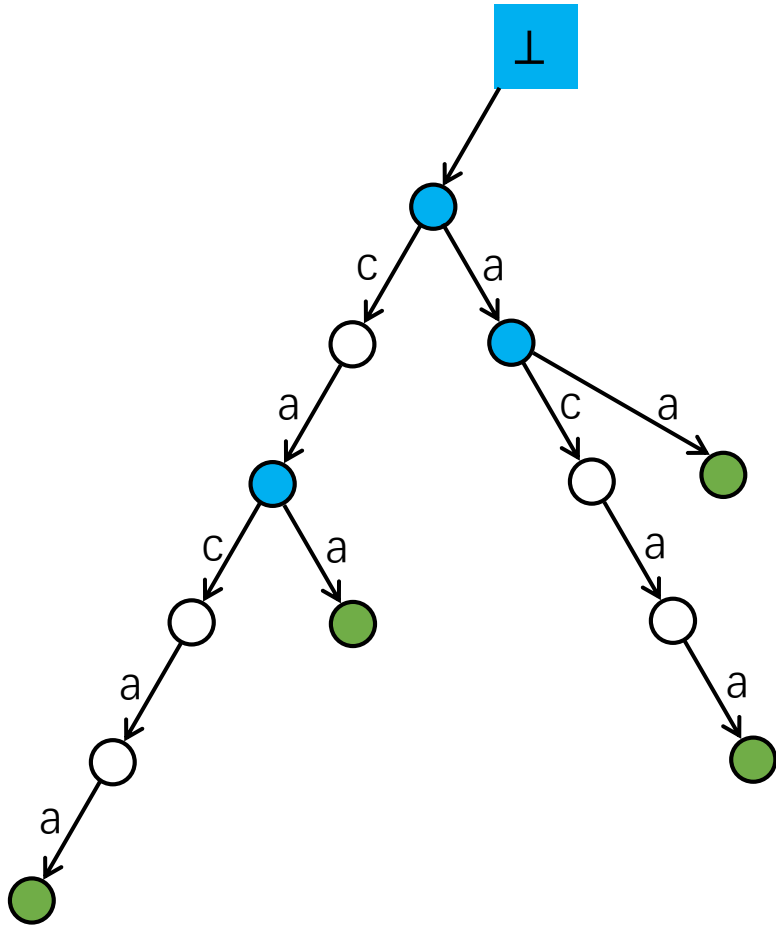
$STree(T)$ differs from $STrie(T)$ mainly in 4 aspects:

1. the set of explicit states $Q' \cup \{\perp\}$,
2. the generalized transition g' ,
3. the modified suffix link f' , and
4. the omitted accepting states.

Explicit and Implicit States

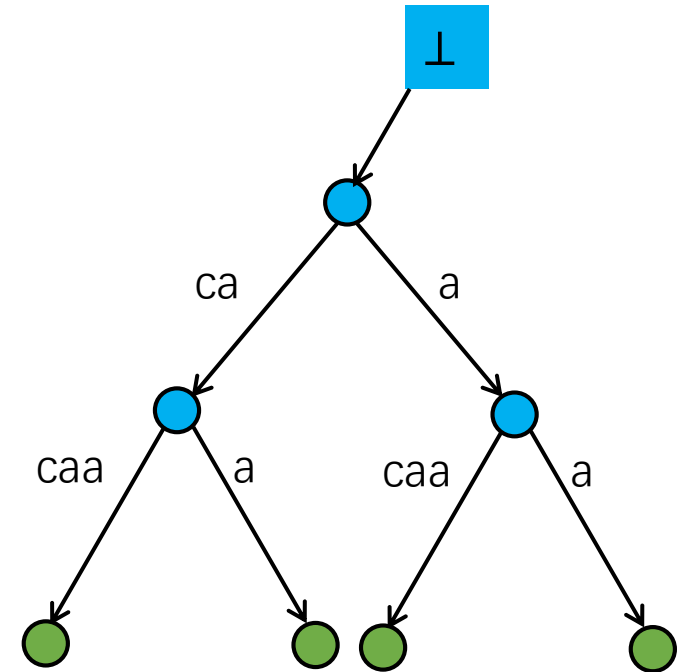
Definitions Related to States	
Concept	Definition
branching states	states from which there are at least two transitions
leaves	states from which there are no transitions
explicit states	all branching states and leaves
implicit states	states from which there are one transition (states other than explicit states)





The suffix trie of string *cacao*

- branching states
- leaves
- implicit states



The suffix tree of string *cacao*

Generalized Transitions

Recall that $g: Q \cup \{\perp\} * \Sigma \rightarrow Q$ is the transition function of $STrie(T)$. For example, $g(\overline{aca}, a) = \overline{acaa}$.

Because the edge two adjacent states in $STree(T)$ can represent a string instead of a letter, we need a generalized version of transitions.

Here, g' is defined as

$$g': Q' \cup \{\perp\} * \Sigma^* \rightarrow Q', \quad (1)$$

where Σ^* is the set of strings on alphabet Σ . For example, $aab \in \{a, b\}^*$.

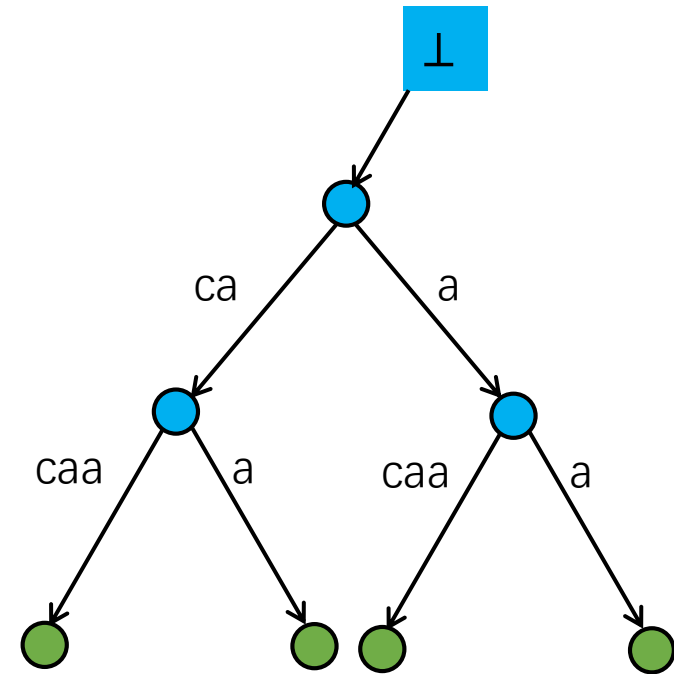
Examples

$$g'(\overline{ca}, caa) = \overline{cacaa}$$

$$g'(root, ca) = \overline{ca}$$

$$g'(\perp, c) = root$$

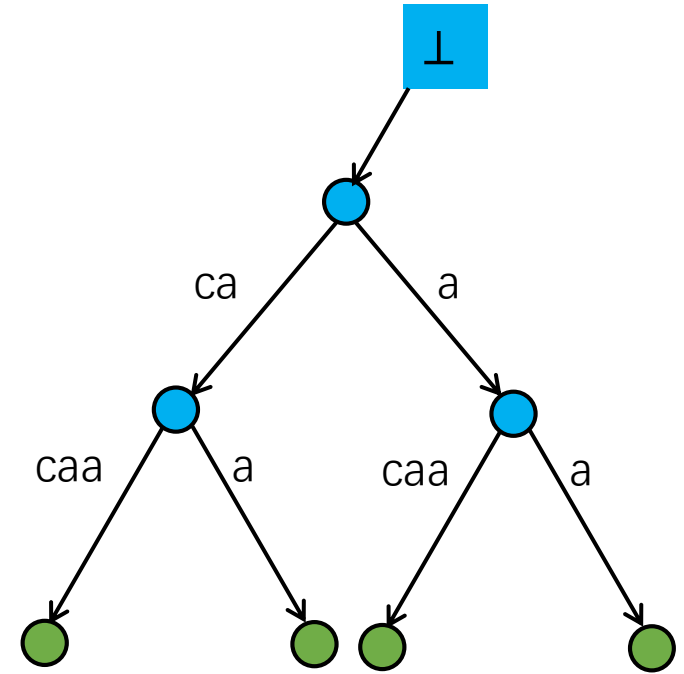
- To save memory, we use a pair of **pointers** (k, p) to store string $t_k t_{k+1} \cdots t_p$. k is the **left pointer** and p is the **right pointer**. For example, $g'(\overline{ca}, caa) = g'(\overline{ca}, (3, 5)) = \overline{cacaa}$.
- For \perp and $g'(\perp, l)$ ($\forall l = t_i \in \Sigma$), we use **negative indexes** to represent letter l . For example, $g'(\perp, l) = g'(\perp, (-i, -i)) = root$.



The suffix tree of string *cacaa*

Note:

- $g'(s, (k, p)) = r$ is called a **t_k -transition**.
- $g'(s, w) = r$ indicates that s and r are **adjacent** states.
- $g'(s, \epsilon) = g'(s, (p + 1, p))$.
- If more than one pair of pointers can represent a string, (e.g. $ca = (1,2) = (3,4)$), we choose the pair s.t. k is the **smallest**. In this example, we choose $(k, p) = (1,2)$.



The suffix tree of string *cacaa*

Lemma 2.1

For each letter a , each state s has **at most one** a -transition.

Proof:

Suppose there exist two a -transitions for state s , namely $g'(s, ax) = r$ and $g'(s, ay) = t$.

There is no other states on the paths from s to r and from s to t . But now ax and ay are two paths connecting $s - r$ and $s - t$, thus $g(s, a)$ is a branching state in $STrie(T)$, which indicates that $g'(s, a) \in Q'$. Also, $g'(s, a)$ is on both paths from s to r and from s to t , which leads to a contradiction.

Q.E.D.

Lemma 2.2

$STree(T)$ is of **linear** size in $|T|$.

Proof:

We need to consider the number of states and transitions.

Since Q' is composed of branching states and leaves, and each leaf corresponds to a suffix, Q' has at most $|T|$ leaves. Now we prove Q' has at most $|T| - 1$ branching states.

Based on the basic property of trees, if we suppose each branching state (i.e., internal node) has k ($k \geq 2$) child nodes and denote the number of branching states by i , we have

$$2i + 1 \leq ki + 1 \leq |T| + i. \quad (1)$$

The expression on the right is an upper bound of $|Q'|$. From Eq. (1), we can derive that

$$i \leq |T| - 1. \quad (2)$$

Lemma 2.2

$STree(T)$ is of **linear** size in $|T|$.

Proof:

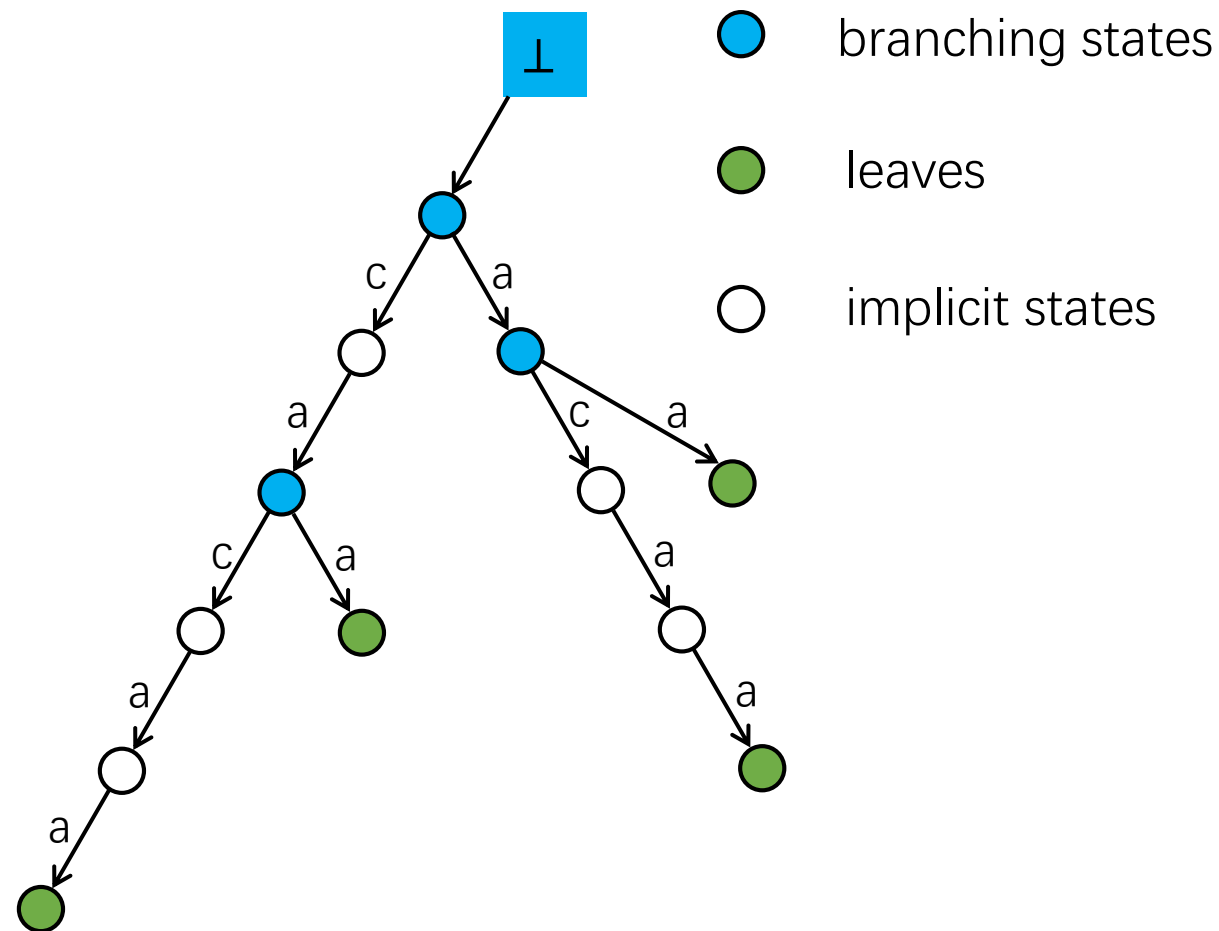
In $STree(T)$, there at most nodes (states), so the number of edges (transitions) is at most $|T| + (|T| - 1) - 1 = 2(|T| - 1)$.

Apparently, the numbers of both states and transitions are of linear size in $|T|$, so $STree(T)$ is of linear size in $|T|$.

Q.E.D.

Reference Pairs

- In a similar form of $g'(s, w) = r$, we refer to an explicit or implicit state r by a **reference pair** (s, w) .
- s is some **explicit** state that is an **ancestor** of r
- w is the string spelled out by the transitions from s to r in the corresponding suffix trie



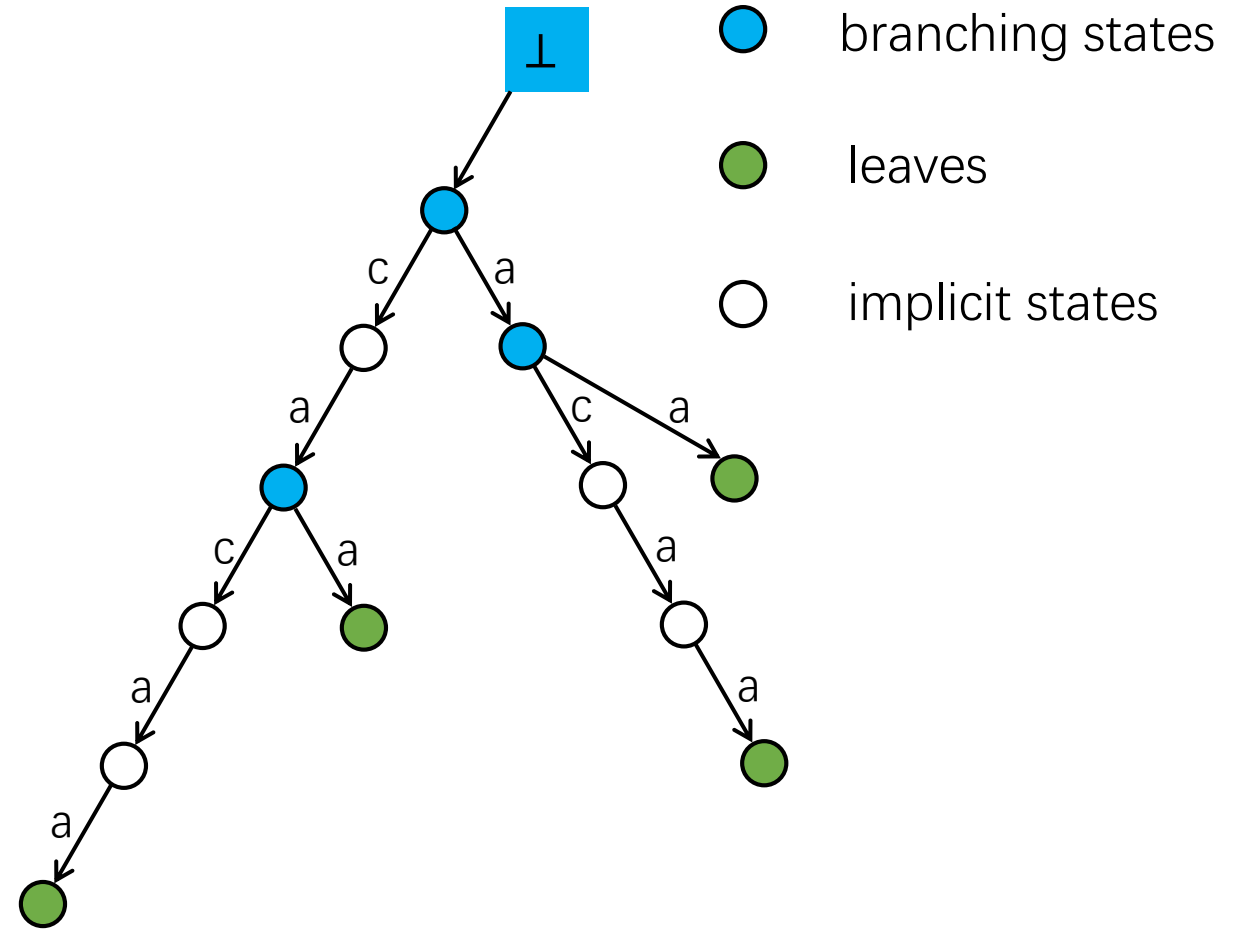
The suffix trie of string *caca*

Example

state r	reference pair (s, w)
\bar{c}	$(root, c)$
\overline{ac}	(\bar{a}, c)
\overline{ac}	$(root, ac)$

Note

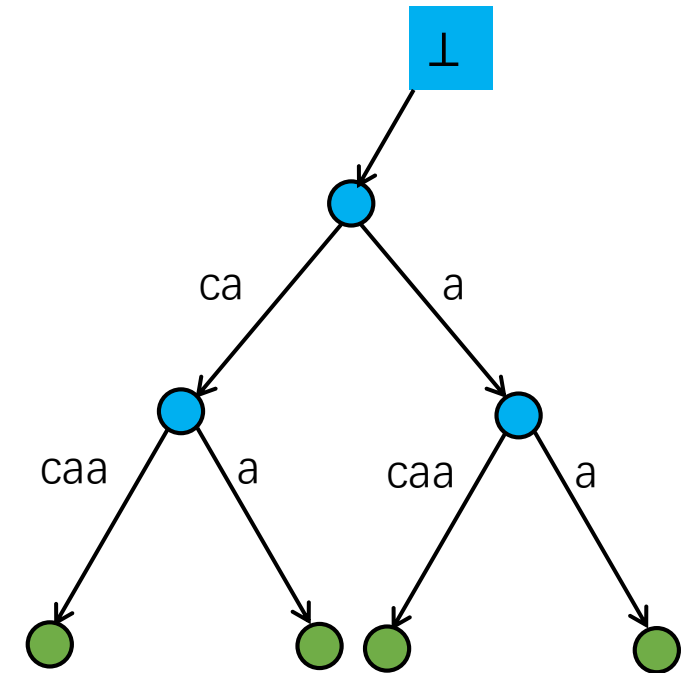
- s is not necessarily the closest ancestor of r .
- When s is the closest ancestor of r , (s, w) is **canonical**.



The suffix trie of string *cacaa*

■ Suffix Function f'

- Recall that in the context of suffix tries, $f: Q \rightarrow Q \cup \{\perp\}$ is the suffix function. For example, $f(\overline{acaa}) = \overline{caa}$.
- In suffix trees, f' is only defined on **branching states** in the same form. For example, $f'(\overline{ca}) = \overline{a}$, $f'(root) = \perp$.



The suffix tree of string *caca*

Claim 2.3

f' is well-defined: if \bar{x} is a branching state, then $f'(\bar{x})$ is also a branching state.

Proof:

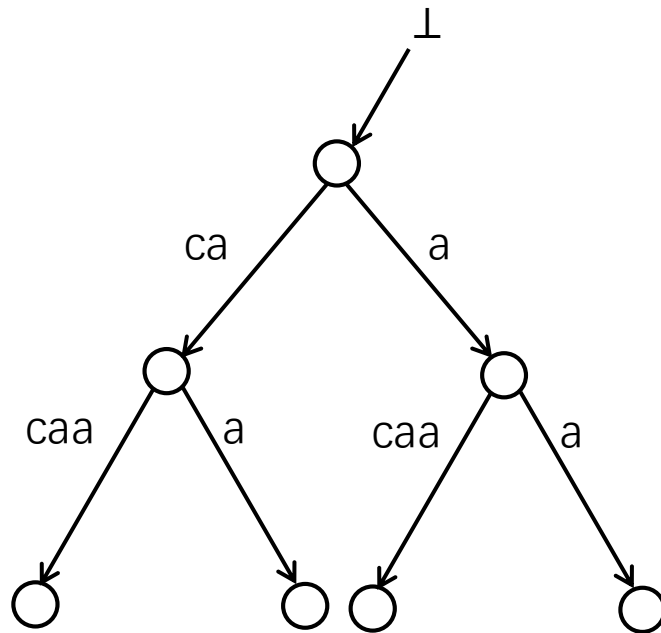
We denote x by ay , where a is a letter. If $\bar{x} = root$, we have proved that $f'(root) = \perp$ in the chapter of suffix tries. If $\bar{x} \neq root$ is a branching state then there exist at least two transitions denoted by $g(\bar{x}, b)$ and $g(\bar{x}, c)$, which indicates xb and xc are two substrings of T .

Because y is a suffix of x , yb and yc are also two substrings of T , which means states \overline{yb} and \overline{yc} exist. By definition, $f(\bar{x}) = \bar{y}$ is also a branching state.

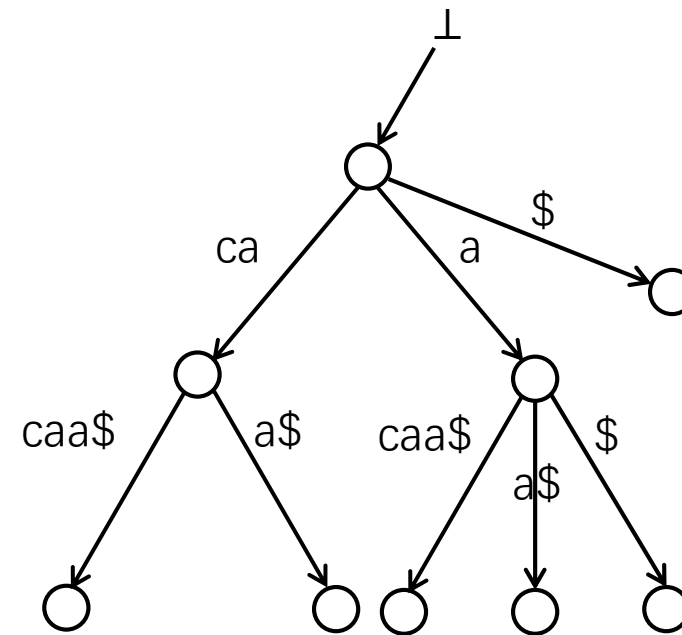
Q.E.D.

■ Omitted Accepting States

The set of accepting states F is omitted for convenience. When explicit accepting states are necessary, we can add the **terminal symbol** at the end of T so that the leaves are **in one-to-one correspondence** with the suffixes.



The suffix tree of string *caca*



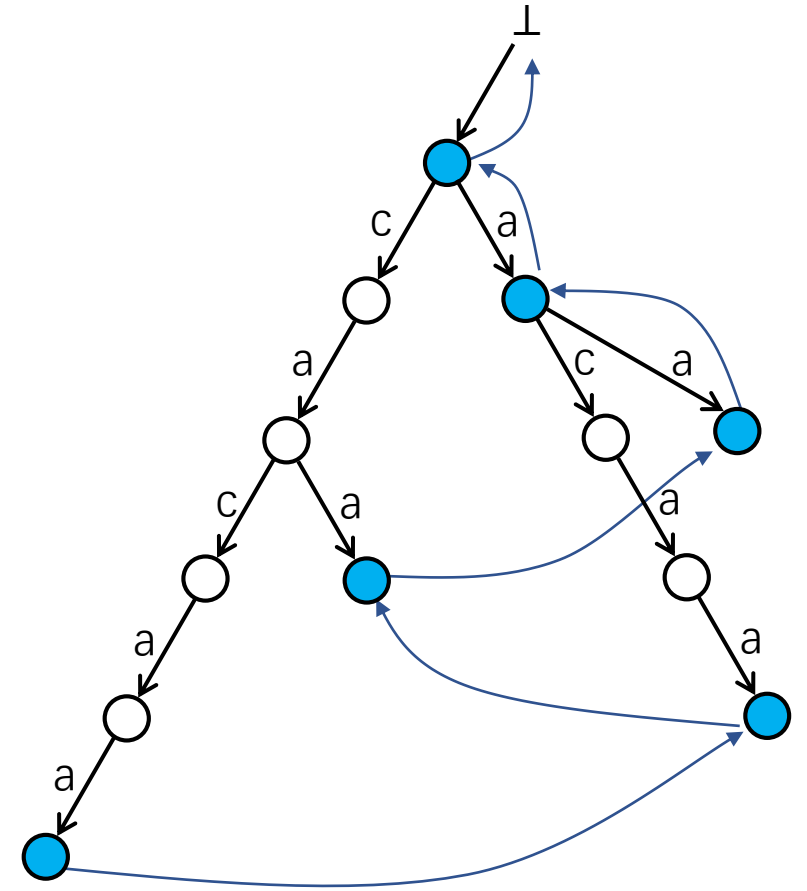
The suffix tree of string *caca\$*

Comments

- We only store explicit states in suffix trees to save space so that $STree(T)$ is of linear size in $|T|$.
- In $STree(T)$, each leaf represents a suffix but some suffixes correspond to branching states. However, in $STree(T\$)$, the leaves are **in one-to-one correspondence** with the suffixes.

On-Line Construction

We start by reviewing the on-line construction of suffix tries. When constructing suffix tries, it's important to traverse the boundary path with the suffix function f .



The boundary path of $STrie(cacaoa)$

Another Observation: A Shortcut

Fact:

the new states we need to add in $STrie(T^4)$: $\overline{caca}, \overline{aca}$ (blue)

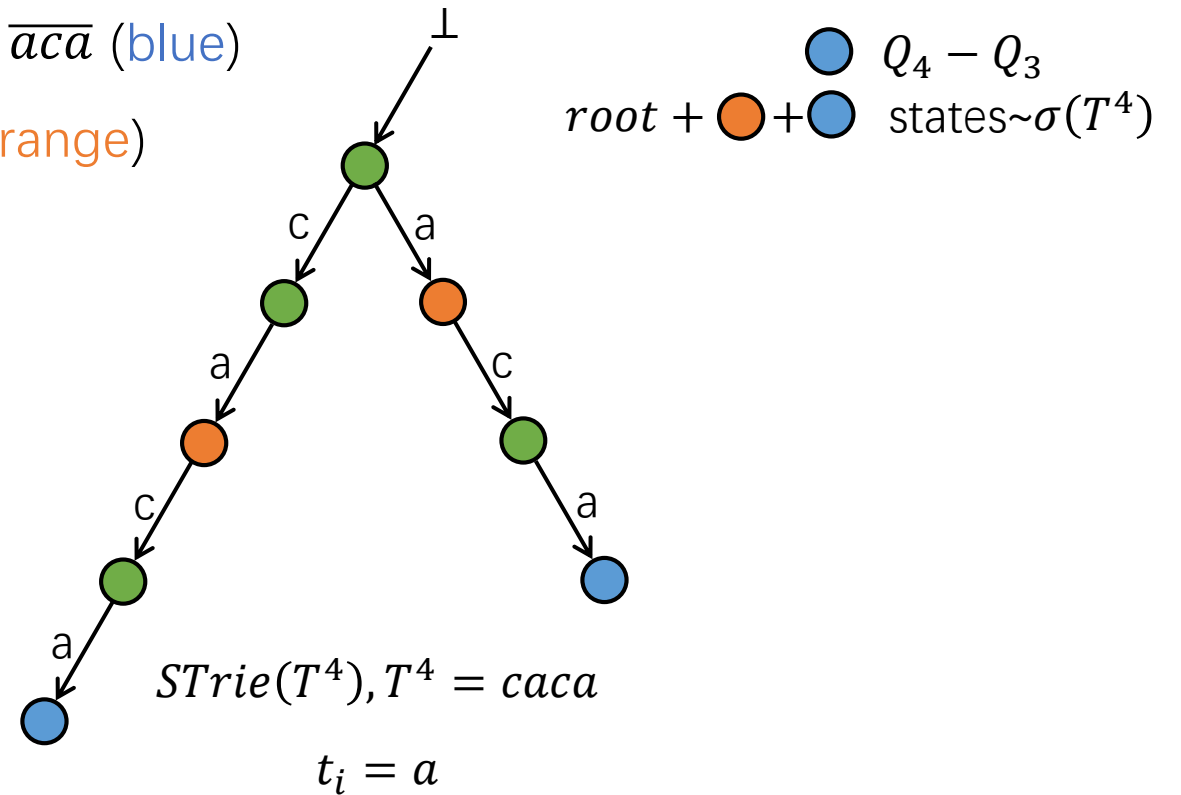
the states that already exist in $STrie(T^{33})$: $\overline{ca}, \overline{a}$ (orange)

Observation:

ca and a are both **shorter** than $caca$ and aca

Shortcut:

If we do not want to traverse the whole boundary path, then can we **stop at a certain state**?

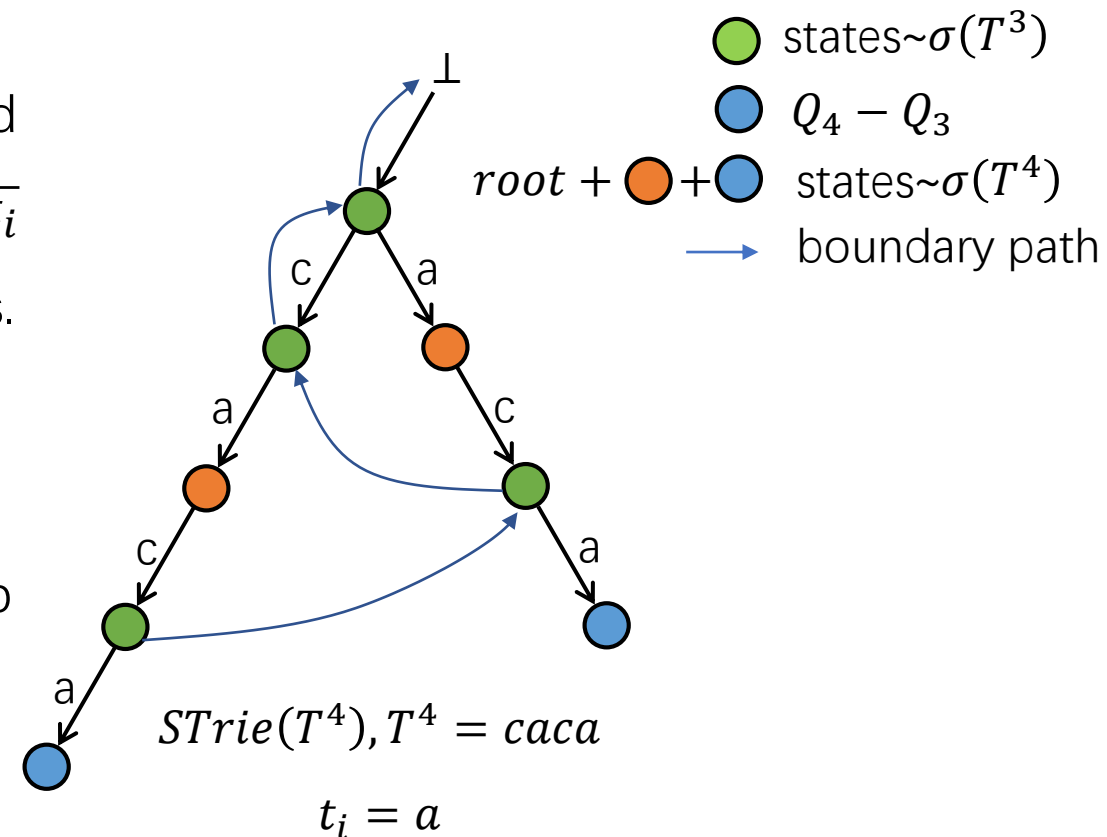


The answer is **YES!**

Shortcut:

The traversal along the boundary path can be stopped immediately when the first state \bar{z} is found s.t. state $\overline{zt_i}$ (and hence also transition $g(\bar{z}, t_i) = \overline{zt_i}$) already exists.

In the figure, $i = 4, t_i = a, z = c, \bar{z} = \bar{c}$, and $g(\bar{z}, t_i) = \overline{zt_i} = \overline{ca}$ already exists in Q_3 . So we can stop the traversal immediately at \bar{c} .



Lemma 1.1

The traversal along the boundary path can be stopped immediately when the first state \bar{z} is found s.t. state $\overline{zt_i}$ (and hence also transition $g(\bar{z}, t_i) = \overline{zt_i}$) already exists.

Proof

(First, we need to prove that such state \bar{z} always exists.)

Such state \bar{z} exists because at least the auxiliary state \perp satisfies the requirement.

From the definition of \perp we have

$$g(\perp, t_i) = g(\overline{t_i^{-1}}, t_i) = \overline{t_i^{-1}t_i} = \bar{\epsilon} = \text{root}. \quad (1)$$

Because ϵ is suffix of any string, the state *root* always exists. Thus, \perp satisfies the requirement and such \bar{z} exists.

Lemma 1.2

The traversal along the boundary path can be stopped immediately when the first state \bar{z} is found s.t. state $\overline{zt_i}$ (and hence also transition $g(\bar{z}, t_i) = \overline{zt_i}$) already exists.

Proof

(Next, because the traversal is to add the t_i transition when necessary, to prove that such \bar{z} is a proper terminal, we need to show that all other states on the boundary path already have the t_i transition.)

If $g(\bar{z}, t_i) = \overline{zt_i}$ already exists in Q_{i-1} , then zt_i is a substring of T^{i-1} (recall that there exists a one-to-one correspondence between Q and all substrings).

Consequently, for any remaining state \bar{z}' along the boundary path, z' is the suffix of z , and $z't_i$ is a suffix of zt_i , thus a substring of T^{i-1} , which indicates that $\overline{z't_i} \in Q_{i-1}$.

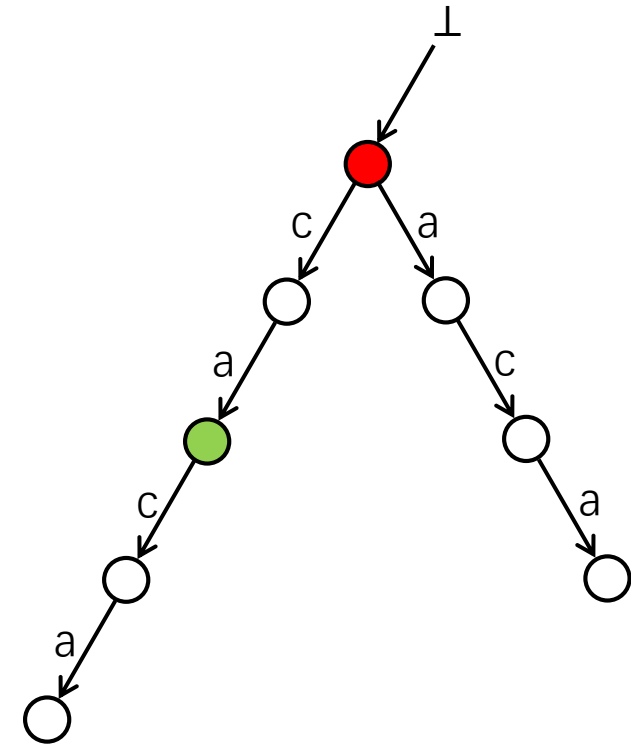
Namely, the transition $g(\bar{z}', t_i) = \overline{z't_i}$ also exist.

Q.E.D.

We define the prefix $t_1 \cdots t_i$ of T by T^i . When we construct $STrie(T^i)$ from $STrie(T^{i-1})$, we define the states that correspond to the suffixes of T^{i-1} by $s_k = \overline{t_k \cdots t_{i-1}}$. Specially, $s_i = root$ and $s_{i+1} = \perp$. There are two important states among them.

Let j be the smallest index s.t. s_j is **not a leaf**, and let j' be the smallest index s.t. $s_{j'}$ **has a t_i -transition**.

For example, To construct $STrie(cacaa)$ from $STrie(caca)$, $s_j = \overline{c\bar{a}}$, $s_{j'} = root$.



$STrie(T^4), T^4 = caca$

Lemma 2.3

Both j and j' are well-defined and $j \leq j'$.

.

Proof:

Since s_1 is a leaf, and $s_{i+1} = \perp$ is a non-leaf, there exists a smallest index j s.t. s_j is not a leaf. Because $s_{i+1} = \perp$ has a t_i -transition, there also exists a smallest index j' s.t. $s_{j'}$ has a t_i -transition.

By definition, $s_{j'}$ has a t_i -transition, so $s_{j'}$ is not a leaf. This indicates $j \leq j'$.

Q.E.D.

Lemma 2.4

(1) The on-line construction of suffix tries adds to $STrie(T^{i-1})$ a t_i -transition for each s_h where $1 \leq h < j'$.

(2) For $1 \leq h < j$, the transition expands an old branch that ends at s_h .

(3) For $j \leq h < j'$, the transition initiates a new branch.

.

Proof:

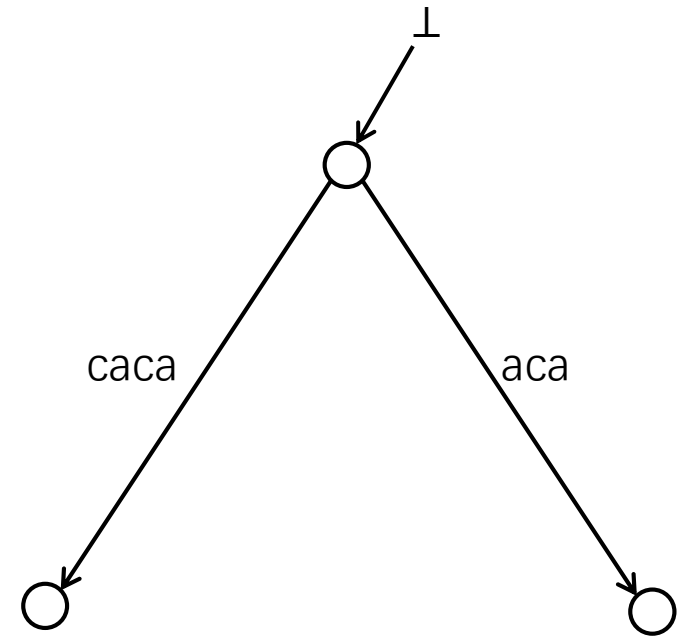
We have proved (1) in lemma 1.1 when talking about the shortcut.

(2) is obvious because by the definition of s_j , s_h is a leaf when $1 \leq h < j$.

(3) is also obvious since by the definition of s_j and $s_{j'}$, s_h is a non-leaf and the t_i -transition needs to be created in a new branch when $j \leq h < j'$.

Q.E.D.

- In terms of suffix trees, we call s_j the **active point** and $s_{j'}$ the **end point**.
- They can be explicit or implicit states. For example, $s_j = \overline{ca}$ is implicit while $s_{j'} = \text{root}$ is explicit.

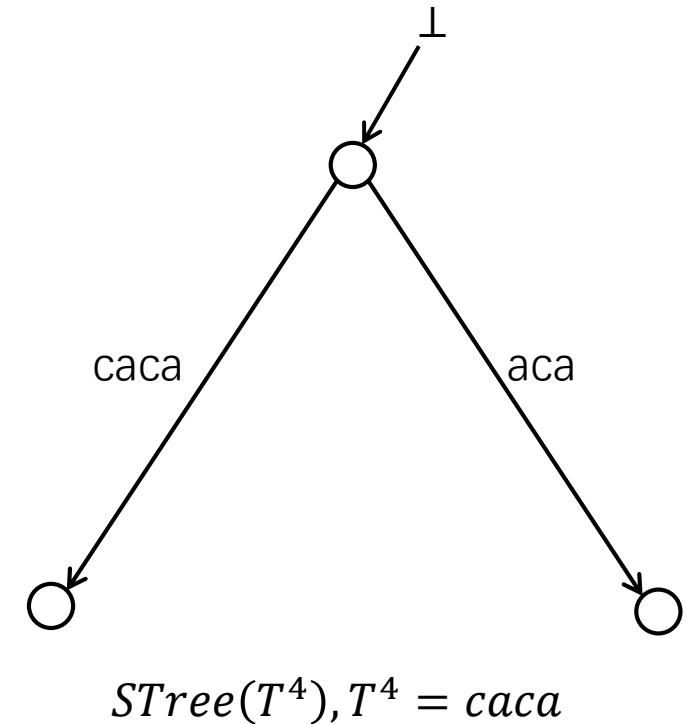


$STree(T^4), T^4 = caca$

Lemma 2.4 shows we need to insert two types of t_i -transition.
In terms of suffix trees, the operations are described as follows.

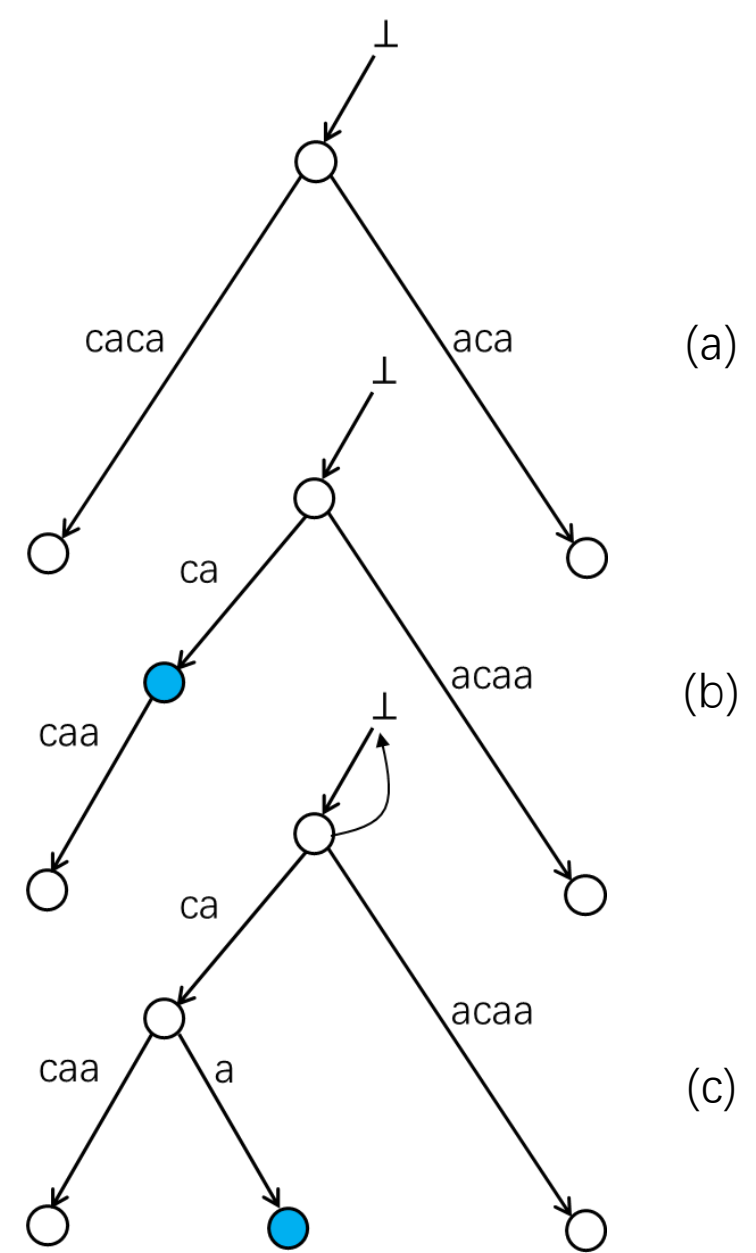
Extend a branch

- Extending a branch means updating the right pointer p . For convenience, we introduce the **open transition** $g'(s, (k, \infty))$ that always leads to a leaf.
- For example, in $STree(caca)$, the transitions from *root* are denoted by $g'(root, (1, \infty)) = \overline{caca}$ and $g'(root, (2, \infty)) = \overline{aca}$. When we begin to construct $STree(cacaa)$, the two leaves automatically become $g'(root, (1, \infty)) = \overline{cacaa}$ and $g'(root, (2, \infty)) = \overline{acaa}$.



Initiate a new branch

- If s_h is explicit, we simply add a new branch.
- If s_h is implicit, first make s_h explicit and then create a new branch. In this case, the suffix function f' also needs to be updated.
- For example, in $STree(caca)$, (a) $s_h = \overline{ca}$ is an implicit state. To initiate $g'(s_h, a)$, first (b) make s_h explicit, then (c) create a new branch.



Initiate a new branch from
an implicit state

Find s_{h+1}

- We find s_{h+1} by suffix links and reference pairs.
- If the reference pair of s_h is $(\bar{x}, (k, i - 1))$ (the right pointer is $i - 1$ because s_h is a suffix of T^{i-1}), then the reference pair of s_{h+1} is $(f'(\bar{x}), (k, i - 1))$
- In practice, we define a function *canonicalize*($s, (k, p)$) that finds the canonical reference pair of s_{h+1} . We will talk more about it later.

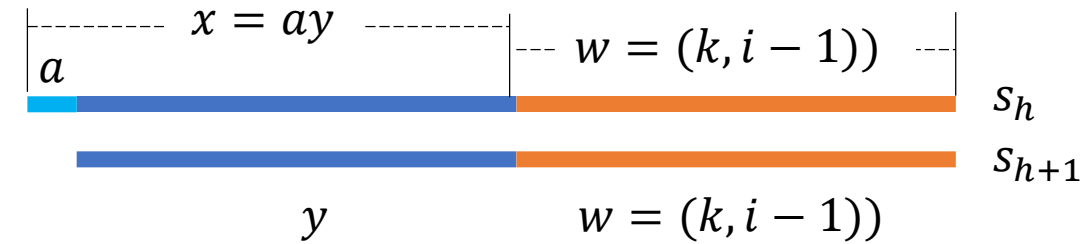


Illustration of $s_{h+1} = (f'(\bar{x}), (k, i - 1))$

Iteration: From End to Start

In each iteration, we add a new transition to each state s_h that is along the path from s_j to $s_{j'}$. However, how do we find the active point for the next iteration?

We first show some alternative definitions of active point and end point.

Lemma 2.5

s_j is the active point of $STree(T^i)$ iff the corresponding string $str_j = t_j t_{j+1} \dots t_i$ is the longest suffix of T^i that occurs at least twice in T^i .

Proof:

\Rightarrow

Suppose s_j is the active point of $STree(T^i)$. Since s_j is not a leaf, we denote the leaf on the branch where s_j exists by s_l . In $STrie(T^i)$, each leaf corresponds to a suffix, so str_l is also a suffix. This means str_j is a substring (not a suffix) of str_l , thus str_j occurs at least twice in T^i .

By definition, along the boundary path, s_j is the non-leaf with the smallest index j , so str_j is the longest suffix of T^i that occurs at least twice in T^i .

Lemma 2.5

s_j is the active point of $STree(T^i)$ iff the corresponding string $str_j = t_j t_{j+1} \dots t_i$ is the longest suffix of T^i that occurs at least twice in T^i .

.

Proof:

\Leftarrow

Suppose $str_j = t_j t_{j+1} \dots t_i$ is the longest suffix of T^i that occurs at least twice in T^i .

If the corresponding state s_j is a leaf of $STrie(T^i)$, since str_j occurs at least twice in T^i , str_j must be a substring of another suffix str_k of T^i . Thus, on the same branch, s_k is also a leaf, which leads to a contradiction. Consequently, s_j is not a leaf.

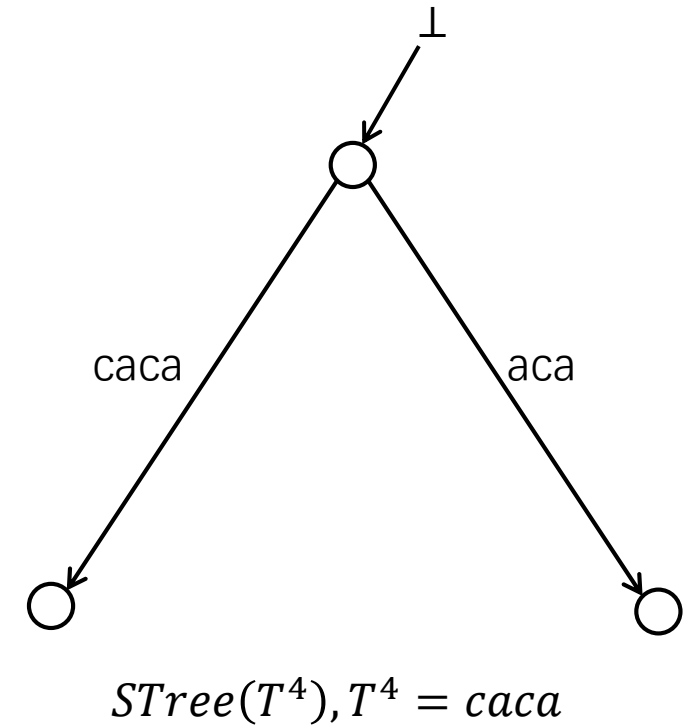
Since str_j is the longest among such suffixes, j is the smallest. Thus, s_j is the active point.

Q.E.D.

Example

In $STree(caca)$, \overline{ca} is the active point (implicit).

In $T^4 = caca$, ca is the longest suffix of T^4 that occurs at least twice in T^4 .



Lemma 2.6

$s_{j'}$ is the end point of $STree(T^i)$ iff the corresponding string $str_{j'} = t_j, t_{j+1} \dots t_i$ is the longest suffix of T^i s.t. that $t_j, t_{j+1} \dots t_i t_{i+1}$ is a substring of T^i .

.

Proof:

\Rightarrow

Suppose $s_{j'}$ is the end point of $STree(T^i)$.

By definition, $str_{j'}$ is the longest suffix of T^i that has a t_{i+1} -transition, which means $str_{j'}, t_{i+1}$ is a substring of T^i . Thus, $str_{j'} = t_j, t_{j+1} \dots t_i$ is the longest suffix of T^i s.t. that $t_j, t_{j+1} \dots t_i t_{i+1}$ is a substring of T^i .

Lemma 2.6

$s_{j'}$ is the end point of $STree(T^i)$ iff the corresponding string $str_{j'} = t_{j'}t_{j'+1} \dots t_i$ is the longest suffix of T^i s.t. that $t_{j'}t_{j'+1} \dots t_it_{i+1}$ is a substring of T^i .

Proof:

\Leftarrow

Suppose $str_{j'} = t_{j'}t_{j'+1} \dots t_i$ is the longest suffix of T^i s.t. that $t_{j'}t_{j'+1} \dots t_it_{i+1}$ is a substring of T^i .

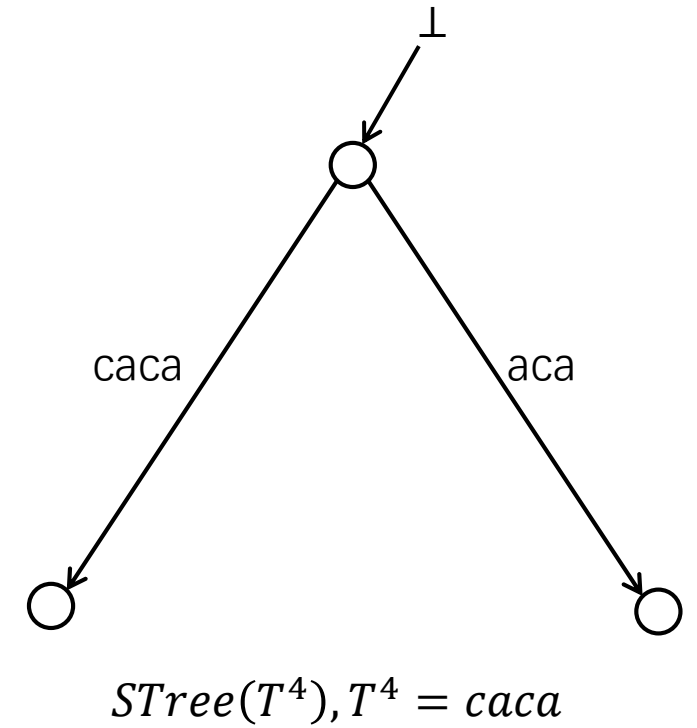
Since $str_{j'}t_{i+1}$ is a substring of T^i , $s_{j'}$ has a t_{i+1} -transition. Among all such suffixes, $str_{j'}$ is the longest, which indicates that index j' is the smallest. Consequently, $s_{j'}$ is the end point of $STree(T^i)$.

Q.E.D.

Example

In $STree(caca)$, $root$ is the end point.

In $T^4 = caca$, ϵ is the longest suffix of T^4 s.t. that $\epsilon t_5 = a$ is a substring of T^4 .



Lemma 2.7

If $s_{j'}$ is the end point of $STree(T^{i-1})$, then $g(s_{j'}, t_i)$ is the active point of $STree(T^i)$.
Namely, if $(s, (k, i - 1))$ is the reference pair of the end point of $STree(T^{i-1})$, then $(s, (k, i))$ is the reference pair of the active point of $STree(T^i)$.

Proof:

By Lemma 2.6, $str_{j', t_i} = t_{j'}, t_{j'+1} \dots t_{i-1} t_i$ is a substring of T^{i-1} , so str_{j', t_i} is also a substring of T^i . Obviously, str_{j', t_i} is a suffix of T^i , which means str_{j', t_i} occurs at least twice in T^i .

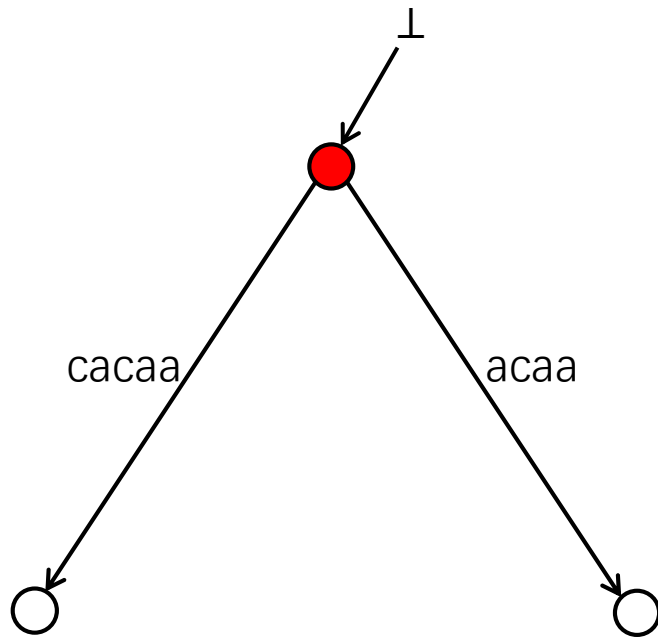
Also, $str_{j'}$ is the longest suffix of T^{i-1} s.t. that $t_{j'}, t_{j'+1} \dots t_i$ is a substring of T^{i-1} . Thus, str_{j', t_i} is the longest suffix of T^i that occurs at least twice in T^i . By Lemma 2.5, $g(s_{j'}, t_i)$ is the active point of $STree(T^i)$.

Q.E.D.

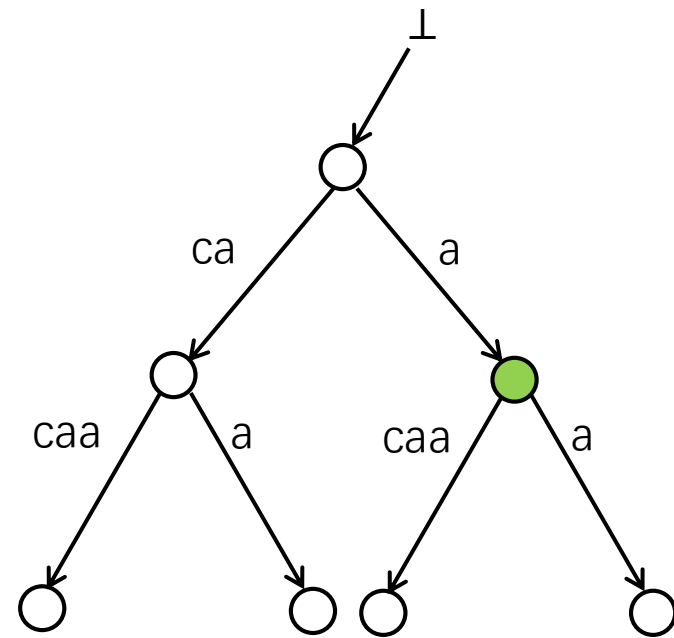
Example

In $STree(caca)$, $root$ is the end point.

In $STree(cacaa)$, $g(root, a) = \bar{a}$ is the active point.



$STree(T^4), T^4 = caca$



$STree(T^5), T^5 = cacaa$

Comments

- There are two types of operations when inserting new t_i -transitions: extending an existing branch and initiating a new branch.
- Before inserting a new t_i -transition, we still need to define a function $test_and_split(s, (k, p), t)$ to check whether s_h already has a t_i -transition (namely has reached the end point $s_{j'}$).
- An open transition extends an existing branch automatically. Practically, the open transition $g'(s, (k, \infty))$ can be replaced with $g'(s, (k, n))$, where $n = |T|$.
- When initiating a new branch, first make s_h explicit and then create a new branch.

■ Comments

- We find s_{h+1} by reference pairs and suffix links. To acquire the canonical reference pair, we still need to define a function *canonize*($s, (k, p)$).
- To prepare for the traversal of the next iteration, we find the active point by $g(s_j, t_i)$.

Algorithms

We need three functions:

- (1) *update* that constructs $S\text{Tree}(T^i)$ from $S\text{Tree}(T^{i-1})$,
- (2) *test_and_split*($s, (k, p), t$) that checks whether s_h already has a t_i -transition (namely has reached the end point $s_{j'}$), and
- (3) *canonize*($s, (k, p)$) that finds the canonical reference pair of $(s, (k, p))$.

■ *test_and_split*($s, (k, p), t$)

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (**true**, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (**false**, r)

else

if there is no t -transition from s **then return** (**false**, s)

else return (**true**, s)

- This function tests whether or not a state with **canonical** reference pair $(s, (k, p))$ has a t -transition.
- The first output parameter is a bool variable:
true: $(s, (k, p))$ has a t -transition
false: $(s, (k, p))$ does not have a t -transition
- The second output parameter is an explicit state.

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

else

if there is no t -transition from s **then return** (false, s)

else return (true, s)

➤ When $k > p$, $(s, (k, p)) = s$ is an explicit state.

***test_and_split*($s, (k, p), t$)**

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

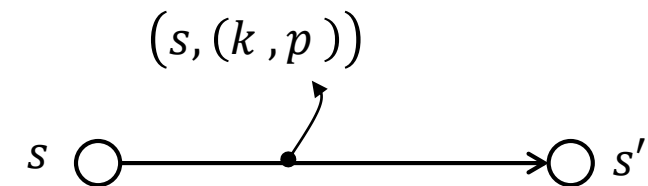
return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)

- When $k \leq p$, first find the t_k -transition $g'(s, (k', p')) = s'$. This is because $(s, (k, p))$ can be an implicit state, but $(s, (k', p'))$ is explicit.
- Because the input is a canonical reference pair, s is the closest ancestor of $(s, (k, p))$.



$(s, (k, p))$ can be an implicit state

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

else

if there is no t -transition from s **then return** (false, s)

else return (true, s)

- We can find such t_k -transition in constant time because (1) we suppose $|\Sigma|$ is independent of $|T|$, and (2) for each letter $a \in \Sigma$, each state s has at most one a -transition (Lemma 2.1).

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

else

if there is no t -transition from s **then return** (false, s)

else return (true, s)

- $t = t_{k'+p-k+1}$ tests whether $(s, (k, p))$ has a t -transition.
- Sometimes $t_{k+p-k+1}$ also works. However, by definition of g' , $k' \leq k$, which avoids potential problems of the range of index.

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

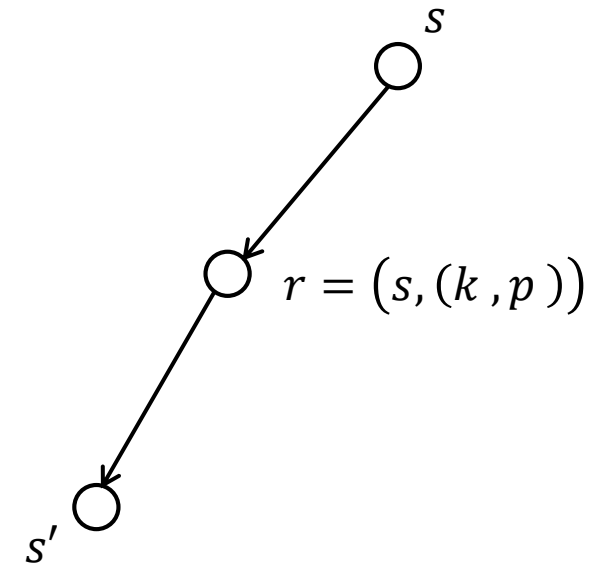
return (false, r)

else

if there is no t -transition from s **then return** (false, s)

else return (true, s)

- When $(s, (k, p))$ is an implicit state, make it explicit.



■ ***canonicalize***($s, (k, p)$)

```
canonicalize( $s, (k, p)$ )  
if  $p < k$  then return ( $s, k$ );  
else  
    find the  $t_k$ -transition  $g'(s, (k', p')) = s'$  from  $s$ ;  
    while  $p' - k' \leq p - k$  do  
         $k \leftarrow k + p' - k' + 1$ ;  
         $s \leftarrow s'$ ;  
        if  $k \leq p$  then  
            find the  $t_k$ -transition  
             $g'(s, (k', p')) = s'$  from  $s$ ;  
    return ( $s, k$ ).
```

- This function finds the canonical reference pair of $(s, (k, p))$.
- The first output parameter is the closest ancestor of $(s, (k, p))$
- The second output parameter is the left pointer (because the right pointer does not change).

canonize($s, (k, p)$)

if $p < k$ **then** **return** (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition

$g'(s, (k', p')) = s'$ from s ;

return (s, k) .

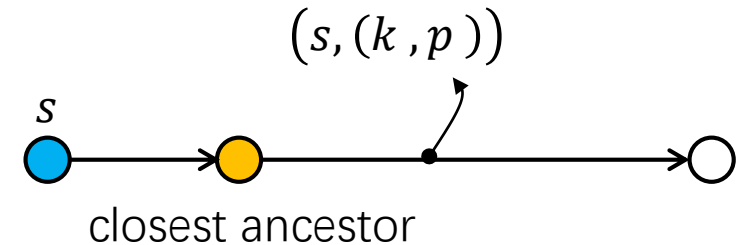
- When $k > p$, $(s, (k, p))$ is a canonical reference pair.

```

canonize( $s, (k, p)$ )
if  $p < k$  then return ( $s, k$ );
else
    find the  $t_k$ -transition  $g'(s, (k', p')) = s'$  from  $s$ ;
    while  $p' - k' \leq p - k$  do
         $k \leftarrow k + p' - k' + 1$ ;
         $s \leftarrow s'$ ;
        if  $k \leq p$  then
            find the  $t_k$ -transition
             $g'(s, (k', p')) = s'$  from  $s$ ;
    return ( $s, k$ ).

```

- When $k \leq p$, update the reference pair until it becomes a canonical reference pair.



s can be far from the closest ancestor

canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

$k \leftarrow k + p' - k' + 1$;

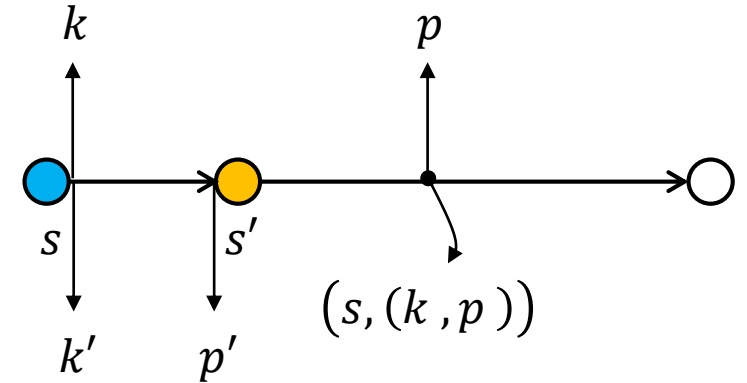
$s \leftarrow s'$;

if $k \leq p$ then

 find the t_k -transition

$g'(s, (k', p')) = s'$ from s ;

return (s, k) .



canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

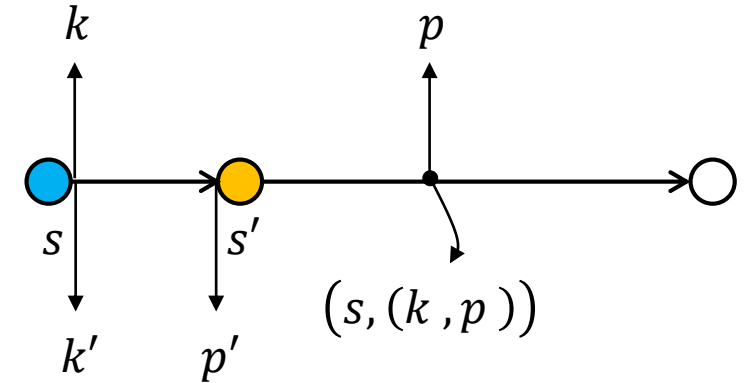
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

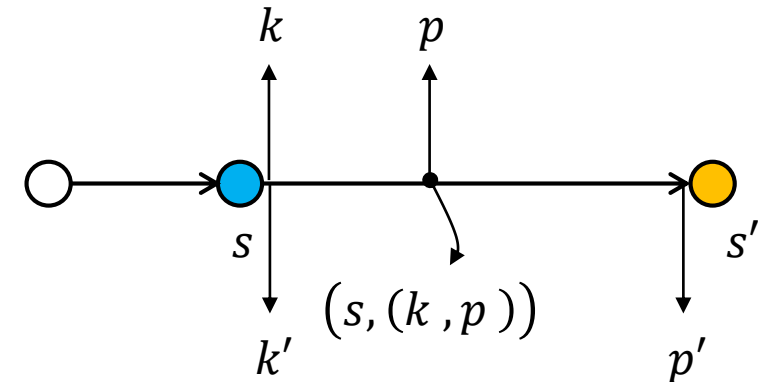
if $k \leq p$ then

find the t_k -transition
 $g'(s, (k', p')) = s'$ from s ;

return (s, k) .



iter 1



canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

$k \leftarrow k + p' - k' + 1$;

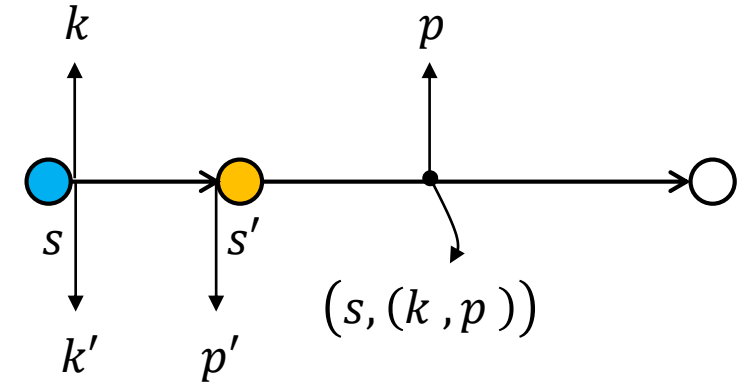
$s \leftarrow s'$;

if $k \leq p$ then

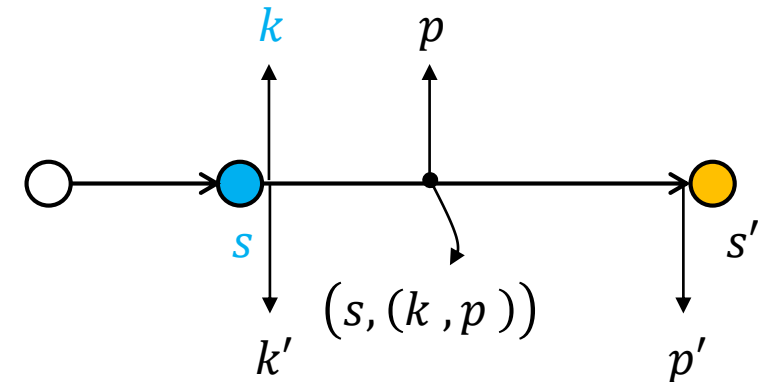
find the t_k -transition

$g'(s, (k', p')) = s'$ from s ;

return (s, k) .



iter 1



canonize($s, (k, p)$)

if $p < k$ **then return** (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition

$g'(s, (k', p')) = s'$ from s ;

return (s, k).

Note

- When *canonize* is called, the while loop is executed **at most once**. We will give the reason in time complexity analysis.

canonize(s, (k, p))

```
if  $p < k$  then return  $(s, k)$ ;
else
    find the  $t_k$ -transition  $g'(s, (k', p')) = s'$  from  $s$ ;
    while  $p' - k' \leq p - k$  do
         $k \leftarrow k + p' - k' + 1$ ;
         $s \leftarrow s'$ ;
    if ...
```

test_and_split(s, (k, p), t)

```
...
else
    replace the  $t_k$ -transition above by transitions
     $g'(s, (k', k' + p - k)) = r$  and
     $g'(r, (k' + p - k + 1, p')) = s'$ ,
    where  $r$  is a new state;
    return (false,  $r$ )
else
    if there is no  $t$ -transition from  $s$  then return (false,  $s$ )
    else return (true,  $s$ )
```

Note

- Different from *test_and_split*, we update k by $k + p' - k' + 1$ instead of $k' + p' - k' + 1 = p' + 1$. This is because in *test_and_split*, the left and right pointers are **updated simultaneously**. However, in *canonize*, the right pointer p **remains the same**.
- Since $k' \leq k$, although (k, p) and $(k', k' + p - k)$ refers to the same string, (k, p) and (k', p) are different.
- For example, in string *caca*, let $k' = 1$, $(k, p) = (3, 4)$. We have $t_1 = t_3 = c$, but $(k', p) = (1, 4)$ refers to *caca* instead of *ca*.

***canonize*($s, (k, p)$)**

if $p < k$ **then return** (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if ...

Note

- Because in *canonize*, the right pointer p remains the same, we always have $k \leq p$. Thus, we will not meet any problem about the range of index.

***test_and_split*($s, (k, p), t$)**

...

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (**false**, r)

else

if there is no t -transition from s **then return** (**false**, s)

else return (**true**, s)

■ *update*

```
(s, (k, i - 1)) is the canonical reference pair for the active point;  
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);  
oldr ← root;  
while end_point == false do  
    create new transition  $g'(r, (i, \infty)) = r'$   
    where  $r'$  is a new state;  
    if oldr ≠ root then create new suffix link  $f'(\text{oldr}) = r$ ;  
    oldr ← r;  
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));  
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);  
if oldr ≠ root then create new suffix link  $f'(\text{oldr}) = s$ ;  
(s, k) ← canonize(s, (k, i));
```

- This function constructs $STree(T^i)$ from $STree(T^{i-1})$.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- First find the active point of $S\mathit{Tree}(T^{i-1})$ (Lemma 2.7).
- s is the end point of $S\mathit{Tree}(T^{i-2})$.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- *test_and_split* tests whether or not a state with **canonical** reference pair $(s, (k, p))$ has a t -transition.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- This assignment guarantees that if the while loop is not executed, we do not need to create a new suffix link (in this case, $end_point == true$).


```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- When $(s, (k, i - 1))$ is not an end point, create a t_i -transition.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- Recall that our purpose is to add a t_i -transition to each s_h . When s_h is an implicit state, we make it explicit and create new suffix links.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

➤ $oldr$ and r represent the traversal of s_h .

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

- When we jump out of the while loop, s is the end point of $STree(T^{i-1})$.

```

( $s, (k, i - 1)$ ) is the canonical reference pair for the active point;
( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$ 
    where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
    ( $s, k$ )  $\leftarrow canonize(f'(s), (k, i - 1))$ ;
    ( $end\_point, r$ )  $\leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
    ( $s, k$ )  $\leftarrow canonize(s, (k, i))$ ;

```

➤ Find the active point for the next iteration.

Comments

- *test_and_split*($s, (k, p), t$) tests whether or not a state with canonical reference pair $(s, (k, p))$ has a t -transition. This function can also change an implicit state to an explicit state.
- Some *canonize*($s, (k, p)$) is executed before *test_and_split*, which guarantees that the input reference pair of *test_and_split* is canonical.
- *update* constructs suffix trees in **linear time**.

Execution on an Example

STree(T^0)

$T = caciaa$

$T^0 = \epsilon$

create states *root* and \perp ;

for $i \leftarrow 1, \dots, 5$ do

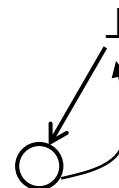
 create new transition $g'(\perp, (-i, -i)) = \text{root}$;

 create new suffix link $f'(\text{root}) = \perp$;

$s = \text{root}$;

$k = 1$;

$i = 0$;



g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>

f'	
input	output
<i>root</i>	\perp

$STree(T^0) \rightarrow STree(T^1)$

$T^0 = \epsilon$

$T^1 = c$

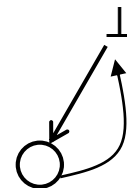
$i = 0$

$s = root$

$k = 1$

$i \leftarrow 1$

g'		f'	
input	output	input	output
$(\perp, (-i, -i))$	$root$	$root$	\perp



$i \leftarrow i + 1;$

$(s, (k, i - 1))$ is the canonical reference pair for the active point;

$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

$oldr \leftarrow root;$

while $end_point == false$ **do**

 create new transition $g'(r, (i, \infty)) = r'$ where r' is a new state;

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = r;$

$oldr \leftarrow r;$

$(s, k) \leftarrow canonize(f'(s), (k, i - 1));$

$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = s;$

$(s, k) \leftarrow canonize(s, (k, i));$

STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

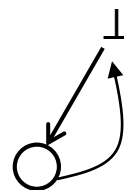
$s = root$

$k = 1$

$(end_point, r) \leftarrow$
 $test_and_split(root, (1, 0), c)$

g'	
input	output
$(\perp, (-i, -i))$	$root$

f'	
input	output
$root$	\perp



```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
     $(s, k) \leftarrow canonize(s, (k, i));$ 
  
```

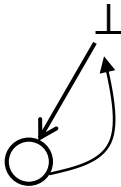
$$STree(T^0) \rightarrow STree(T^1)$$

$$\begin{aligned} T^0 &= \epsilon & s &= root \\ T^1 &= c & k &= 1 \\ i &= 1 & p &= 0 \\ s &= root & t &= c \\ k &= 1 \end{aligned}$$

return (false, *root*)

g'	
input	output
$(\perp, (-i, -i))$	$root$

f'	
input	output
$root$	\perp



***test_and_split*(*s*, (*k*, *p*), *t*)**

if *k* ≤ *p* **then**

find the *t_k*-transition *g'*(*s*, (*k'*, *p'*)) = *s'* from *s*;

if *t* = *t_{k'+p-k+1}* **then return** (true, *s*)

else

replace the *t_k*-transition above by transitions

$$g'(s, (k', k' + p - k)) = r \text{ and}$$

$$g'(r, (k' + p - k + 1, p')) = s',$$

where *r* is a new state;

return (false, *r*)

else

if there is no *t* -transition from *s* **then return** (false, *s*)

else return (true, *s*)

$STree(T^0) \rightarrow STree(T^1)$

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

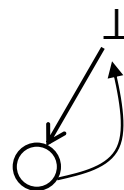
$s = root$

$k = 1$

$(end_point, r) \leftarrow (false, root)$

g'	
input	output
$(\perp, (-i, -i))$	$root$

f'	
input	output
$root$	\perp



```

i ← i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i − 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i − 1));
    (end_point, r) ← test_and_split(s, (k, i − 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
  
```

STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = root$

$k = 1$

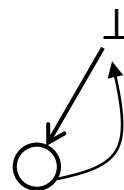
$end_point = false$

$r = root$

oldr \leftarrow *root*

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>

f'	
input	output
<i>root</i>	\perp



```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(\textit{oldr}) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(\textit{oldr}) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
    
```

STree(T^0) \rightarrow ***STree***(T^1)

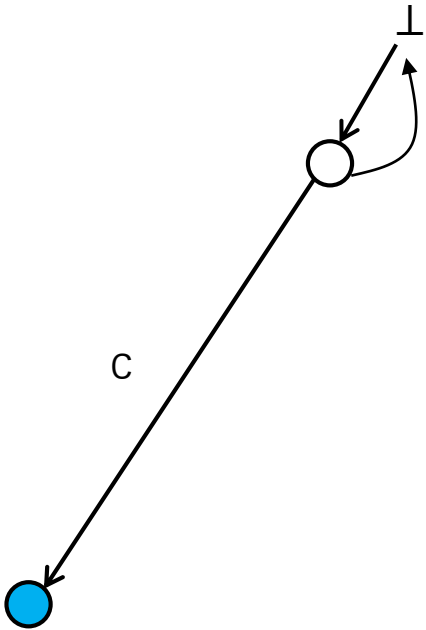
$T^0 = \epsilon$
 $T^1 = c$
 $i = 1$
 $s = root$
 $k = 1$
 $end_point = false$
 $r = root$
 $oldr = root$

create r'

```
 $i \leftarrow i + 1;$   
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;  
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$   
 $oldr \leftarrow r;$   
while  $end\_point == false$  do  
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;  
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$   
     $oldr \leftarrow r;$   
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$   
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$   
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$   
 $(s, k) \leftarrow canonize(s, (k, i));$ 
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = root$

$k = 1$

$end_point = false$

$r = root$

$oldr = root$

oldr \leftarrow *root*

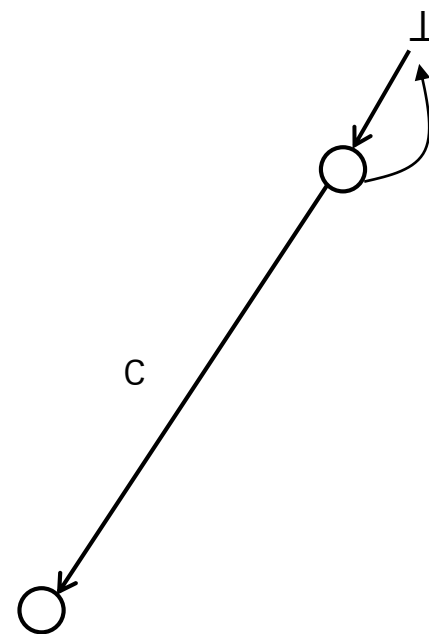
g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(root, (1, \infty))$	\bar{c}

f'	
input	output
<i>root</i>	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i - 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));

```



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = root$

$k = 1$

$end_point = false$

$r = root$

$oldr = root$

$(s, k) \leftarrow canonize(\perp, (1, 0))$

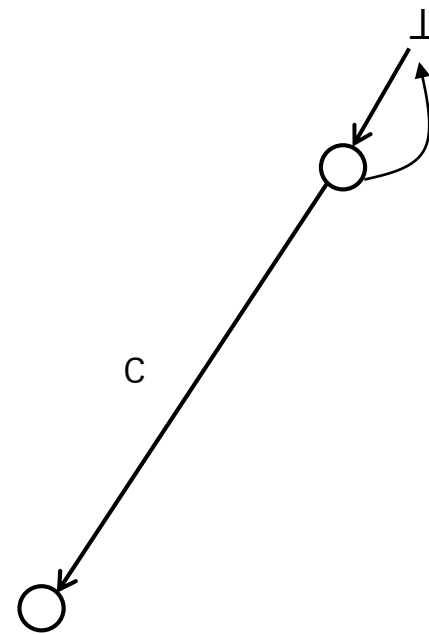
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
    (s, k) ← canonize(s, (k, i));

```



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$ $s = \perp$
 $T^1 = c$ $k = 1$
 $i = 1$ $p = 0$
 $s = root$
 $k = 1$
 $end_point = false$
 $r = root$
 $oldr = root$

return ($\perp, 1$)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

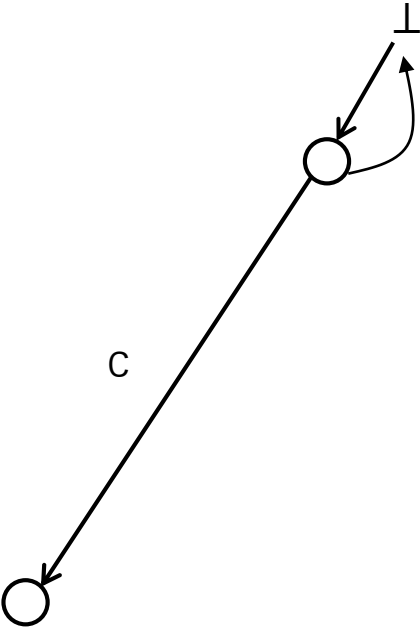
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = root$

$k = 1$

$end_point = false$

$r = root$

$oldr = root$

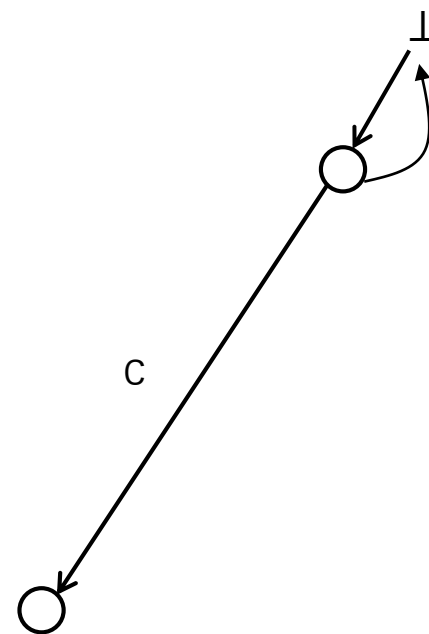
$(s, k) \leftarrow (\perp, 1)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1$ ;
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
 $oldr \leftarrow root$ ;
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
     $oldr \leftarrow r$ ;
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1))$ ;
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
 $(s, k) \leftarrow canonize(s, (k, i))$ ;
  
```



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = \perp$

$k = 1$

$end_point = false$

$r = root$

$oldr = root$

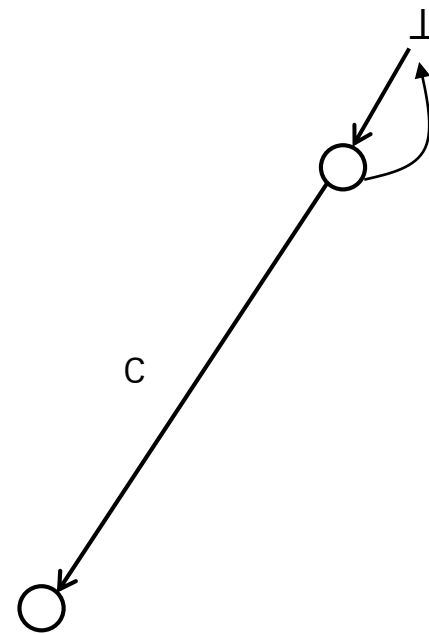
$(end_point, r) \leftarrow test_and_split(\perp, (1, 0), c)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
    (s, k) ← canonize(s, (k, i));
  
```



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$ $s = \perp$
 $T^1 = c$ $k = 1$
 $i = 1$ $p = 0$
 $s = \perp$ $t = c$
 $k = 1$
end_point = false
 $r = root$
 $oldr = root$

return (true, \perp)

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

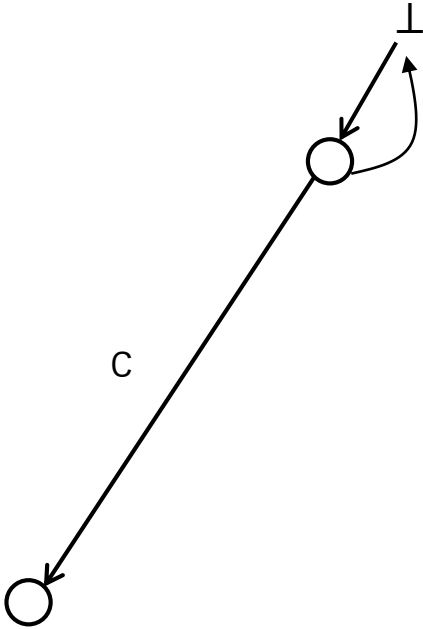
else

if there is no t -transition from s then return (false, s)

else return (true, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = \perp$

$k = 1$

$end_point = false$

$r = root$

$oldr = root$

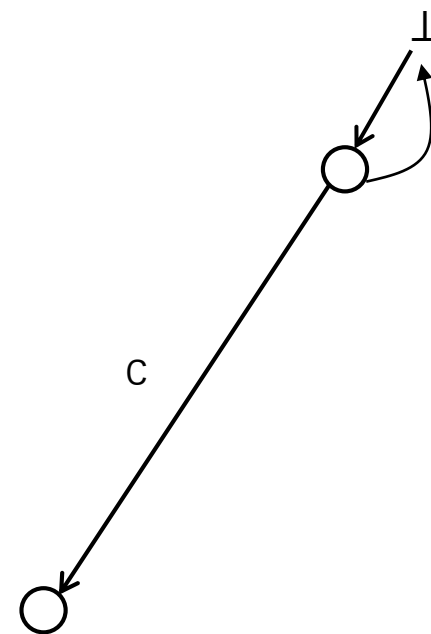
$(end_point, r) \leftarrow (true, \perp)$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r ;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
    (s, k) ← canonize(s, (k, i));
    
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = \perp$

$k = 1$

$end_point = true$

$r = \perp$

$oldr = root$

$(s, k) \leftarrow canonize(\perp, (1, 1))$

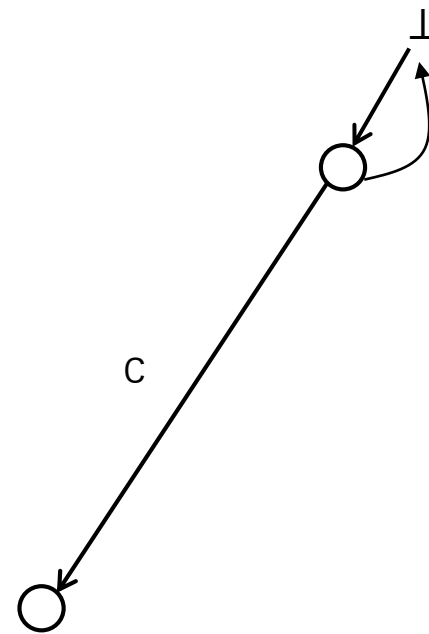
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



$$STree(T^0) \rightarrow STree(T^1)$$
$$T^0 = \epsilon \quad S = \perp$$
$$T^1 = c \quad k = 1$$
$$i = 1 \qquad p = 1$$
$$S = \perp$$
$$k = 1$$

```
end_point = true
```

$$r = \perp$$
$$oldr = root$$
$$(k', p') = (-1, -1)$$
$$s' = root$$
$$\mathit{canonize}(s, (k, p))$$

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

```
while  $p' - k' \leq p - k$  do
```

$$k \leftarrow k + p' - k' + 1;$$
$$S \leftarrow S';$$

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

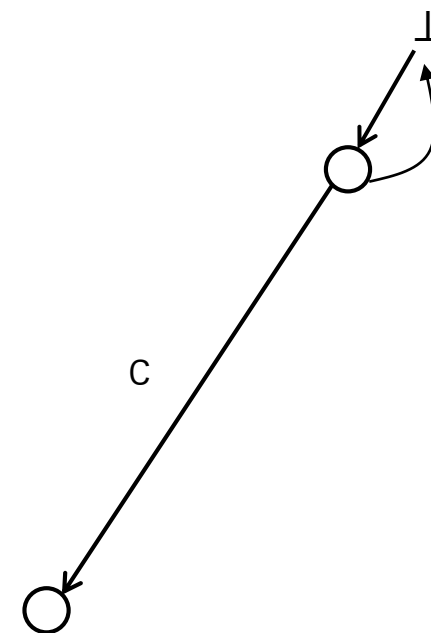
```

return (s, k).

```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$ $s = \perp$ $s' = root$
 $T^1 = c$ $k = 1$ $k' = -1$
 $i = 1$ $p = 1$ $p' = -1$
 $s = \perp$
 $k = 1$
 $end_point = true$
 $r = \perp$
 $oldr = root$

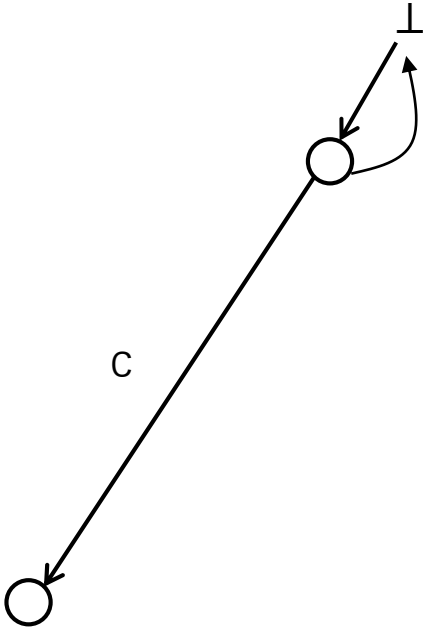
$k \leftarrow 2$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k) .



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$ $s = \perp$ $s' = root$
 $T^1 = c$ $k = 2$ $k' = -1$
 $i = 1$ $p = 1$ $p' = -1$
 $s = \perp$
 $k = 1$
 $end_point = true$
 $r = \perp$
 $oldr = root$

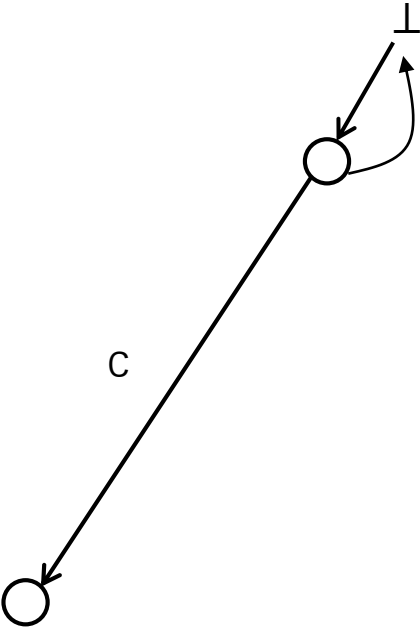
$s \leftarrow root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ **do**
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ **then**
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$ $s = root$ $s' = root$
 $T^1 = c$ $k = 2$ $k' = -1$
 $i = 1$ $p = 1$ $p' = -1$
 $s = \perp$
 $k = 1$
 $end_point = true$
 $r = \perp$
 $oldr = root$

return ($root, 2$)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

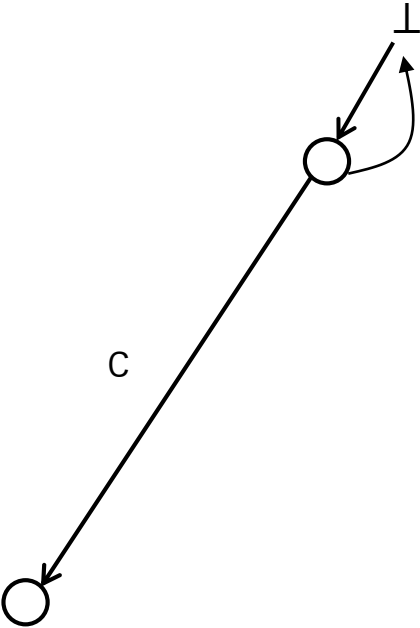
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = \perp$

$k = 1$

$end_point = true$

$r = \perp$

$oldr = root$

$(s, k) \leftarrow (root, 2)$

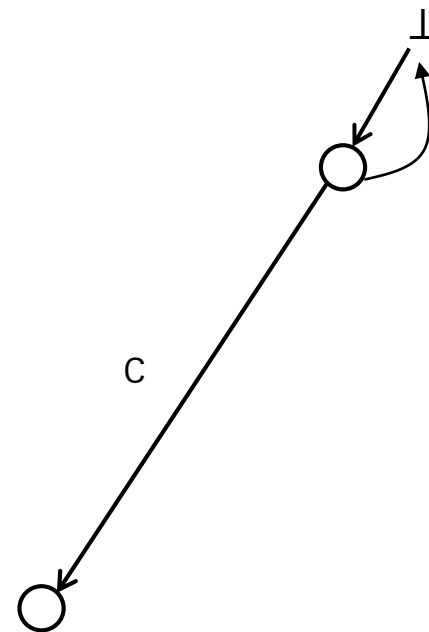
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^0) \rightarrow ***STree***(T^1)

$T^0 = \epsilon$

$T^1 = c$

$i = 1$

$s = root$

$k = 2$

$end_point = true$

$r = \perp$

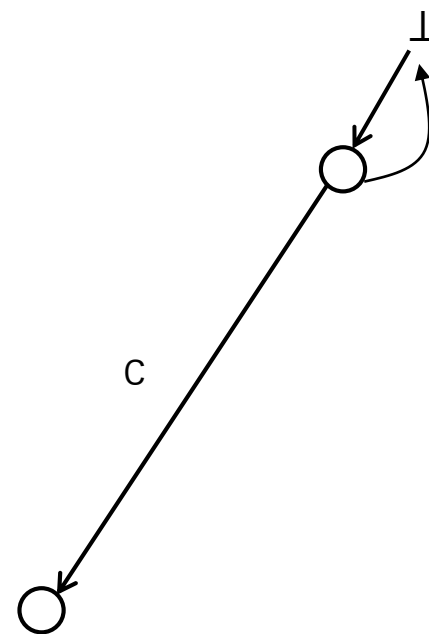
$oldr = root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\bar{c}

f'	
input	output
$root$	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 1$

$s = root$

$k = 2$

$end_point = true$

$r = \perp$

$oldr = root$

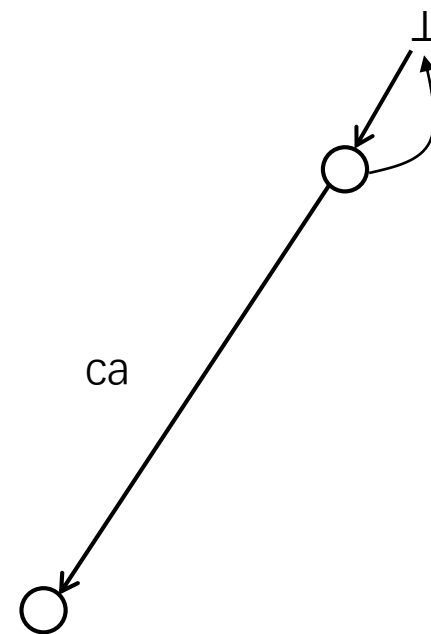
$i \leftarrow 2$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 
  
```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = root$

$k = 2$

$end_point = true$

$r = \perp$

$oldr = root$

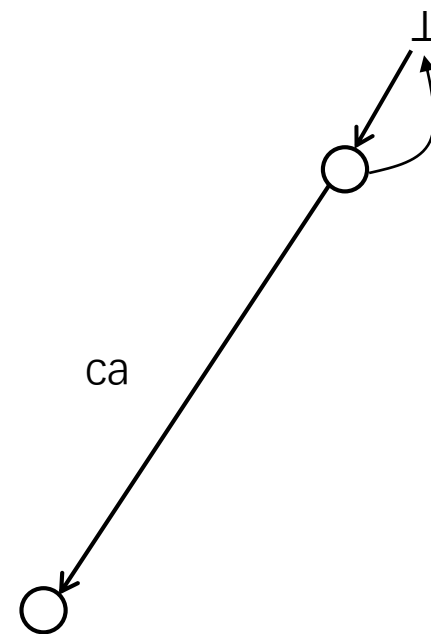
$(end_point, r) \leftarrow test_and_split(root, (2, 1), a)$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
    
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = root$
 $T^2 = ca$ $k = 2$
 $i = 2$ $p = 1$
 $s = root$ $t = a$
 $k = 2$
 $end_point = true$
 $r = \perp$
 $oldr = root$

return (false, *root*)

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

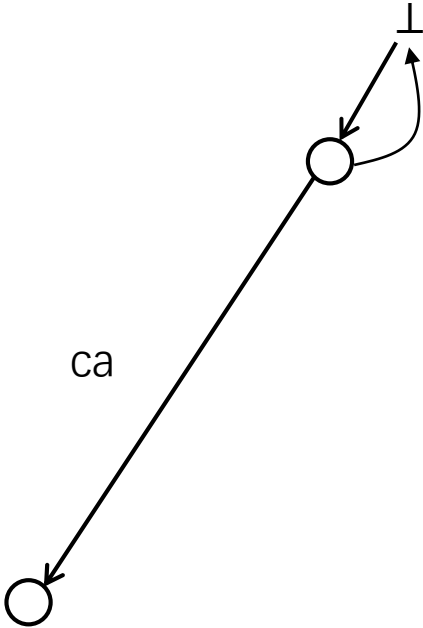
else

if there is no t -transition from s then return (false, s)

else return (true, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = root$

$k = 2$

$end_point = true$

$r = \perp$

$oldr = root$

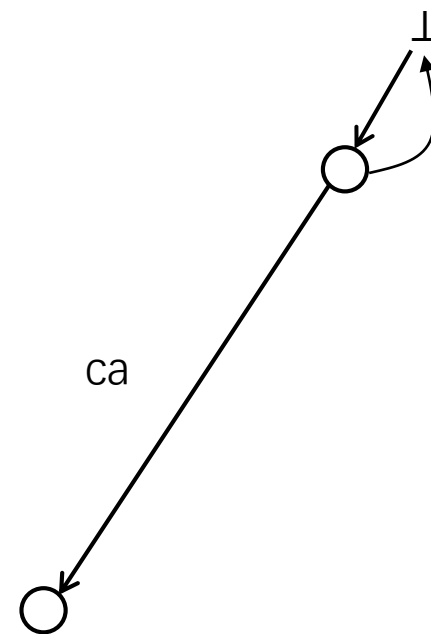
$(end_point, r) \leftarrow (false, root)$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
    
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	$\overline{c\bar{a}}$

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = root$

$k = 2$

$end_point = false$

$r = root$

$oldr = root$

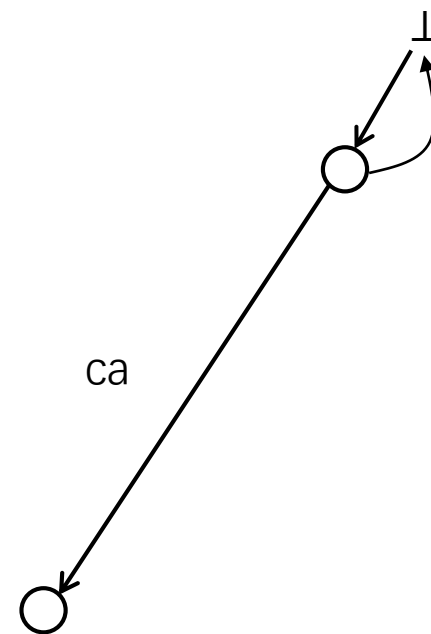
oldr \leftarrow *root*

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(root, (1, \infty))$	$\overline{c\bar{a}}$

f'	
input	output
<i>root</i>	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
oldr  $\leftarrow$  root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i - 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^1) \rightarrow ***STree***(T^2)

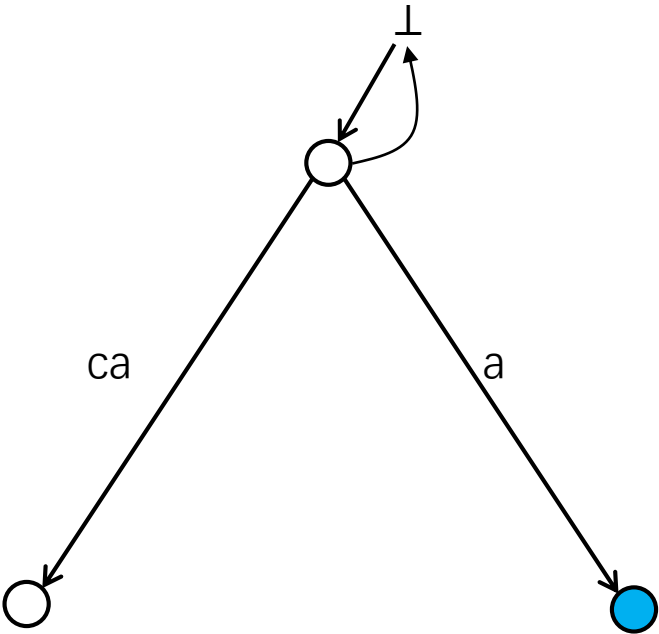
$T^1 = c$
 $T^2 = ca$
 $i = 2$
 $s = root$
 $k = 2$
 $end_point = false$
 $r = root$
 $oldr = root$

create r'

$i \leftarrow i + 1;$
 $(s, (k, i - 1))$ is the canonical reference pair for the active point;
 $(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$
 $oldr \leftarrow root;$
while $end_point == false$ **do**
 create new transition $g'(r, (i, \infty)) = r'$ where r' is a new state;
 if $oldr \neq root$ **then** create new suffix link $f'(oldr) = r;$
 $oldr \leftarrow r;$
 $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$
 $(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$
if $oldr \neq root$ **then** create new suffix link $f'(oldr) = s;$
 $(s, k) \leftarrow canonize(s, (k, i));$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\overline{a}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \text{root}$

$k = 2$

$\text{end_point} = \text{false}$

$r = \text{root}$

$\text{oldr} = \text{root}$

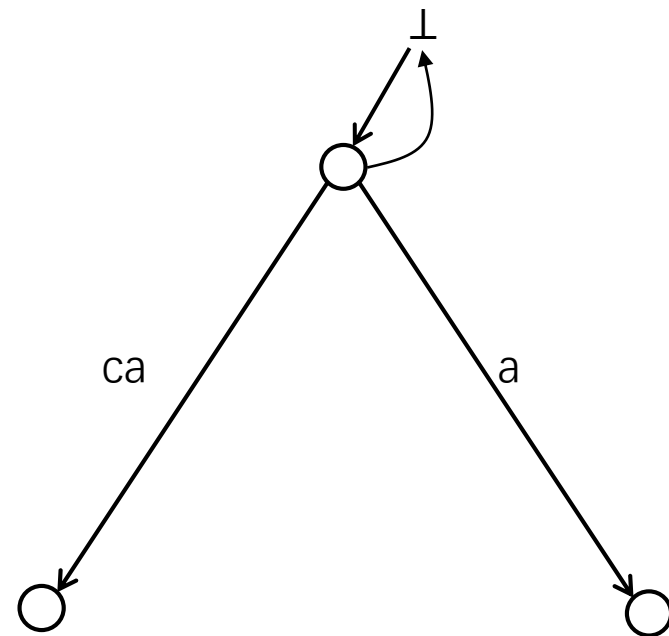
oldr \leftarrow *root*

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(\text{root}, (1, \infty))$	$\overline{c\bar{a}}$
$(\text{root}, (2, \infty))$	\bar{a}

f'	
input	output
<i>root</i>	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(\text{oldr}) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i - 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(\text{oldr}) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = root$

$k = 2$

$end_point = false$

$r = root$

$oldr = root$

$(s, k) \leftarrow canonize(\perp, (2, 1))$

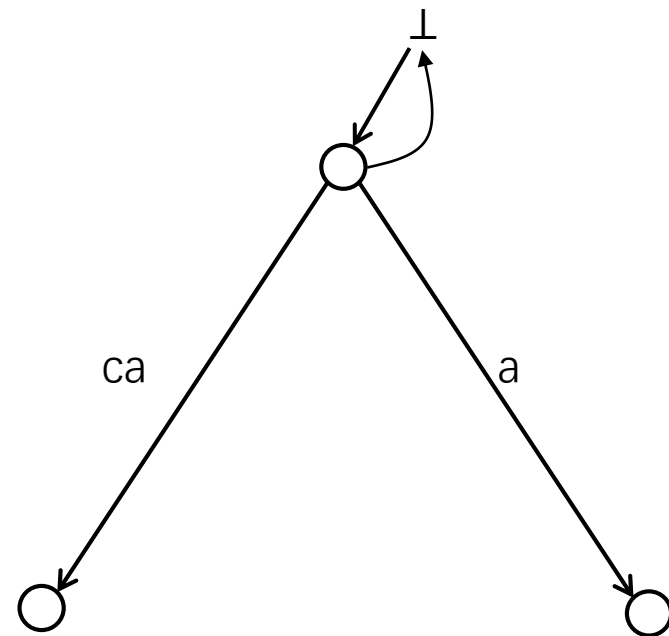
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 

```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = \perp$
 $T^2 = ca$ $k = 2$
 $i = 2$ $p = 1$
 $s = root$
 $k = 2$
 $end_point = false$
 $r = root$
 $oldr = root$

return (\perp , 2)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

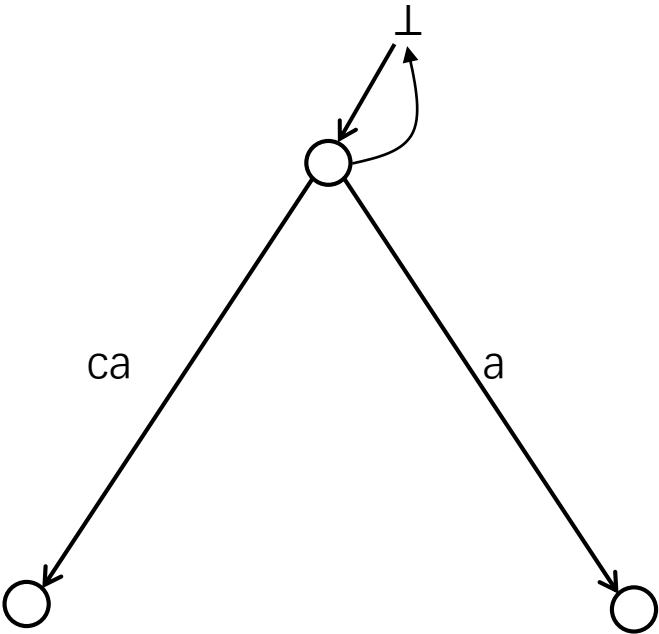
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \text{root}$

$k = 2$

$\text{end_point} = \text{false}$

$r = \text{root}$

$\text{oldr} = \text{root}$

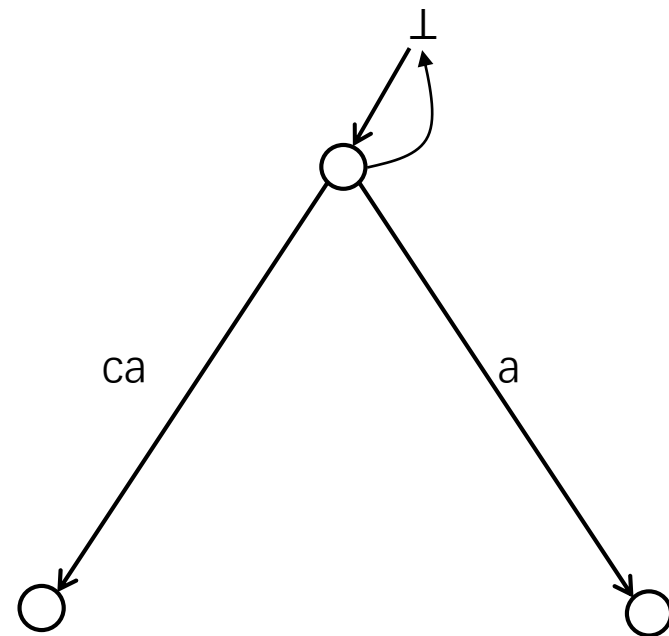
$(s, k) \leftarrow (\perp, 2)$

g'	
input	output
$(\perp, (-i, -i))$	root
$(\text{root}, (1, \infty))$	\overline{ca}
$(\text{root}, (2, \infty))$	\bar{a}

f'	
input	output
root	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(\text{end\_point}, r) \leftarrow \text{test\_and\_split}(s, (k, i - 1), t_i);$ 
 $\text{oldr} \leftarrow \text{root};$ 
while  $\text{end\_point} == \text{false}$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $\text{oldr} \neq \text{root}$  then create new suffix link  $f'(\text{oldr}) = r;$ 
     $\text{oldr} \leftarrow r;$ 
     $(s, k) \leftarrow \text{canonize}(f'(s), (k, i - 1));$ 
     $(\text{end\_point}, r) \leftarrow \text{test\_and\_split}(s, (k, i - 1), t_i);$ 
if  $\text{oldr} \neq \text{root}$  then create new suffix link  $f'(\text{oldr}) = s;$ 
 $(s, k) \leftarrow \text{canonize}(s, (k, i));$ 
  
```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \perp$

$k = 2$

$end_point = false$

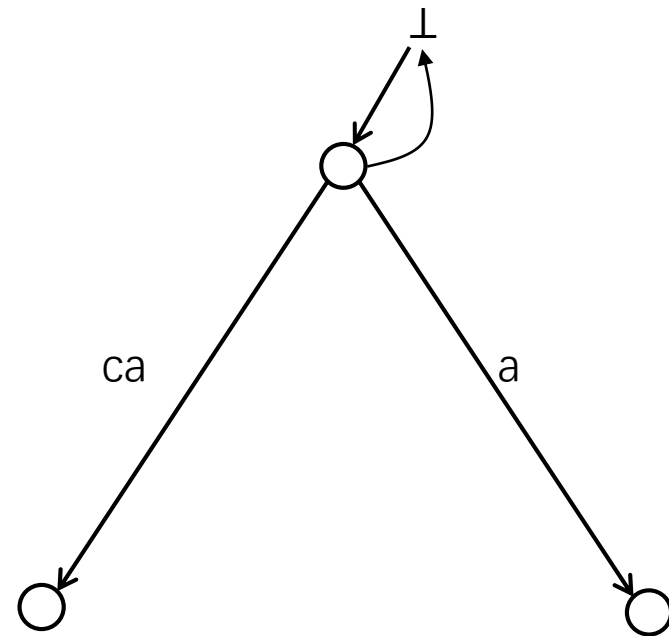
$r = root$

$oldr = root$

$(end_point, r) \leftarrow test_and_split(\perp, (2, 1), a)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
  
```

STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = \perp$
 $T^2 = ca$ $k = 2$
 $i = 2$ $p = 1$
 $s = \perp$ $t = a$
 $k = 2$
 $end_point = false$
 $r = root$
 $oldr = root$

return (true, \perp)

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

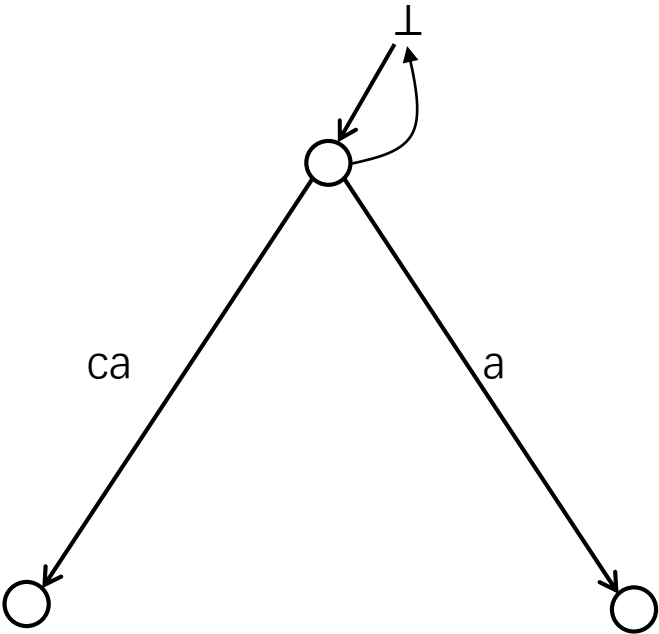
else

if there is no t -transition from s then return (false, s)

else return (true, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	$\overline{c\bar{a}}$
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \perp$

$k = 2$

$end_point = false$

$r = root$

$oldr = root$

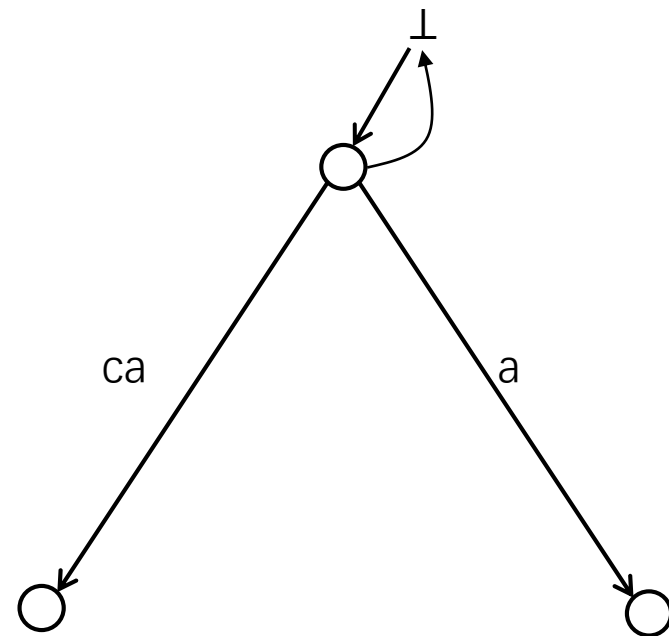
$(end_point, r) \leftarrow (true, \perp)$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
    
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	$\bar{c}\bar{a}$
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \perp$

$k = 2$

$end_point = true$

$r = \perp$

$oldr = root$

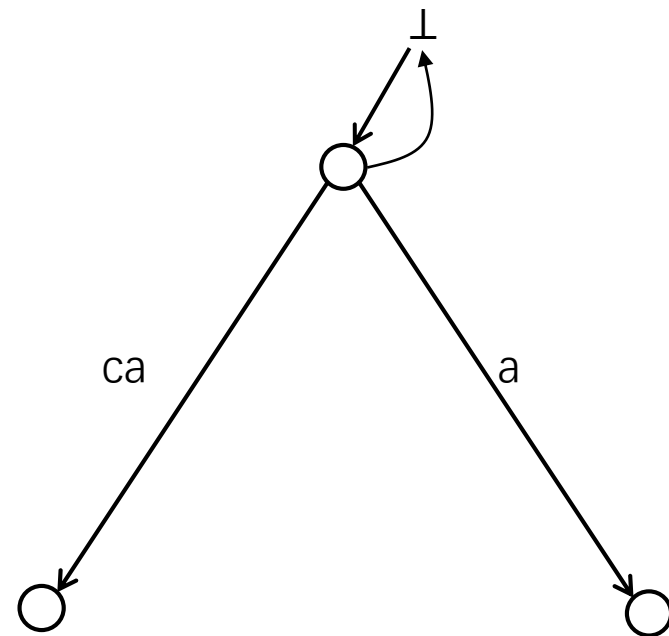
$(s, k) \leftarrow canonize(\perp, (2, 2))$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
  
```



$$STree(T^1) \rightarrow STree(T^2)$$
$$T^1 = c \quad S = \perp$$
$$T^2 = ca \quad k = 2$$
$$i = 2 \qquad p = 2$$
$$S = \perp$$
$$k = 2$$

```
end_point = true
```

$$r = \perp$$
$$oldr = root$$
$$(k', p') = (-2, -2)$$
 $s' = root$
$$\mathit{canonize}(s, (k, p))$$

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

```
while  $p' - k' \leq p - k$  do
```

$$k \leftarrow k + p' - k' + 1;$$
$$S \leftarrow S';$$

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

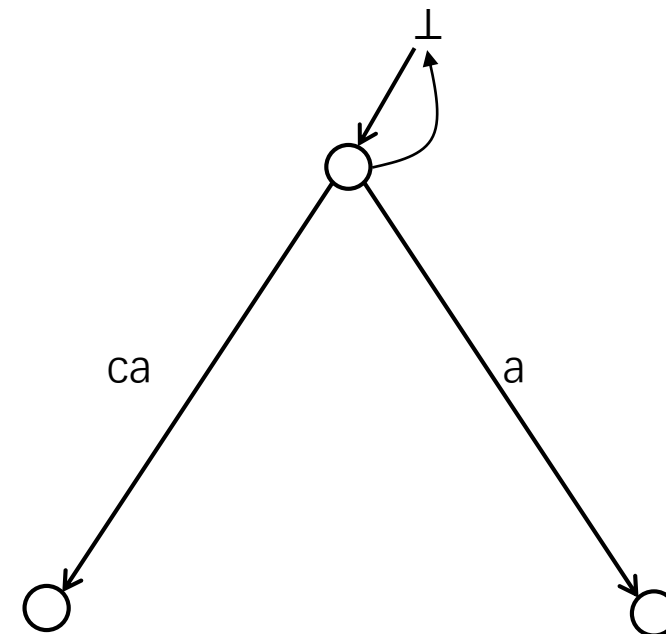
```

return (s, k).

```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\overline{a}

f'	
input	output
$root$	\perp



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = \perp$ $s' = root$
 $T^2 = ca$ $k = 2$ $k' = -2$
 $i = 2$ $p = 2$ $p' = -2$
 $s = \perp$
 $k = 2$
 $end_point = true$
 $r = \perp$
 $oldr = root$

$k \leftarrow 3$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

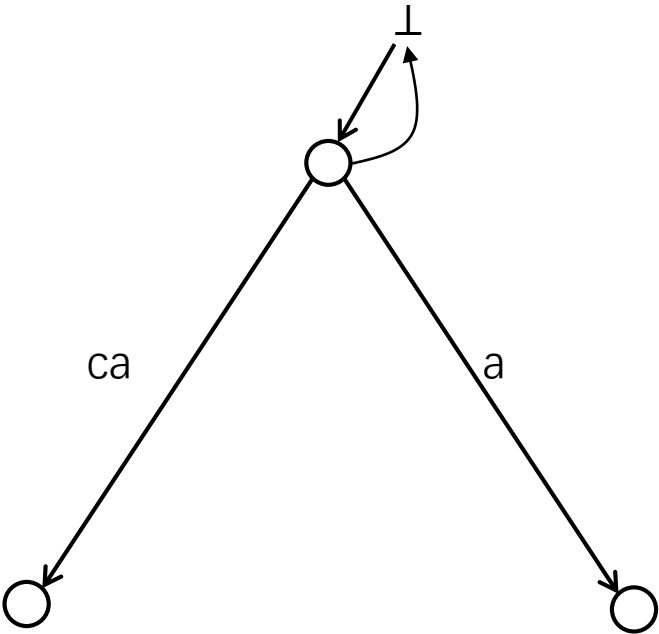
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = \perp$ $s' = root$
 $T^2 = ca$ $k = 3$ $k' = -2$
 $i = 2$ $p = 2$ $p' = -2$
 $s = \perp$
 $k = 2$
 $end_point = true$
 $r = \perp$
 $oldr = root$

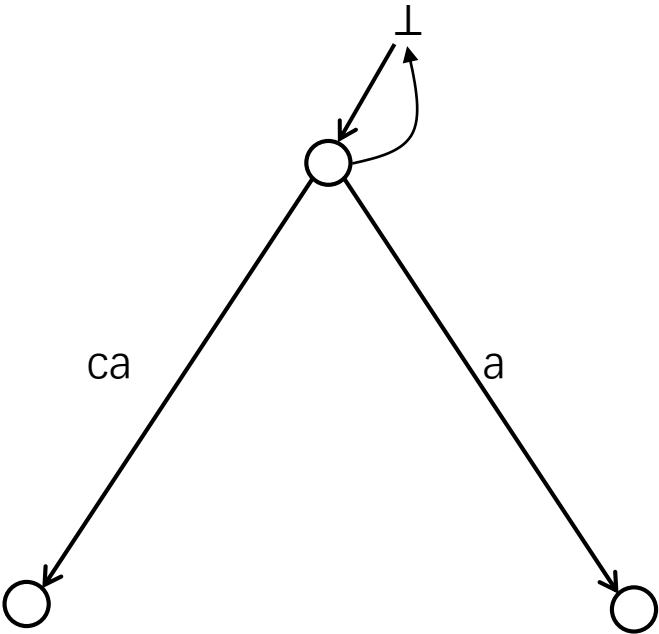
$s \leftarrow root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$ $s = root$ $s' = root$
 $T^2 = ca$ $k = 3$ $k' = -2$
 $i = 2$ $p = 2$ $p' = -2$
 $s = \perp$
 $k = 2$
 $end_point = true$
 $r = \perp$
 $oldr = root$

return (*root*, 3)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

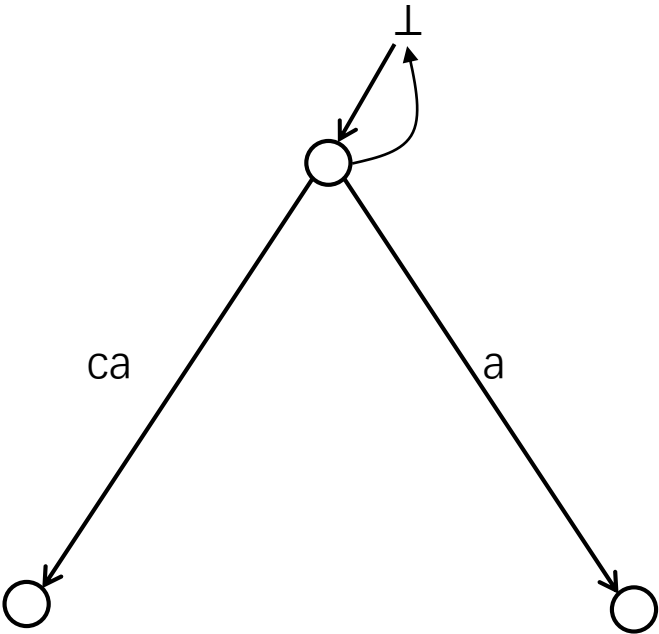
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = \perp$

$k = 2$

$end_point = true$

$r = \perp$

$oldr = root$

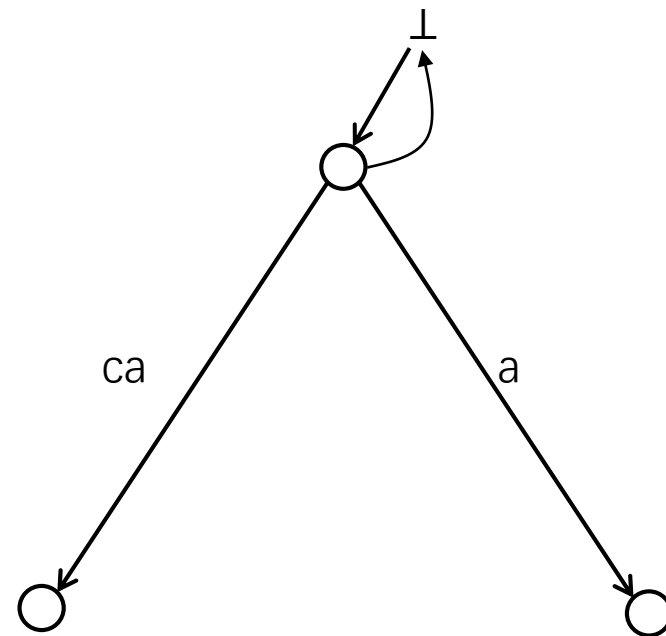
$(s, k) \leftarrow (root, 3)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 
  
```



STree(T^1) \rightarrow ***STree***(T^2)

$T^1 = c$

$T^2 = ca$

$i = 2$

$s = root$

$k = 3$

$end_point = true$

$r = \perp$

$oldr = root$

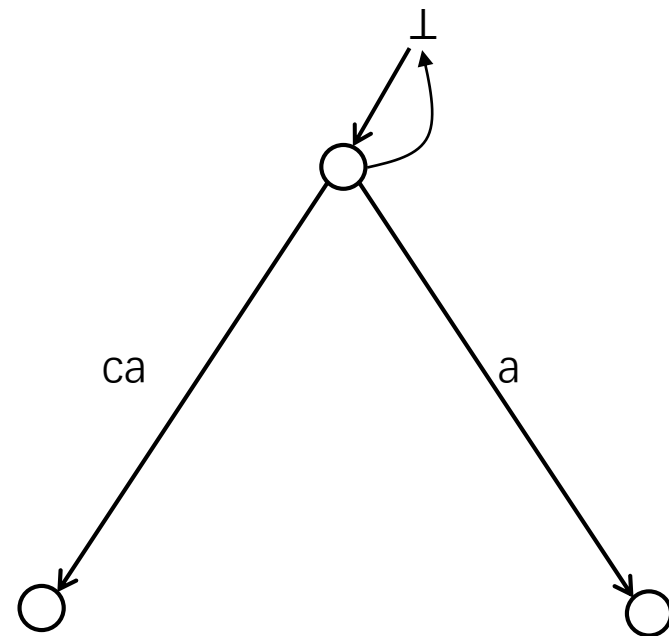
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{ca}
$(root, (2, \infty))$	\bar{a}

f'	
input	output
$root$	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));

```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 2$

$s = root$

$k = 3$

$end_point = true$

$r = \perp$

$oldr = root$

$i \leftarrow 3$

$i \leftarrow i + 1;$

$(s, (k, i - 1))$ is the canonical reference pair for the active point;

$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

$oldr \leftarrow root;$

while $end_point == false$ **do**

create new transition $g'(r, (i, \infty)) = r'$ where r' is a new state;

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = r;$

$oldr \leftarrow r;$

$(s, k) \leftarrow canonize(f'(s), (k, i - 1));$

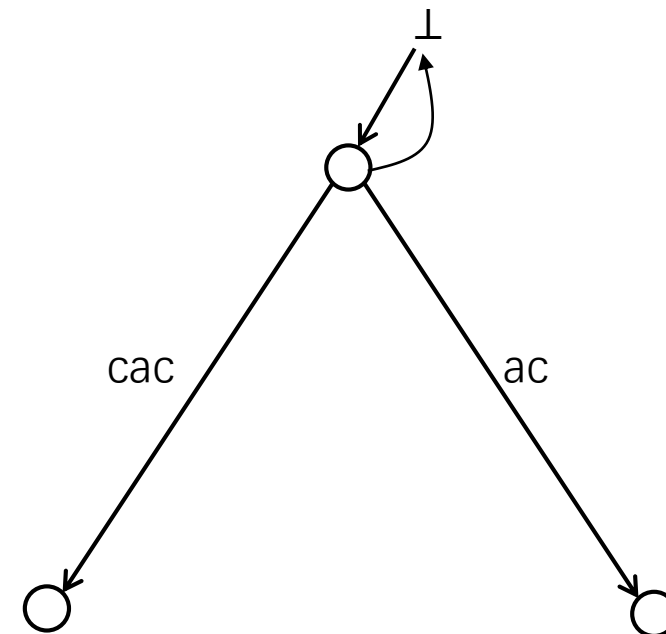
$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = s;$

$(s, k) \leftarrow canonize(s, (k, i));$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = \perp$

$oldr = root$

$(end_point, r) \leftarrow test_and_split(root, (3, 2), c)$

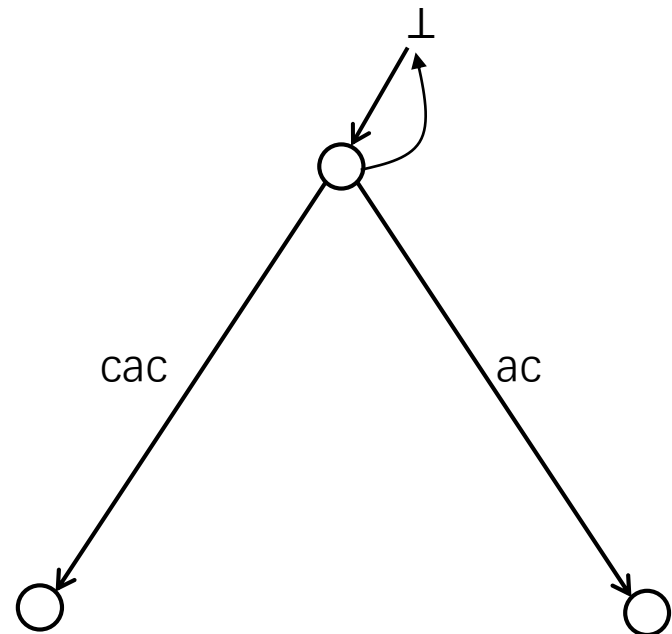
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$ $s = root$
 $T^3 = cac$ $k = 3$
 $i = 3$ $p = 2$
 $s = root$ $t = c$
 $k = 3$
 $end_point = true$
 $r = \perp$
 $oldr = root$

return (true, *root*)

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

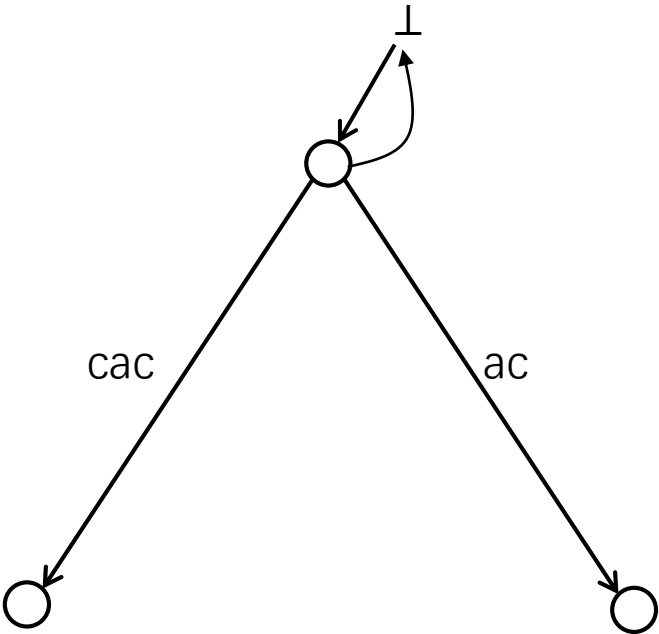
else

if there is no t -transition from s then return (false, s)

else return (true, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp



$STree(T^2) \rightarrow STree(T^3)$

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = \perp$

$oldr = root$

$(end_point, r) \leftarrow (true, root)$

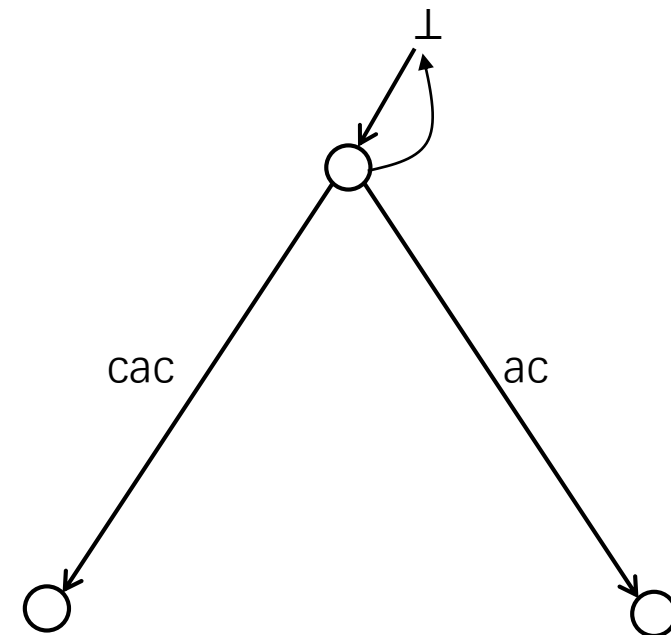
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

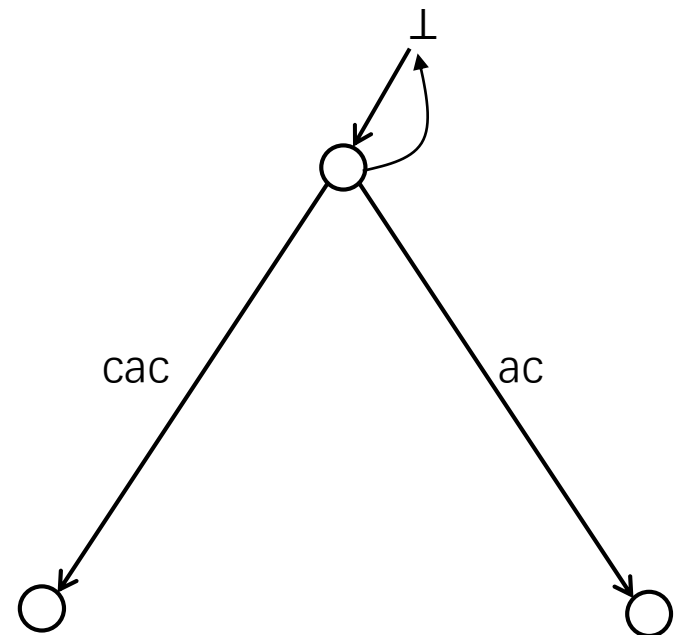
oldr \leftarrow *root*

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
<i>root</i>	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
oldr  $\leftarrow$  root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i - 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(s, k) \leftarrow canonize(root, (3, 3))$

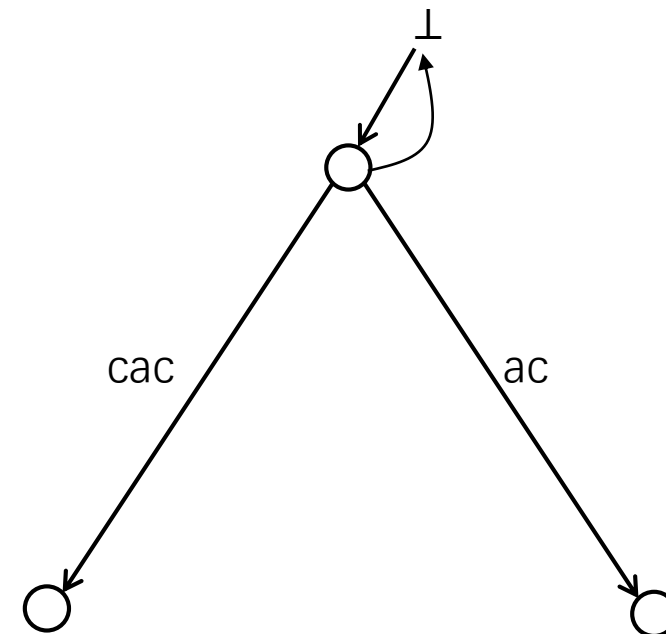
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$ $s = root$
 $T^3 = cac$ $k = 3$
 $i = 3$ $p = 3$
 $s = root$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$

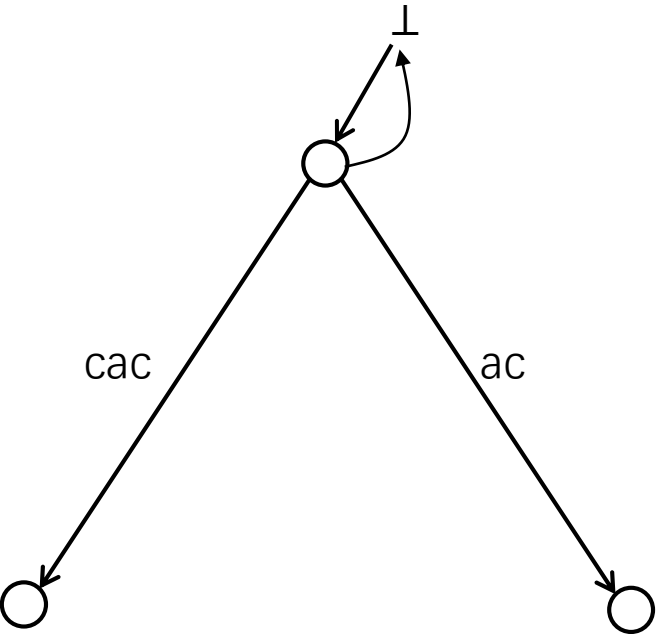
$(k', p') = (1, \infty)$
 $s' = \overline{cac}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).



***S*Tree(T^2) \rightarrow *S*Tree(T^3)**

$T^2 = ca$ $s = \text{root}$ $s' = \overline{cac}$

$T^3 = cac$ $k = 3$ $k' = 1$

$i = 3$ $p = 3$ $p' = \infty$

$s = \text{root}$

$k = 3$

$\text{end_point} = \text{true}$

$r = \text{root}$

$\text{oldr} = \text{root}$

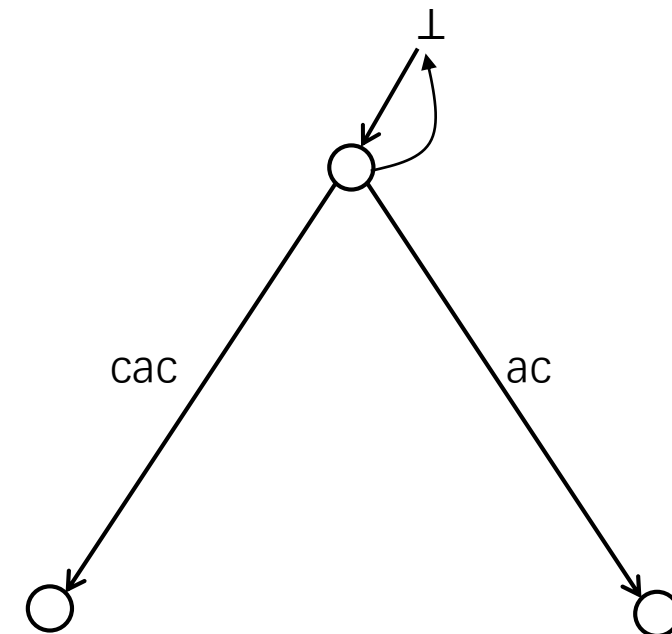
return ($\text{root}, 3$)

g'	
input	output
$(\perp, (-i, -i))$	root
$(\text{root}, (1, \infty))$	\overline{cac}
$(\text{root}, (2, \infty))$	\overline{ac}

f'	
input	output
root	\perp

***canonize*($s, (k, p)$)**

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ **do**
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ **then**
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(s, k) \leftarrow (root, 3)$

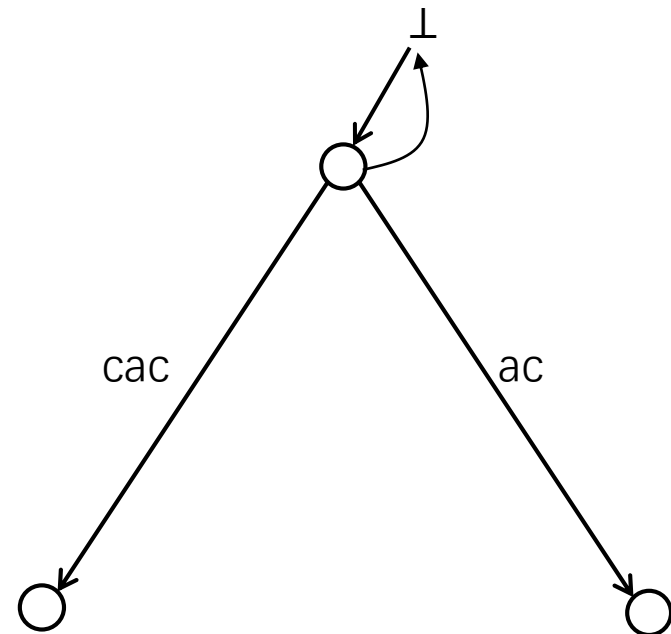
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 

```



STree(T^2) \rightarrow ***STree***(T^3)

$T^2 = ca$

$T^3 = cac$

$i = 3$

$s = root$

$k = 3$

$end_point = true$

$r = root$

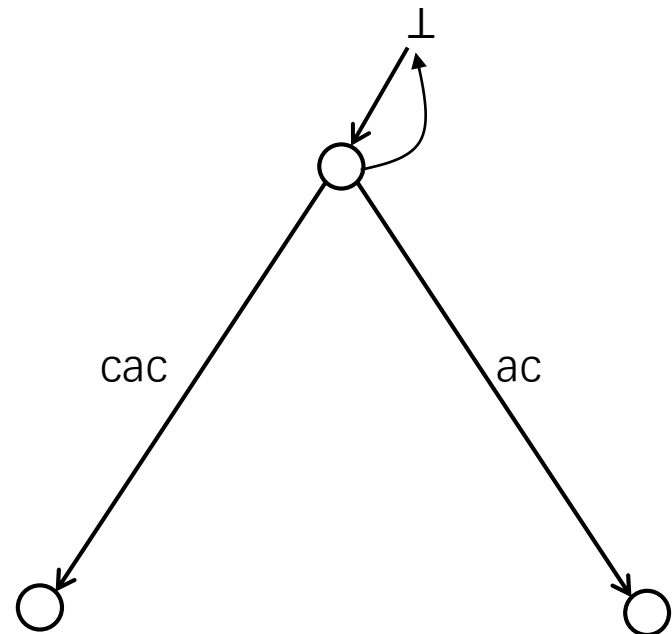
$oldr = root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cac}
$(root, (2, \infty))$	\overline{ac}

f'	
input	output
$root$	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^3) \rightarrow ***STree***(T^4)

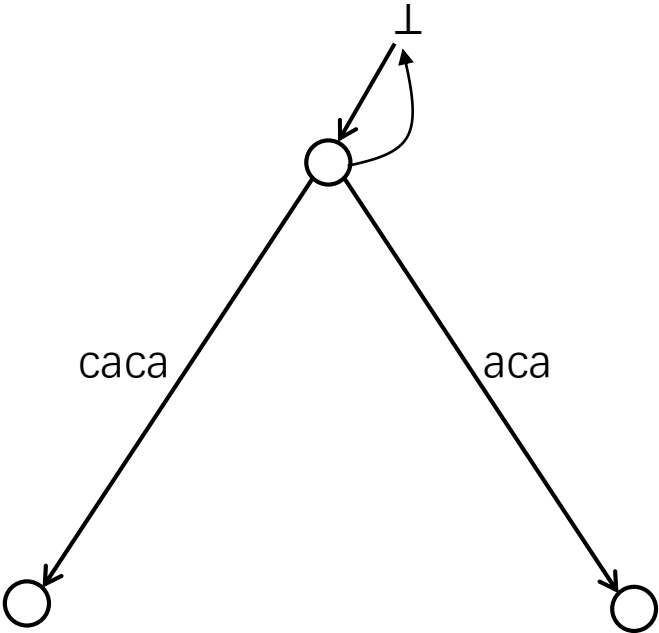
$T^3 = cac$
 $T^4 = caca$
 $i = 3$
 $s = root$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$

$i \leftarrow 4$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

$i \leftarrow i + 1;$
 $(s, (k, i - 1))$ is the canonical reference pair for the active point;
 $(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$
 $oldr \leftarrow root;$
while $end_point == false$ **do**
 create new transition $g'(r, (i, \infty)) = r'$ where r' is a new state;
 if $oldr \neq root$ **then** create new suffix link $f'(oldr) = r;$
 $oldr \leftarrow r;$
 $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$
 $(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$
if $oldr \neq root$ **then** create new suffix link $f'(oldr) = s;$
 $(s, k) \leftarrow canonize(s, (k, i));$



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(end_point, r) \leftarrow test_and_split(root, (3, 3), a)$

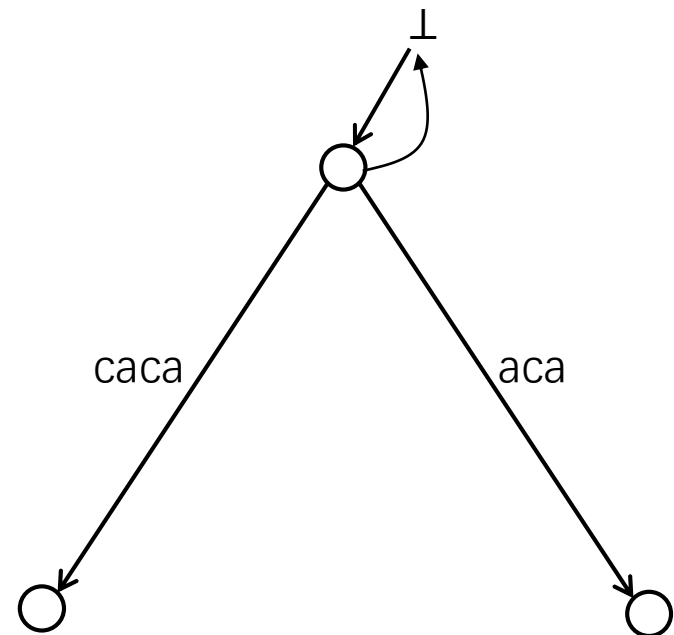
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$ $s = root$
 $T^4 = caca$ $k = 3$
 $i = 4$ $p = 3$
 $s = root$ $t = a$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$
 $(k', p') = (1, \infty)$
 $s' = \overline{caca}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

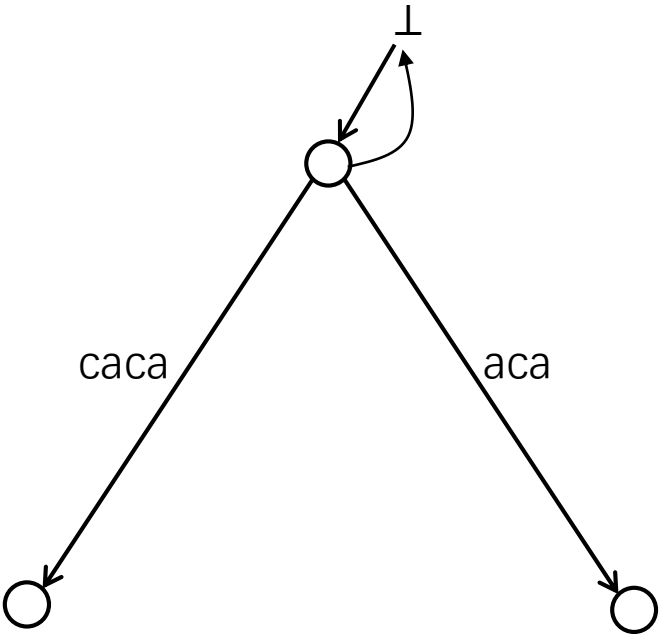
where r is a new state;

return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$ $s = root$ $s' = \overline{cac\bar{a}}$
 $T^4 = caca$ $k = 3$ $k' = 1$
 $i = 4$ $p = 3$ $p' = \infty$
 $s = root$ $t = a$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$

return (*true*, *root*)

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (*true*, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (*false*, r)

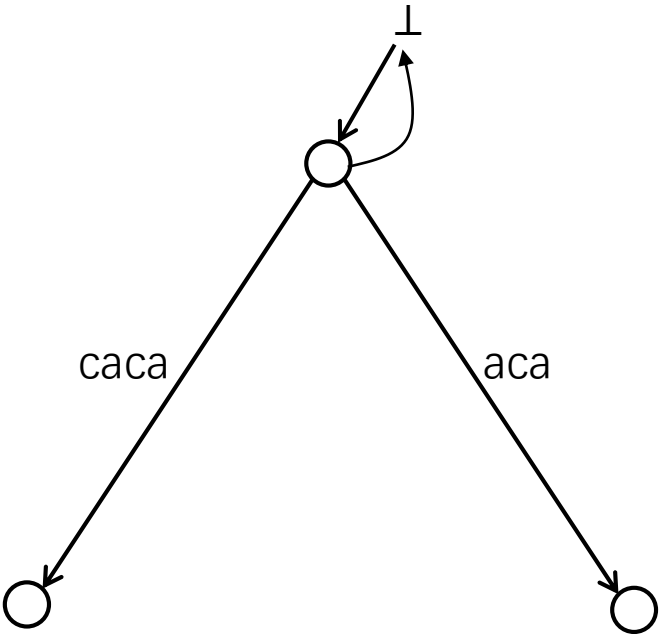
else

if there is no t -transition from s then return (*false*, s)

else return (*true*, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	$\overline{cac\bar{a}}$
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(end_point, r) \leftarrow (true, root)$

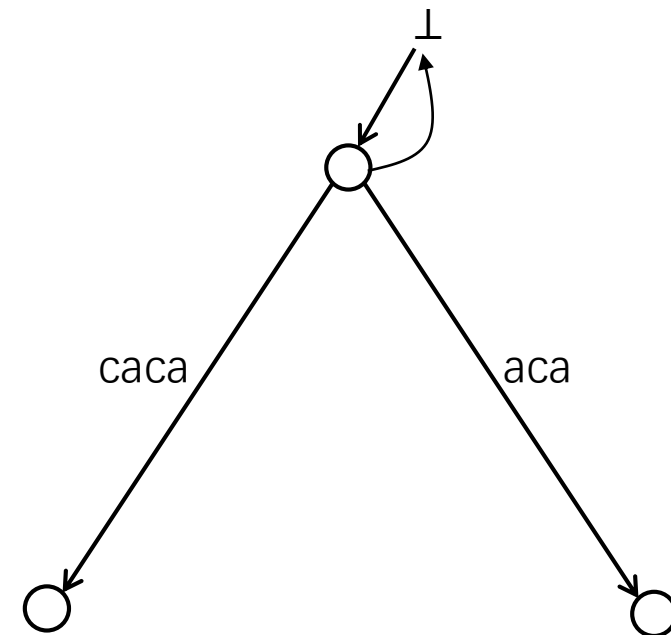
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



$STree(T^3) \rightarrow STree(T^4)$

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$oldr \leftarrow root$

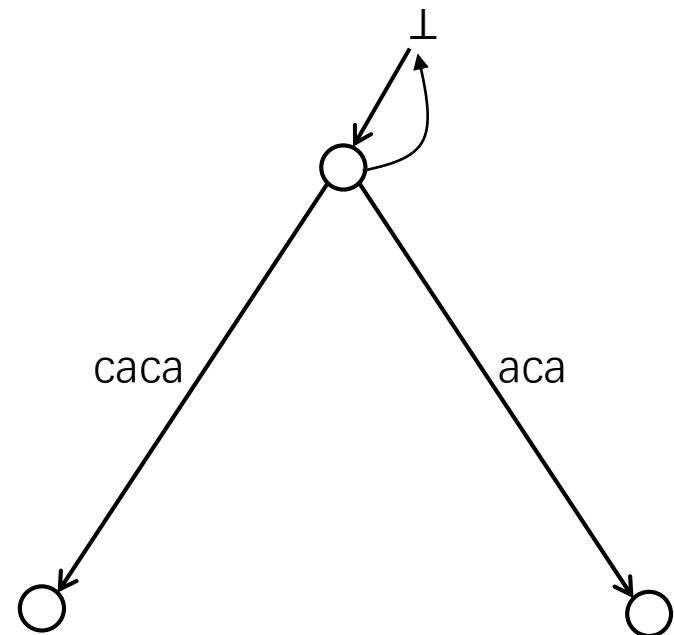
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize(f'(s), (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(s, k) \leftarrow canonize(root, (3, 4))$

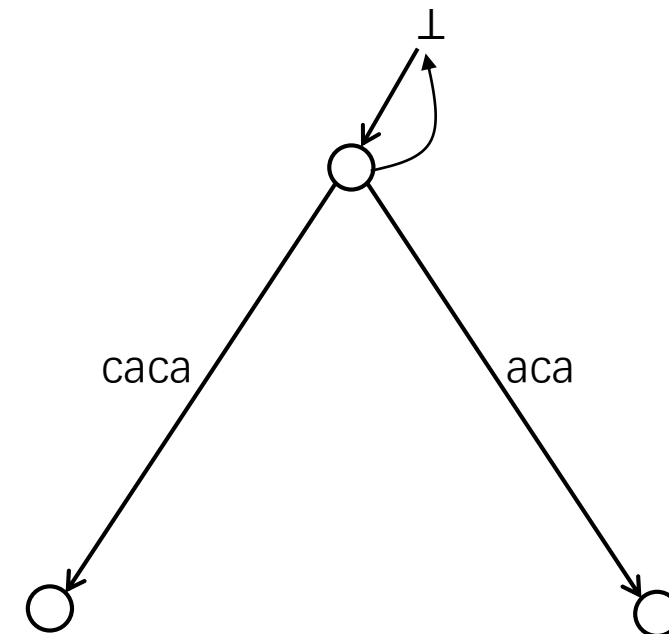
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize(f'(s), (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac \quad s = root$
 $T^4 = caca \quad k = 3$
 $i = 4 \quad p = 4$
 $s = root$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$

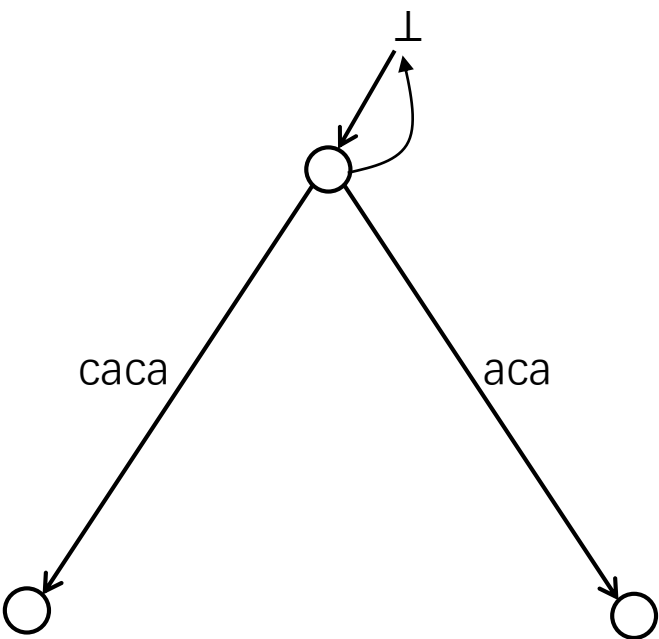
$(k', p') = (1, \infty)$
 $s' = \overline{caca}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).



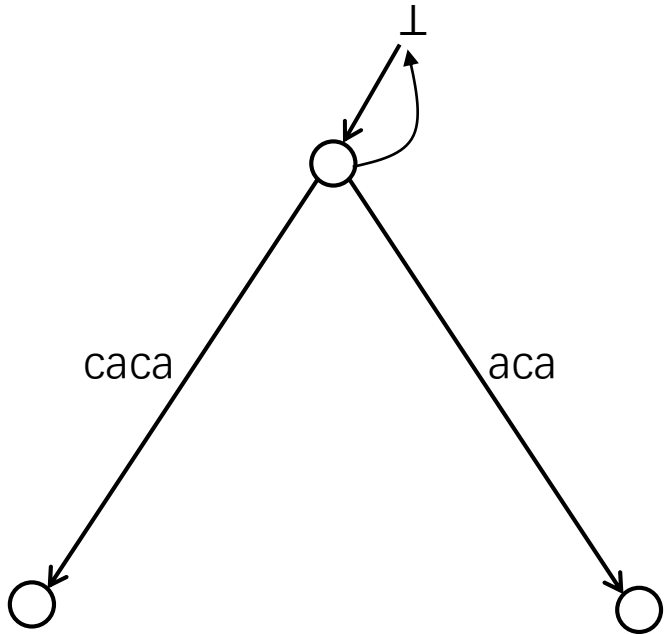
STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$ $s = root$ $s' = \overline{caca}$
 $T^4 = caca$ $k = 3$ $k' = 1$
 $i = 4$ $p = 4$ $p' = \infty$
 $s = root$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$

return (*root*, 3)

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
<i>root</i>	\perp



canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ **do**
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ **then**
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).

STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

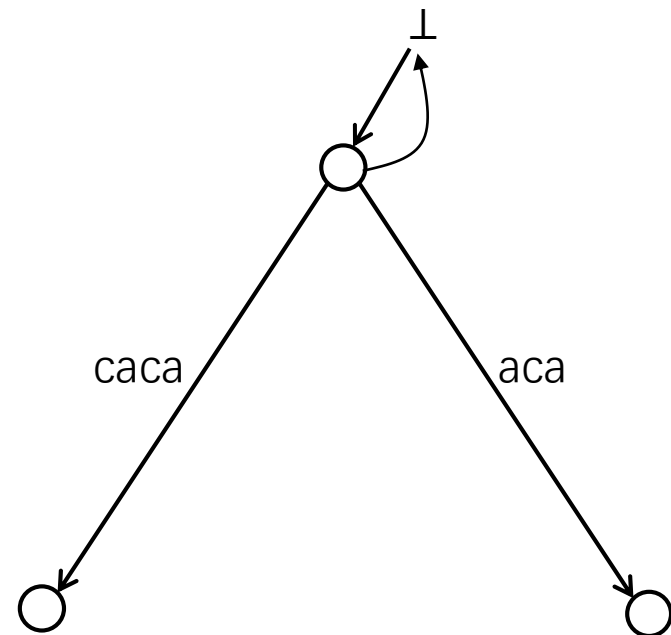
$(s, k) \leftarrow (root, 3)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 
  
```



STree(T^3) \rightarrow ***STree***(T^4)

$T^3 = cac$

$T^4 = caca$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

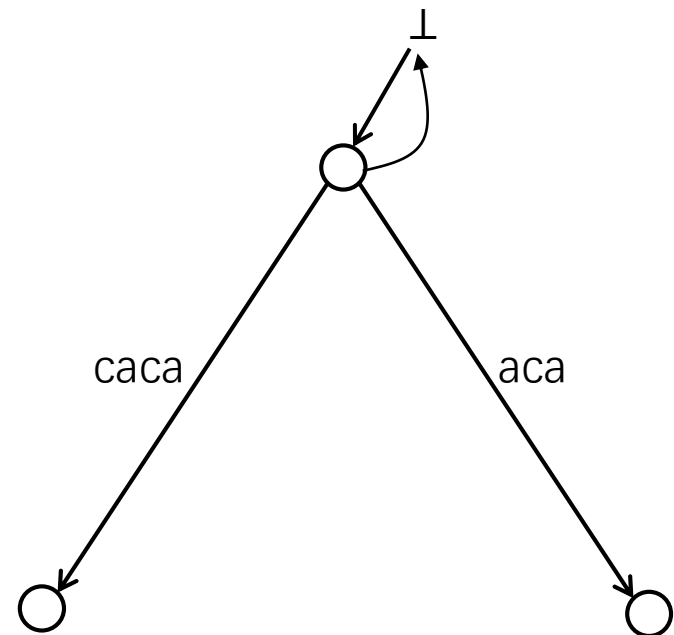
$oldr = root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{caca}
$(root, (2, \infty))$	\overline{aca}

f'	
input	output
$root$	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 4$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

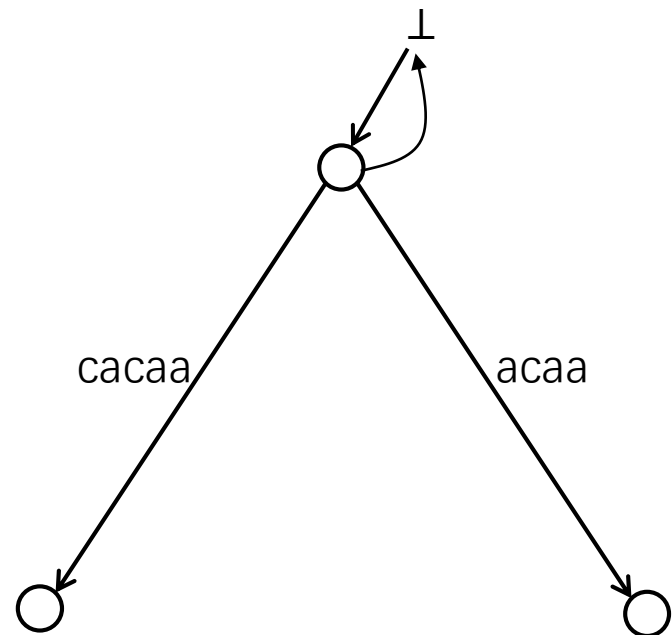
$i \leftarrow 5$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}

f'	
input	output
$root$	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
oldr  $\leftarrow$  root ;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r ;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i − 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i − 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(end_point, r) \leftarrow test_and_split(root, (3, 4), a)$

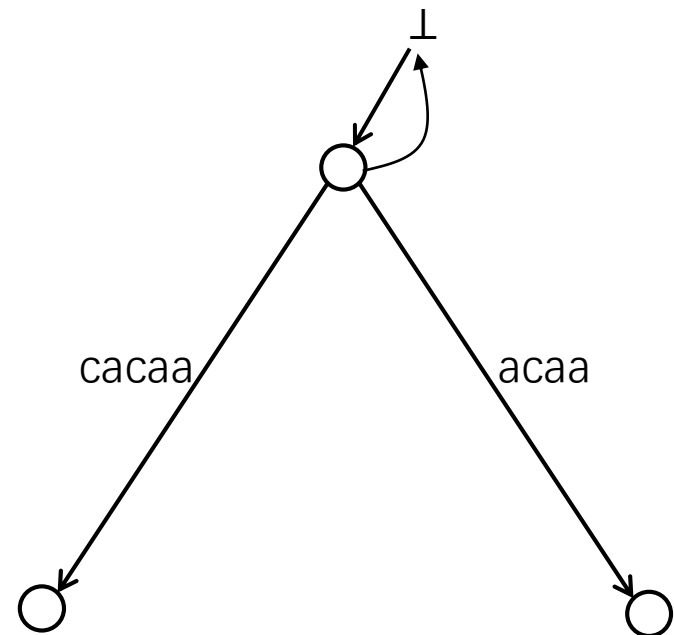
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$
 $T^5 = cacia$ $k = 3$
 $i = 5$ $p = 4$
 $s = root$ $t = a$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$
 $(k', p') = (1, \infty)$
 $s' = \overline{cacia}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, \infty))$	\overline{cacia}
$(root, (2, \infty))$	\overline{acia}

f'	
input	output
$root$	\perp

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

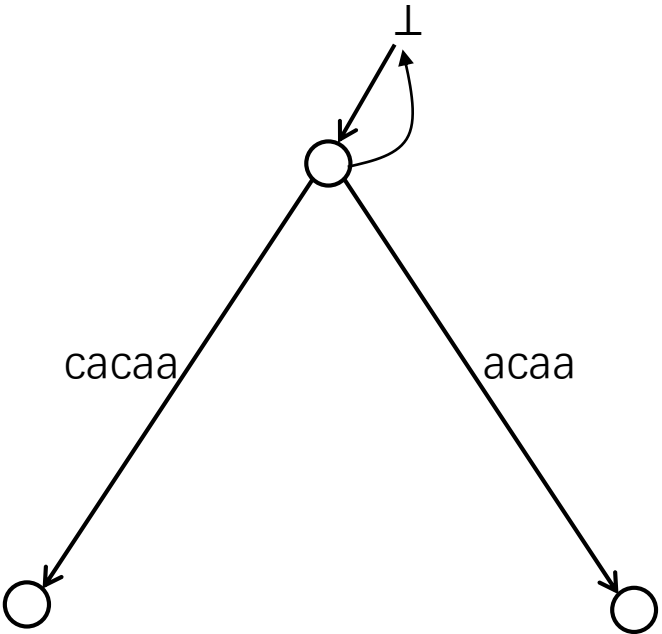
where r is a new state;

return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$ $s' = \overline{cacaa}$
 $T^5 = cacaa$ $k = 3$ $k' = 1$
 $i = 5$ $p = 4$ $p' = \infty$
 $s = root$ $t = a$
 $k = 3$
 $end_point = true$
 $r = root$
 $oldr = root$
 $g'(root, (1,2)) = r = \overline{ca}$
 $g'(\overline{ca}, (3, \infty)) = \overline{cacaa}$

test_and_split($s, (k, p), t$)

if $k \leq p$ then

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

 if $t = t_{k'+p-k+1}$ then return (true, s)

 else

 replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

 where r is a new state;

 return (false, r)

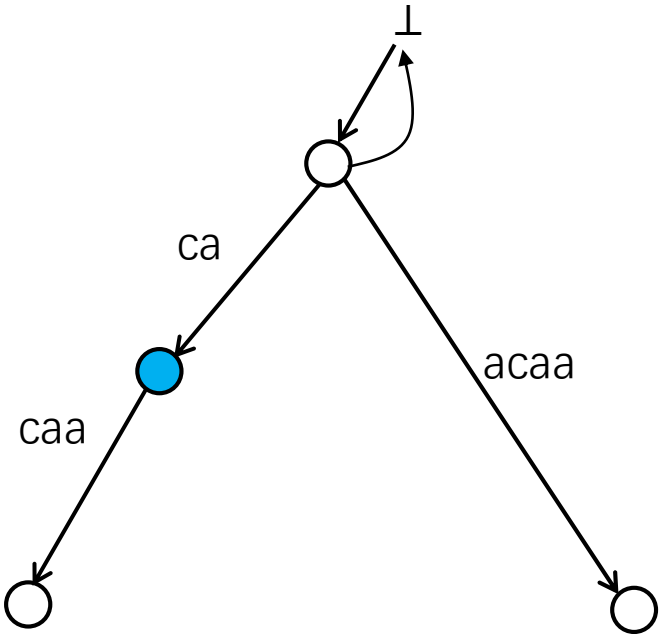
else

 if there is no t -transition from s then return (false, s)

 else return (true, s)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}

f'	
input	output
$root$	\perp



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = \text{root}$ $s' = \overline{cacaa}$
 $T^5 = cacaa$ $k = 3$ $k' = 1$
 $i = 5$ $p = 4$ $p' = \infty$
 $s = \text{root}$ $t = a$ $r = \overline{ca}$
 $k = 3$
 $\text{end_point} = \text{true}$
 $r = \text{root}$
 $\text{oldr} = \text{root}$

g'	
input	output
$(\perp, (-i, -i))$	root
$(\text{root}, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(\text{root}, (2, \infty))$	\overline{acaa}

f'	
input	output
root	\perp

return (false, \overline{ca})

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s$,

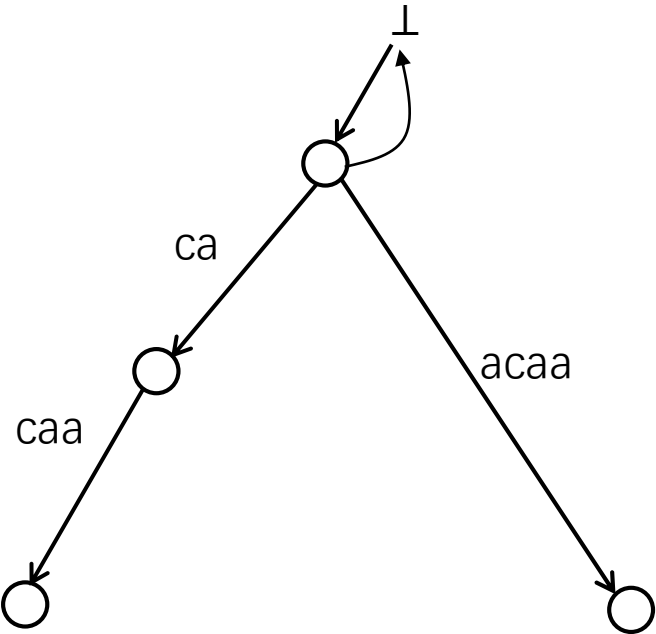
where r is a new state;

return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)



$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = true$

$r = root$

$oldr = root$

$(end_point, r) \leftarrow (false, \overline{ca})$

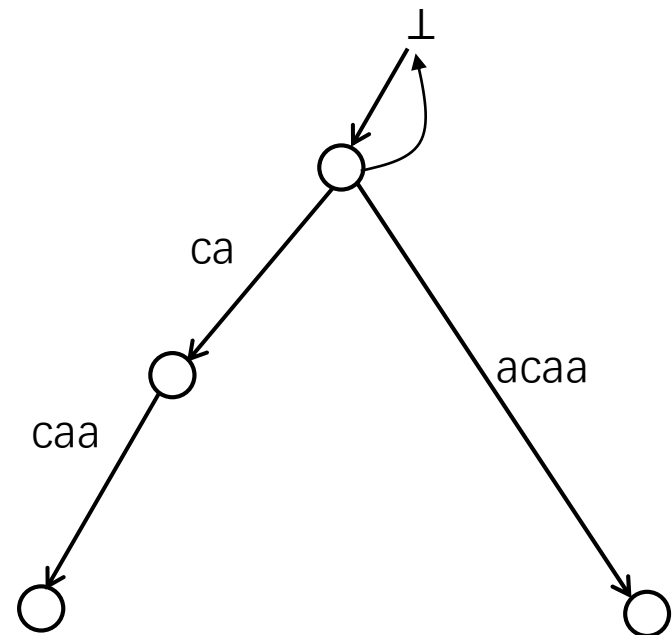
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = false$

$r = \overline{c\bar{a}}$

$oldr = root$

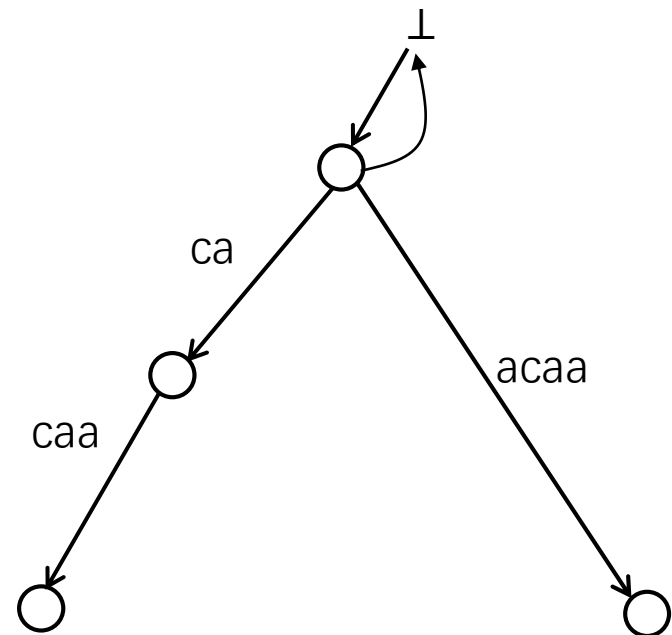
oldr \leftarrow *root*

g'	
input	output
$(\perp, (-i, -i))$	<i>root</i>
$(root, (1, 2))$	$\overline{c\bar{a}}$
$(\overline{c\bar{a}}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}

f'	
input	output
<i>root</i>	\perp

```

i  $\leftarrow$  i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
oldr  $\leftarrow$  root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr  $\neq$  root then create new suffix link  $f'(oldr) = r$ ;
    oldr  $\leftarrow$  r;
    (s, k)  $\leftarrow$  canonize( $f'(s)$ , (k, i - 1));
    (end_point, r)  $\leftarrow$  test_and_split(s, (k, i - 1), ti);
if oldr  $\neq$  root then create new suffix link  $f'(oldr) = s$ ;
(s, k)  $\leftarrow$  canonize(s, (k, i));
  
```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = false$

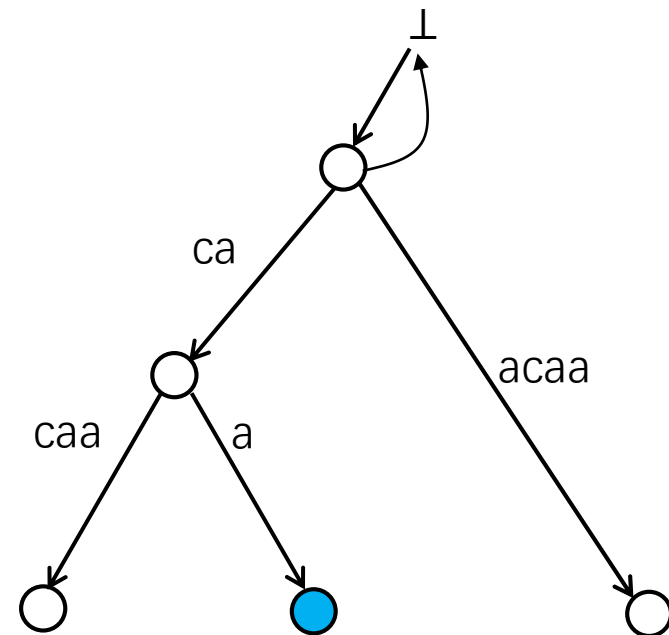
$r = \overline{c\bar{a}}$

$oldr = root$

create r'

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\overline{c\bar{a}}$
$(\overline{c\bar{a}}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{c\bar{a}}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
 $(s, k) \leftarrow canonize(s, (k, i));$ 

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacia$

$i = 5$

$s = root$

$k = 3$

$end_point = false$

$r = \overline{ca}$

$oldr = root$

$oldr \leftarrow \overline{ca}$

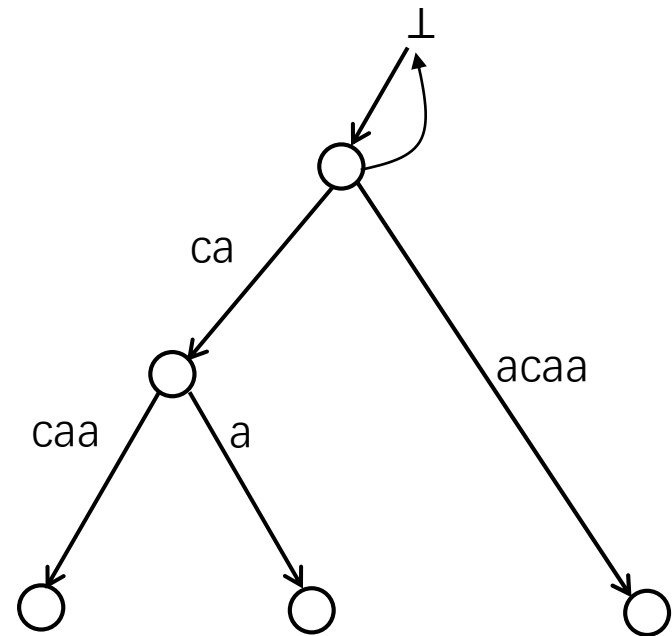
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = false$

$r = \overline{ca}$

$oldr = \overline{ca}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

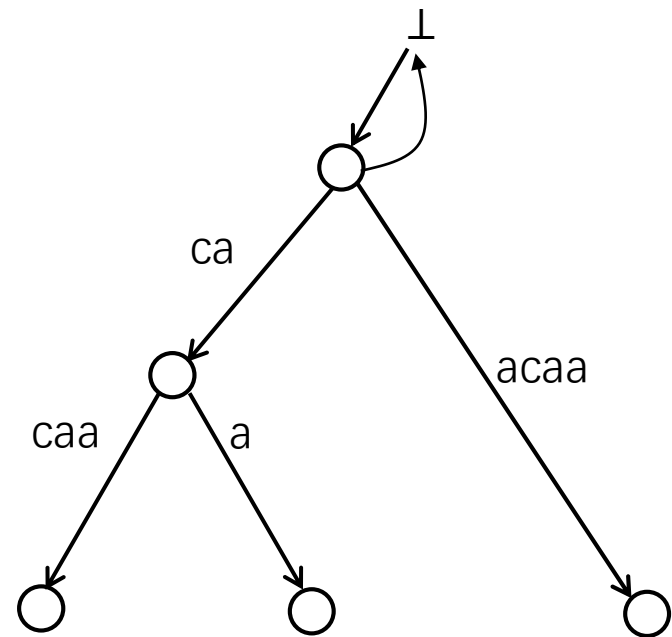
f'	
input	output
$root$	\perp

$(s, k) \leftarrow canonize(\perp, (3, 4))$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



$$STree(T^4) \rightarrow STree(T^5)$$
$$T^4 = caca \quad s = \perp$$
$$T^5 = caca a \quad k = 3$$
$$i = 5 \qquad p = 4$$
$$s = root$$
 $k = 3$

```
end_point = false
```

$$r = \overline{ca}$$
$$oldr = \overline{ca}$$
$$(k', p') = (-3, -3)$$
$$s' = root$$
$$\mathit{canonize}(s, (k, p))$$

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

```
while  $p' - k' \leq p - k$  do
```

$$k \leftarrow k + p' - k' + 1;$$
$$S \leftarrow S';$$

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

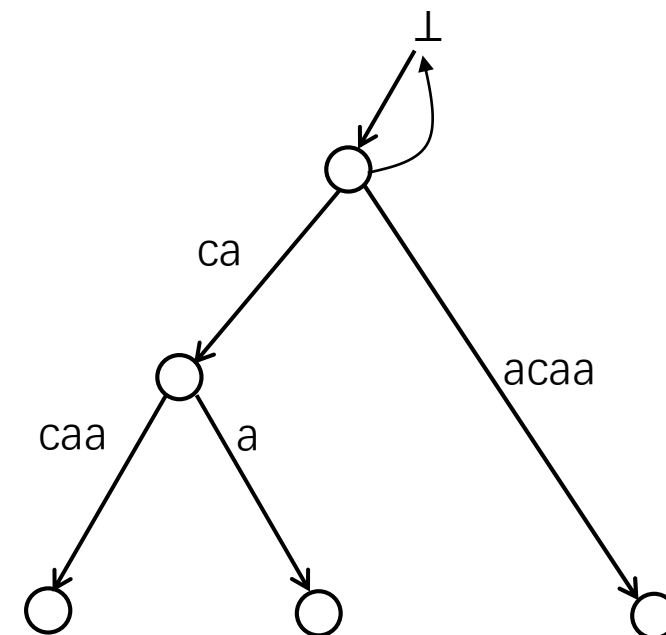
```

return (s, k).

```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = \perp$ $s' = root$
 $T^5 = cacaa$ $k = 3$ $k' = -3$
 $i = 5$ $p = 4$ $p' = -3$
 $s = root$
 $k = 3$
 $end_point = false$
 $r = \overline{ca}$
 $oldr = \overline{ca}$

$k \leftarrow 4$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp

canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

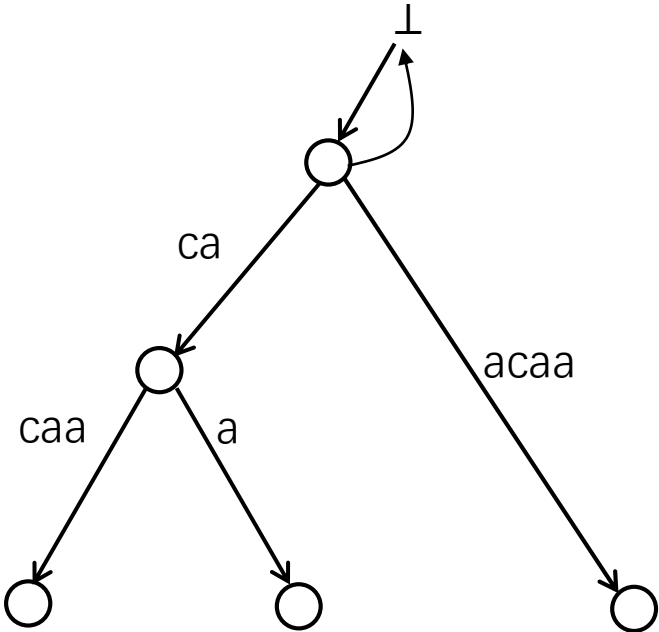
$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).



$$STree(T^4) \rightarrow STree(T^5)$$
$$T^4 = caca \quad s = \perp \quad s' = root$$
$$T^5 = caca a \quad k = 4 \quad k' = -3$$
$$i = 5 \qquad p = 4 \qquad p' = -3$$
$$s = root$$
 $k = 3$

```
end_point = false
```

$$r = \overline{ca}$$
$$oldr = \overline{ca}$$
$$s \leftarrow root$$

canonize(*s*, (*k*, *p*))

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$$k \leftarrow k + p' - k' + 1;$$
$$S \leftarrow S';$$

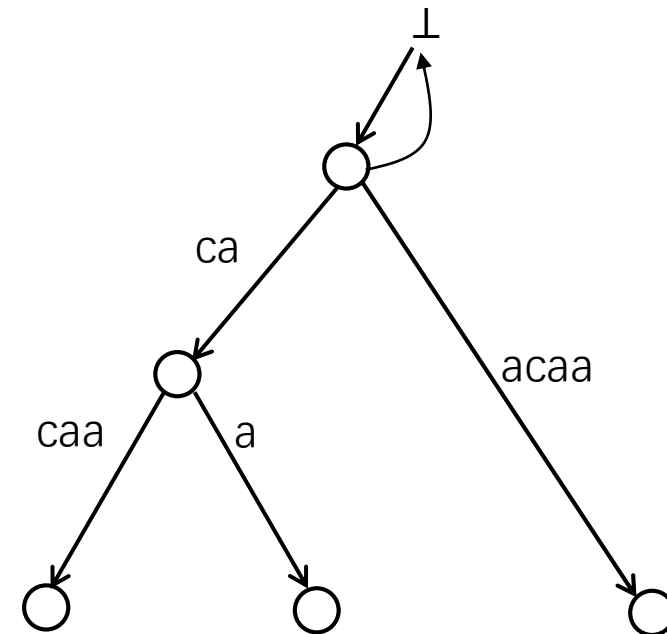
if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

```
return (s, k).
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



$$STree(T^4) \rightarrow STree(T^5)$$
$$T^4 = caca \quad s = root \quad s' = root$$
$$T^5 = cacaa \quad k = 4 \quad k' = -3$$
$$i = 5 \qquad p = 4 \qquad p' = -3$$
$$s = root$$
 $k = 3$

```
end_point = false
```

$$r = \overline{ca}$$
$$oldr = \overline{ca}$$
$$(k', p') = (2, \infty)$$
$$s' = \overline{acaa}$$
$$\mathit{canonize}(s, (k, p))$$

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$$k \leftarrow k + p' - k' + 1;$$
$$S \leftarrow S';$$

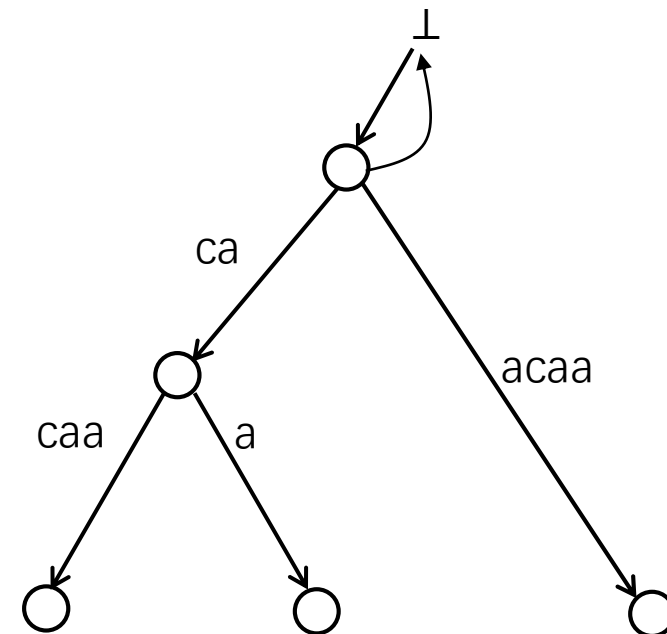
if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

```
return (s, k).
```

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



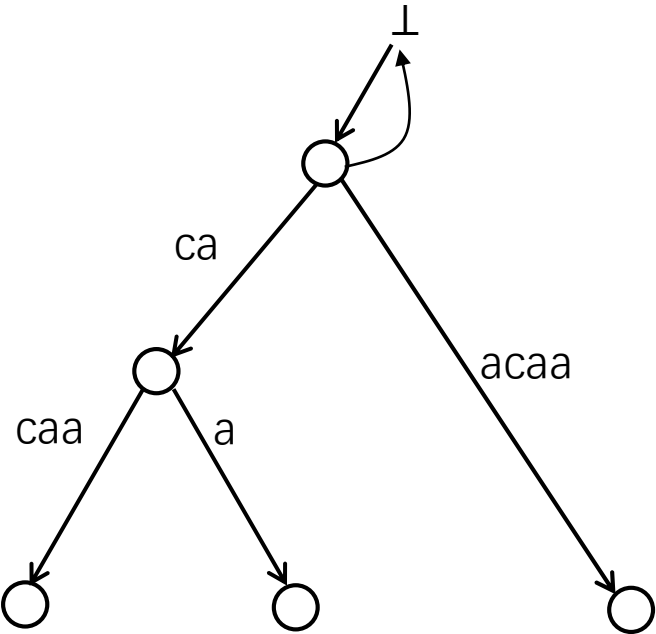
STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$ $s' = \overline{acaa}$
 $T^5 = cacaa$ $k = 4$ $k' = 2$
 $i = 5$ $p = 4$ $p' = \infty$
 $s = root$
 $k = 3$
 $end_point = false$
 $r = \overline{ca}$
 $oldr = \overline{ca}$

return ($root, 4$)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



canonize($s, (k, p)$)

if $p < k$ then return (s, k);
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ **do**
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ **then**
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k).

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 3$

$end_point = false$

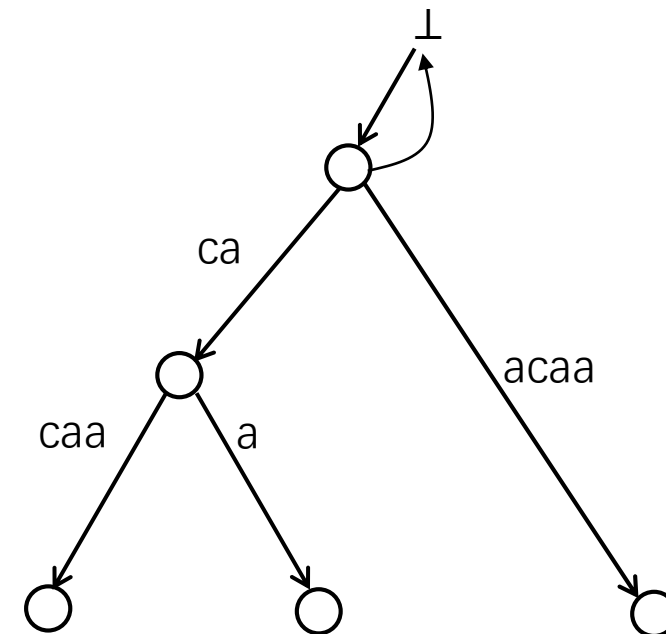
$r = \overline{ca}$

$oldr = \overline{ca}$

$(s, k) \leftarrow (root, 4)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



```

i ← i + 1;
(s, (k, i − 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i − 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i − 1));
    (end_point, r) ← test_and_split(s, (k, i − 1), ti);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
  
```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

$r = \overline{ca}$

$oldr = \overline{ca}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

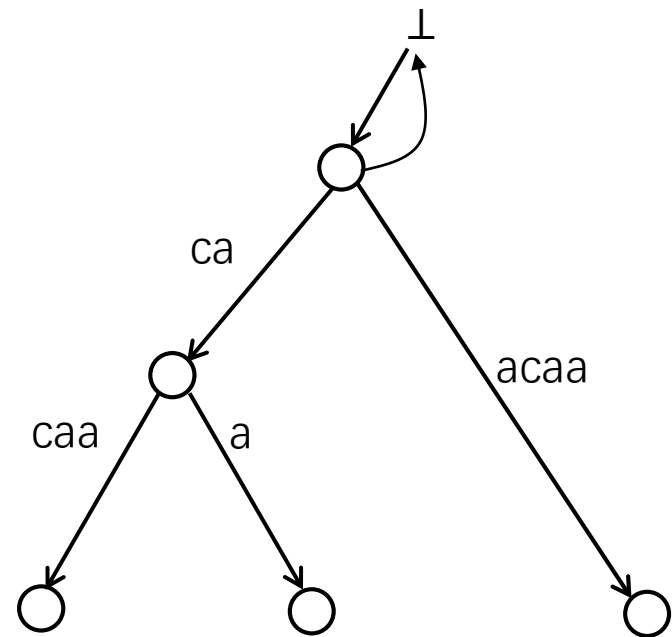
f'	
input	output
$root$	\perp

$(end_point, r) \leftarrow test_and_split(root, (4, 4), a)$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$
 $T^5 = cacaa$ $k = 4$
 $i = 5$ $p = 4$
 $s = root$ $t = a$
 $k = 4$
 $end_point = false$
 $r = \overline{c\overline{a}}$
 $oldr = \overline{c\overline{a}}$
 $(k', p') = (2, \infty)$
 $s' = \overline{acaa}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\overline{c\overline{a}}$
$(\overline{c\overline{a}}, (3, \infty))$	\overline{cacaa}
$(root, (2, \infty))$	\overline{acaa}
$(\overline{c\overline{a}}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp

test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

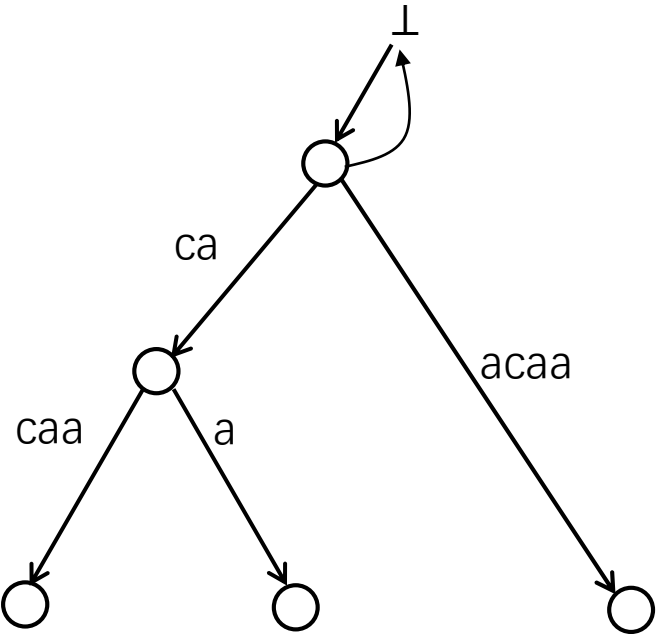
where r is a new state;

return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)



$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$ $s = root$ $s' = \overline{acaa}$

$T^5 = cacaa$ $k = 4$ $k' = 2$

$i = 5$ $p = 4$ $p' = \infty$

$s = root$ $t = a$

$k = 4$

$end_point = false$

$r = \overline{ca}$

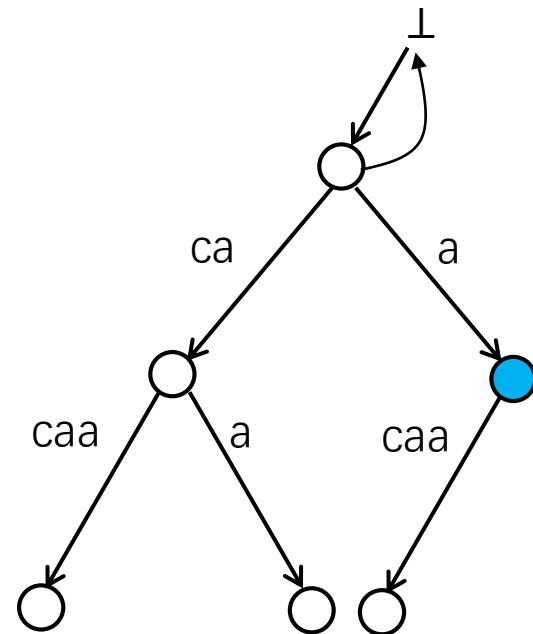
$oldr = \overline{ca}$

$g'(root, (2,2)) = r = \overline{a}$

$g'(\overline{a}, (3, \infty)) = \overline{acaa}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2,2))$	\overline{a}
$(\overline{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ then return (true, s)

else

replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

where r is a new state;

return (false, r)

else

if there is no t -transition from s then return (false, s)

else return (true, s)

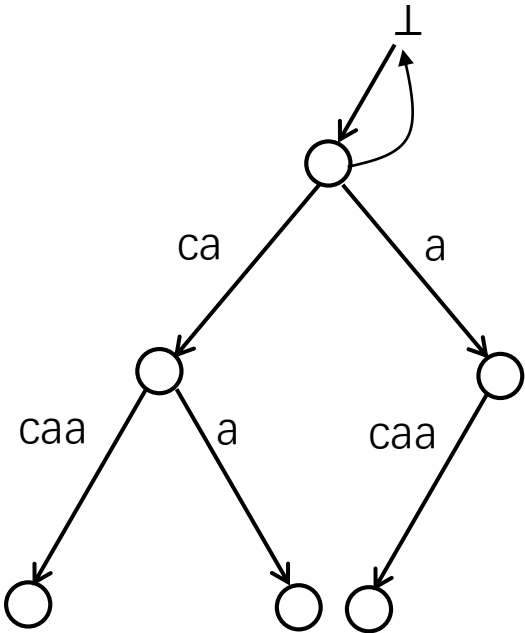
STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$ $s' = \overline{acaa}$
 $T^5 = cacaa$ $k = 4$ $k' = 2$
 $i = 5$ $p = 4$ $p' = \infty$
 $s = root$ $t = a$ $r = \bar{a}$
 $k = 4$
 $end_point = false$
 $r = \overline{ca}$
 $oldr = \overline{ca}$

return (false, \bar{a})

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2,2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}

f'	
input	output
$root$	\perp



test_and_split($s, (k, p), t$)

if $k \leq p$ then

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
if $t = t_{k'+p-k+1}$ then return (true, s)
else
replace the t_k -transition above by transitions
 $g'(s, (k', k' + p - k)) = r$ and
 $g'(r, (k' + p - k + 1, p')) = s'$,
where r is a new state;
return (false, r)

else

if there is no t -transition from s then return (false, s)
else return (true, s)

$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$

$T^5 = cacia$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

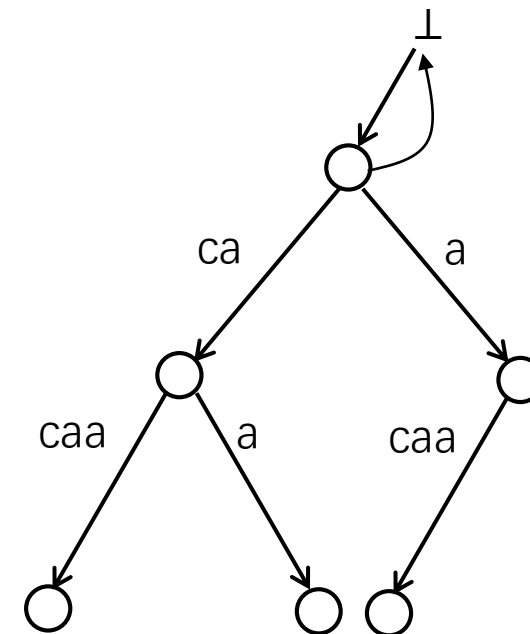
$r = \overline{ca}$

$oldr = \overline{ca}$

$(end_point, r) \leftarrow (false, \overline{a})$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{caca}
$(root, (2, 2))$	\overline{a}
$(\overline{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caaa}

f'	
input	output
$root$	\perp



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));
  
```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacia$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

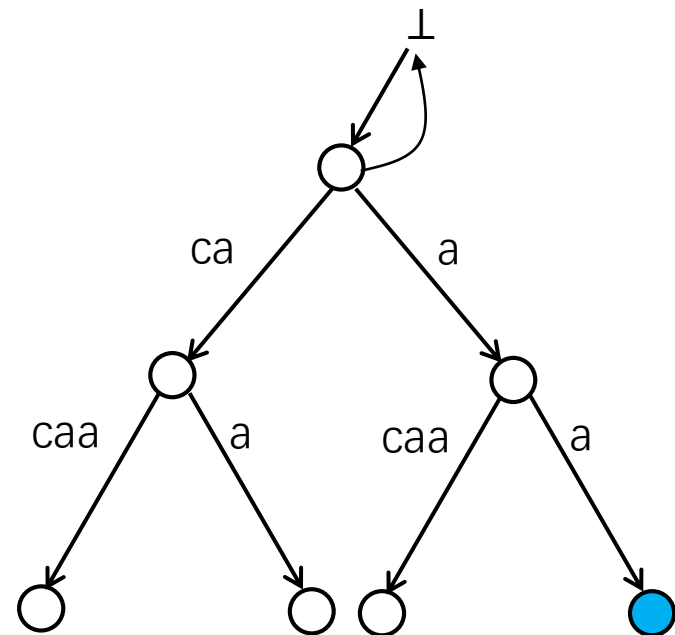
$r = \bar{a}$

$oldr = \overline{ca}$

create r'

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp



```

 $i \leftarrow i + 1;$ 
 $(s, (k, i - 1))$  is the canonical reference pair for the active point;
 $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
 $oldr \leftarrow root;$ 
while  $end\_point == false$  do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r;$ 
     $oldr \leftarrow r;$ 
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i);$ 
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s;$ 
     $(s, k) \leftarrow canonize(s, (k, i));$ 

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

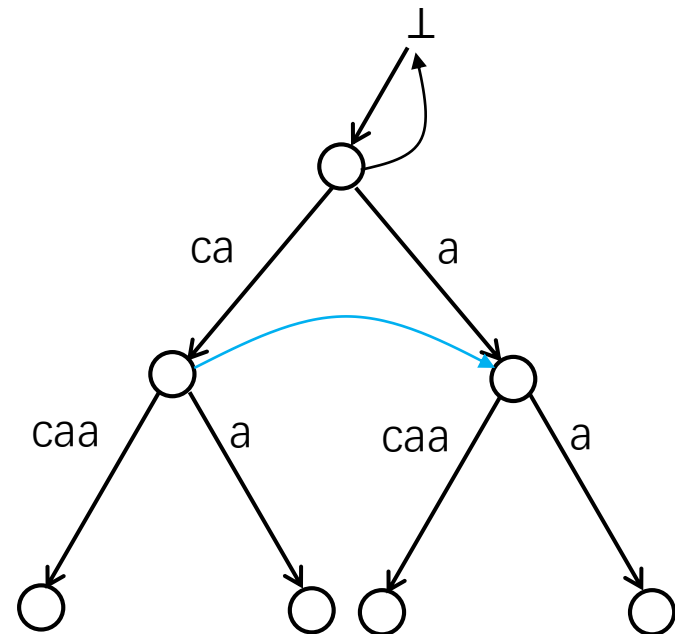
$r = \bar{a}$

$oldr = \overline{ca}$

create $f'(\overline{ca}) = \bar{a}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), ti);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if  $oldr \neq root$  then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), ti);
if  $oldr \neq root$  then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

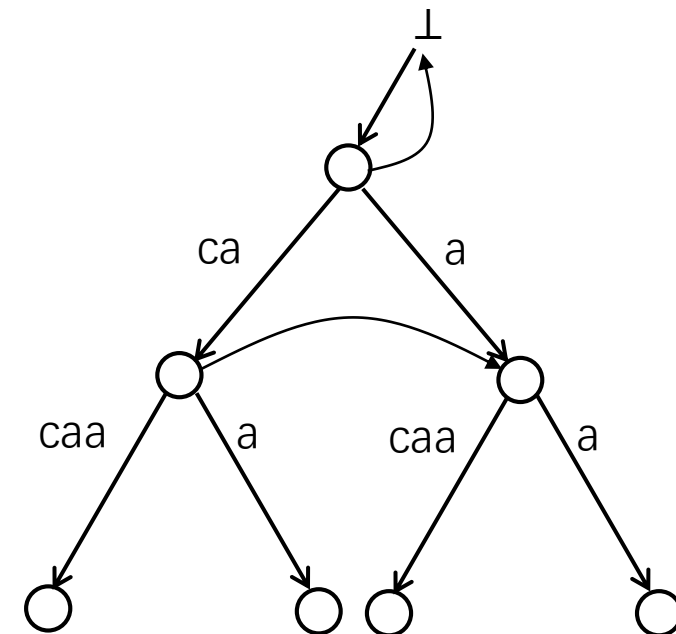
$r = \bar{a}$

$oldr = \overline{ca}$

$oldr \leftarrow \bar{a}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

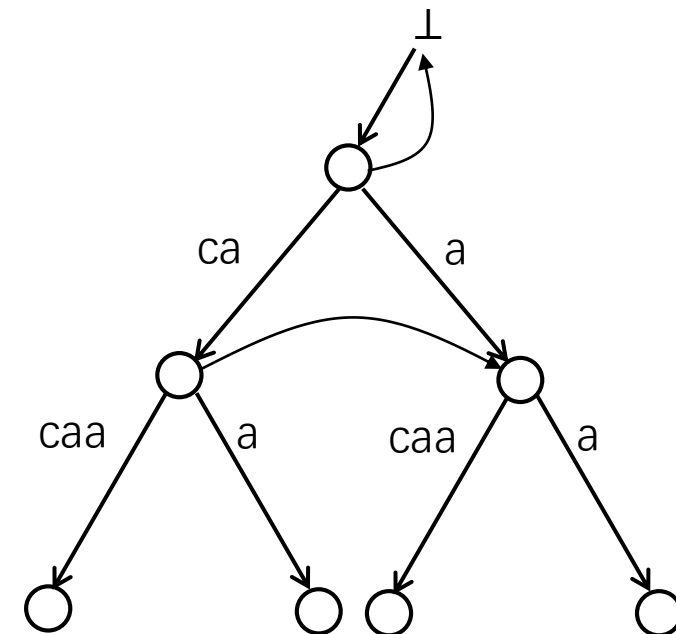
$r = \bar{a}$

$oldr = \bar{a}$

$(s, k) \leftarrow canonize(\perp, (4, 4))$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
     $(s, k) \leftarrow canonize(f'(s), (k, i - 1));$ 
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

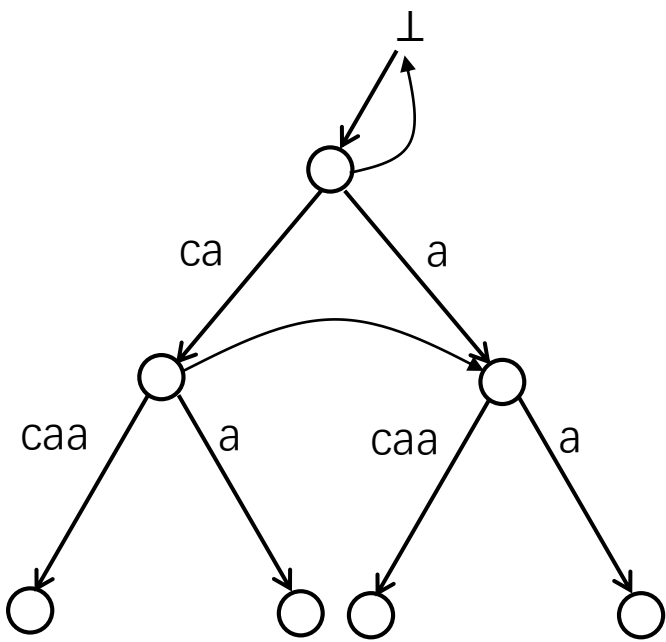
STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = \perp$
 $T^5 = cacaa$ $k = 4$
 $i = 5$ $p = 4$
 $s = root$
 $k = 4$
 $end_point = false$
 $r = \bar{a}$
 $oldr = \bar{a}$

$(k', p') = (-4, -4)$
 $s' = root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k) .

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$
 $T^5 = caciaa$
 $i = 5$
 $s = root$
 $k = 4$
 $end_point = false$
 $r = \bar{a}$
 $oldr = \bar{a}$

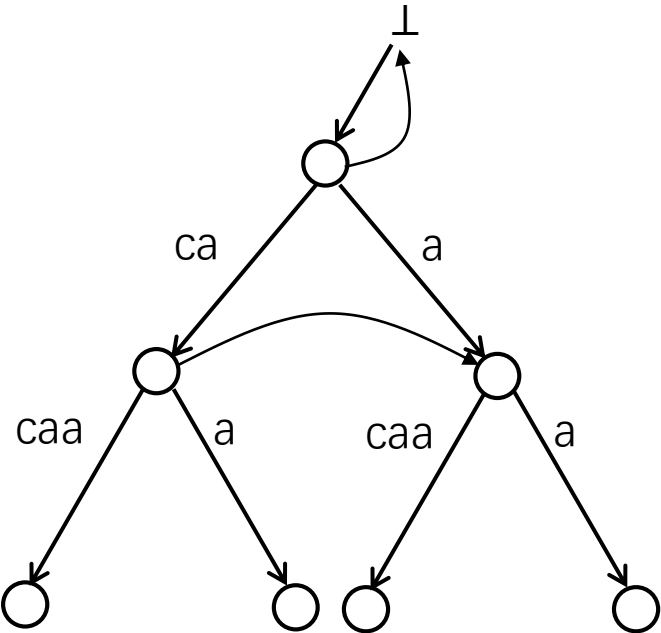
$s = \perp$
 $k = 4$
 $p = 4$

$s' = root$
 $k' = -4$
 $p' = -4$

$k \leftarrow 5$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2,2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caaa}
$(\bar{a}, (5, \infty))$	\overline{aaa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



$canonize(s, (k, p))$
if $p < k$ then return (s, k) ;
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k) .

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$
 $T^5 = cacaa$
 $i = 5$
 $s = root$
 $k = 4$
 $end_point = false$
 $r = \bar{a}$
 $oldr = \bar{a}$

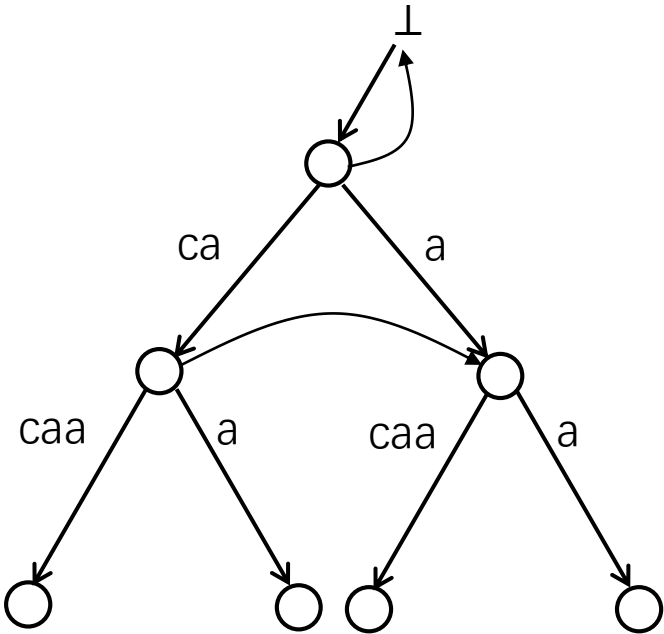
$s = \perp$
 $k = 5$
 $p = 4$

$s' = root$
 $k' = -4$
 $p' = -4$

$s \leftarrow root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2,2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



$canonize(s, (k, p))$
if $p < k$ then return (s, k) ;
else
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 while $p' - k' \leq p - k$ do
 $k \leftarrow k + p' - k' + 1$;
 $s \leftarrow s'$;
 if $k \leq p$ then
 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;
 return (s, k) .

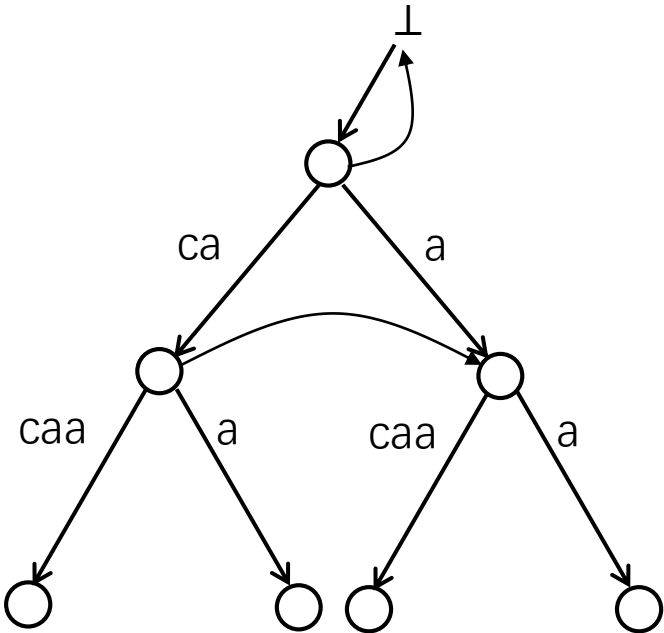
STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$ $s' = root$
 $T^5 = cacaa$ $k = 5$ $k' = -4$
 $i = 5$ $p = 4$ $p' = -4$
 $s = root$
 $k = 4$
 $end_point = false$
 $r = \bar{a}$
 $oldr = \bar{a}$

return ($root, 5$)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caaa}
$(\bar{a}, (5, \infty))$	\overline{aaa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}



canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 4$

$end_point = false$

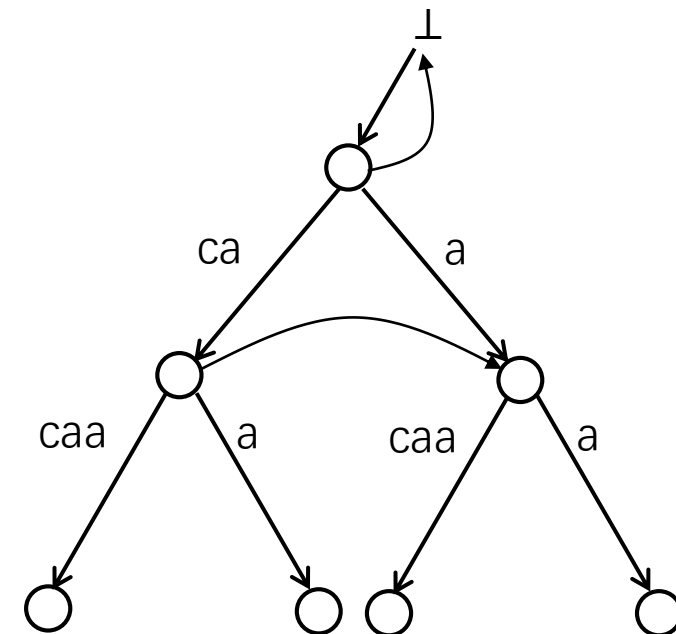
$r = \bar{a}$

$oldr = \bar{a}$

$(s, k) \leftarrow (root, 5)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	$\bar{c}\bar{a}\bar{c}\bar{a}\bar{a}$
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	$\bar{a}\bar{c}\bar{a}\bar{a}$
$(\bar{c}\bar{a}, (5, \infty))$	$\bar{c}\bar{a}\bar{a}$
$(\bar{a}, (5, \infty))$	$\bar{a}\bar{a}$

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize(f'(s), (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 5$

$end_point = false$

$r = \bar{a}$

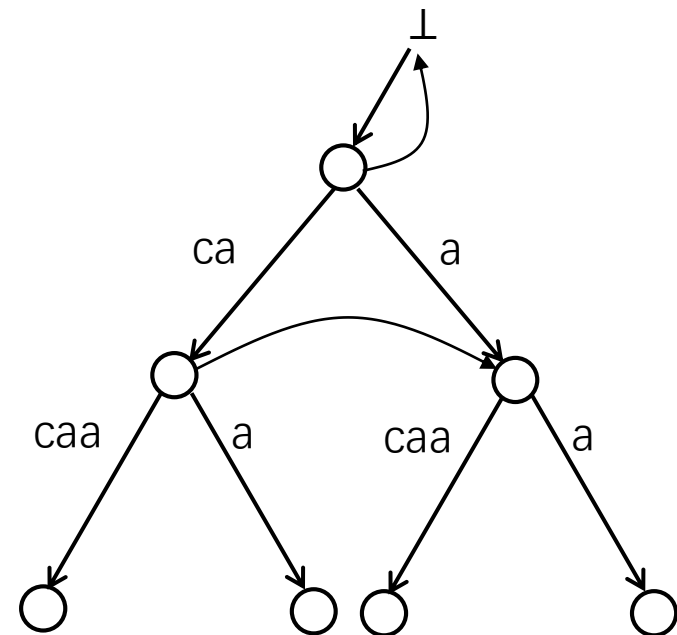
$oldr = \bar{a}$

$(end_point, r) \leftarrow$

$test_and_split(root, (5, 4), a);$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	$\bar{c}\bar{a}\bar{c}\bar{a}\bar{a}$
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	$\bar{a}\bar{c}\bar{a}\bar{a}$
$(\bar{c}\bar{a}, (5, \infty))$	$\bar{c}\bar{a}\bar{a}$
$(\bar{a}, (5, \infty))$	$\bar{a}\bar{a}$

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}



$i \leftarrow i + 1;$

$(s, (k, i - 1))$ is the canonical reference pair for the active point;

$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

$oldr \leftarrow root;$

while $end_point == false$ **do**

create new transition $g'(r, (i, \infty)) = r'$ where r' is a new state;

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = r;$

$oldr \leftarrow r;$

$(s, k) \leftarrow canonize(f'(s), (k, i - 1));$

$(end_point, r) \leftarrow test_and_split(s, (k, i - 1), t_i);$

if $oldr \neq root$ **then** create new suffix link $f'(oldr) = s;$

$(s, k) \leftarrow canonize(s, (k, i));$

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$
 $T^5 = cacia$ $k = 5$
 $i = 5$ $p = 4$
 $s = root$ $t = a$
 $k = 5$
 $end_point = false$
 $r = \bar{a}$
 $oldr = \bar{a}$
return (true, root)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1,2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2,2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}

test_and_split($s, (k, p), t$)

if $k \leq p$ **then**

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

if $t = t_{k'+p-k+1}$ **then return** (true, s)

else

 replace the t_k -transition above by transitions

$g'(s, (k', k' + p - k)) = r$ and

$g'(r, (k' + p - k + 1, p')) = s'$,

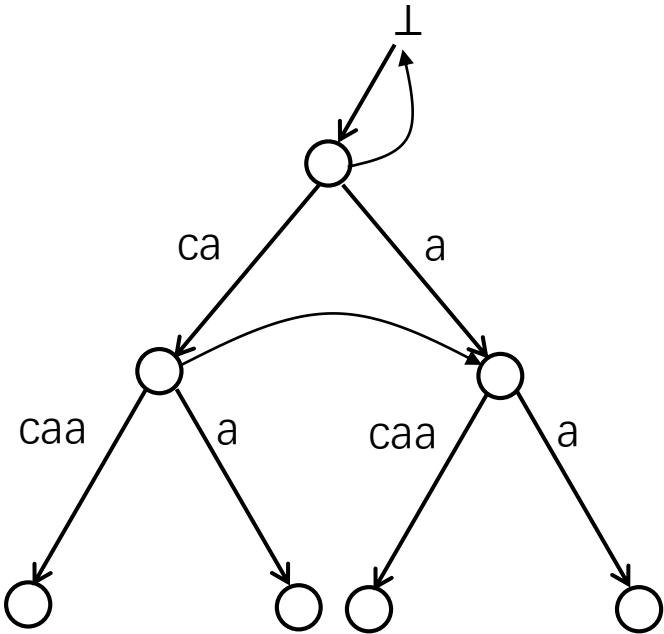
 where r is a new state;

return (false, r)

else

if there is no t -transition from s **then return** (false, s)

else return (true, s)



$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 5$

$end_point = false$

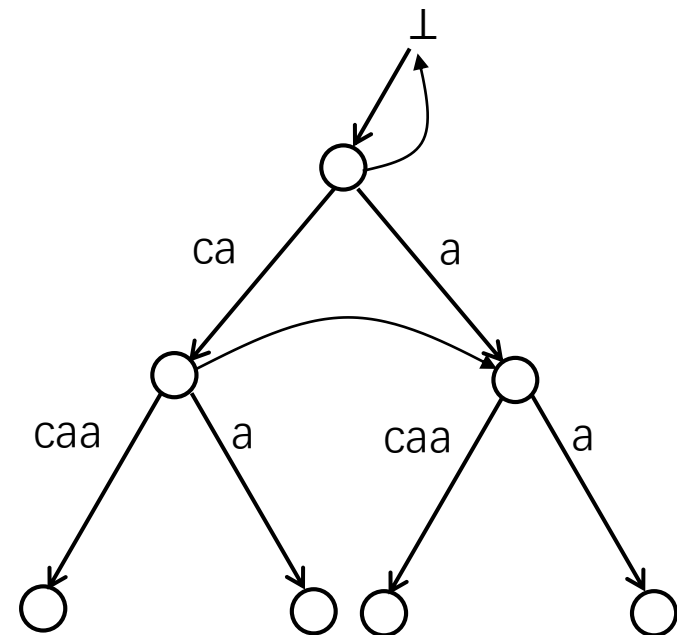
$r = \bar{a}$

$oldr = \bar{a}$

$(end_point, r) \leftarrow (true, root)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	$\bar{c}\bar{a}\bar{c}\bar{a}\bar{a}$
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	$\bar{a}\bar{c}\bar{a}\bar{a}$
$(\bar{c}\bar{a}, (5, \infty))$	$\bar{c}\bar{a}\bar{a}$
$(\bar{a}, (5, \infty))$	$\bar{a}\bar{a}$

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
     $(end\_point, r) \leftarrow test\_and\_split(s, (k, i - 1), t_i)$ ;
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 5$

$end_point = true$

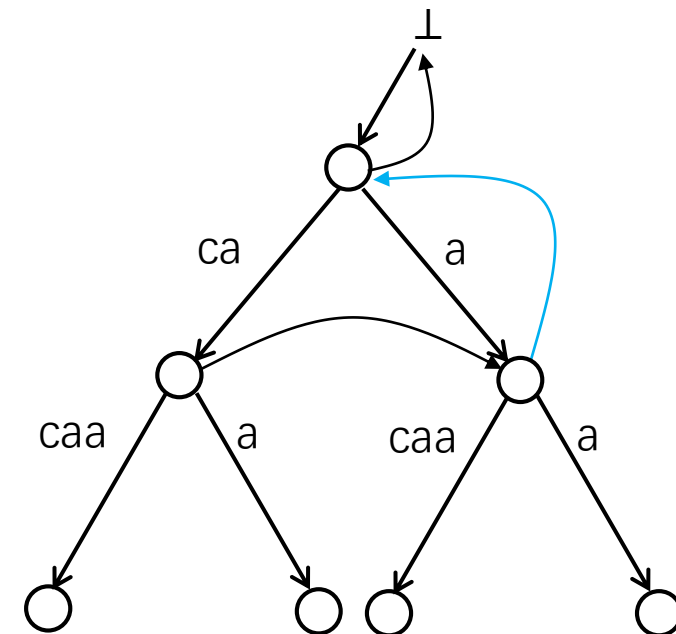
$r = root$

$oldr = \bar{a}$

create $f'(\bar{a}) = root$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}
\bar{a}	$root$



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 5$

$end_point = true$

$r = root$

$oldr = \bar{a}$

$(s, k) \leftarrow canonize(root, (5, 5))$

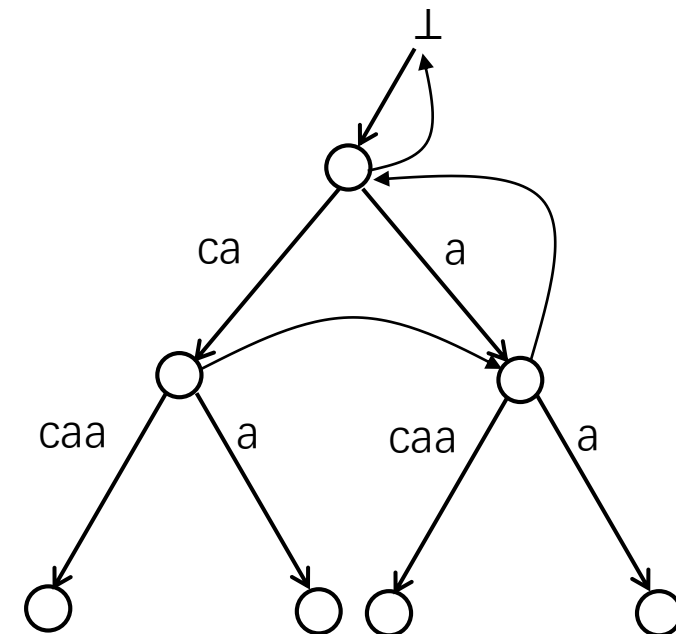
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}
\bar{a}	$root$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = root$

$T^5 = cacaa$ $k = 5$

$i = 5$ $p = 5$

$s = root$

$k = 5$

$end_point = true$

$r = root$

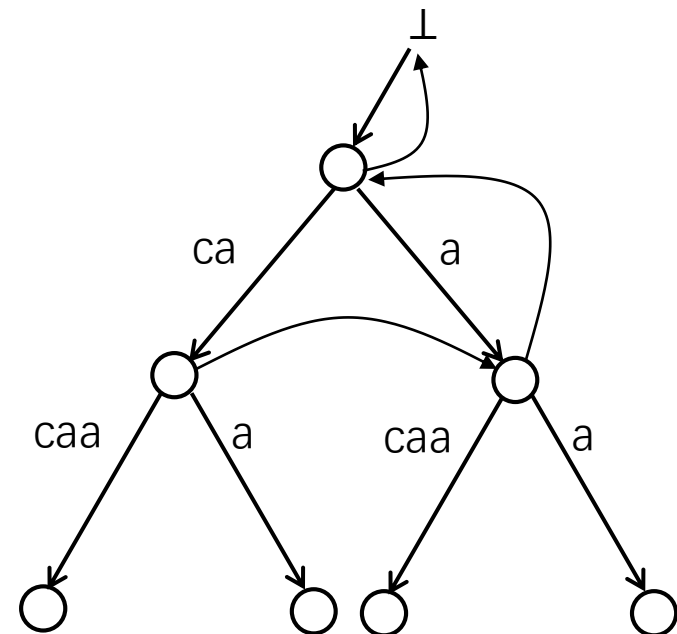
$oldr = \bar{a}$

$(k', p') = (2, 2)$

$s' = \bar{a}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caaa}
$(\bar{a}, (5, \infty))$	\overline{aaa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}
\bar{a}	$root$



canonize($s, (k, p)$)

if $p < k$ then return (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ do

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

 if $k \leq p$ then

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k) .

$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$ $s = root$ $s' = \bar{a}$

$T^5 = cacaa$ $k = 5$ $k' = 2$

$i = 5$ $p = 5$ $p' = 2$

$s = root$

$k = 5$

$end_point = true$

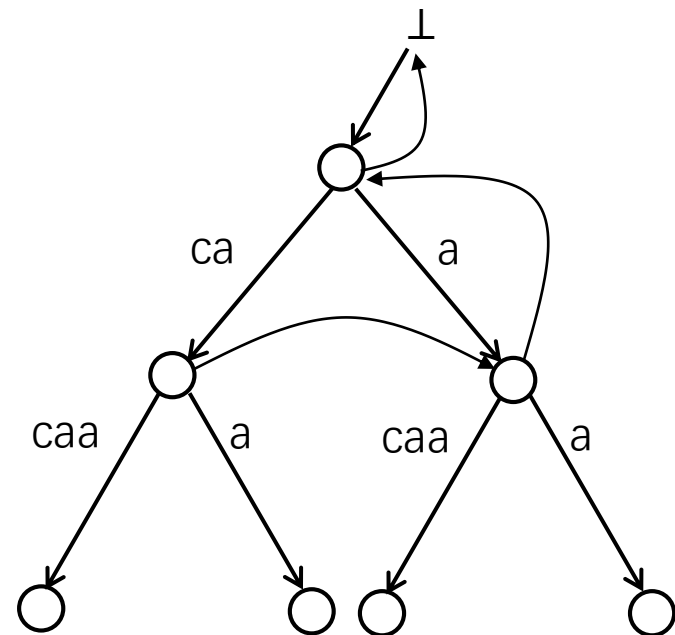
$r = root$

$oldr = \bar{a}$

$k \leftarrow 6$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\bar{c}\bar{a}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}
\bar{a}	$root$



$canonize(s, (k, p))$

if $p < k$ **then return** (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k) .

$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$ $s = root$ $s' = \bar{a}$

$T^5 = cacaa$ $k = 6$ $k' = 2$

$i = 5$ $p = 5$ $p' = 2$

$s = root$

$k = 5$

$end_point = true$

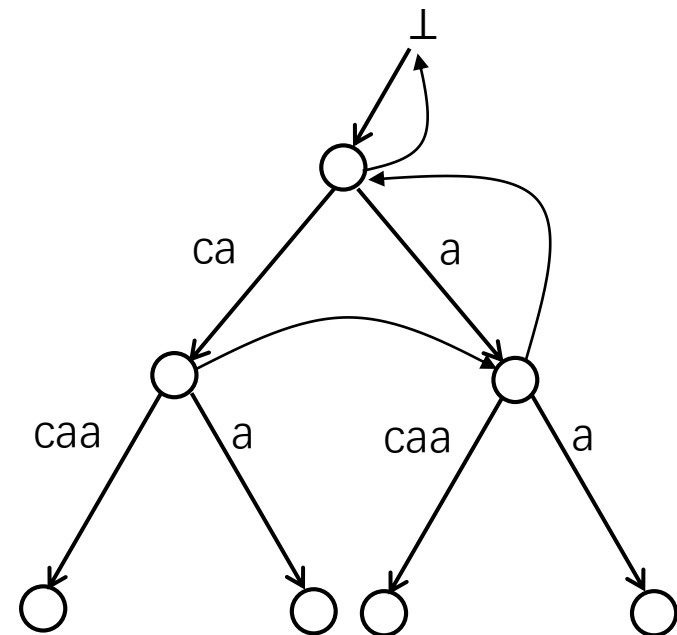
$r = root$

$oldr = \bar{a}$

$s \leftarrow \bar{a}$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\bar{c}\bar{a}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}
\bar{a}	$root$



$canonize(s, (k, p))$

if $p < k$ **then return** (s, k) ;

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k) .

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$ $s = \bar{a}$ $s' = \bar{a}$

$T^5 = cacaa$ $k = 6$ $k' = 2$

$i = 5$ $p = 5$ $p' = 2$

$s = root$

$k = 5$

$end_point = true$

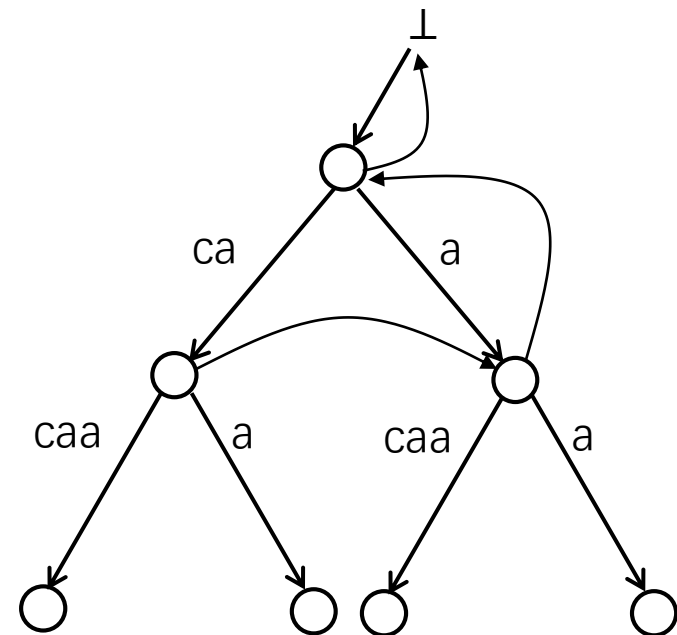
$r = root$

$oldr = \bar{a}$

return ($\bar{a}, 6$)

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	$\bar{c}\bar{a}$
$(\bar{c}\bar{a}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\bar{c}\bar{a}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
$\bar{c}\bar{a}$	\bar{a}
\bar{a}	$root$



canonize($s, (k, p)$)

if $p < k$ then return (s, k);

else

find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

while $p' - k' \leq p - k$ **do**

$k \leftarrow k + p' - k' + 1$;

$s \leftarrow s'$;

if $k \leq p$ **then**

 find the t_k -transition $g'(s, (k', p')) = s'$ from s ;

return (s, k).

$STree(T^4) \rightarrow STree(T^5)$

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = root$

$k = 5$

$end_point = true$

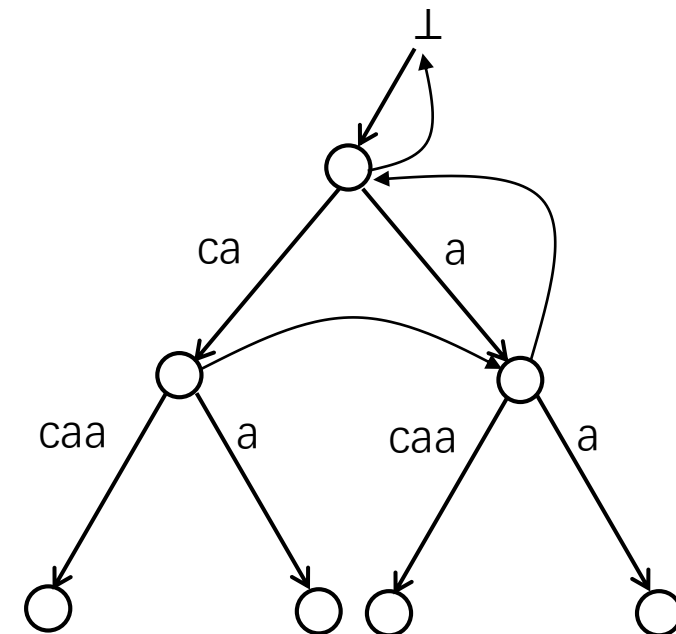
$r = root$

$oldr = \bar{a}$

$(s, k) \leftarrow (\bar{a}, 6)$

g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}
\bar{a}	$root$



```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```

STree(T^4) \rightarrow ***STree***(T^5)

$T^4 = caca$

$T^5 = cacaa$

$i = 5$

$s = \bar{a}$

$k = 6$

$end_point = true$

$r = root$

$oldr = \bar{a}$

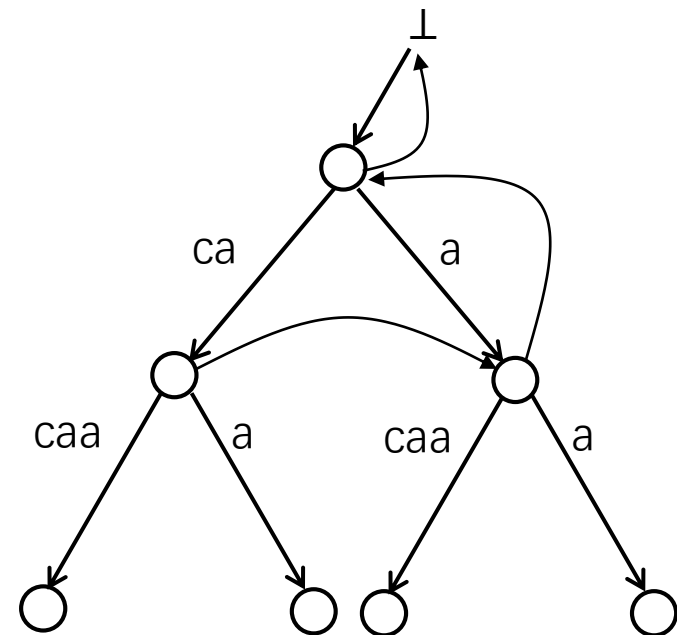
g'	
input	output
$(\perp, (-i, -i))$	$root$
$(root, (1, 2))$	\overline{ca}
$(\overline{ca}, (3, \infty))$	\overline{cacaa}
$(root, (2, 2))$	\bar{a}
$(\bar{a}, (3, \infty))$	\overline{acaa}
$(\overline{ca}, (5, \infty))$	\overline{caa}
$(\bar{a}, (5, \infty))$	\overline{aa}

f'	
input	output
$root$	\perp
\overline{ca}	\bar{a}
\bar{a}	$root$

```

i ← i + 1;
(s, (k, i - 1)) is the canonical reference pair for the active point;
(end_point, r) ← test_and_split(s, (k, i - 1), t_i);
oldr ← root;
while end_point == false do
    create new transition  $g'(r, (i, \infty)) = r'$  where  $r'$  is a new state;
    if oldr ≠ root then create new suffix link  $f'(oldr) = r$ ;
    oldr ← r;
    (s, k) ← canonize( $f'(s)$ , (k, i - 1));
    (end_point, r) ← test_and_split(s, (k, i - 1), t_i);
if oldr ≠ root then create new suffix link  $f'(oldr) = s$ ;
(s, k) ← canonize(s, (k, i));

```



Time Complexity Analysis

Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Proof

The algorithm *update* constructs $STree(T^i)$ base on $STree(T^{i-1})$ and t_i , so it is an on-line algorithm.

While loops appear in *update* and *canonicalize*. To analyze the running time, we divide the total time into two components: the time for *canonicalize* and the rest statements. We will show both are $O(n)$.

For convenience, we call each state s_h between the active point s_j and the end point $s_{j'}$, a visited state ($j \leq h < j'$).

Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Proof

First, consider the running time for statements besides *canonicalize*. We only need to focus on the while loop of *update* because each time *update* is executed, statements outside the while loop run in a constant time. In each loop, *canonicalize* is called once to operate on a visited state. Consequently, the number of while loops corresponds to the number of visited states.

```


update

  
...  
while end_point == false do  
    create new transition  $g'(r, (i, \infty)) = r'$   
    where  $r'$  is a new state;  
    if oldr  $\neq$  root then create new suffix link  $f'(\textit{oldr}) = r$ ;  
    oldr  $\leftarrow r$  ;  
     $(s, k) \leftarrow \textit{canonicalize}(f'(s), (k, i - 1))$ ;  
     $(\textit{end\_point}, r) \leftarrow \textit{test\_and\_split}(s, (k, i - 1), t_i)$ ;  
...  

```

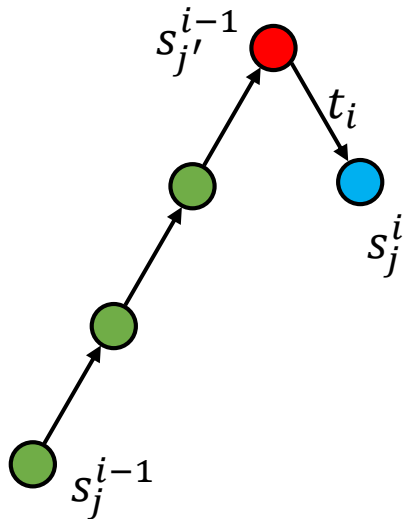
Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Proof

Next, we show that the number of visited states is $O(n)$.

Denote the active point and end point of T^i by s_j^i and $s_{j'}^i$. The traversal path of each visited state is: $s_j^{i-1} \rightarrow \dots \rightarrow s_{j'}^{i-1} \rightarrow g(s_{j'}, t_i) = s_j^i$. Denote the depth of a state (or a node) s by $d(s)$, it's obvious that $d(s_{j'}^{i-1}) - d(s_j^{i-1}) = 1$ and $d(s_j^i) - d(s_{j'}^{i-1}) = 1$. Thus, the number of visited states is:



$$\begin{aligned} & \sum_{i=1}^n d(s_j^{i-1}) - d(s_{j'}^{i-1}) \\ &= \sum_{i=1}^n d(s_j^{i-1}) - (d(s_j^i) - 1) \\ &= d(s_j^0) - d(s_j^n) + n \\ &= O(n). \end{aligned} \tag{1}$$

Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Proof

Finally, we prove the running time for *canonicalize* is also $O(n)$.

Every time *update* is executed, the *canonicalize* outside the while loop is called. We have also proved that the number of while loops is $O(n)$, so *canonicalize* is executed $O(n)$ times. We only need to show that in each execution of *canonicalize*, the running time is bounded by a constant.

```

                                canonicalize(s, (k, p))
if p < k then return (s, k);
else
    find the  $t_k$ -transition  $g'(s, (k', p')) = s'$  from s;
    while  $p' - k' \leq p - k$  do
         $k \leftarrow k + p' - k' + 1$ ;
         $s \leftarrow s'$ ;
        if  $k \leq p$  then
            find the  $t_k$ -transition
             $g'(s, (k', p')) = s'$  from s;
    return (s, k).
```

Theorem 2

$STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.

Proof

In fact, each time *canonicalize* is called, the while loop inside *canonicalize* is executed at most once.

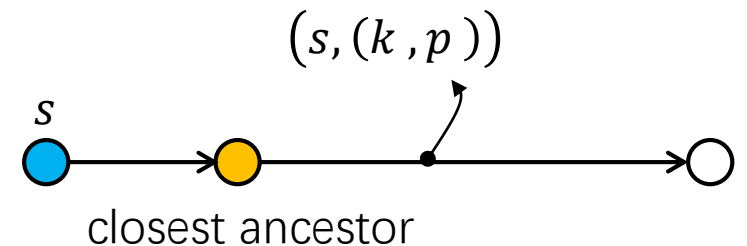
The purpose of the while loop is to decrease the length of $w = (k, p)$. However, the length of w only grows (namely the right pointer increases) by one at the end of *update*. Thus, the total running time of *canonicalize* is $O(n)$.

To sum up, $STree(T)$ can be constructed on-line in time $O(n)$.

Q.E.D.

update

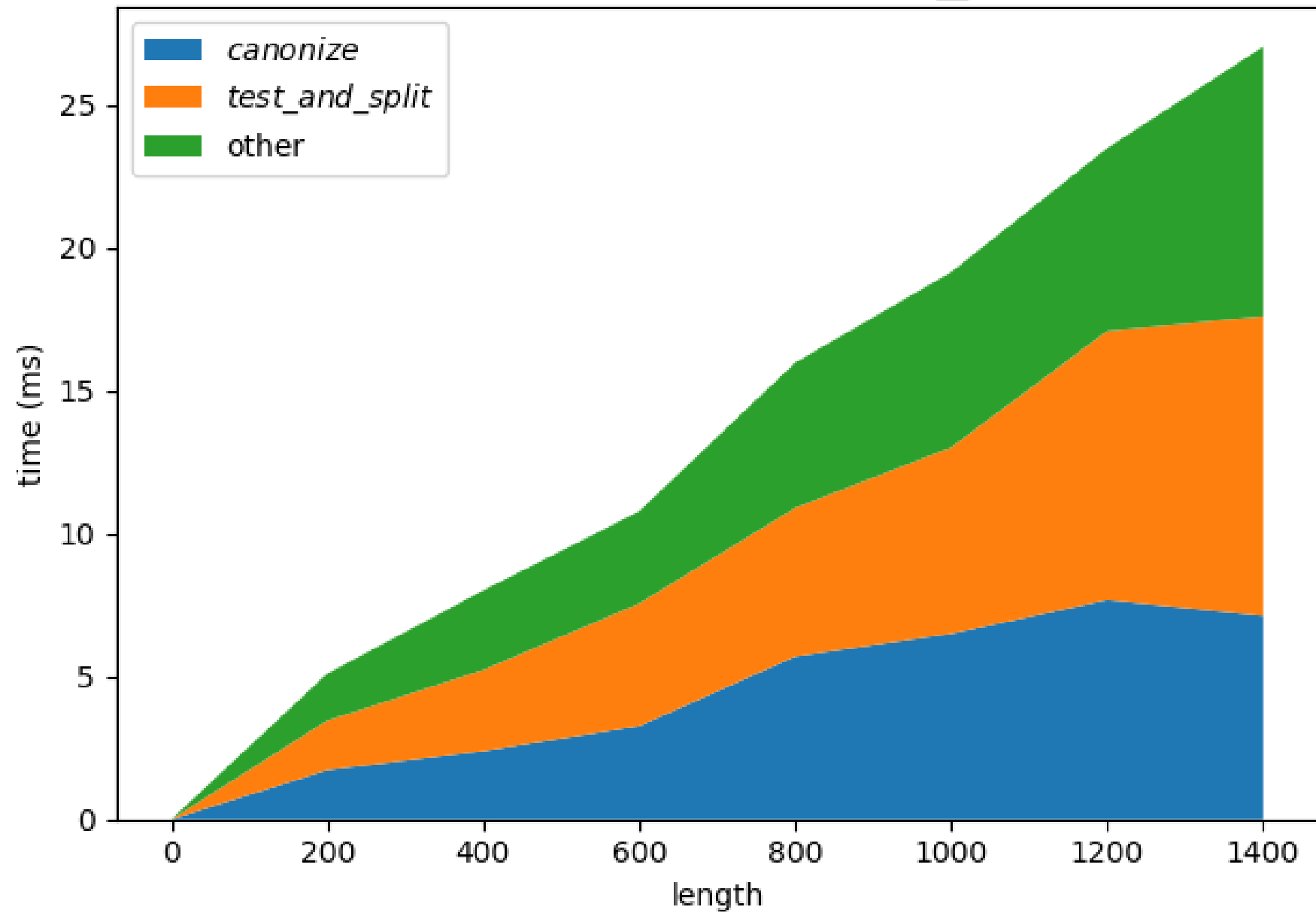
```
...  
while end_point == false do  
  ...  
  if olldr ≠ root then create new suffix link  $f'(\textit{olldr}) = s$ ;  
   $(s, k) \leftarrow \textit{canonicalize}(s, (k, i));$ 
```



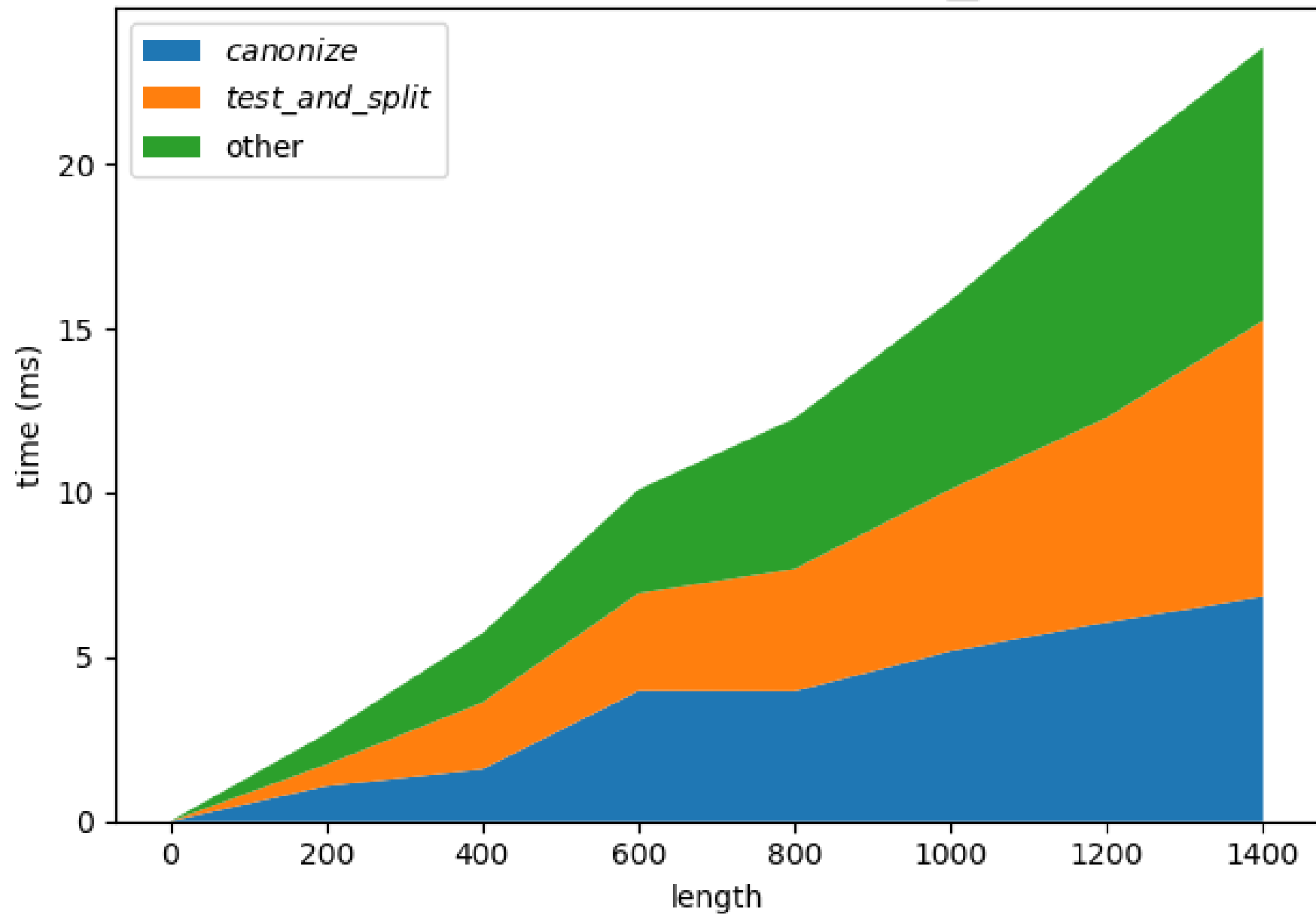
■ Construct Suffix Trees on Laptop

It took **80 seconds** for my laptop to construct the suffix tree (in Python) of a random DNA sequence with length **10^6** .

Average Execution Time (200 Repeats) $\Sigma = \{A, T, C, G\}$



Average Execution Time (200 Repeats) $\Sigma = \{a, \dots, z\}$



■ Comments

- $STree(T)$ can be constructed on-line in time $O(n)$, where $n = |T|$.
- We refer to each string by a reference pair, so the space complexity is also $O(n)$.
- With suffix trees, we can tell whether a pattern is a substring/suffix of T .
- In a suffix tree, there are $O(n)$ states. Can we find another structure that can also compress $STrie(T)$? One possible answer is suffix automata.