Suffix Trees

Pavel Pevzner

Department of Computer Science and Engineering
University of California at San Diego

Algorithms on Strings
Data Structures and Algorithms

This slide desk is incomplete.
For the complete set of frames,
please see our videos in the
Algorithms on Strings course on Coursera
(**Algorithms and Data Structures Specialization**)

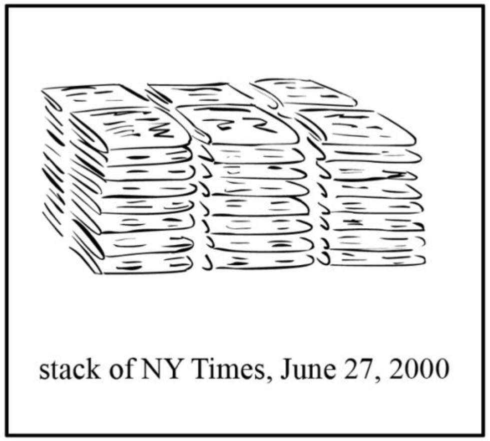
Outline

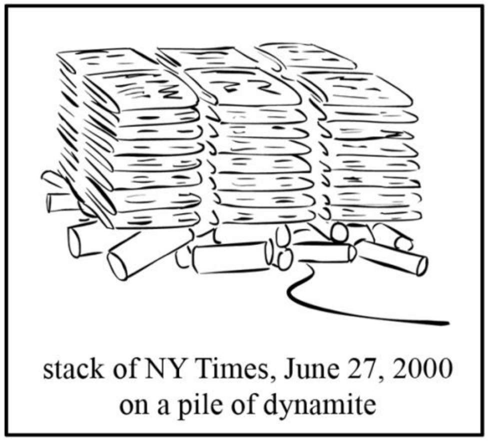
- **From Genome Sequencing to Pattern Matching**
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

The Newspaper Problem

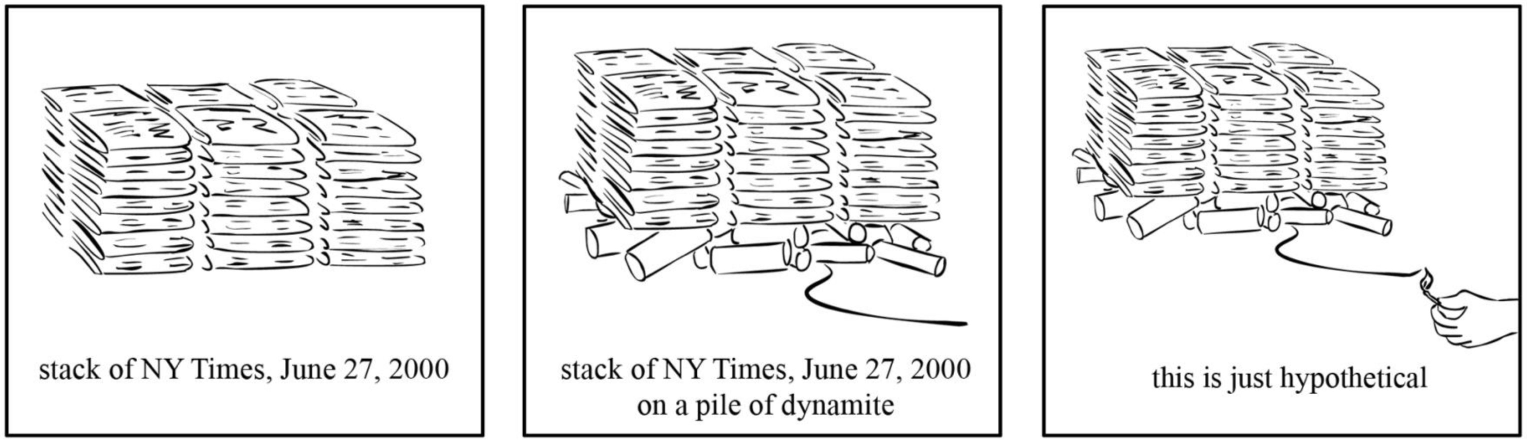
stack of NY Times, June 27, 2000

The Newspaper Problem

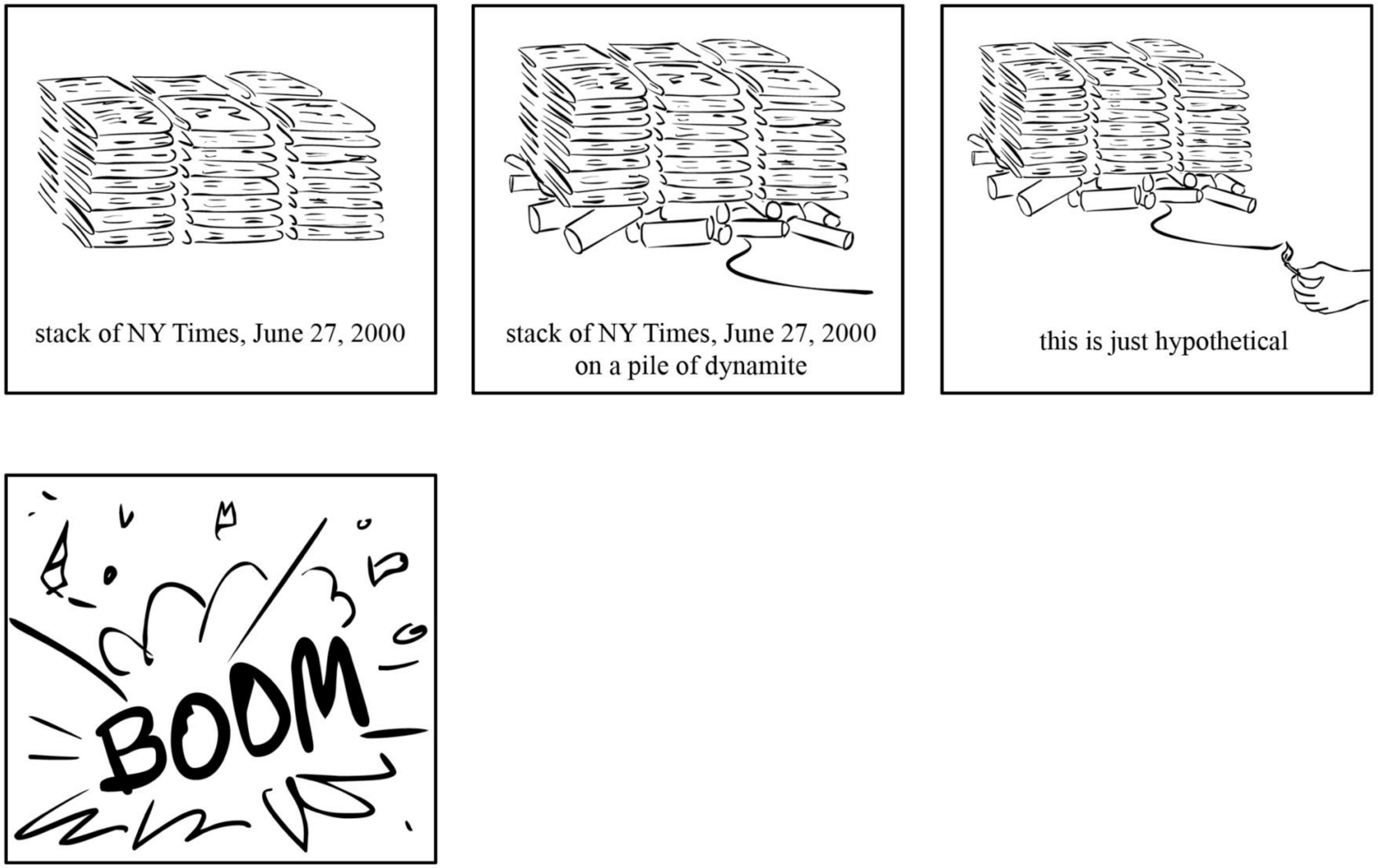




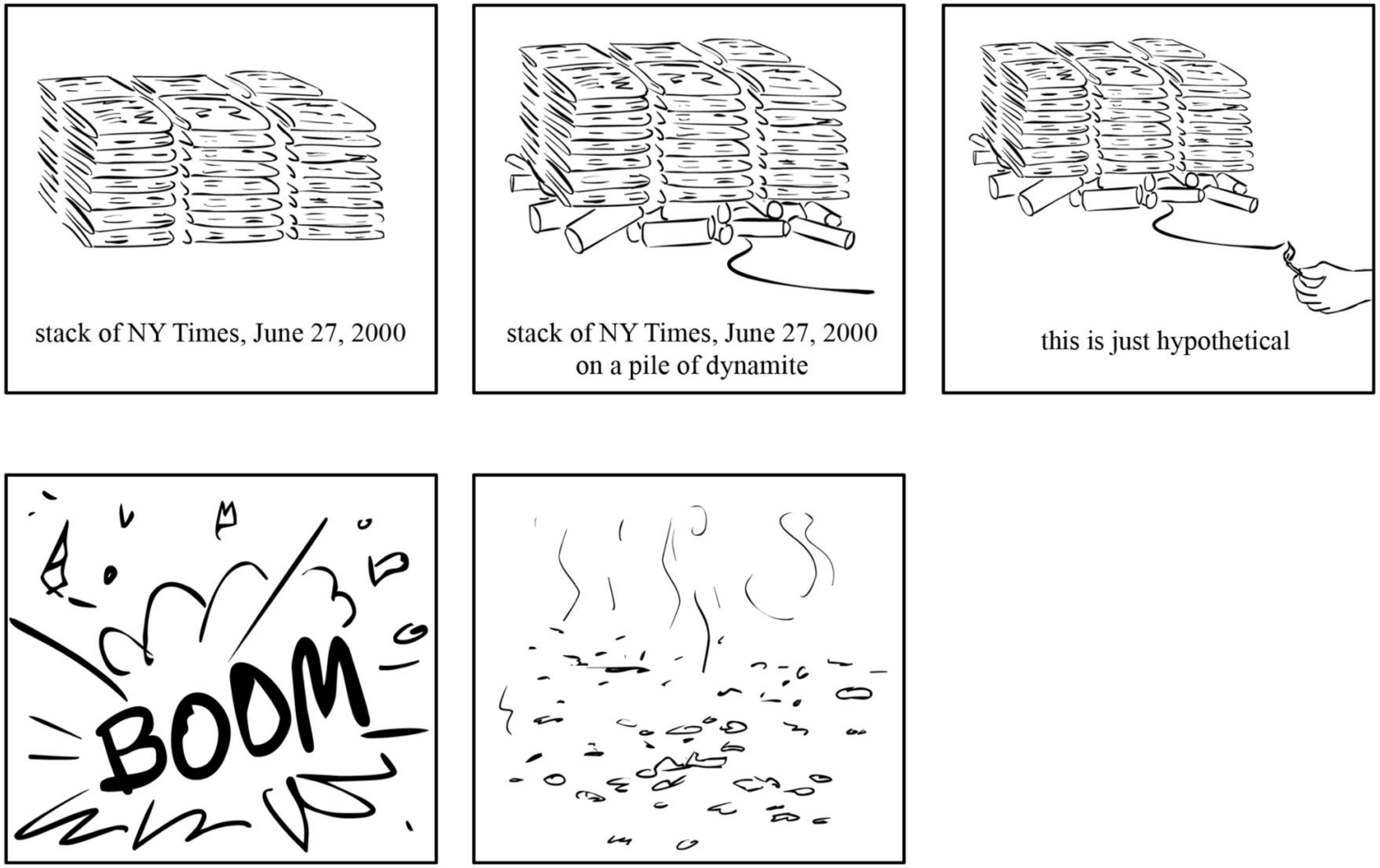
The Newspaper Problem


stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical

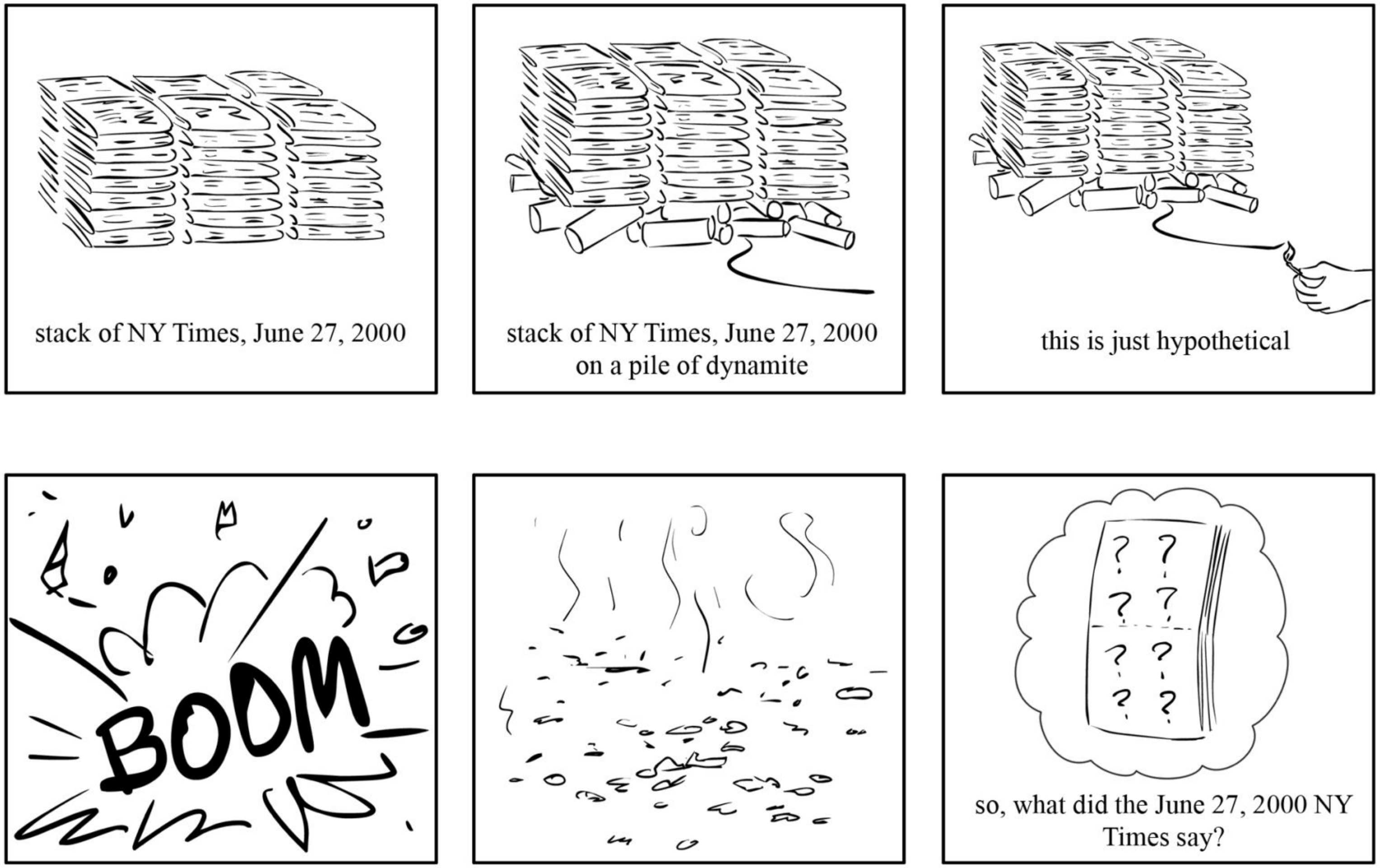
The Newspaper Problem


stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM

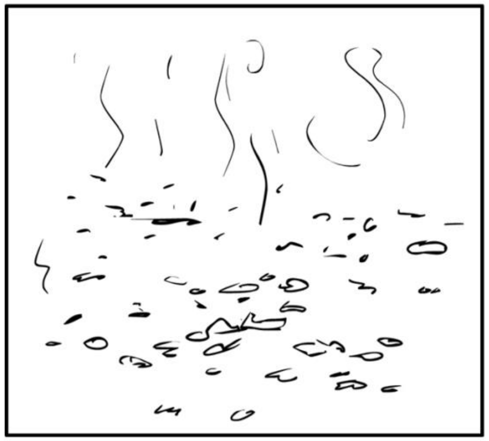
The Newspaper Problem

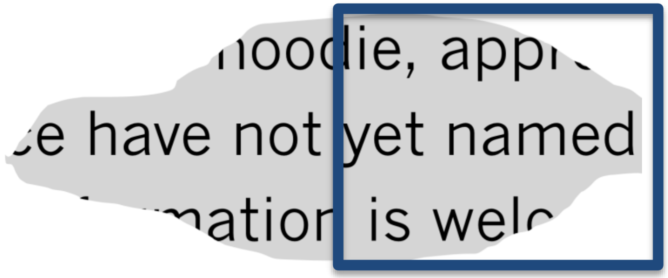

stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM

The Newspaper Problem

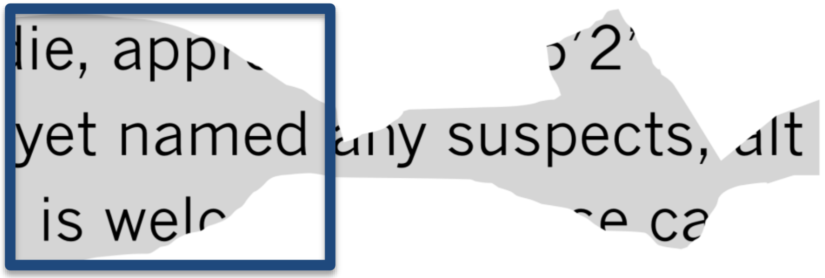

stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM
? ? ? ? ? ? ? ?
so, what did the June 27, 2000 NY Times say?

The Newspaper Problem as an Overlapping Puzzle



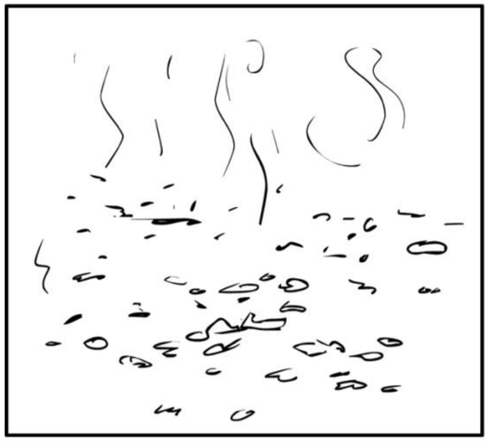


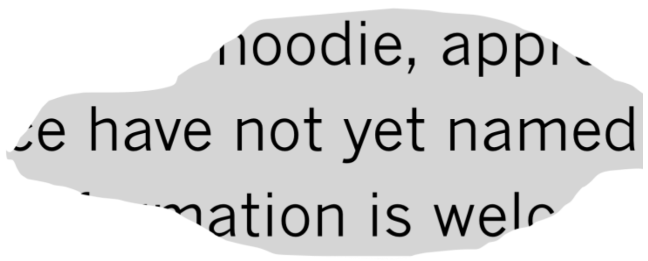
noodie, appr
ce have not yet named
mation is welc



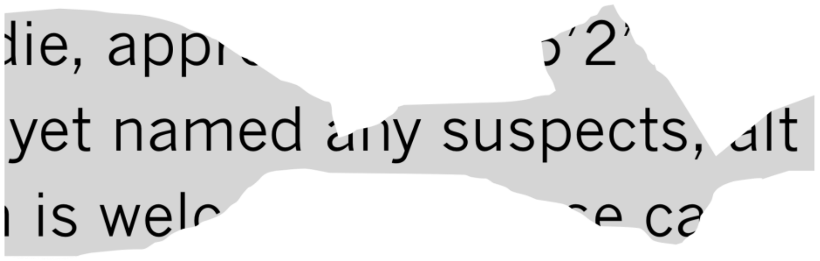
die, appr 5'2'
yet named any suspects, alt
is welc ce ca

The Newspaper Problem as an Overlapping Puzzle





noodie, appr
ce have not yet named
mation is welc



die, appr 5’2’
yet named any suspects, alt
n is welc ce ca

Millions of Copies of a Genome

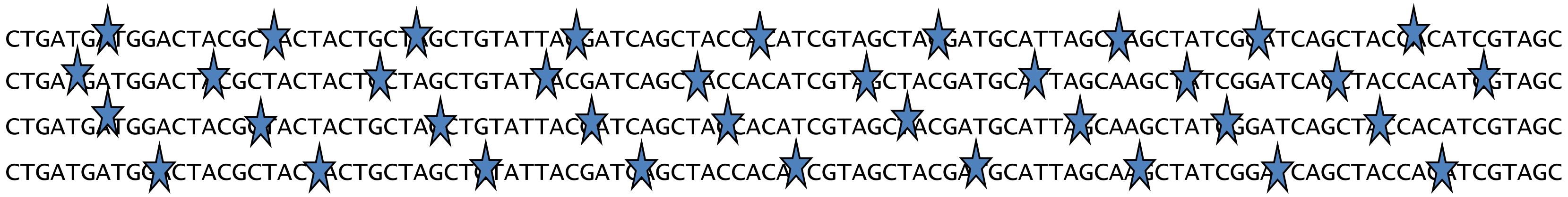
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

Breaking the Genomes at Random Positions



CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

Generating Short Substring (Reads)

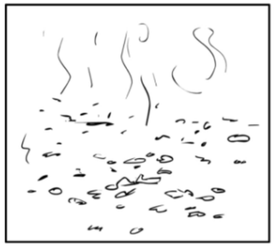
CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

Burning Some Reads



CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

Assembling Genome

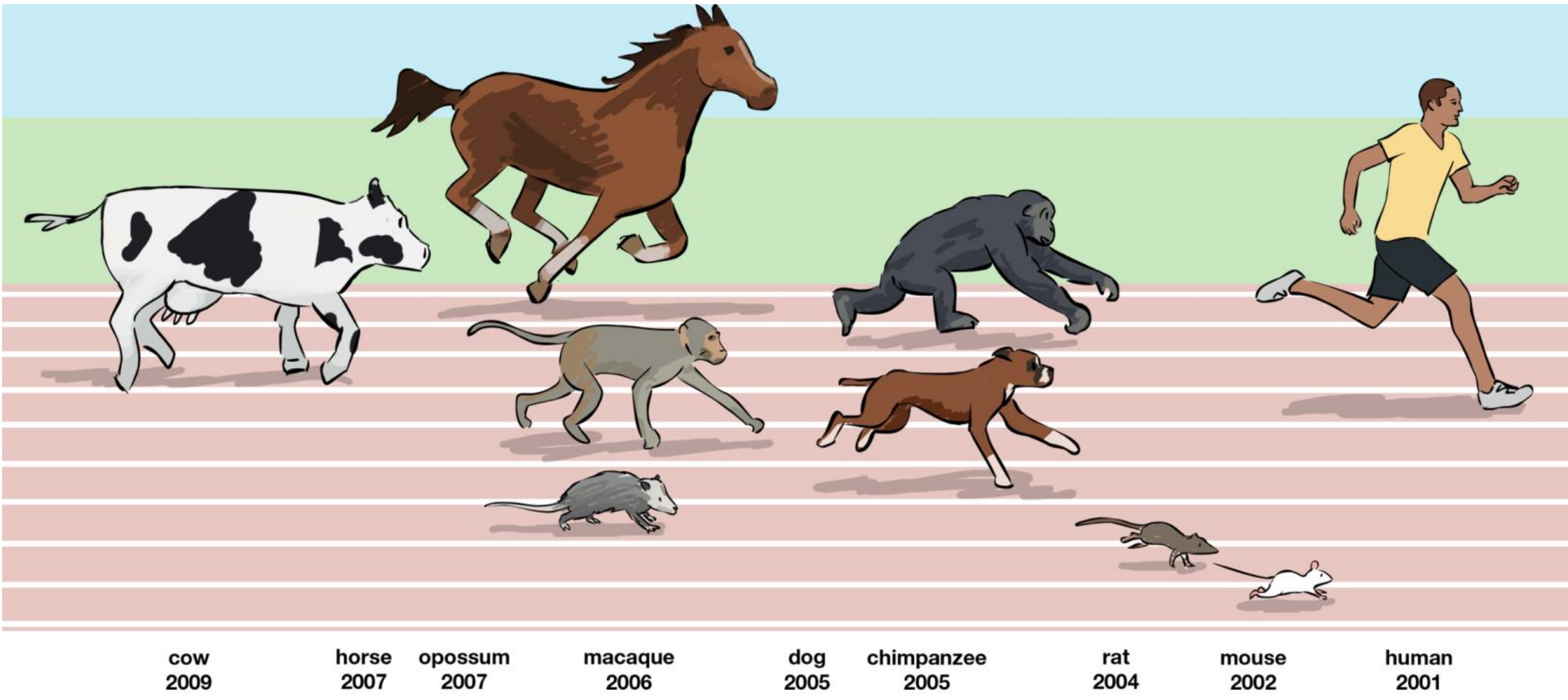
CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

BIOINFORMATICS ALGORITHMS
An Active Learning Approach
by Phillip Compeau & Pavel Pevzner


cow
2009
horse
2007
opossum
2007
macaque
2006
dog
2005
chimpanzee
2005
rat
2004
mouse
2002
human
2001

Why Do We Sequence 1000s of Species?

Applications in

- **medicine** (mouse genome)
- **agriculture** (rice genome)
- **biotechnology** (genomes of energy-producing bacteria),
- etc., etc., etc.

Few Mutations Can Make a Big Difference…

- Different people have slightly different genomes: on average, roughly 1 mutation in 1000 nucleotides.

- The 1 in 1000 nucleotides difference accounts for height, high cholesterol susceptibility, and 7000+ genetic diseases.

```
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGA
TCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTAT
CGATCGATCGATCGATTATCTACGATCGATCGATCGATCA
CTATACGAGCTACTACGTACGTACGATCGCGGGACTATTA
TCGACTACAGATAAAACATGCTAGTACAACAGTATACATA
GCTGCGGGATACGATTAGCTAATAGCTGACGATATCCGAT

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGA
TCAGCTACAACATCGTAGCTACGATGCATTAGCAAGCTAT
CGATCGATCGATCGATTATCTACGATCGATCGATCGATCA
CTATACGAGCTACTACGTACGTACGATCGCGTGACTATTA
TCGACTACAGATGAAACATGCTAGTACAACAGTATACATA
GCTGCGGGATACGATTAGCTAATAGCTGACGATATCCGAT
```

Emergence of Personalized Medicine

- **2010:** Nicholas Volker became the first child to be saved by genome sequencing.
  - Doctors could not diagnose his condition; he went through dozens of surgeries.
  - Sequencing revealed a mutation in a gene linked to a defect in his immune system.
  - This led doctors to use immunotherapy, which saved the child.

From Reference Genome to Personal Genomes

- **Reference human genome** assembled in 2000.

```
CTGAGGATGGACTACGCTACTACTGATAGCTGTTT    reference genome

GAGGA      CCACG       TGA-AG
CTGA    GGACC     ACTAC  A-AGCT         reads
     GATGG   ACGCT            TGTTT

CTGAGGATGGACCACGCTACTACTGA-AGCTGTTT    personal genome
```

Exact Pattern Matching

- Where does a read match the reference genome *exactly*?

- **Pattern Matching Problem:**
  - **Input:** A string *Pattern* (read) and a string *Text* (genome).
  - **Output:** All positions in *Text* where *Pattern* appears as a substring.

Approximate Pattern Matching

- Where does a read match the reference genome *approximately*?

- **Approximate Pattern Matching Problem:**
  - **Input:** A string *Pattern*, a string *Text,* and an integer *d*
  - **Output:** All positions in *Text* where *Pattern* appears as a substring with at most *d* mismatches.

Multiple Pattern Matching

- Where do billions of reads match the reference genome?

- **Multiple Pattern Matching Problem:**
  - **Input:** A set of strings *Patterns* and a string *Text.*
  - **Output:** All positions in *Text* where a string from *Patterns* appears as a substring.

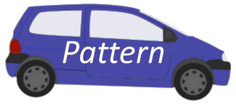
Outline

- From Genome Sequencing to Pattern Matching
- **Brute Force Approach to Pattern Matching**
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

A Brute Force Approach to Exact Pattern Matching

Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC

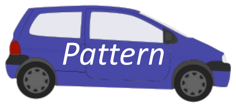


Pattern drives along *Text*

```
p a n a m a b a n a n a s
n a n a
```

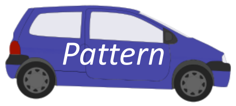
Text

Pattern



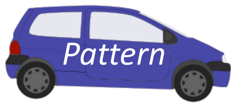
p a n a m a b a n a n a s *Text*

n a n a *Pattern*



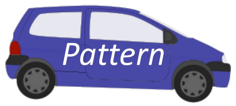
p **a** n a m a b a n a n a s *Text*

n a n a *Pattern*



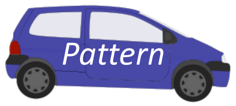
p a **n** a m a b a n a n a s *Text*

n a n a *Pattern*



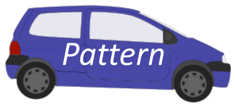
p a **n a** m a b a n a n a s *Text*

n a n a *Pattern*



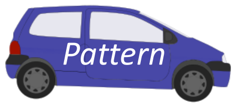
p a **n a m** a b a n a n a s *Text*

n a n a *Pattern*



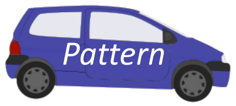
p a n **a** m a b a n a n a s *Text*

n a n a *Pattern*



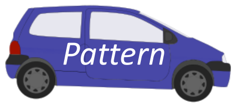
p a n a **m** a b a n a n a n a s *Text*

n a n a *Pattern*



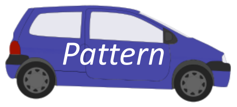
p a n a m **a** b a n a n a s *Text*

n a n a *Pattern*



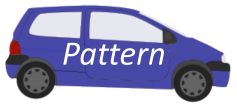
p a n a m a **b** a n a n a s *Text*

n a n a *Pattern*



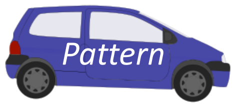
p a n a m a b **a** n a n a s *Text*

n a n a *Pattern*



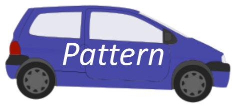
p a n a m a b a **n** a n a s *Text*

n a n a *Pattern*

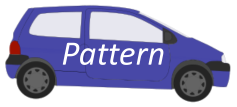


p a n a m a b a **n a** n a s *Text*

n a n a *Pattern*



```
p a n a m a b a **n a n** a s        Text
                n a n a              Pattern
```

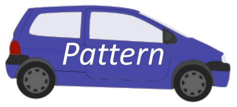


Pattern Found!

p a n a m a b a **n a n a** s *Text*

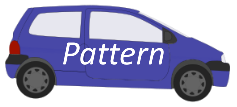
n a n a *Pattern*

•



p a n a m a b a n **a** n a s *Text*

n a n a *Pattern*

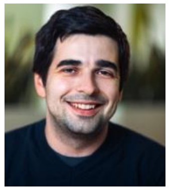


Brute Force Approach Is Fast!

- **single** *Pattern*: O(|*Text*| • |*Pattern*|)

The runtime of the **Knuth-Morris-Pratt** algorithm: O(|*Text*|)

(wait for the lecture on algorithmic challenges in pattern matching)



Brute Force Approach is Slow for **Billions** of Patterns

- **single** *Pattern*: $O(|\textit{Text}| \cdot |\textit{Pattern}|)$
- **multiple** *Patterns*:

$$O\left(\sum_{\substack{\text{all strings } \textit{Pattern} \\ \text{In } \textit{Patterns}}} |\textit{Text}| \cdot |\textit{Pattern}|\right)$$

Brute Force Approach is Slow for Billions of Patterns

- **single** *Pattern*: $O(|Text| \bullet |Pattern|)$
- **multiple** *Patterns*:

$$O\Big(\sum_{\substack{\text{all strings } Pattern \\ \text{In } Patterns}} |Text| \bullet |Pattern|\Big) =$$

$$O(|Text| \bullet |Patterns|)$$

For human genome:

- $|Text| \approx 3*10^9$
- $|Patterns| \approx 10^{12}$

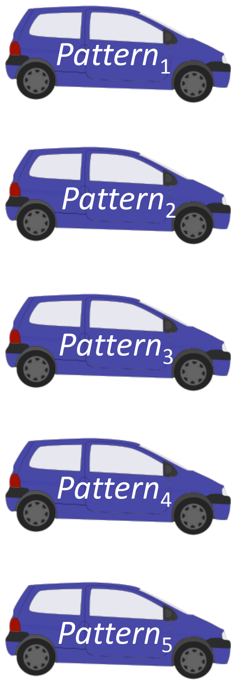
Outline

- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- **Herding Patterns into Trie**
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

Patterns Travel One at a Time

Genome

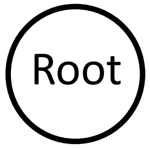
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC



Herding Patterns onto a Bus

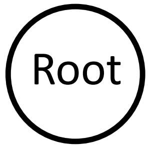
Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC



Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

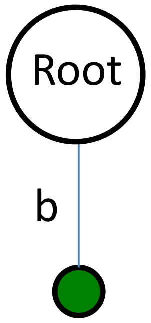
nab

antenna

bandana

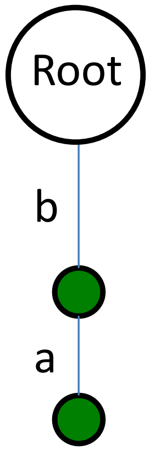
ananas

nana



Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

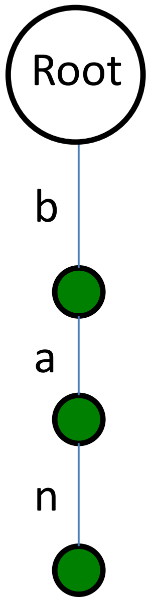
nab

antenna

bandana

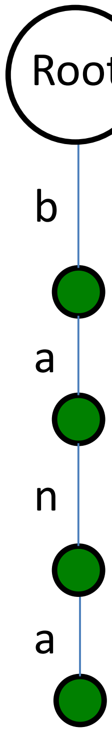
ananas

nana



Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

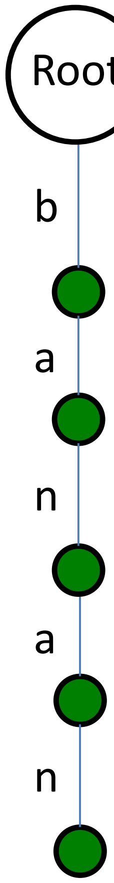
nab

antenna

bandana

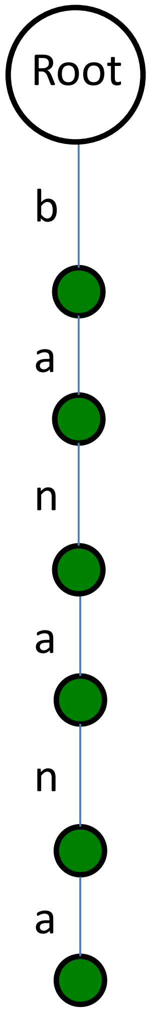
ananas

nana



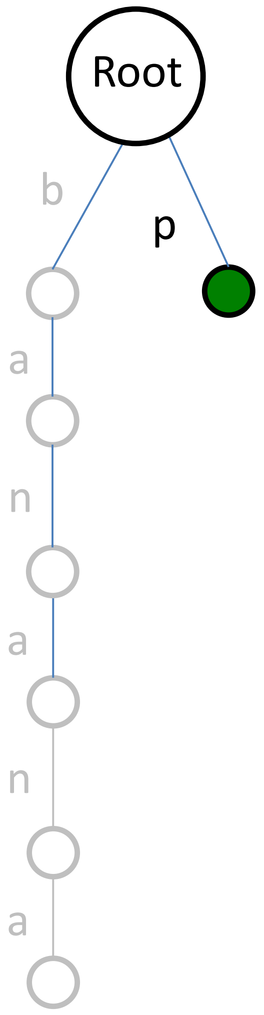
Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

and

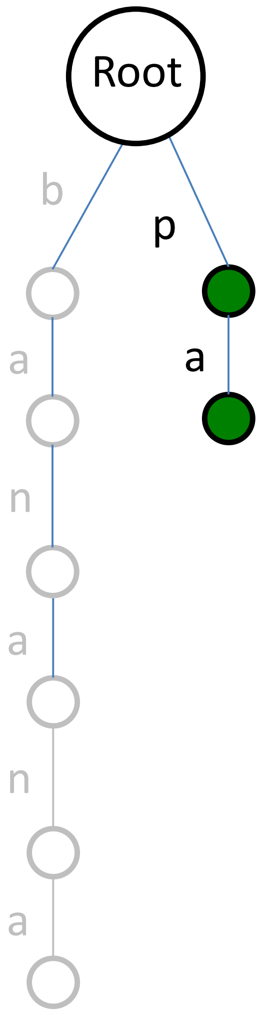
nab

antenna

bandana

ananas

nana



Patterns

banana

pan

and

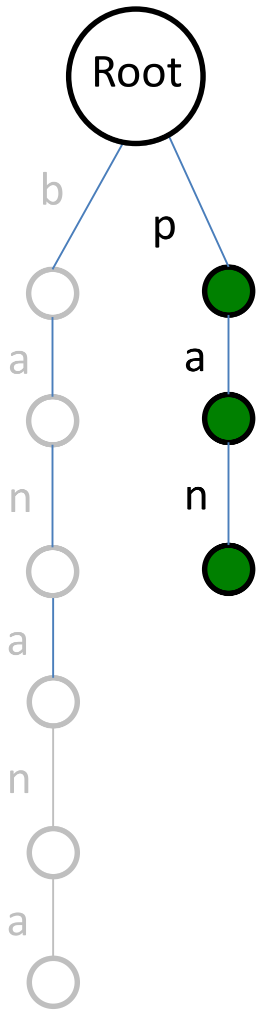
nab

antenna

bandana

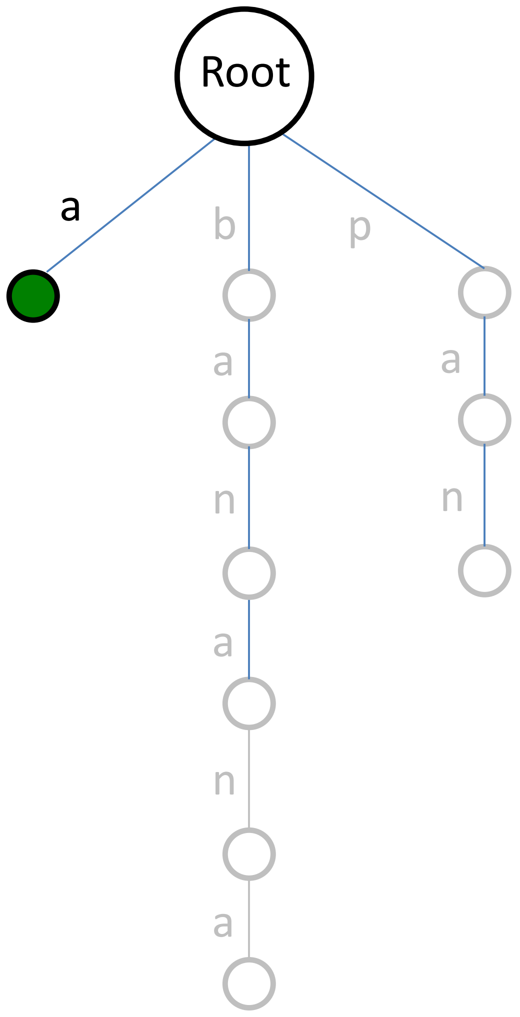
ananas

nana



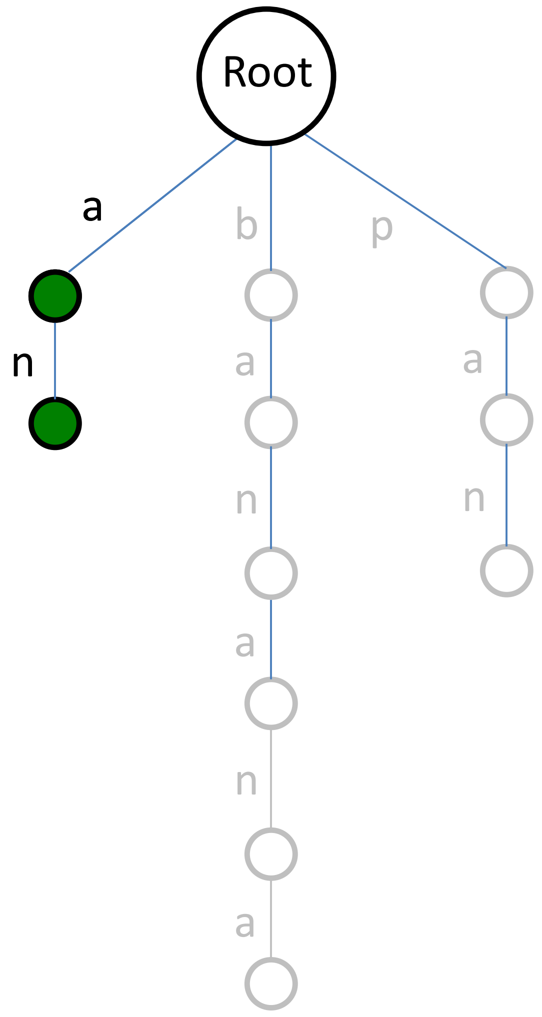
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



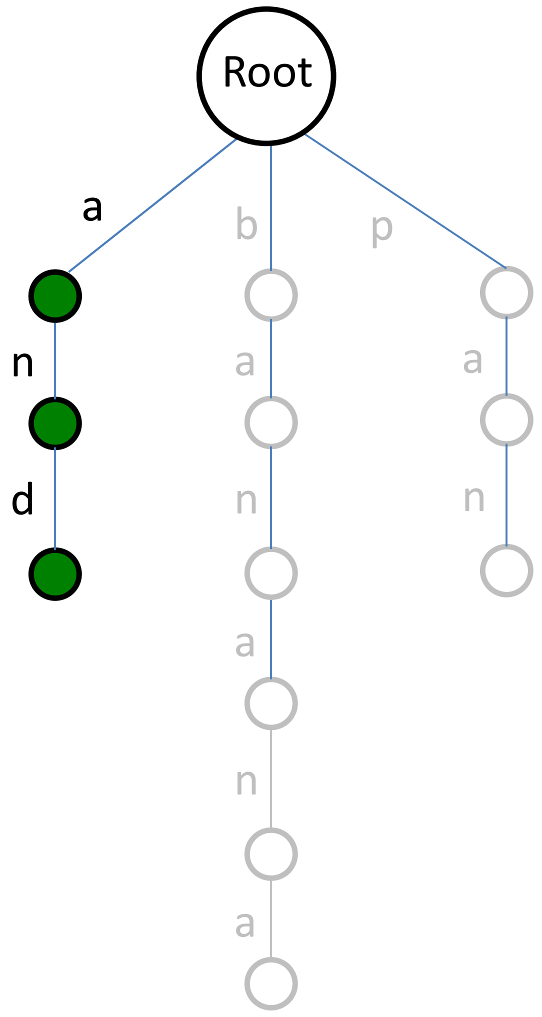
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



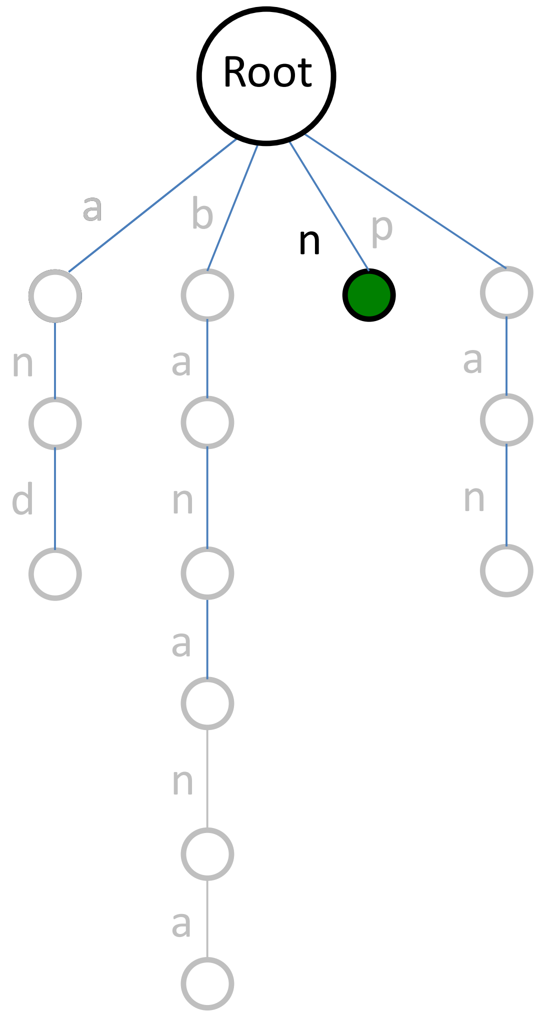
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



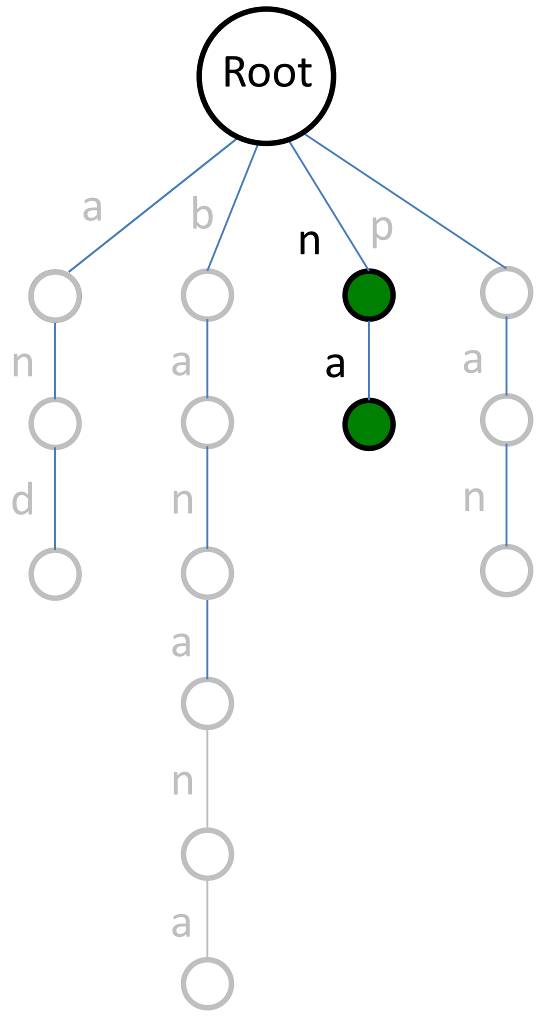
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



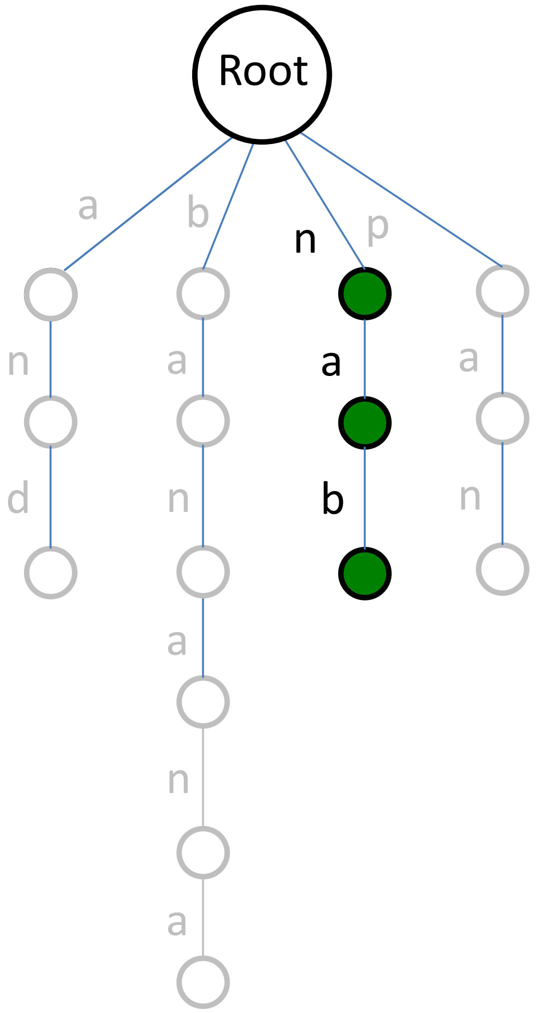
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



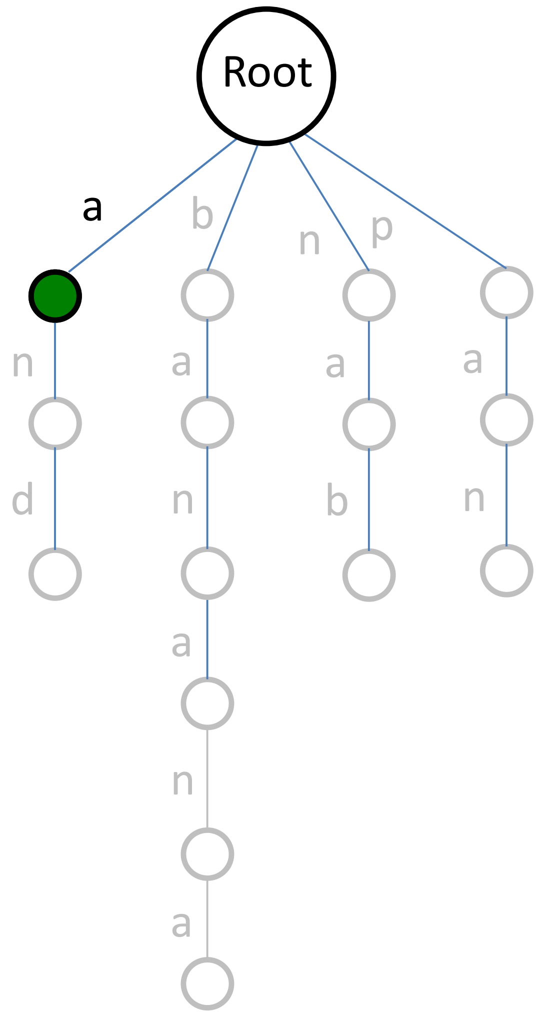
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



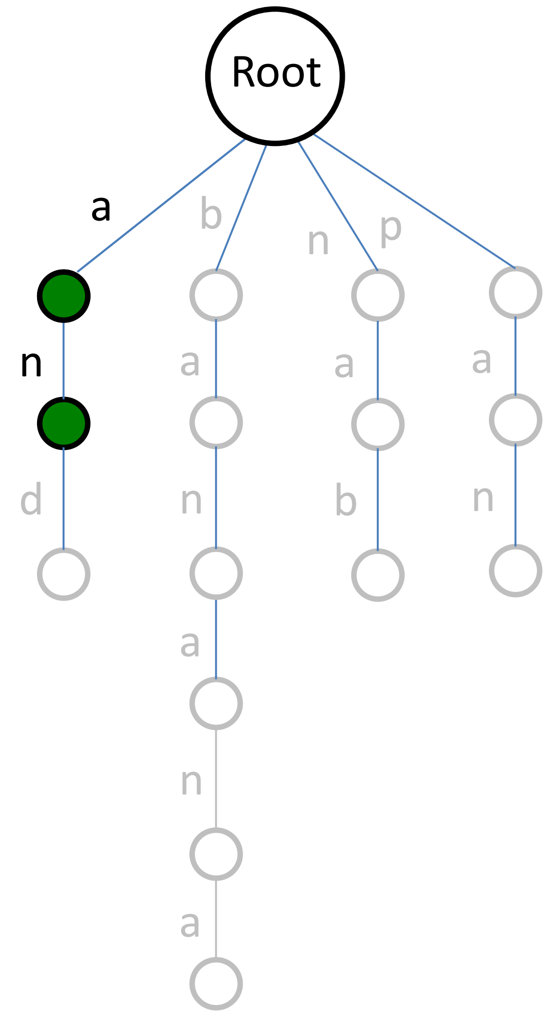
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



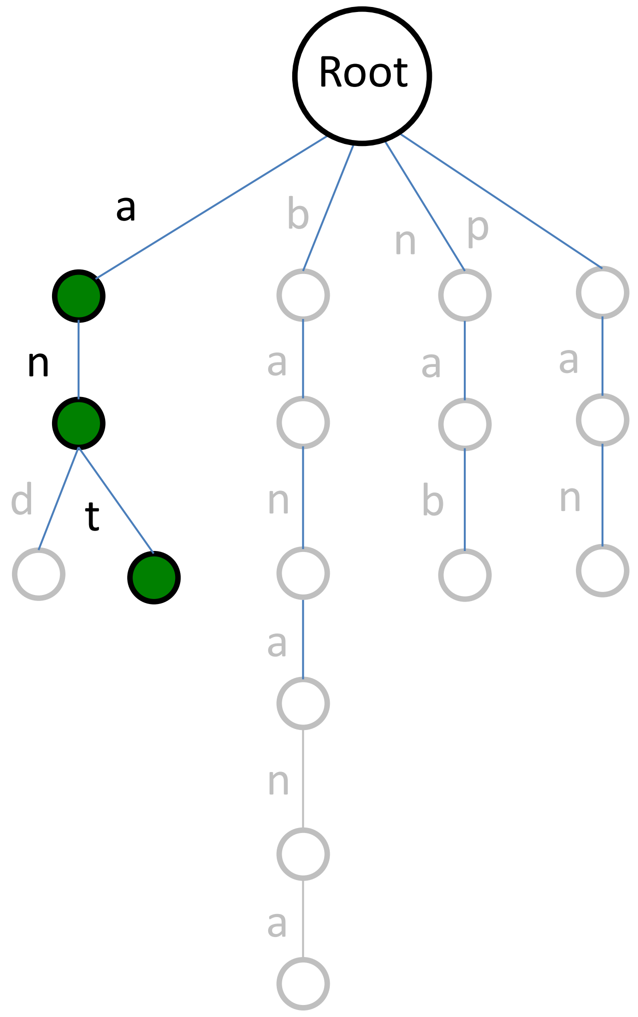
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



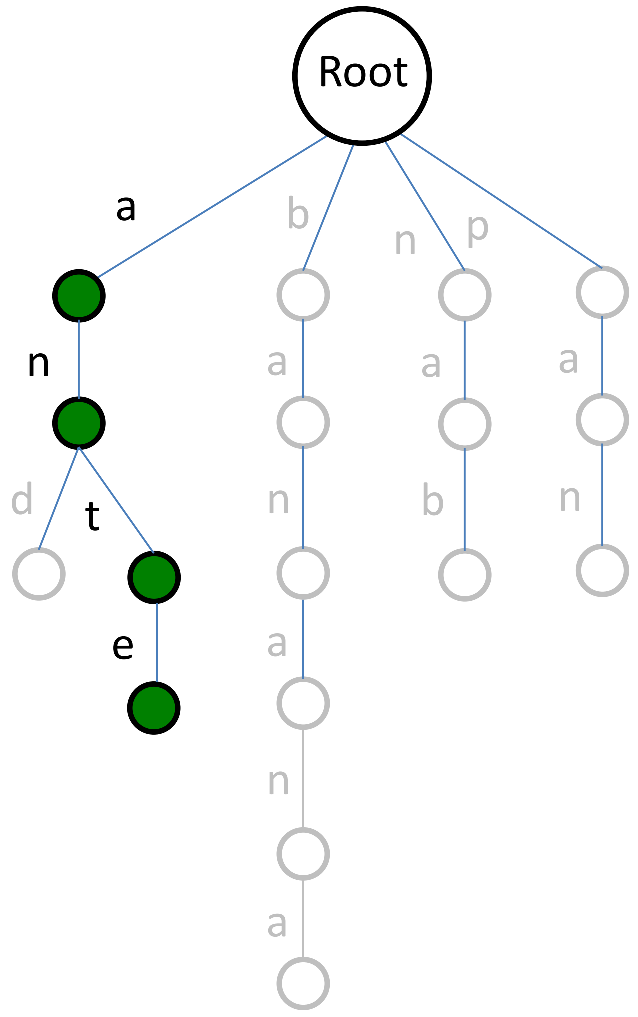
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

and

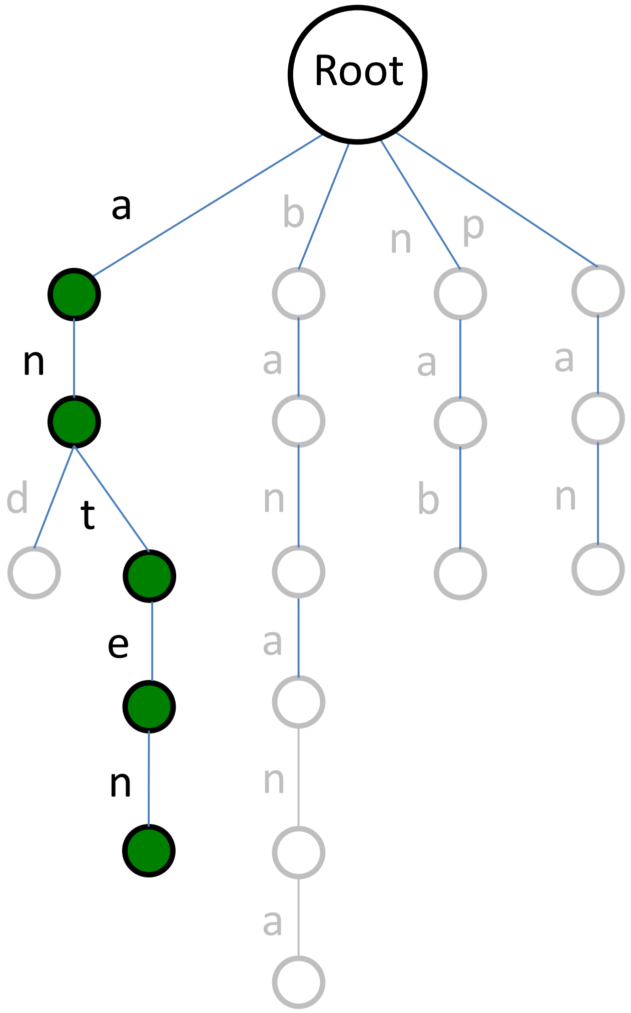
nab

antenna

bandana

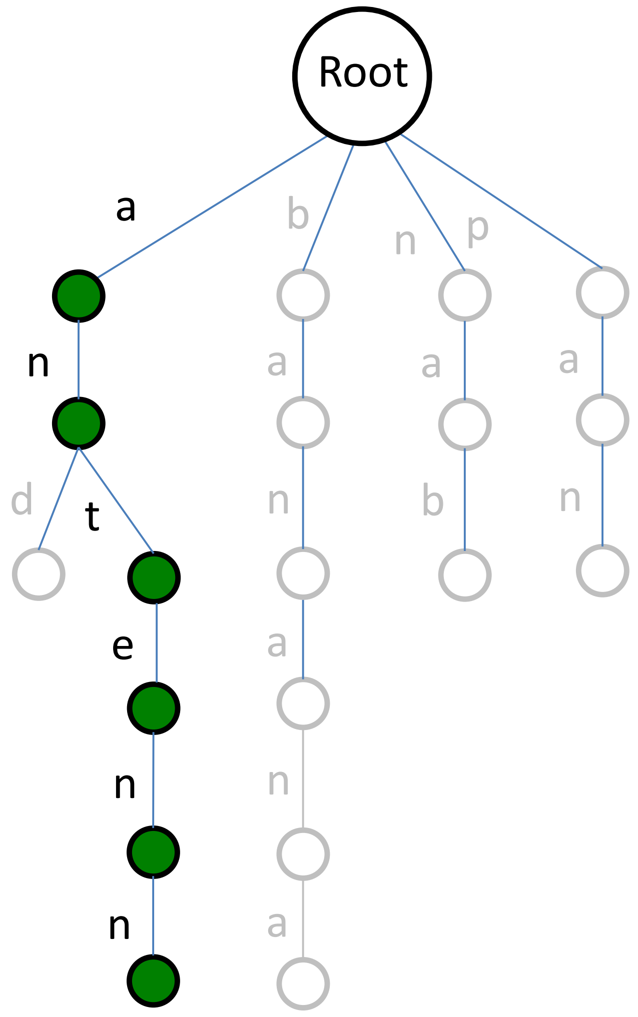
ananas

nana



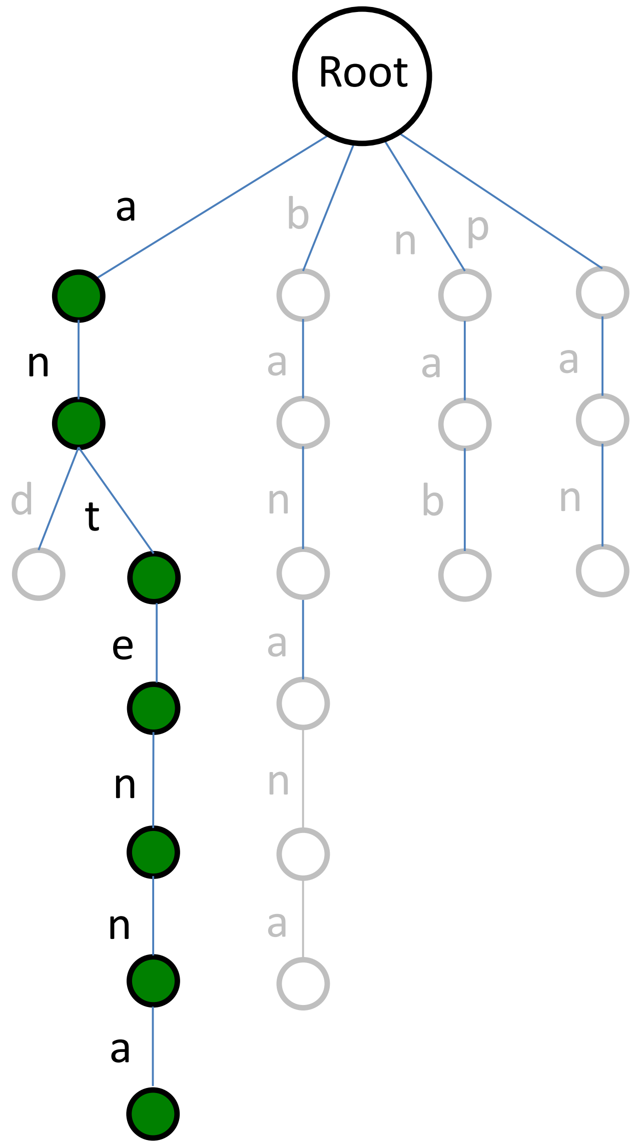
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



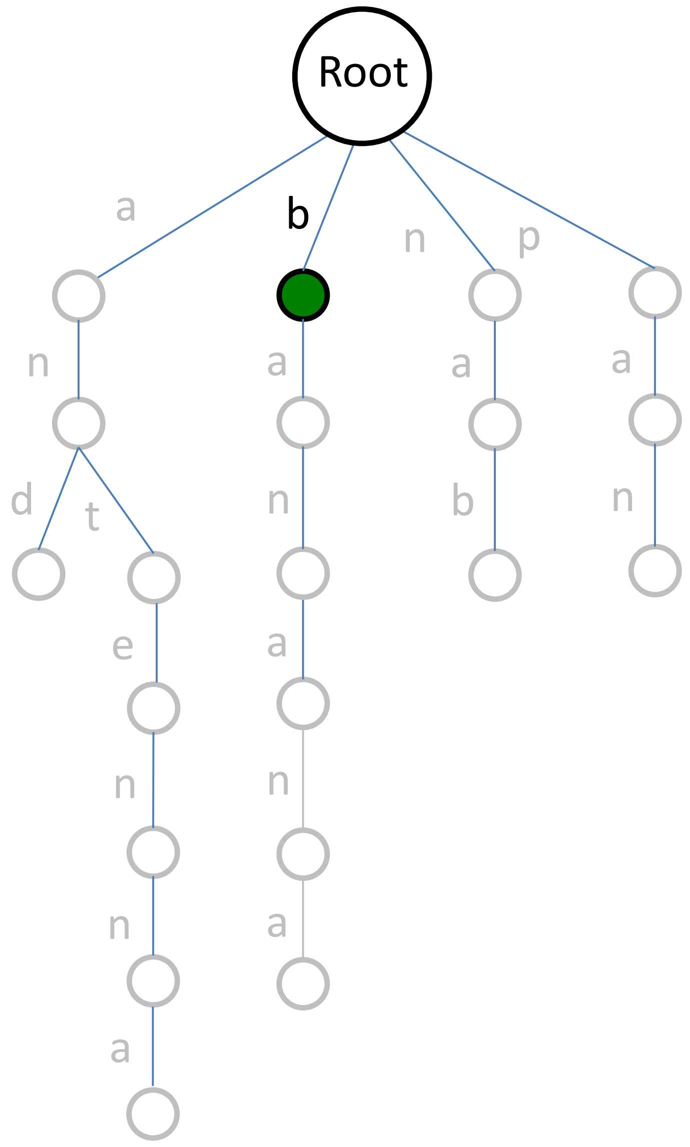
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



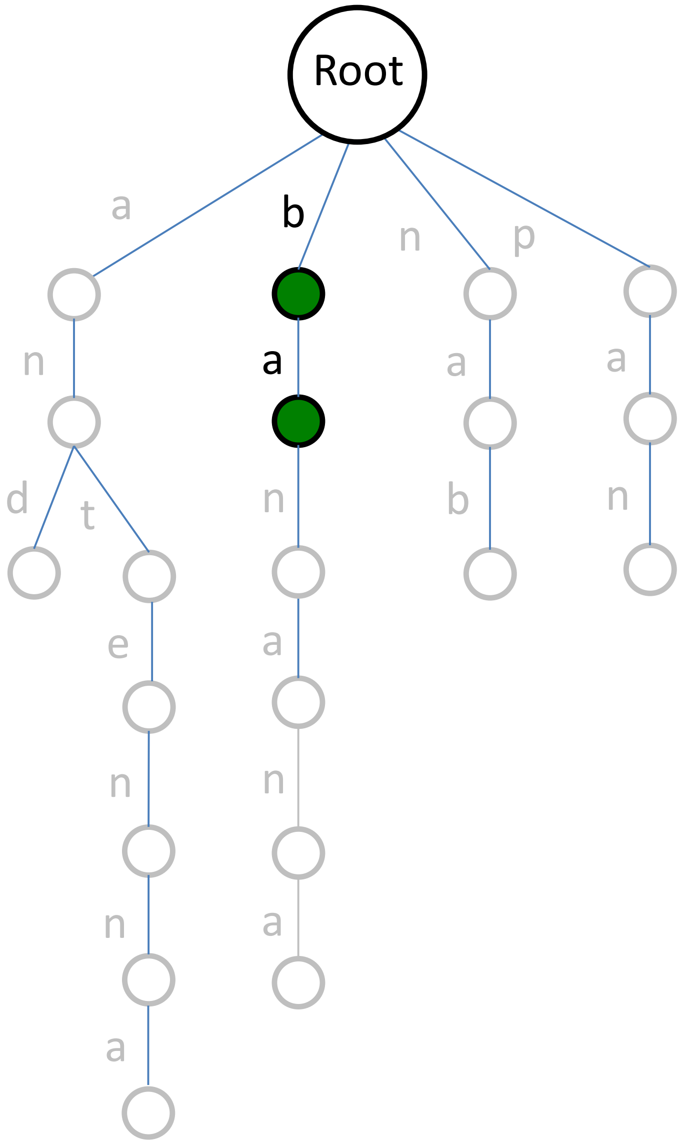
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



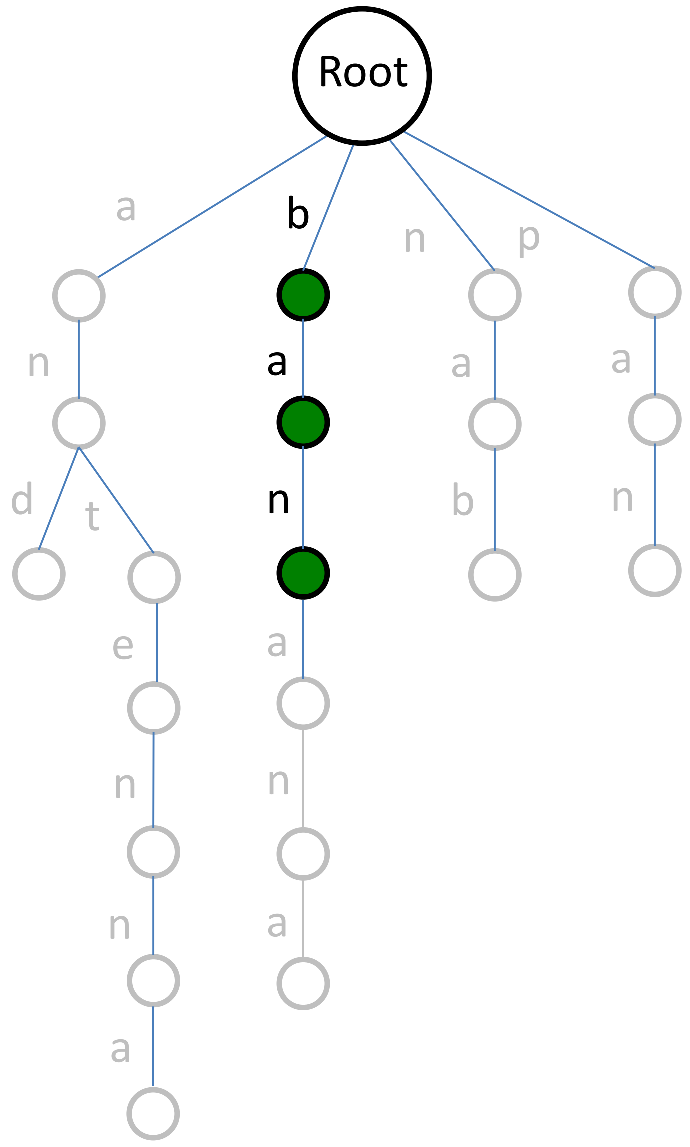
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



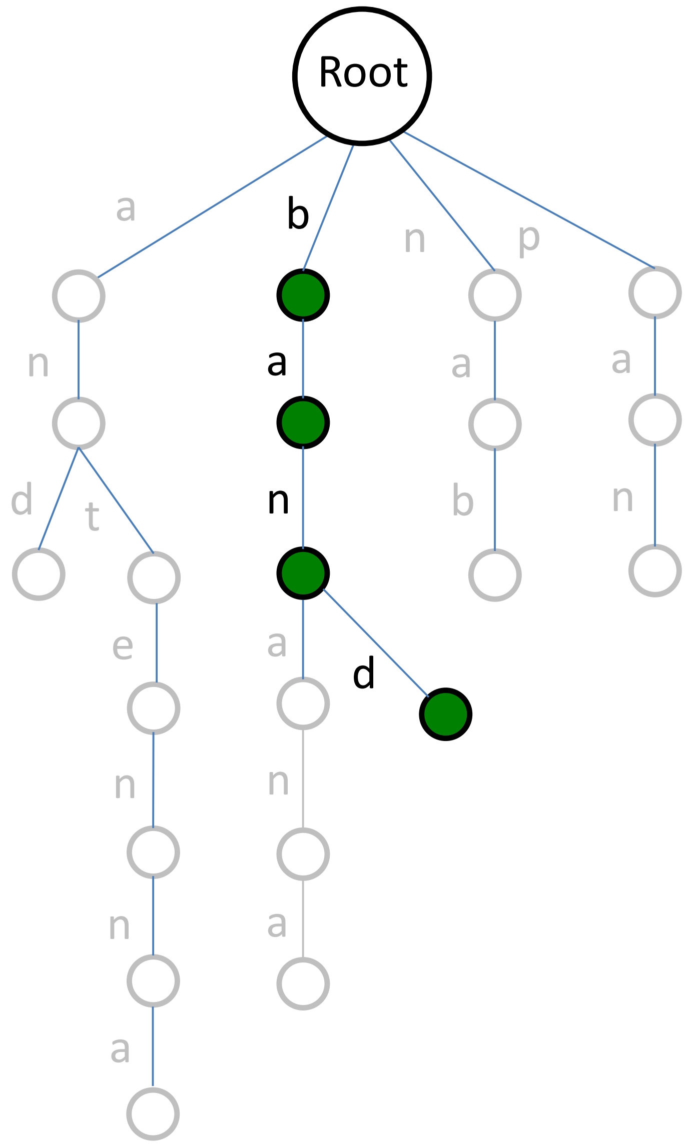
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



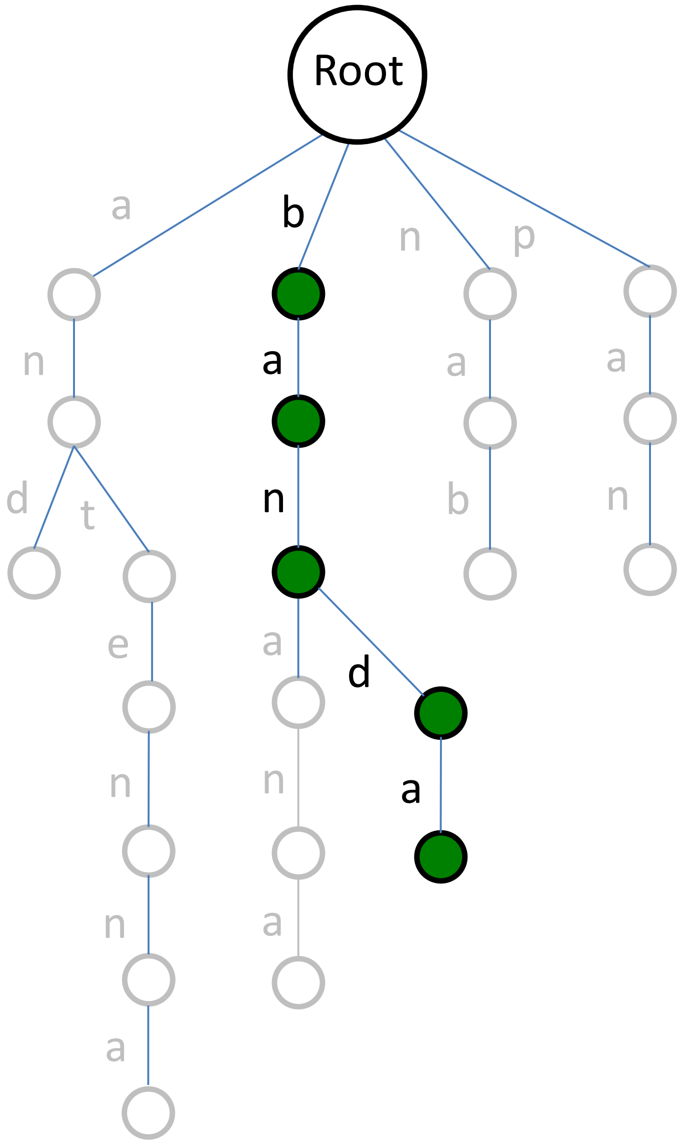
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



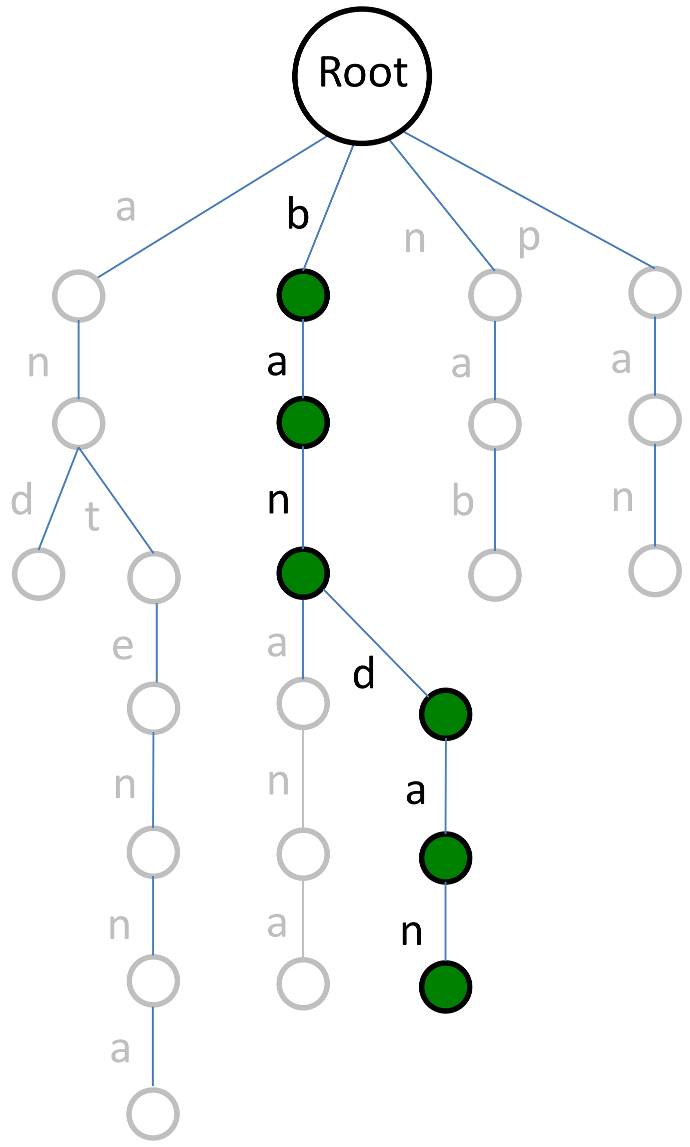
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



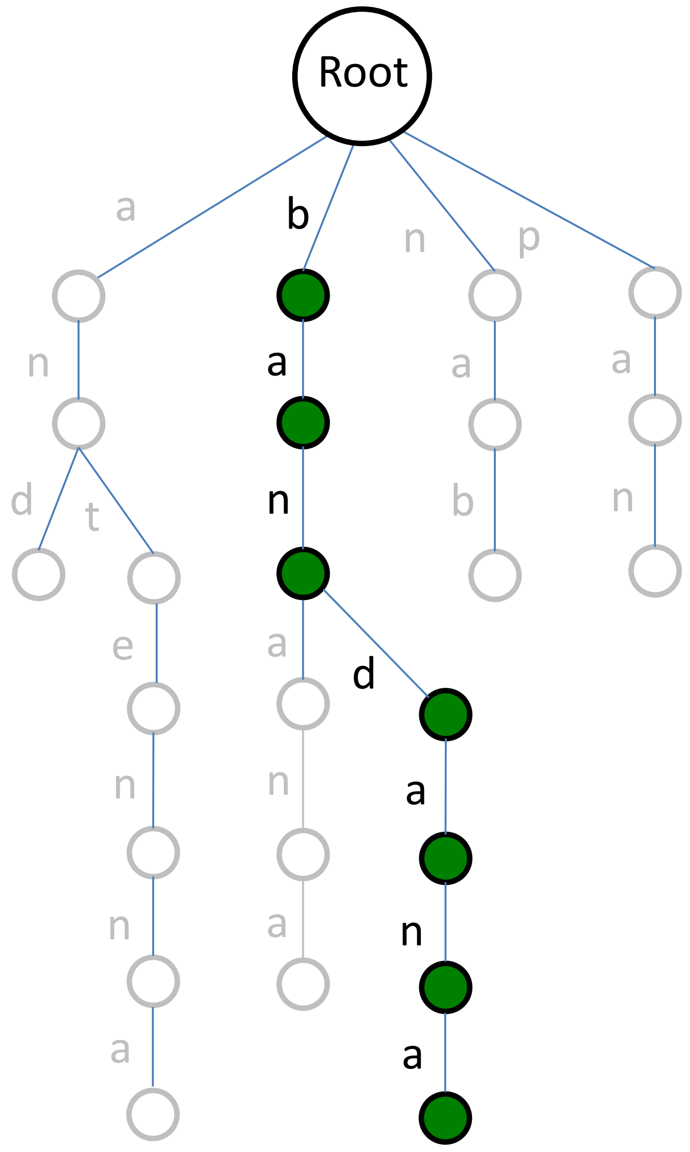
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



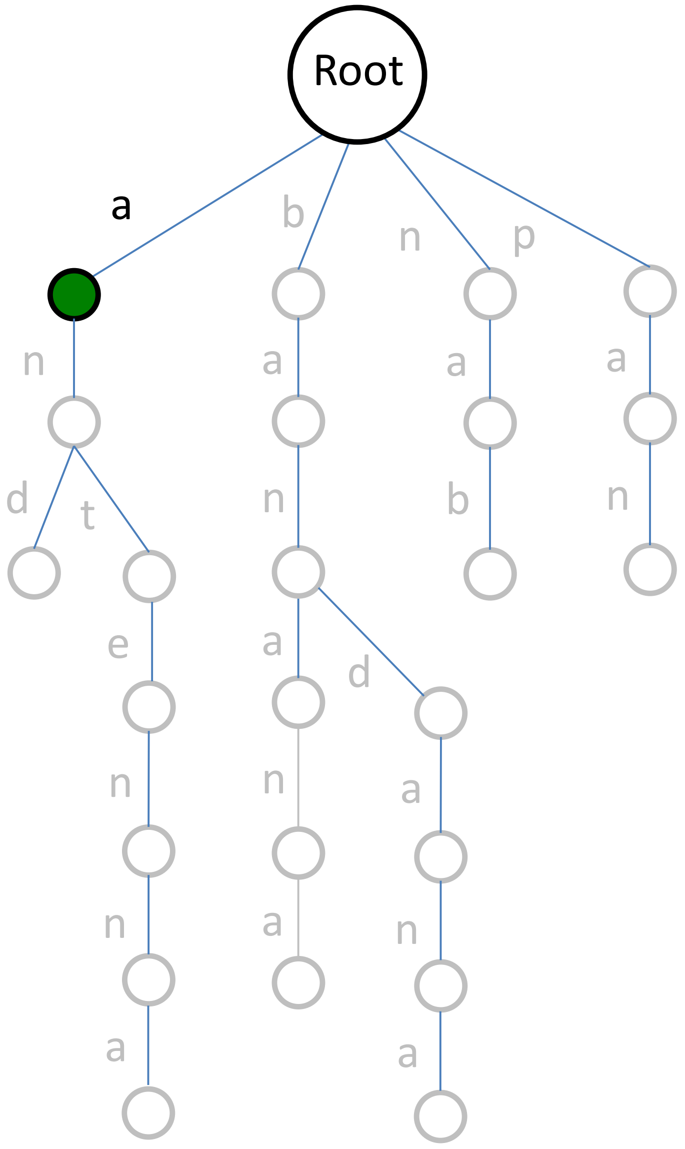
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

and

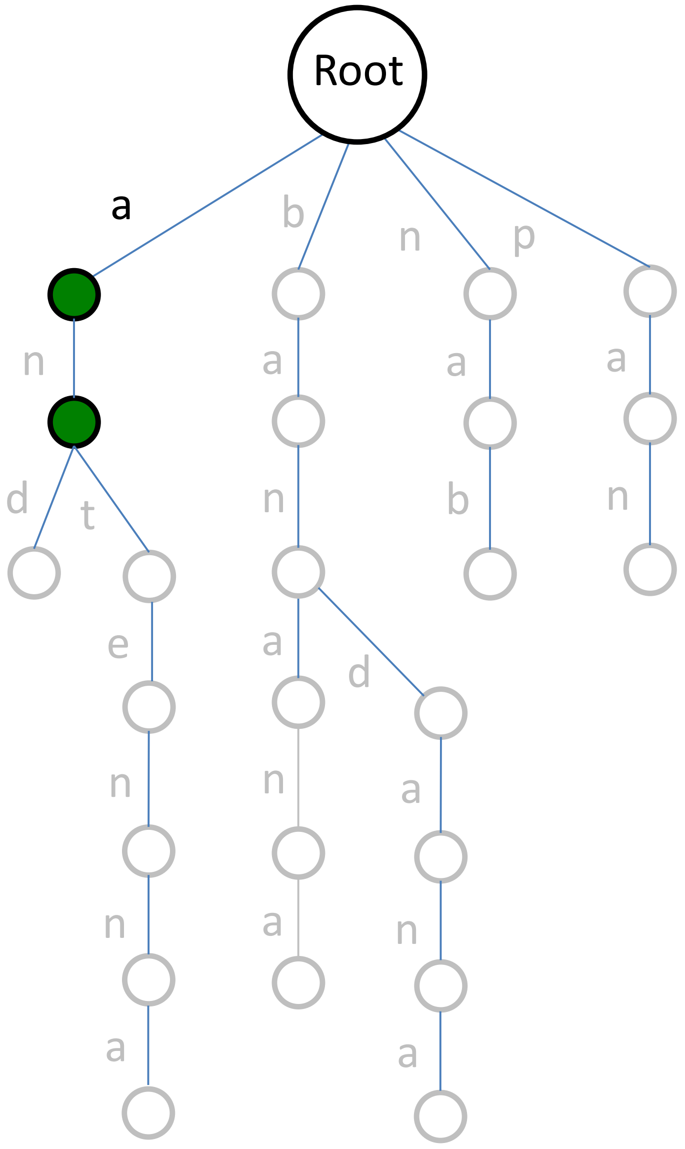
nab

antenna

bandana

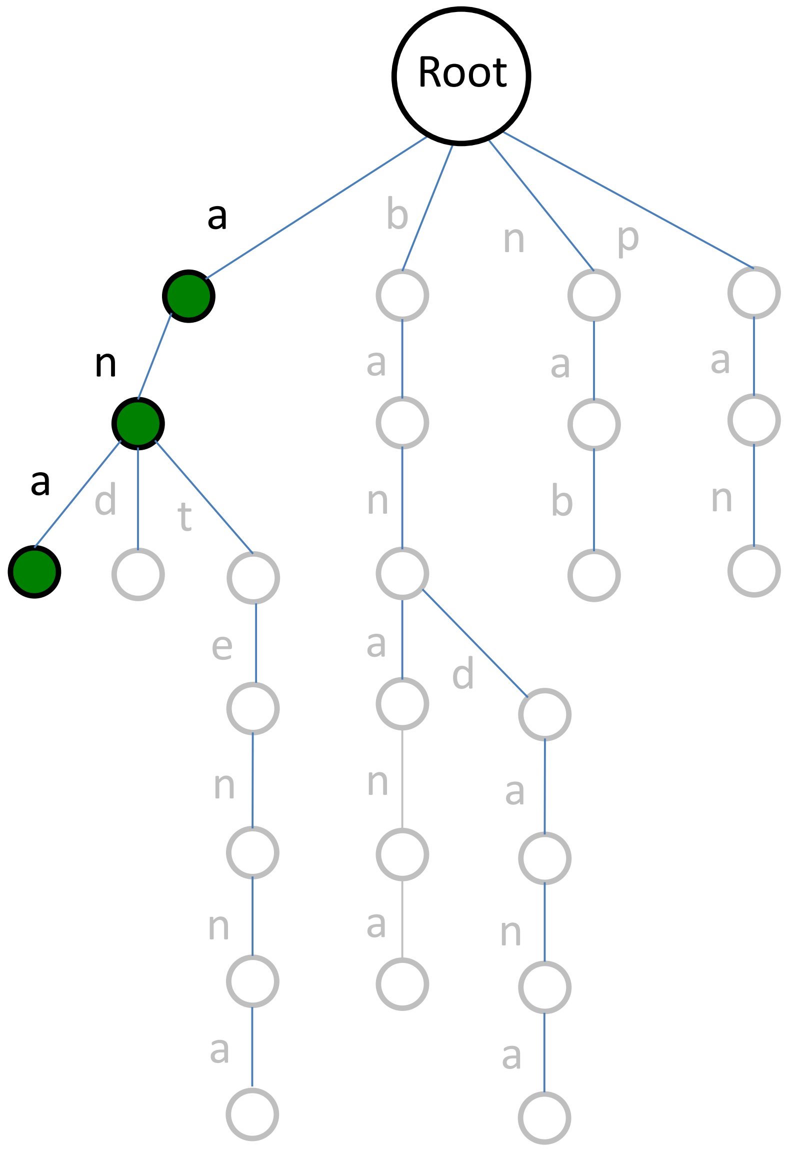
ananas

nana



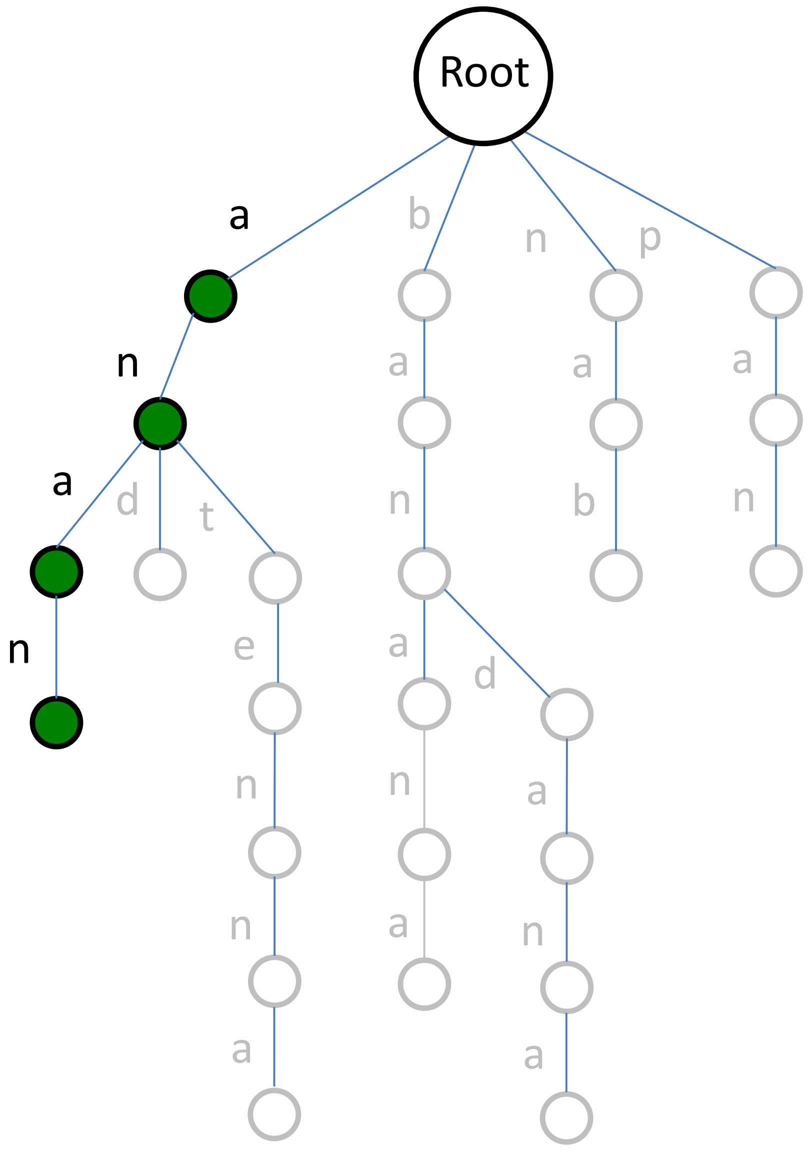
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



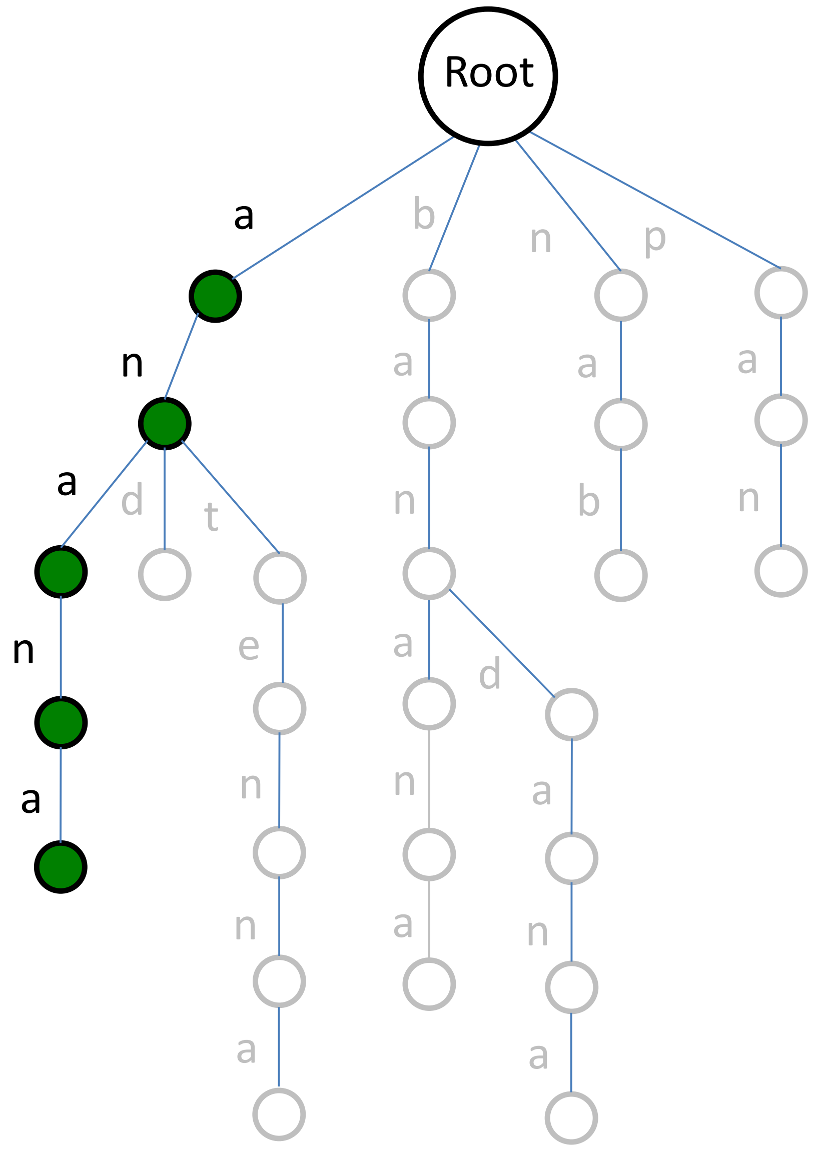
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



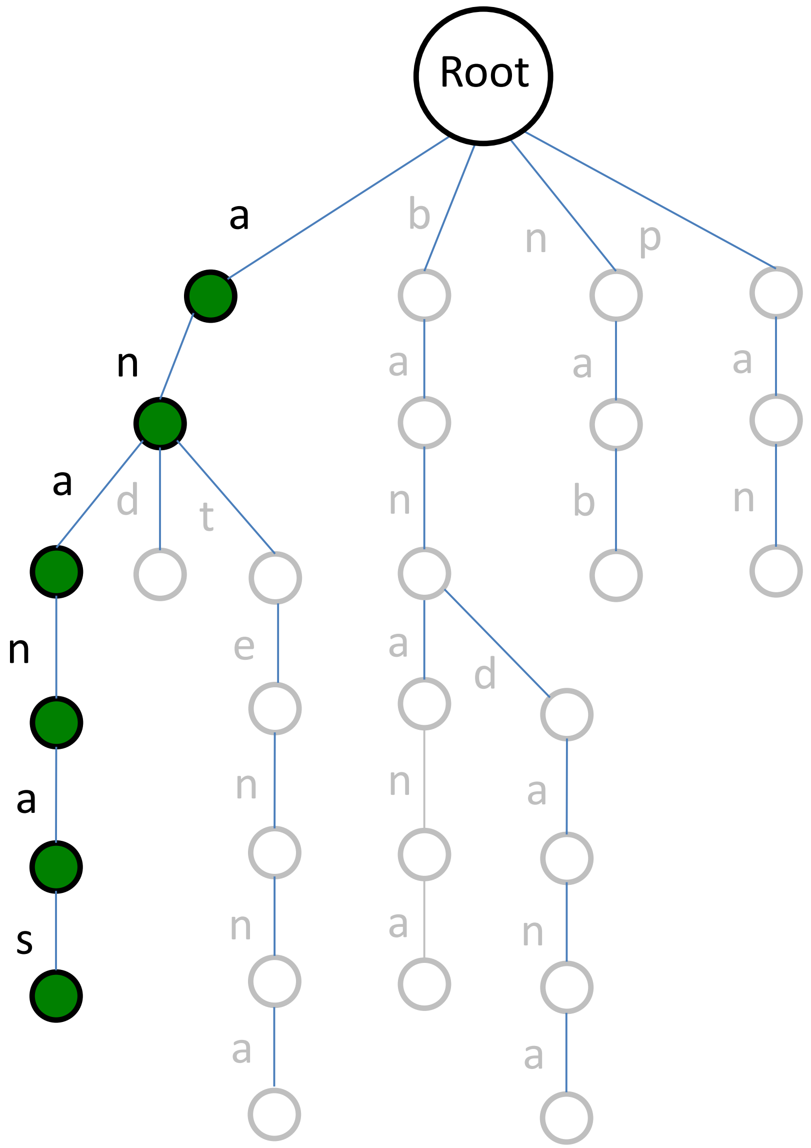
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



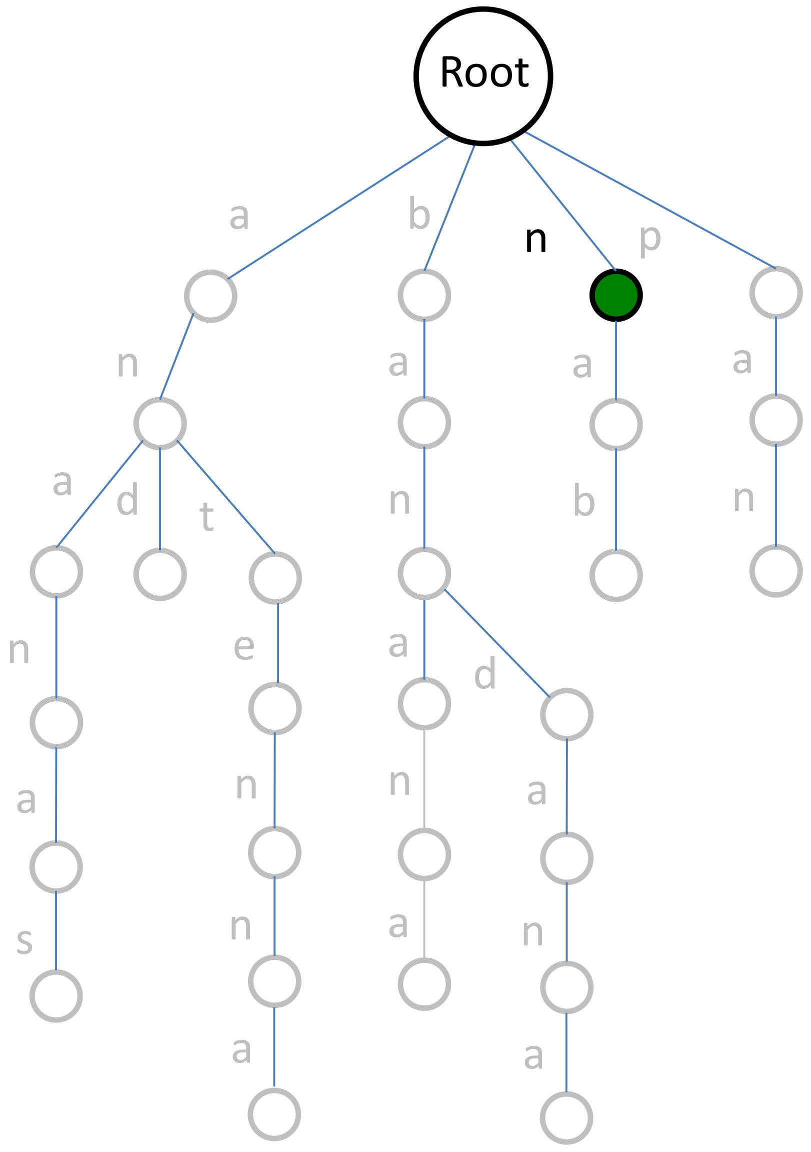
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



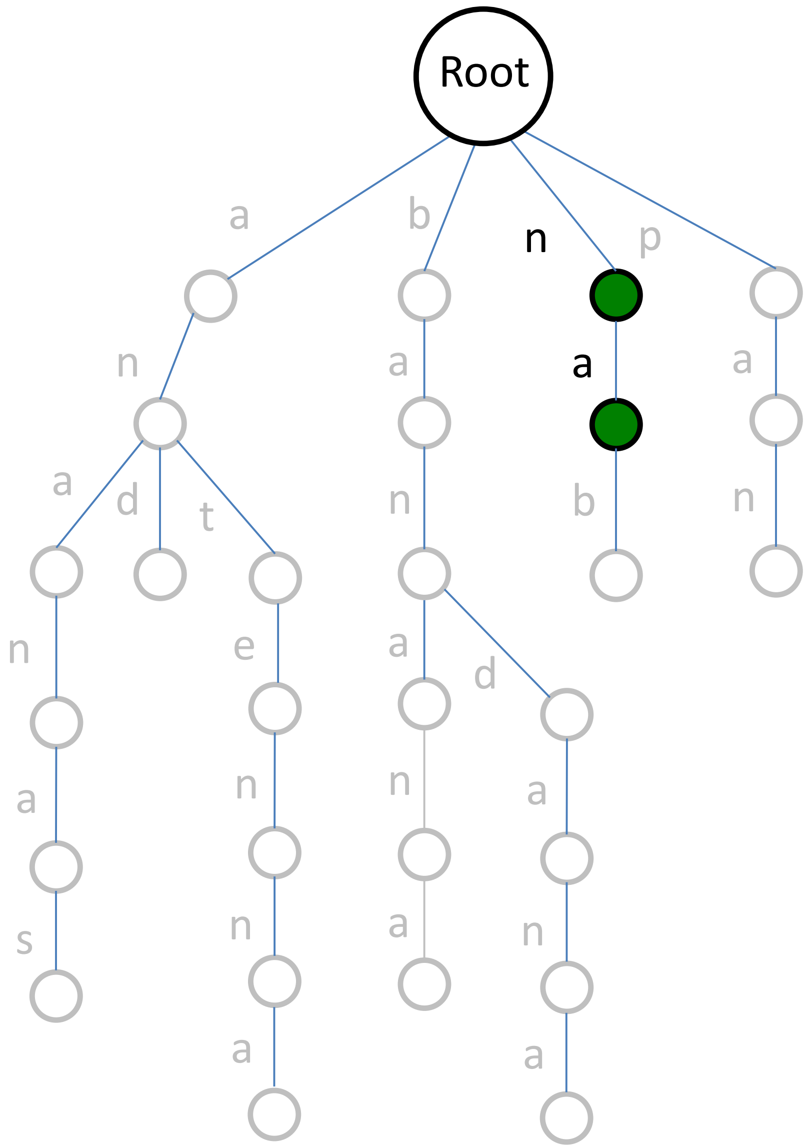
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



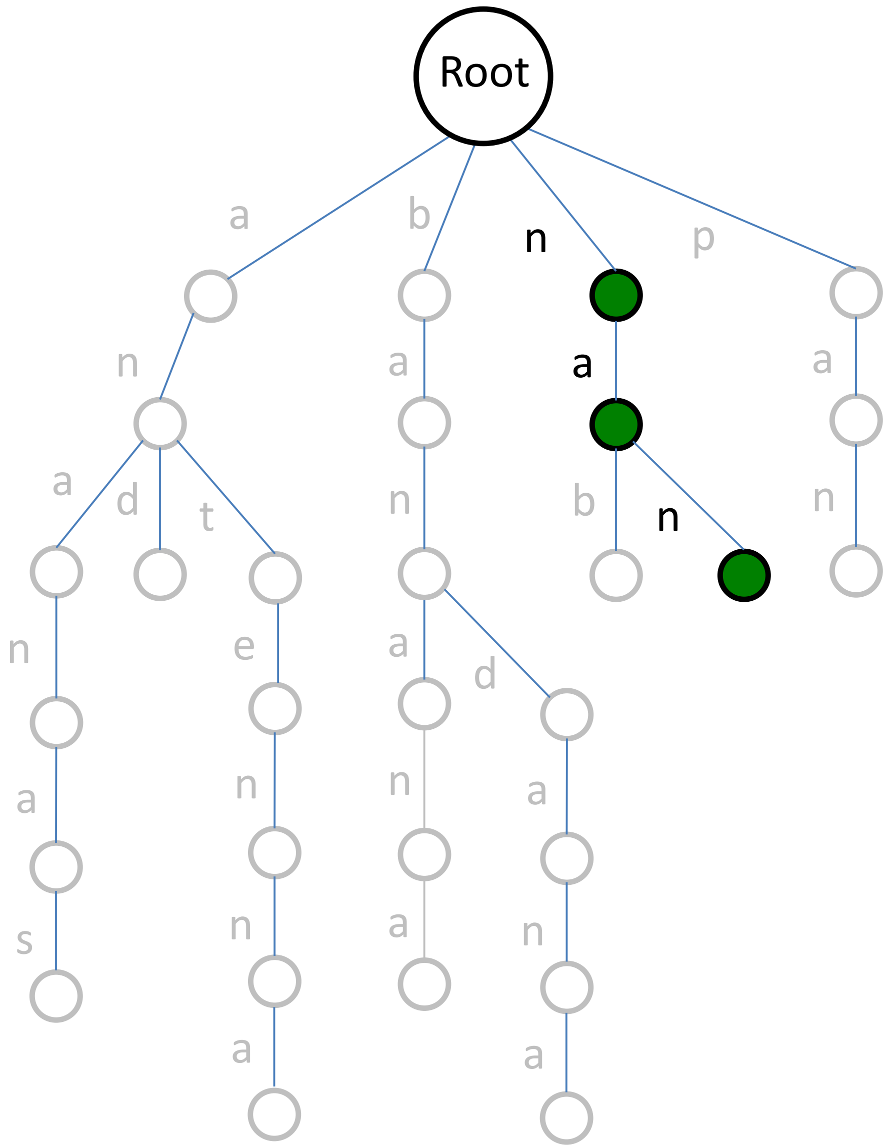
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



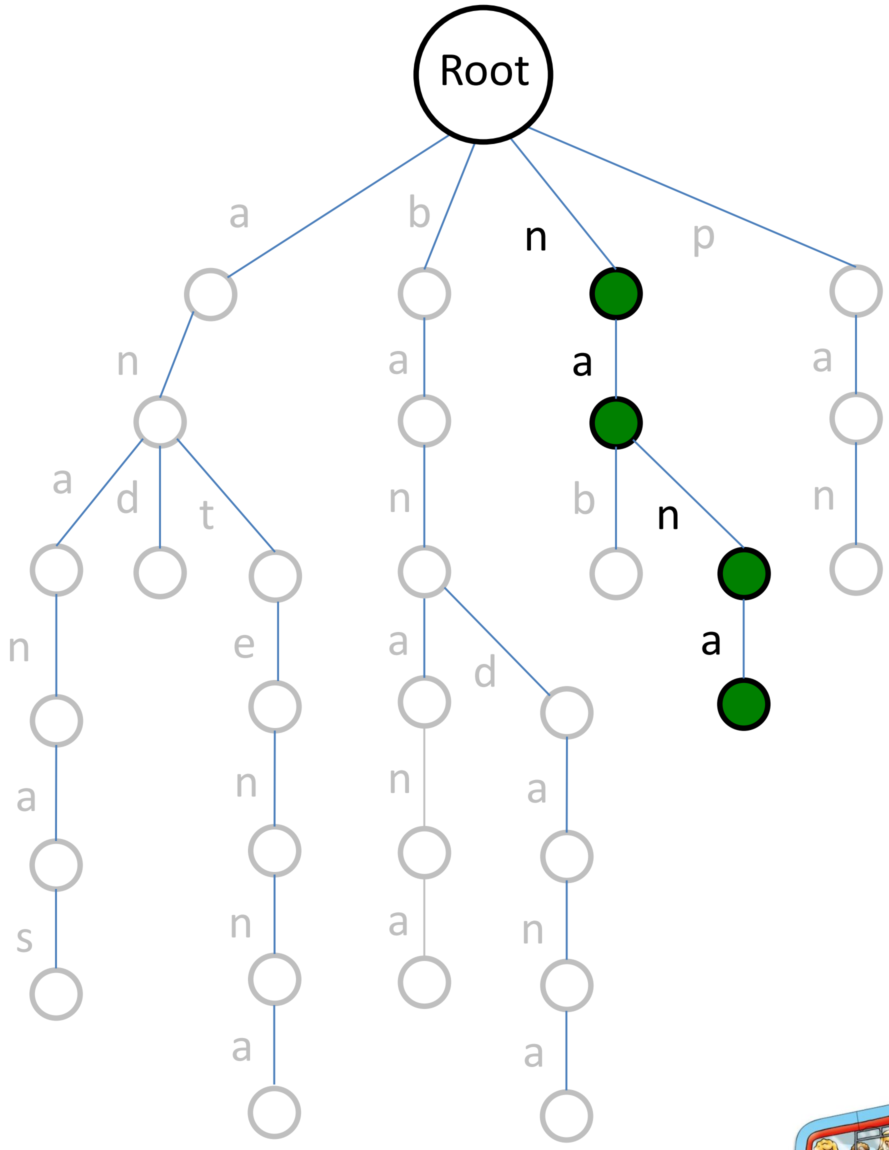
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

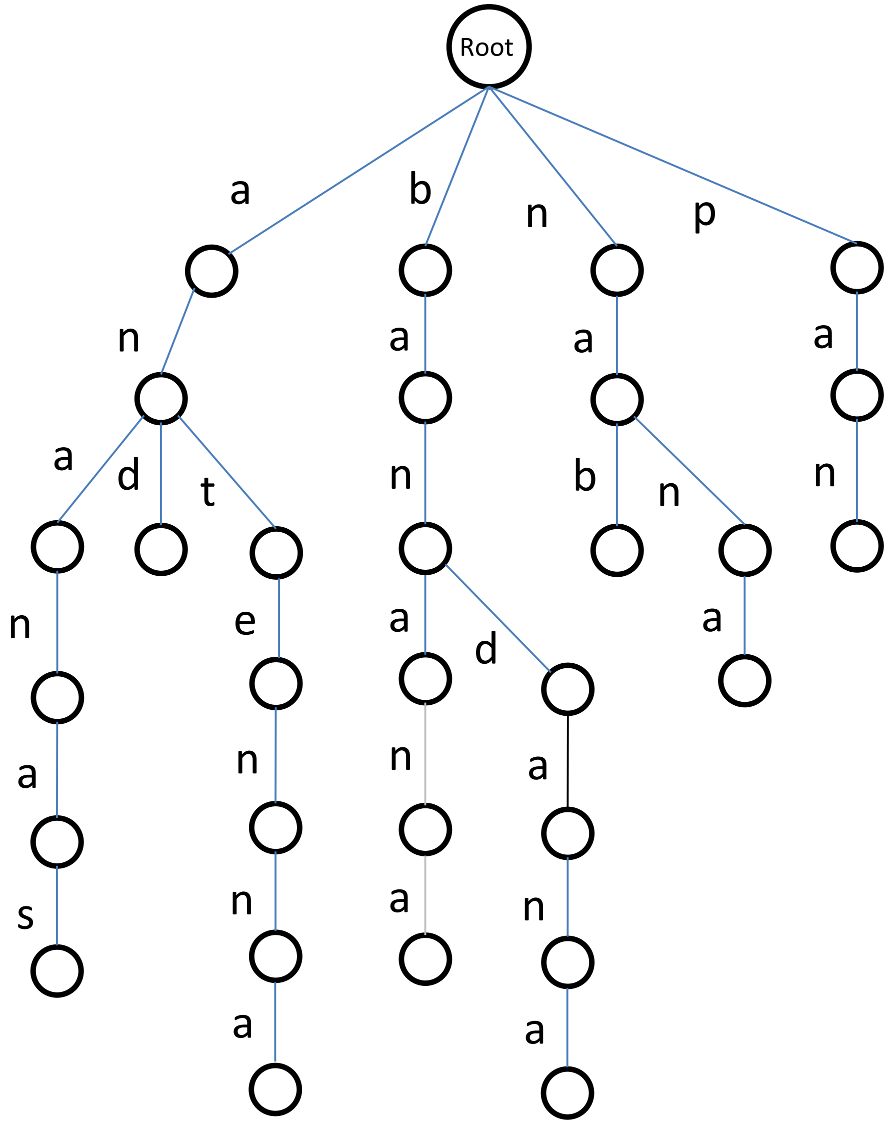


Trie(*Patterns*)

Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

p a n a m a b a n a n a s



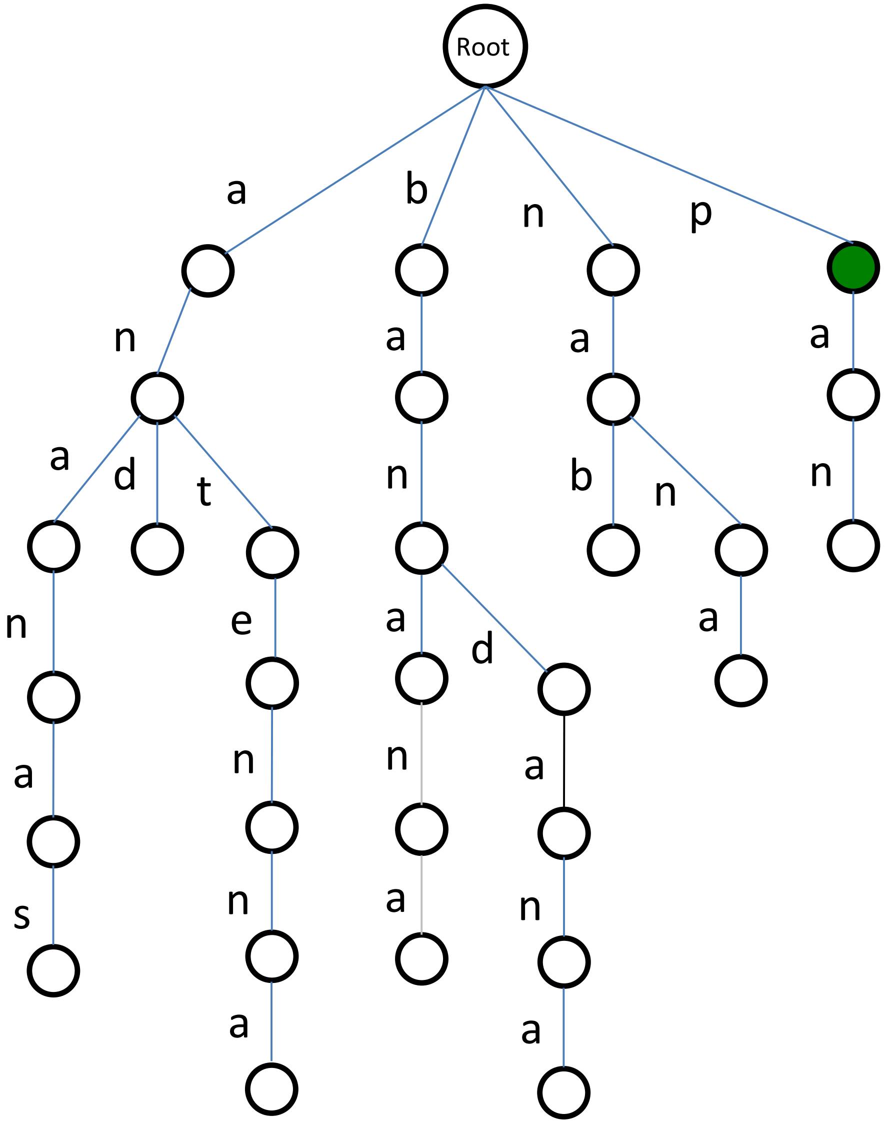
TrieMatching(*Text, Patterns*):

drive Trie(*Patterns*) along *Text* at each position of *Text*

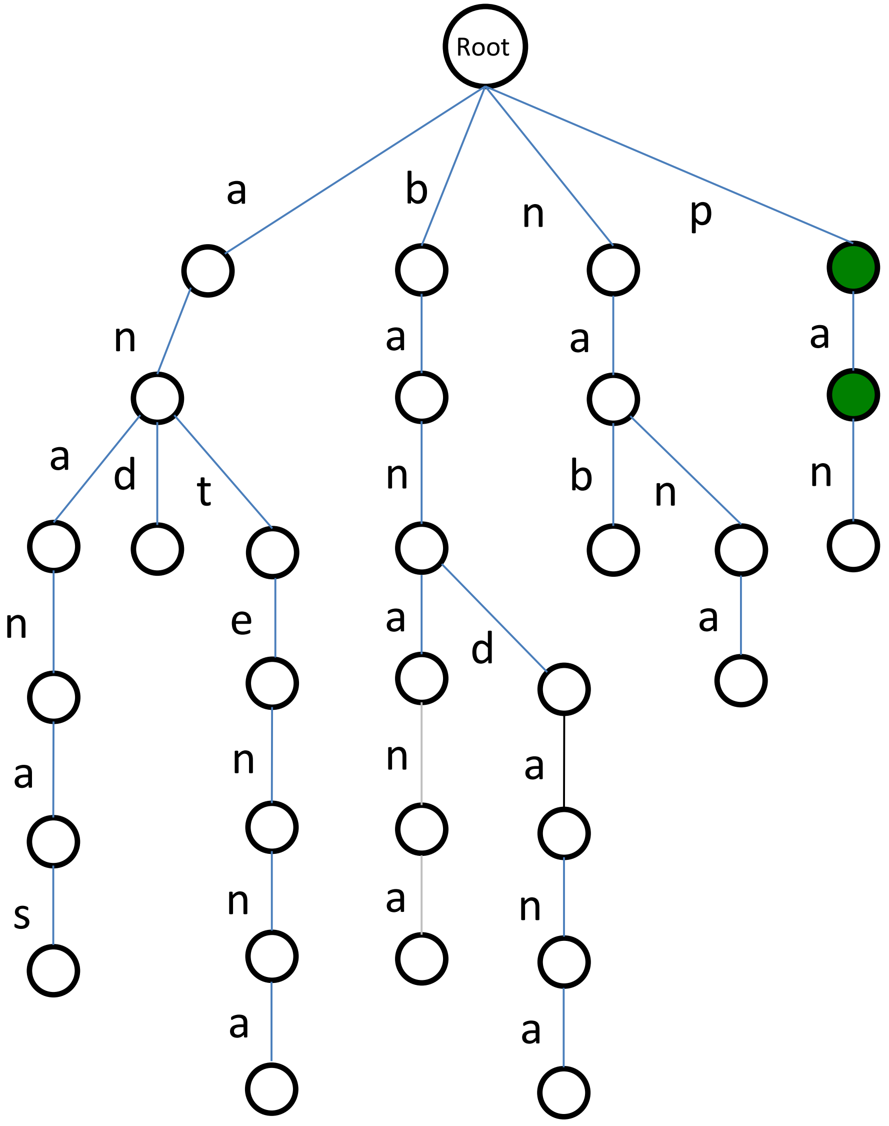
- walk down Trie(*Patterns*) by spelling symbols of *Text*
- a pattern from *Patterns* matches *Text* each time you reach a leaf!

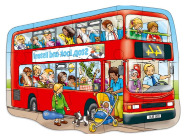
For simplicity, we assume that no pattern is a substring of another pattern

p a n a m a b a n a n a s

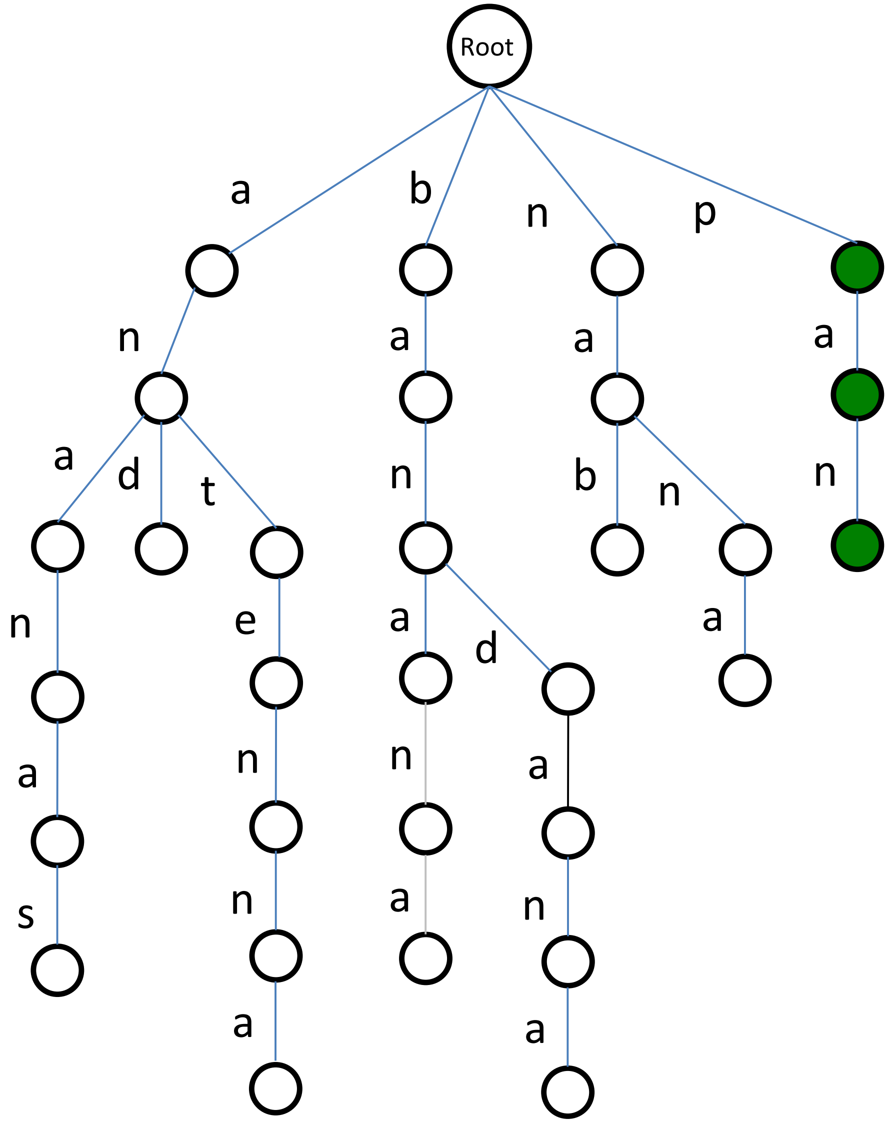


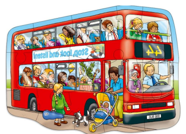
p a n a m a b a n a n a s



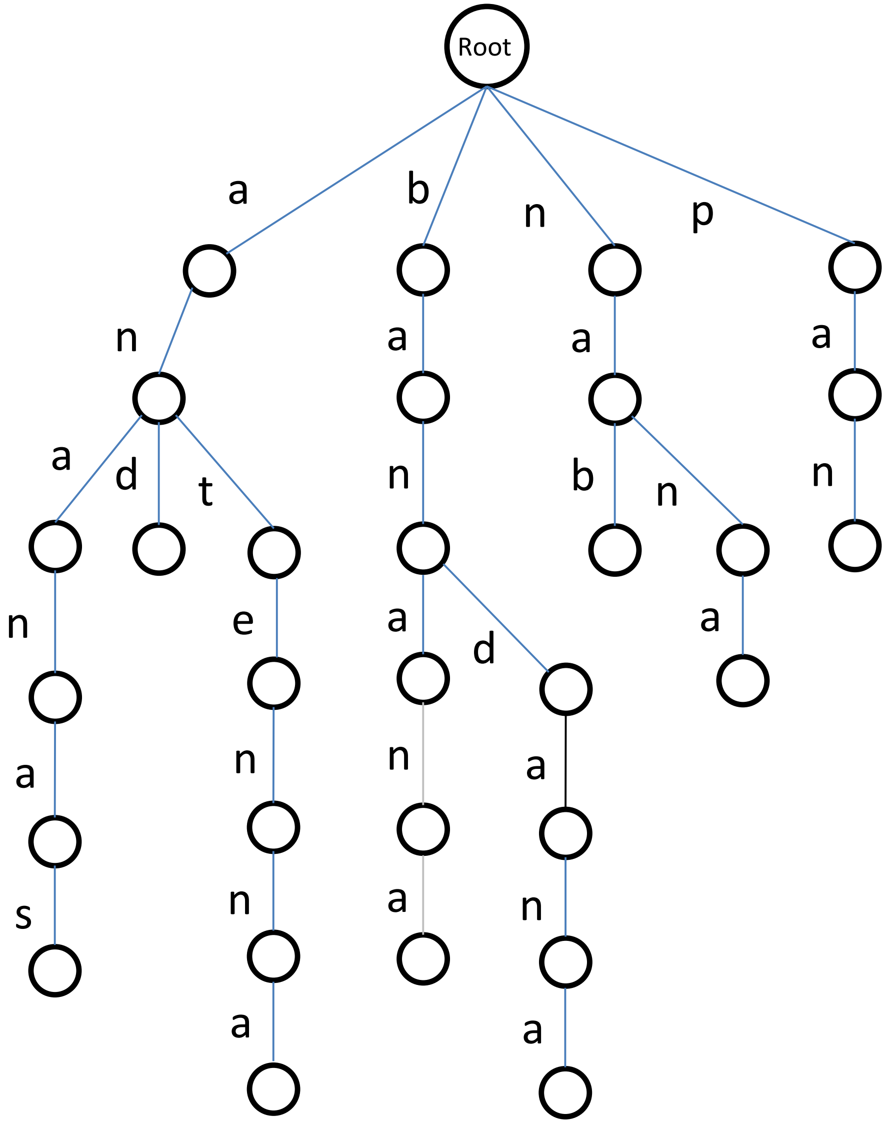


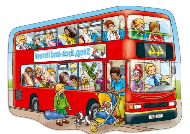
panamabananas



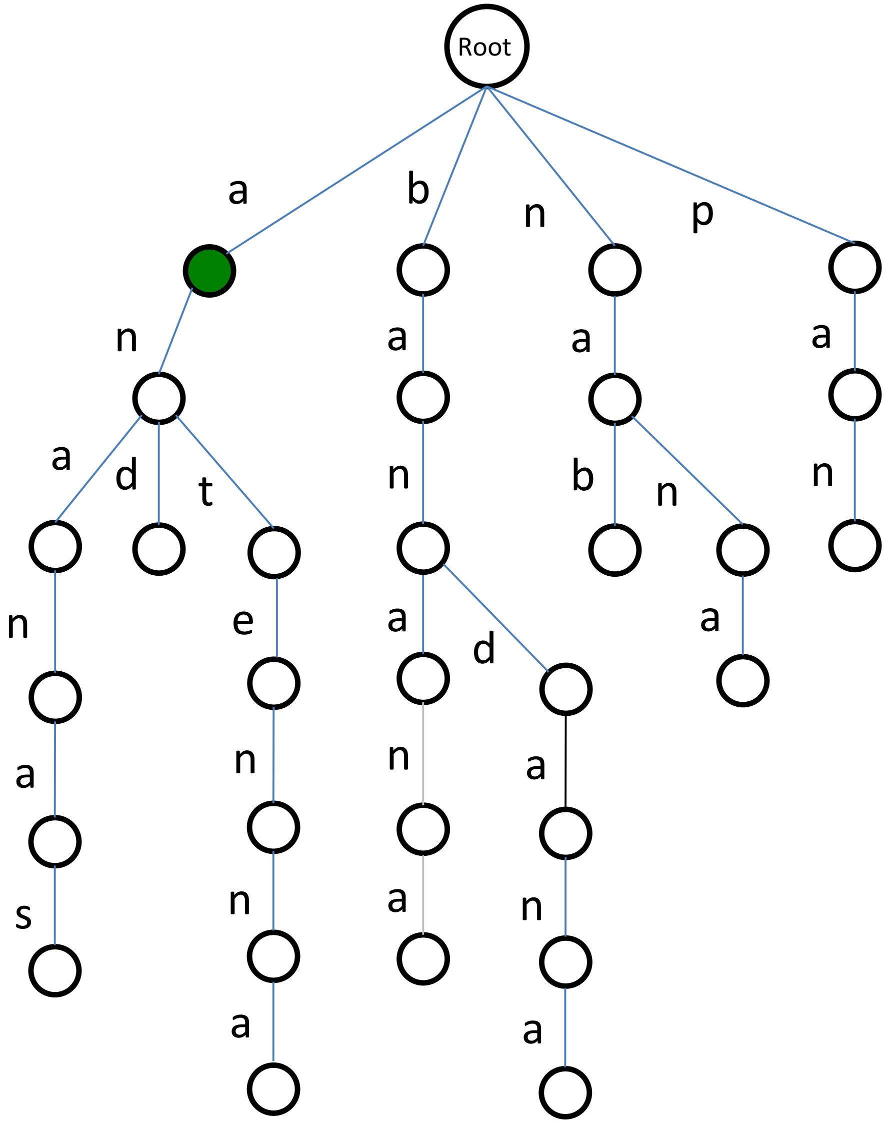


p a n a m a b a n a n a s

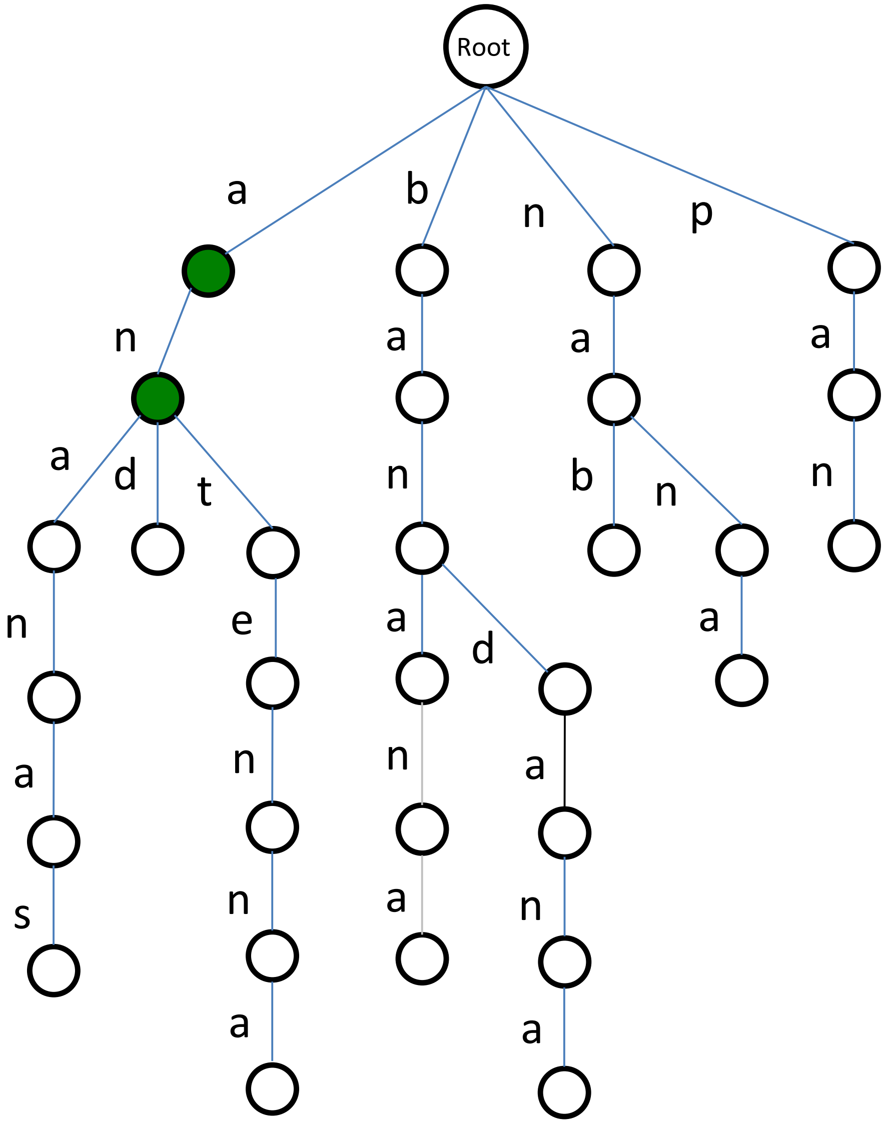




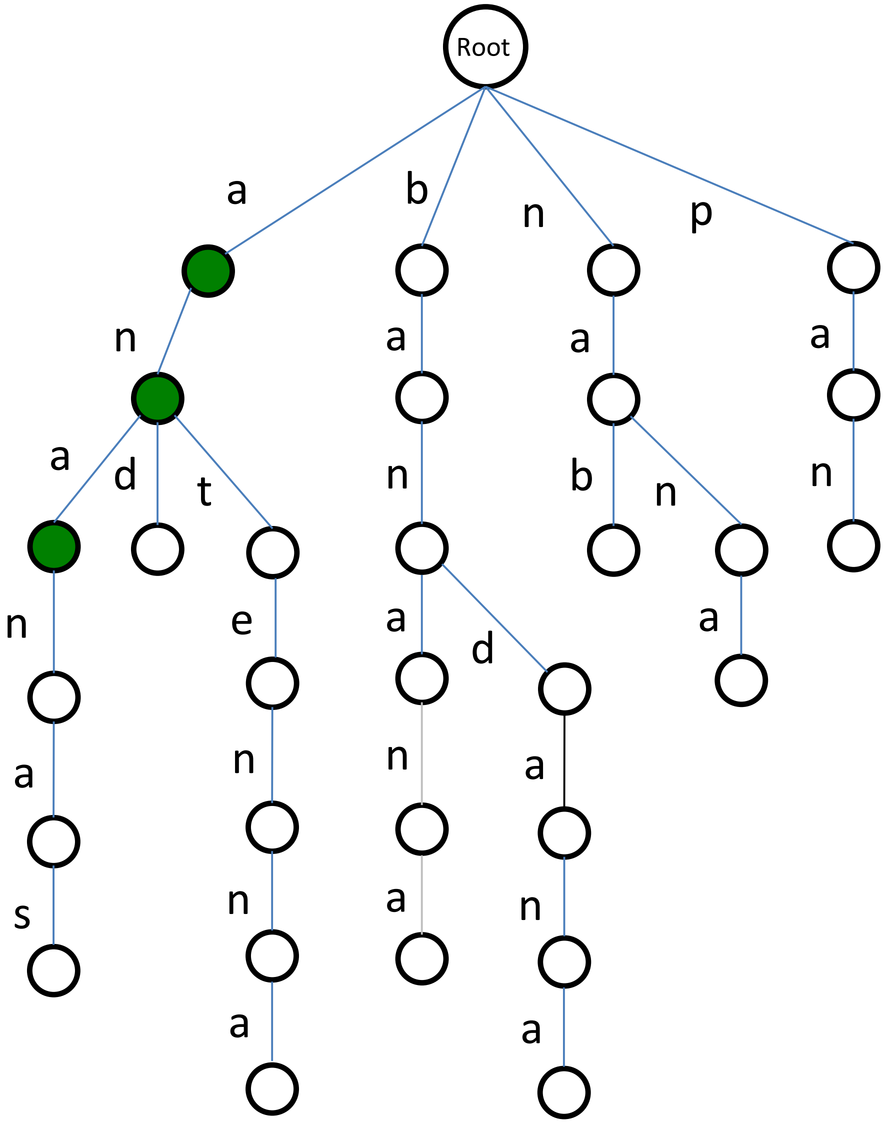
p a n a m a b a n a n a s



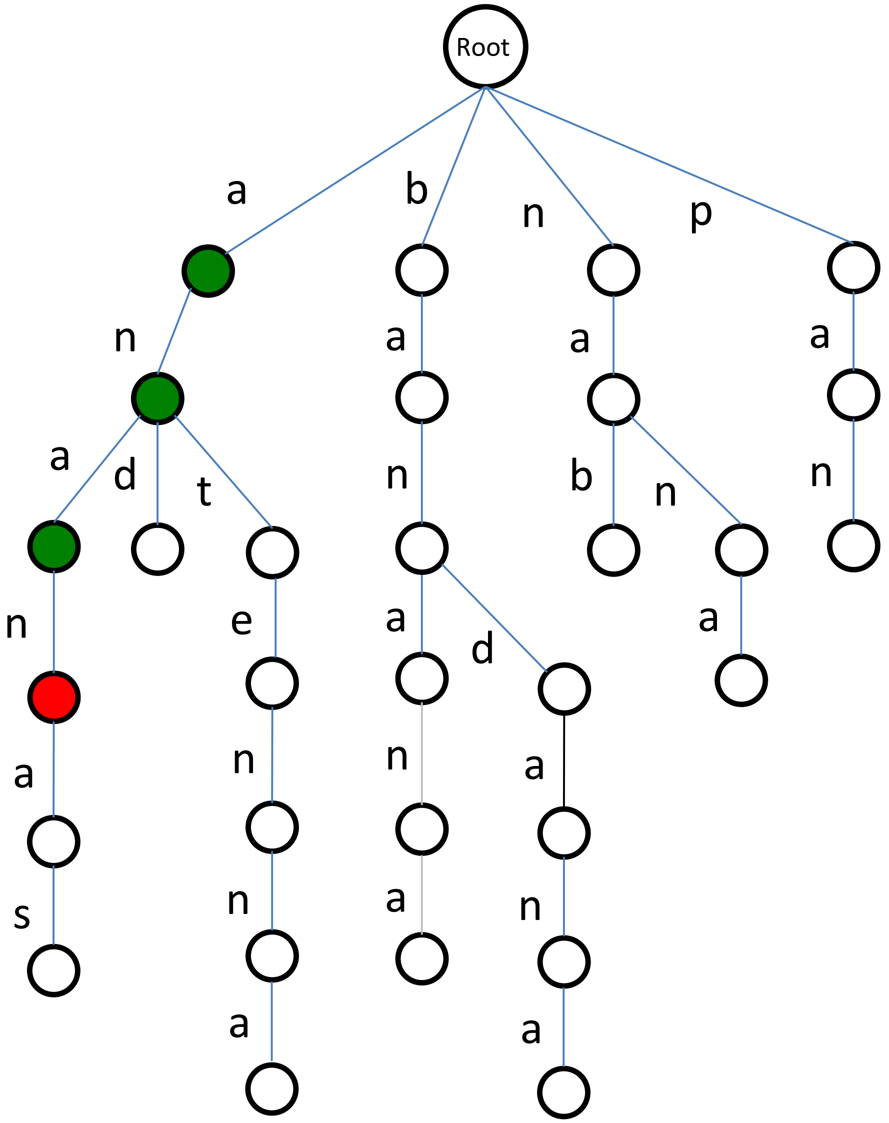
p a n a m a b a n a n a s



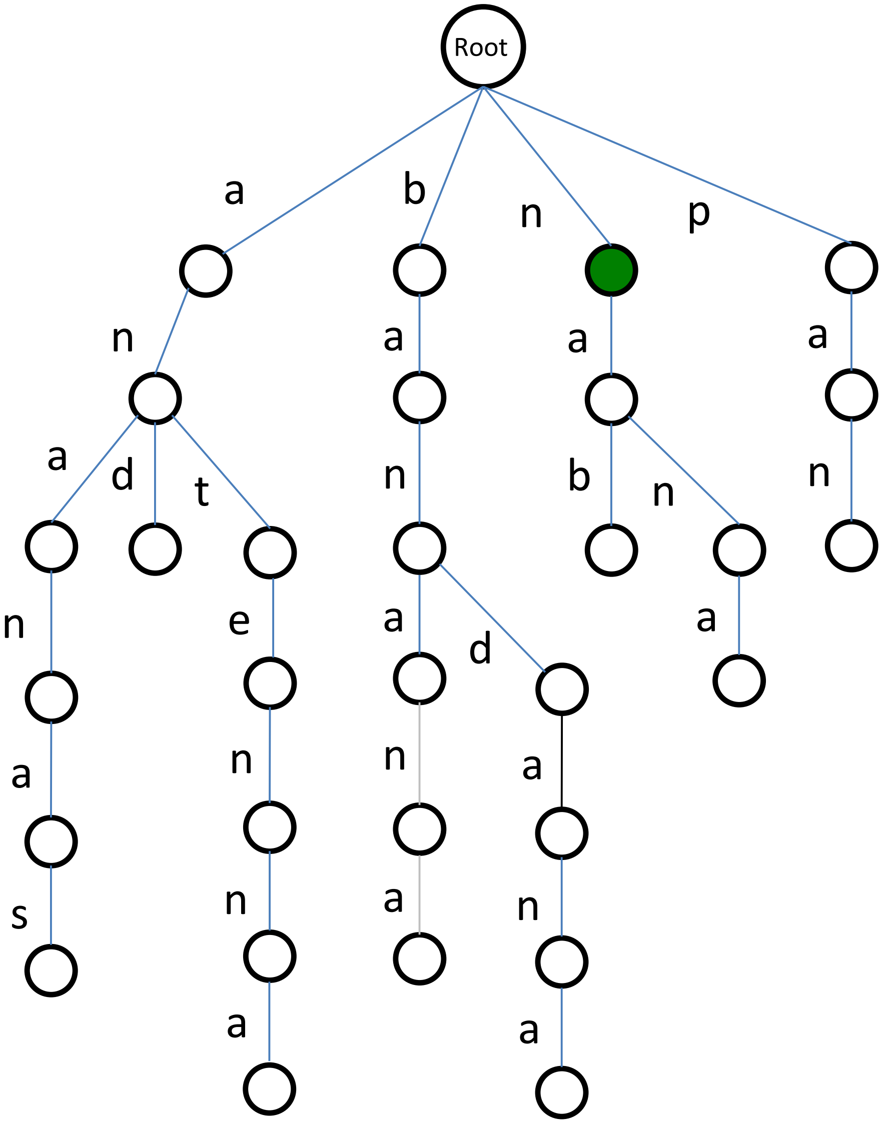
p a n a m a b a n a n a s



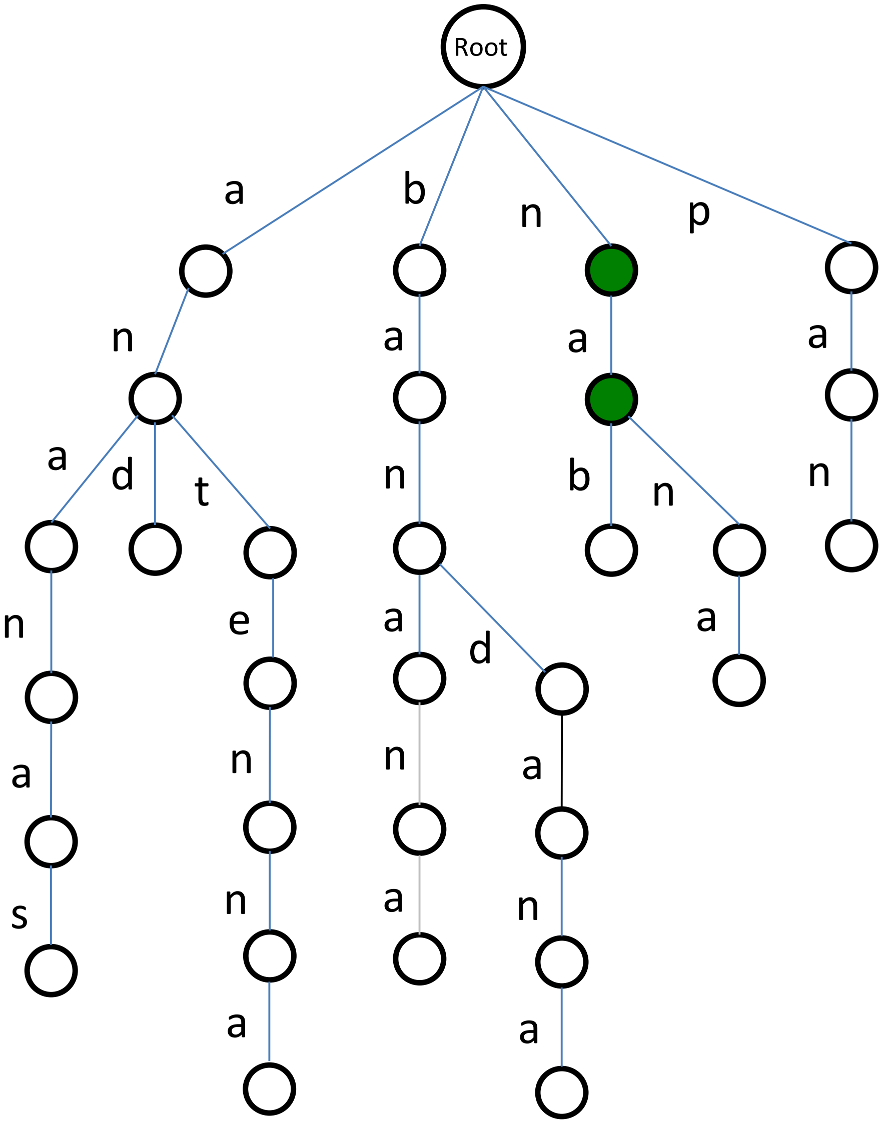
p a n a m a b a n a n a s



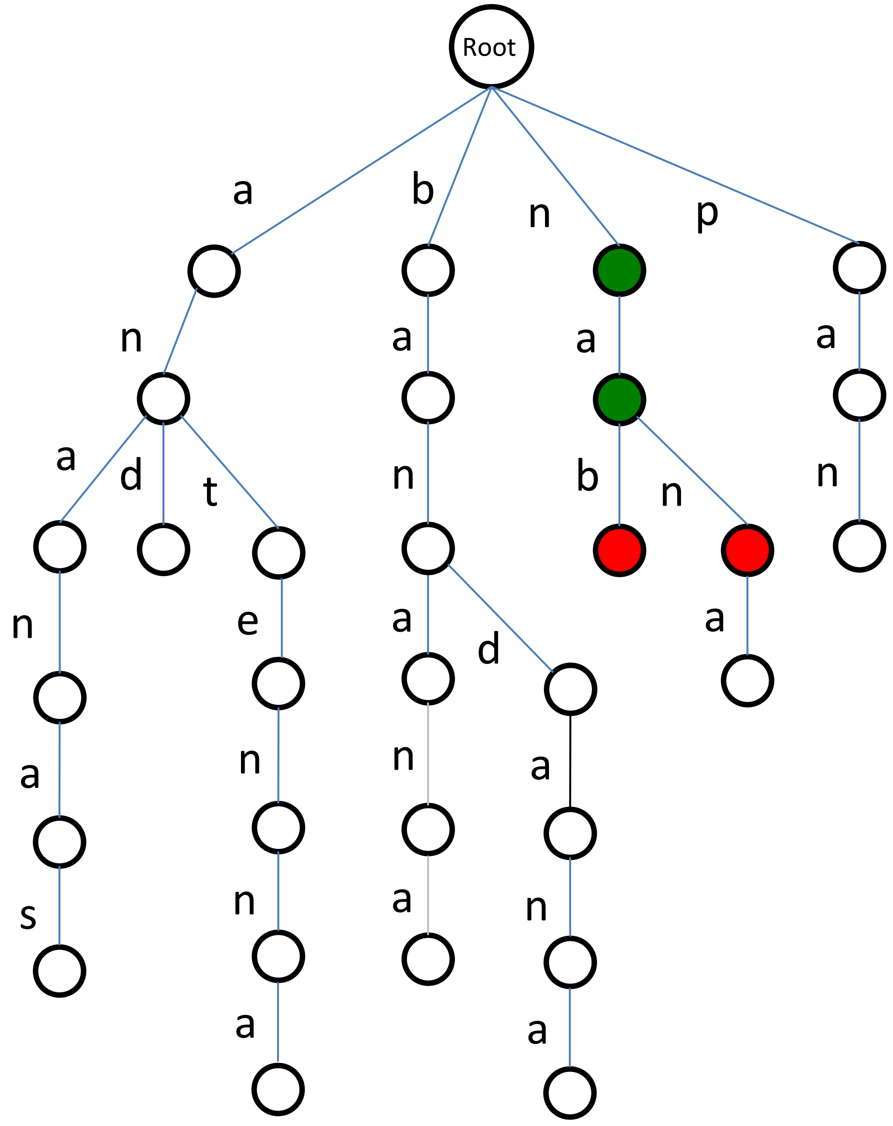
p a **n** a m a b a n a n a s

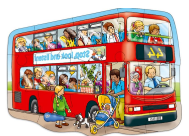


p a **n a** m a b a n a n a s

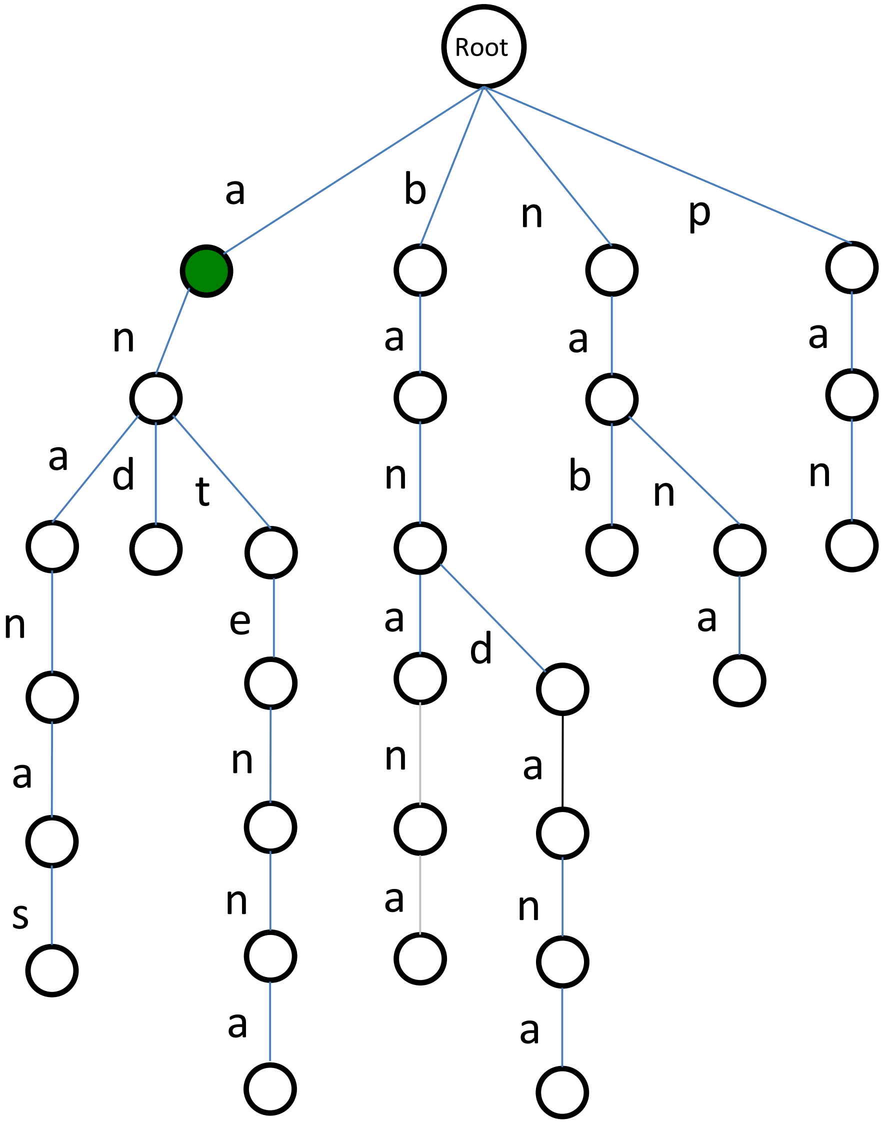


p a n a m a b a n a n a s

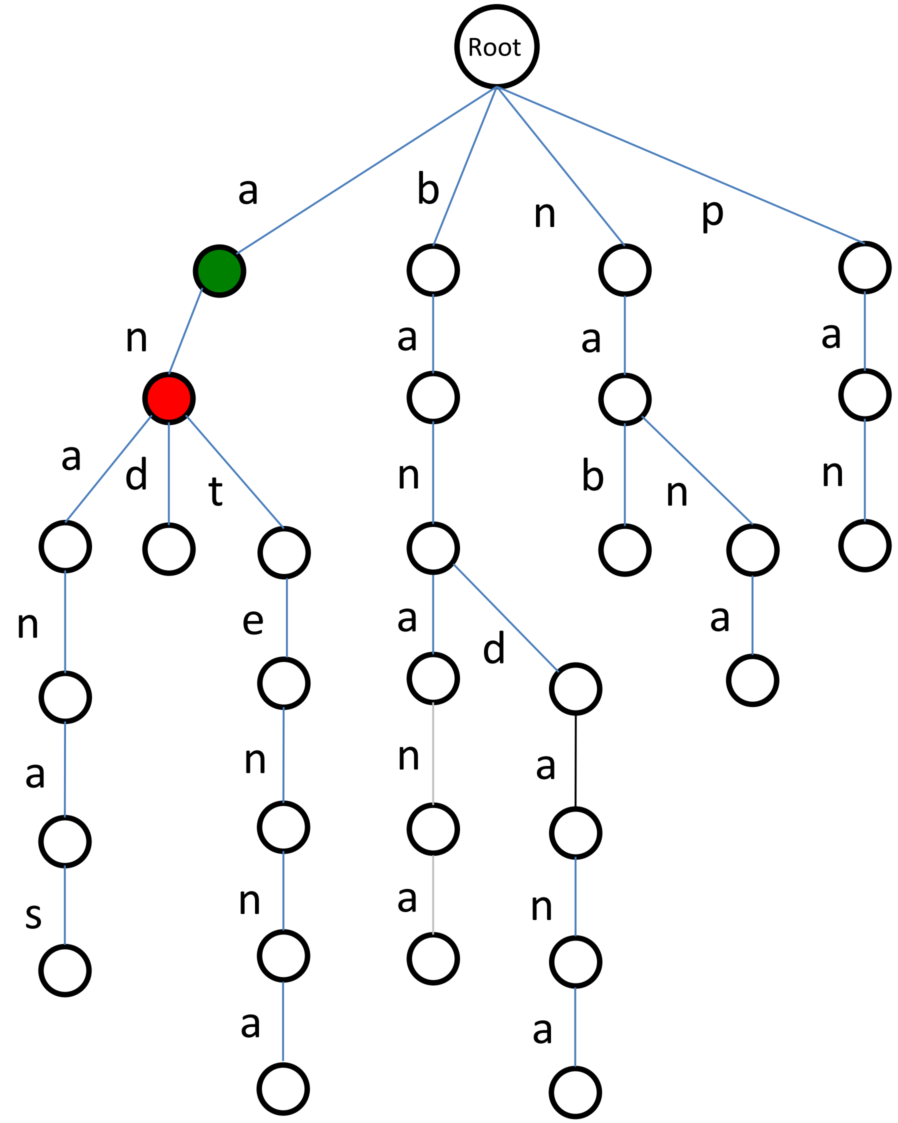




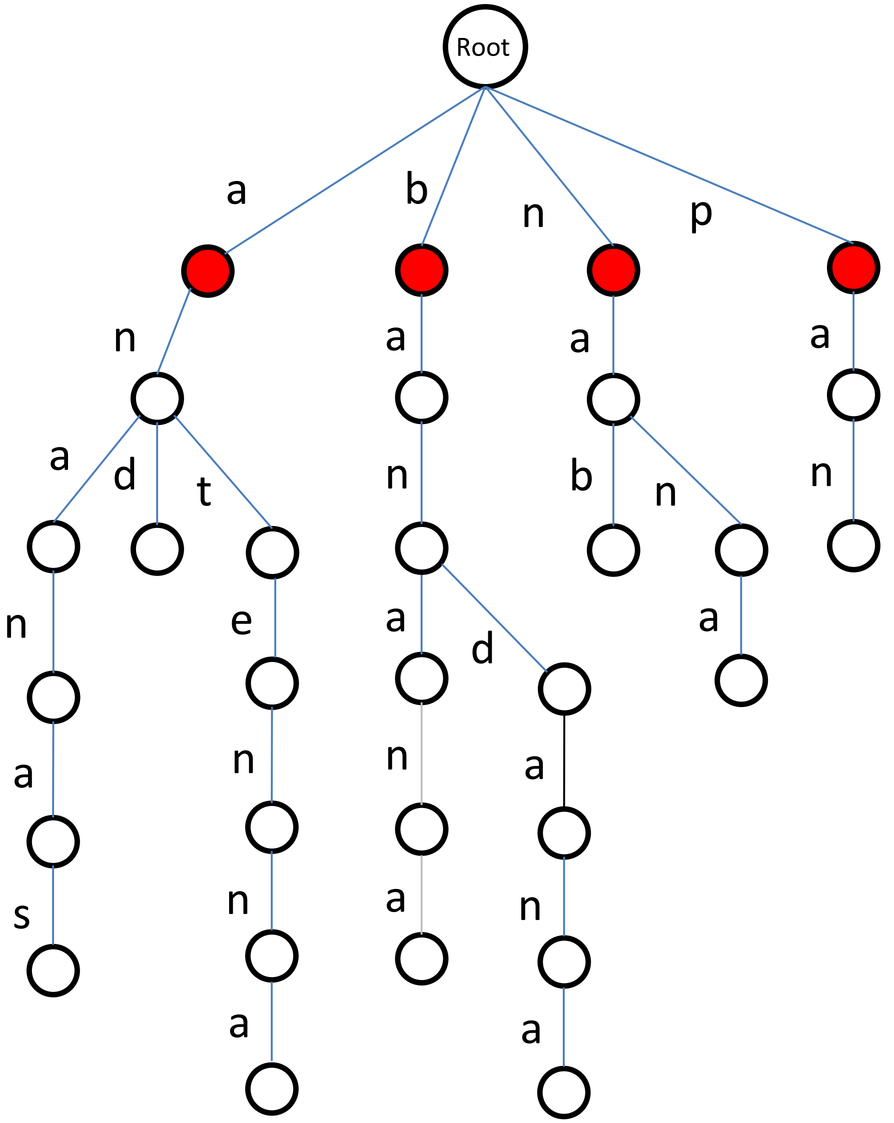
p a n a m a b a n a n a s



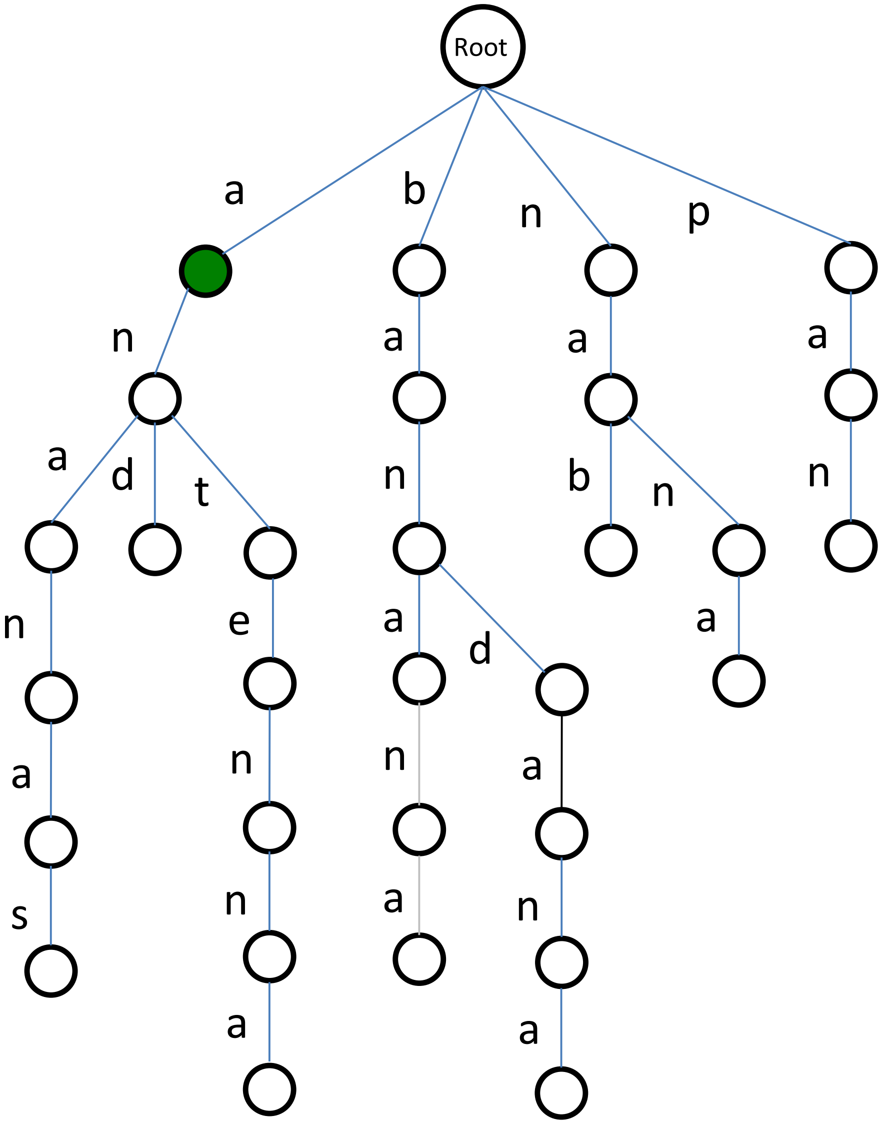
p a n a m a b a n a n a s



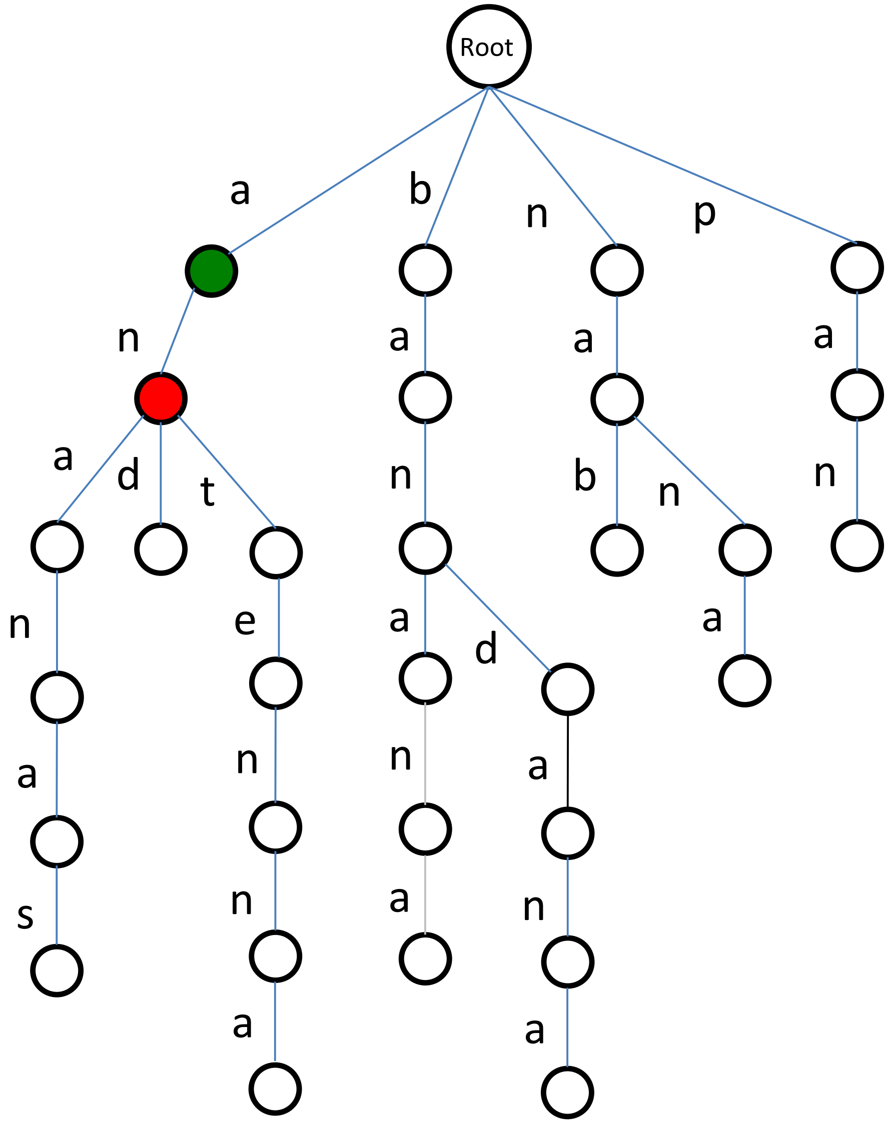
p a n a **m** a b a n a n a s



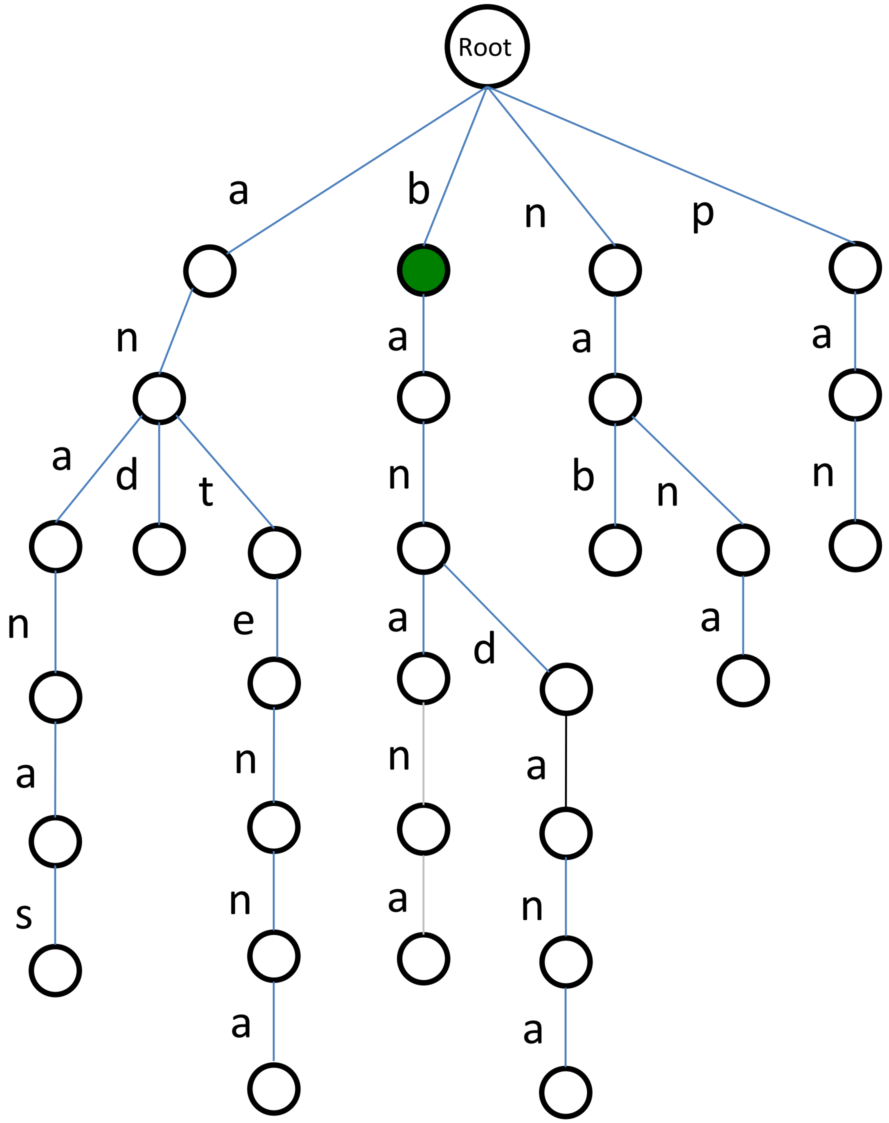
p a n a m a b a n a n a s

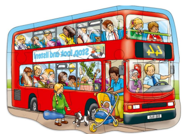


p a n a m **a** **b** a n a n a s

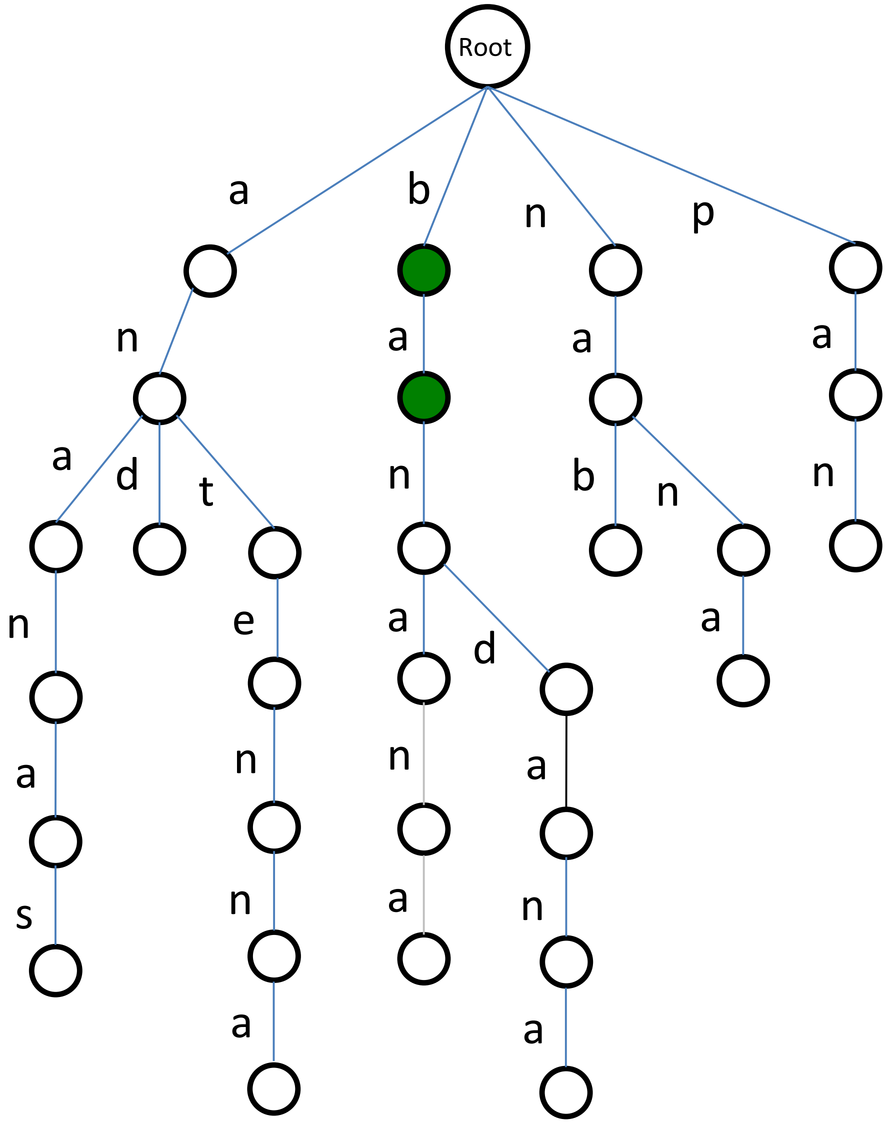


p a n a m a **b** a n a n a s

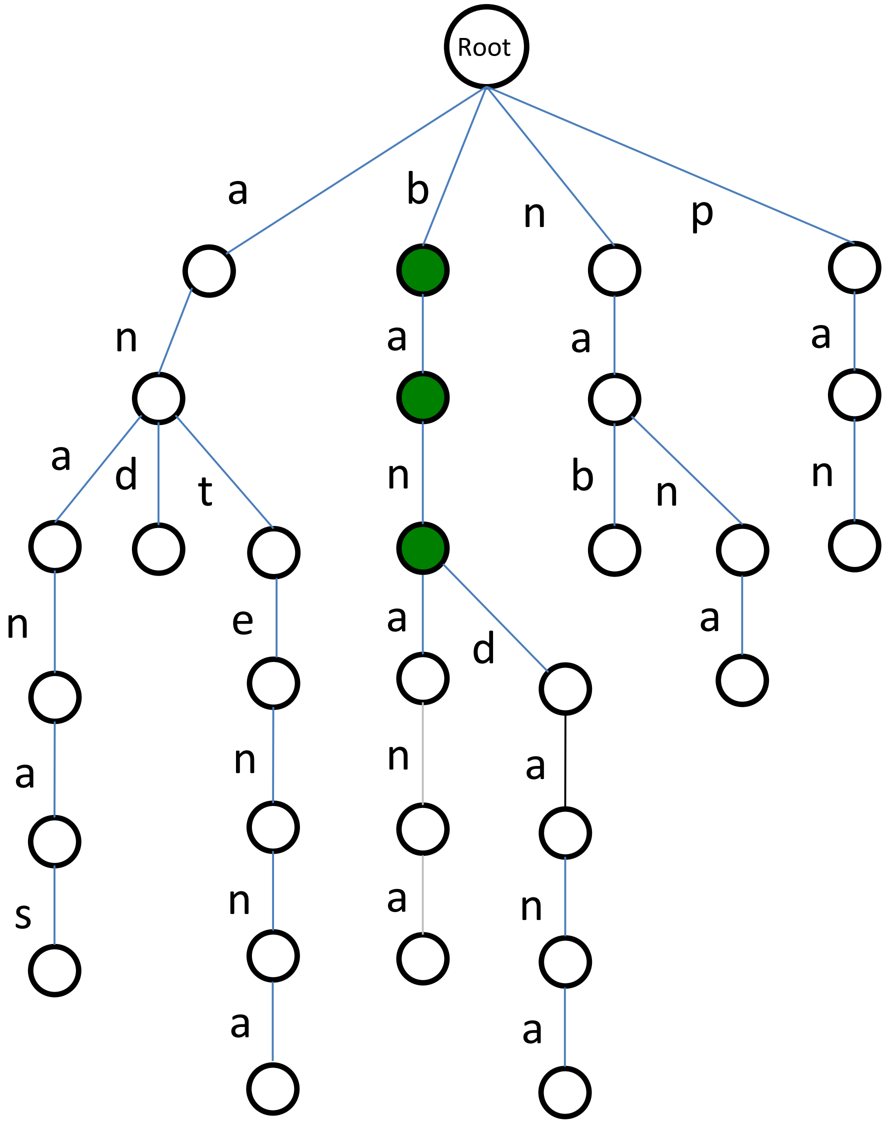


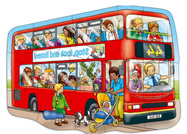


p a n a m a **b a** n a n a s

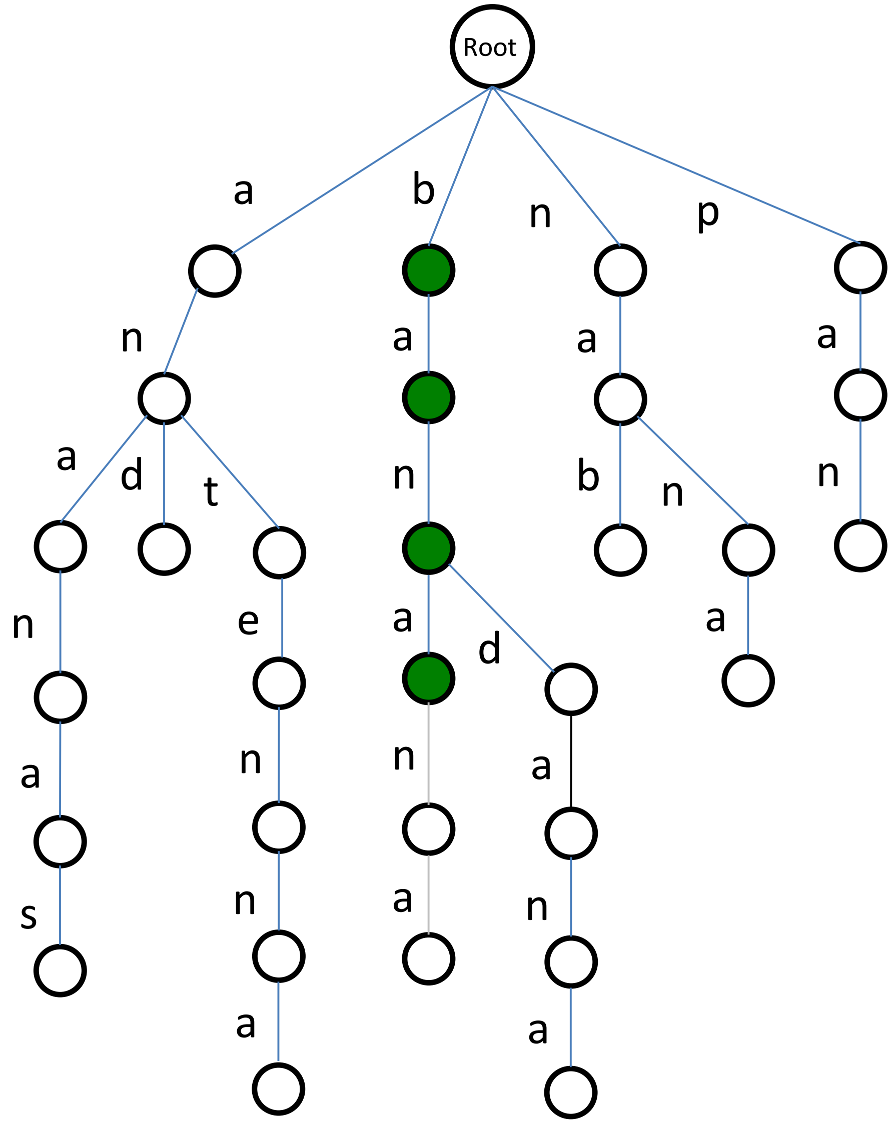


panama**banan**as

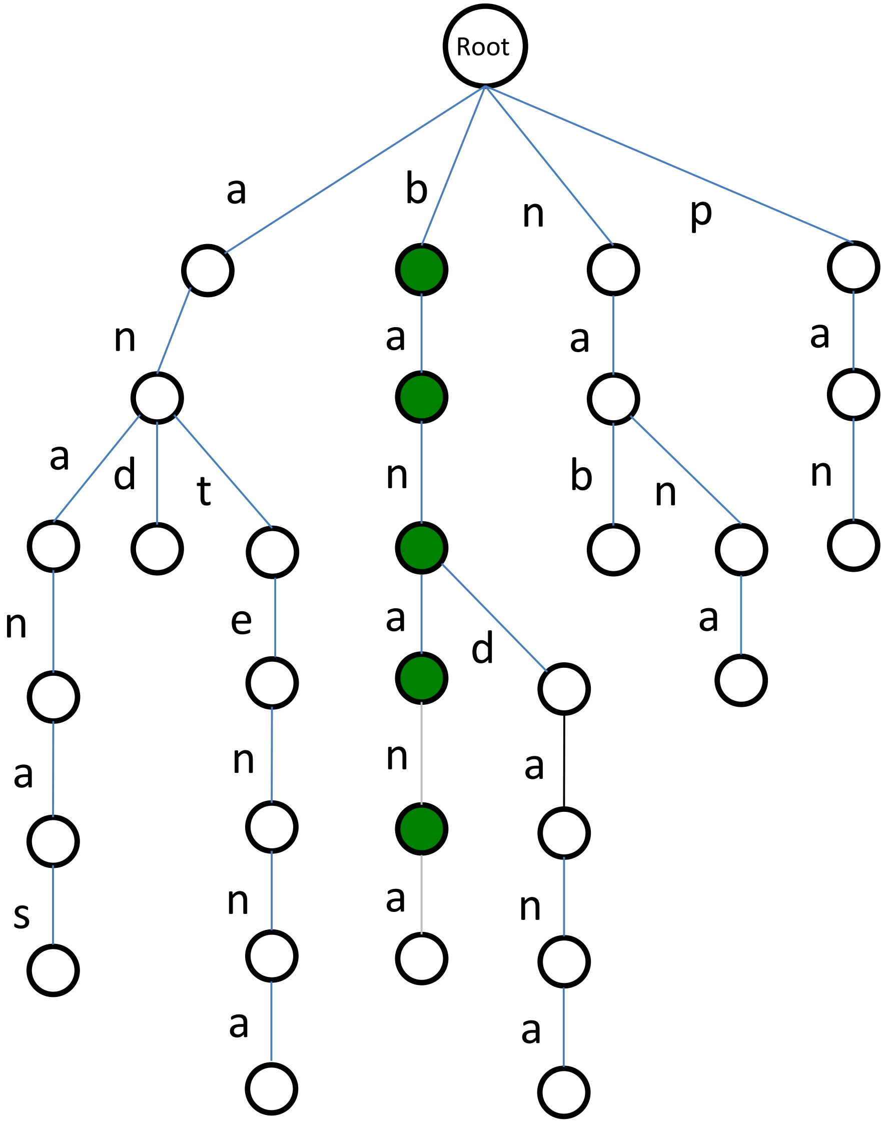




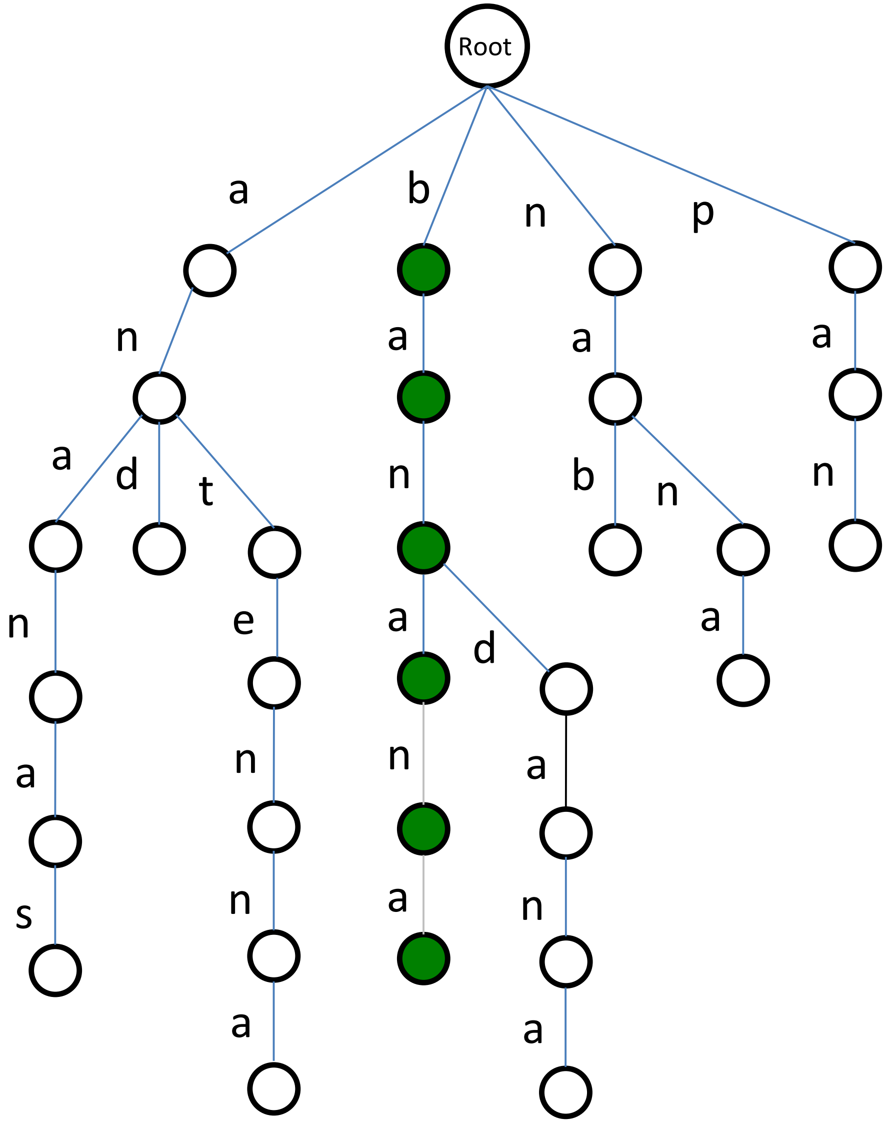
p a n a m a **b a n a** n a s

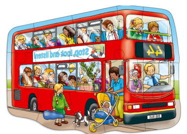


p a n a m a **b a n a n** a s

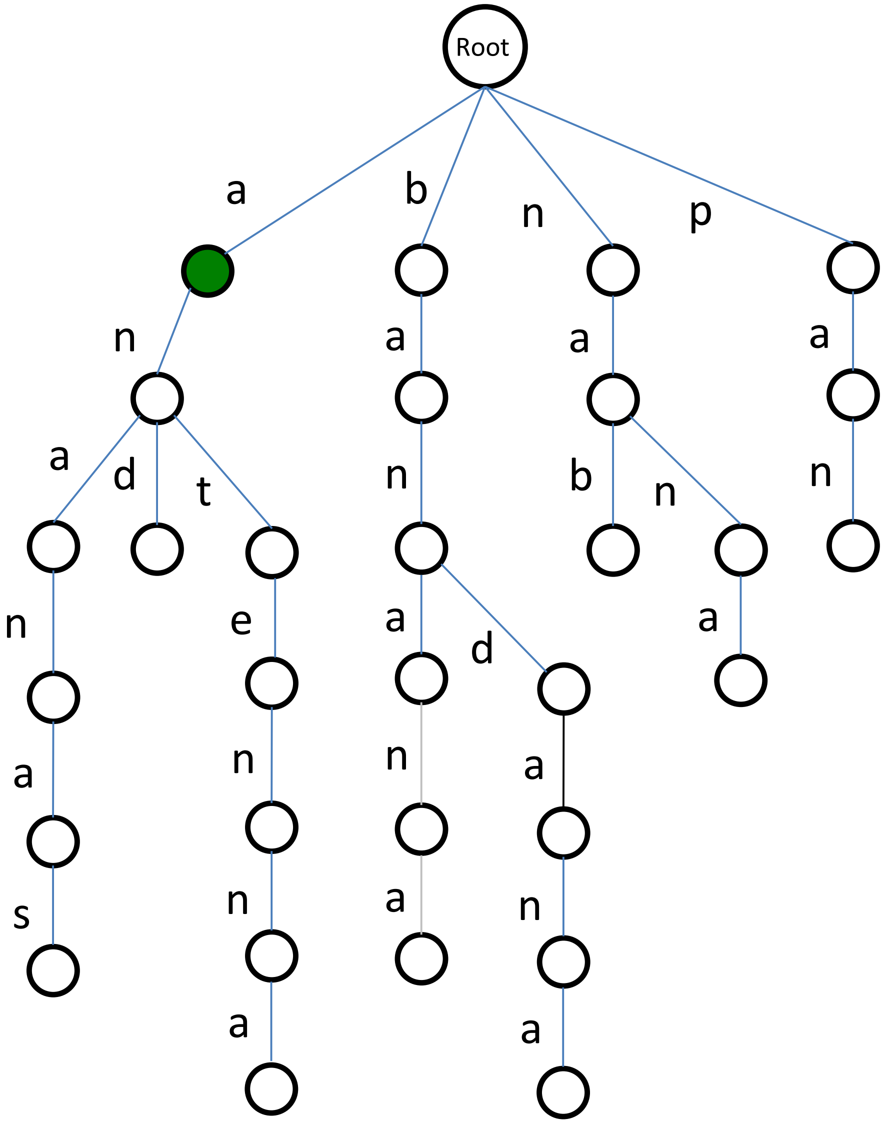


p a n a m a **b a n a n a** s

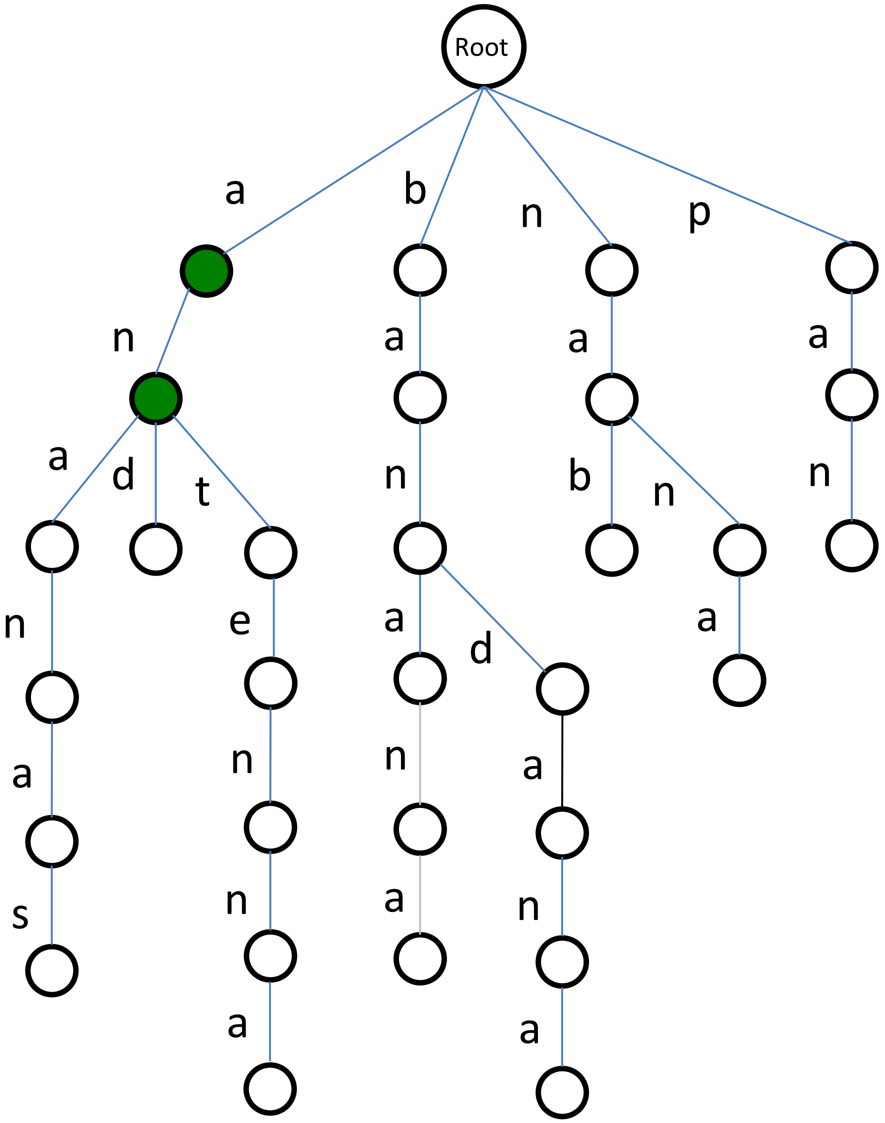




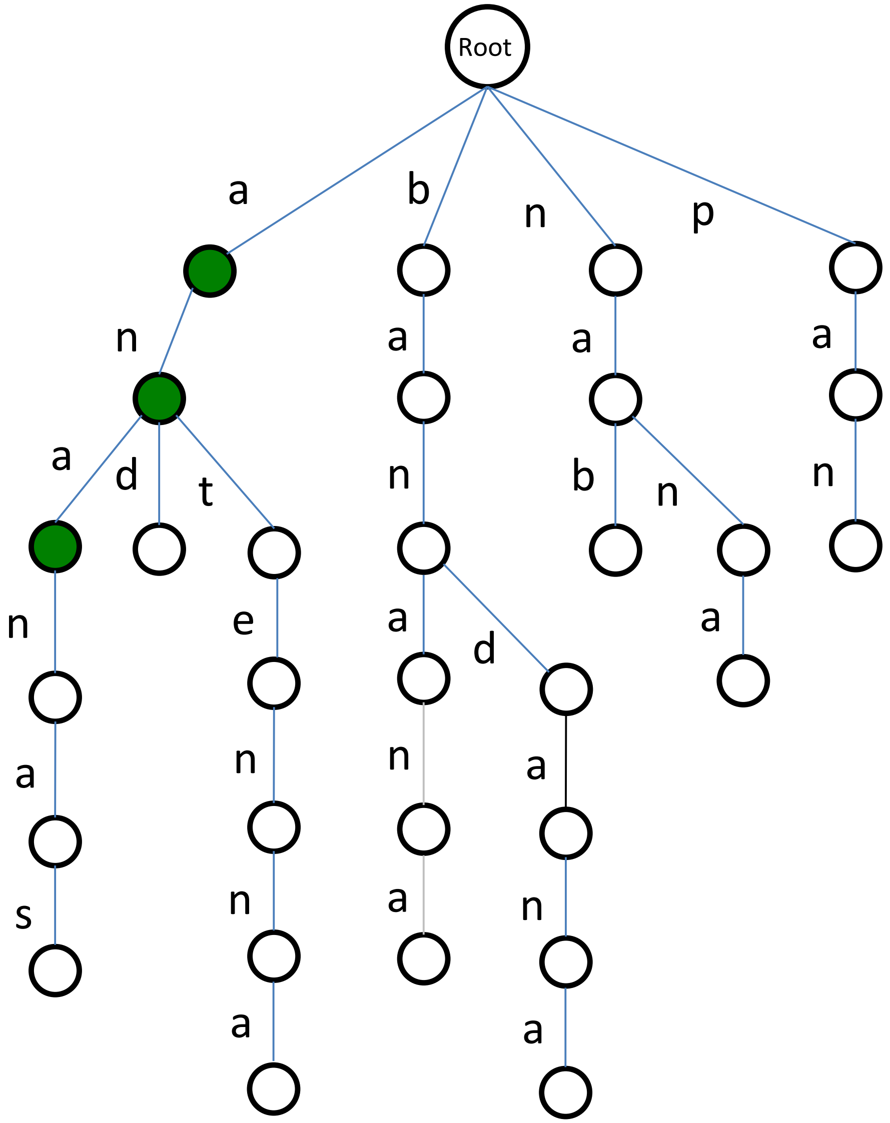
p a n a m a b **a** n a n a s



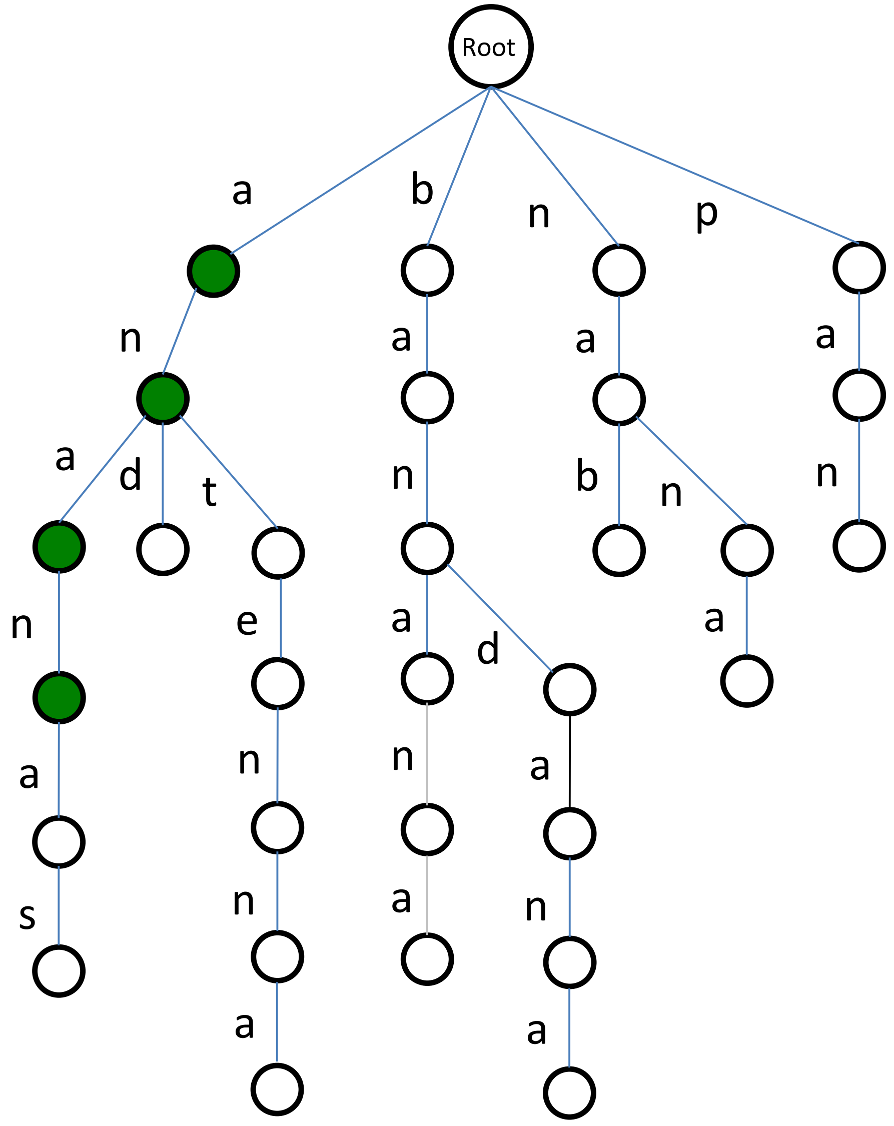
p a n a m a b **a n** a n a s



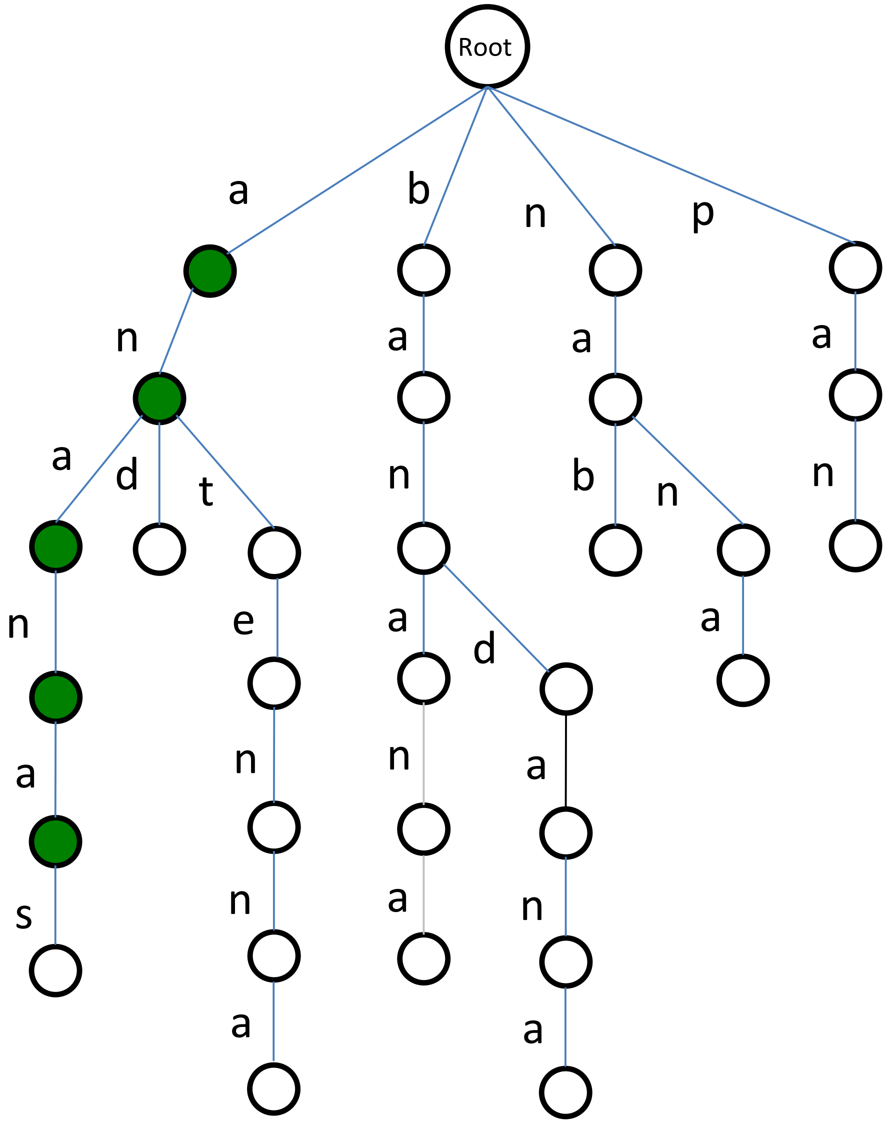
p a n a m a b **a n a** n a s



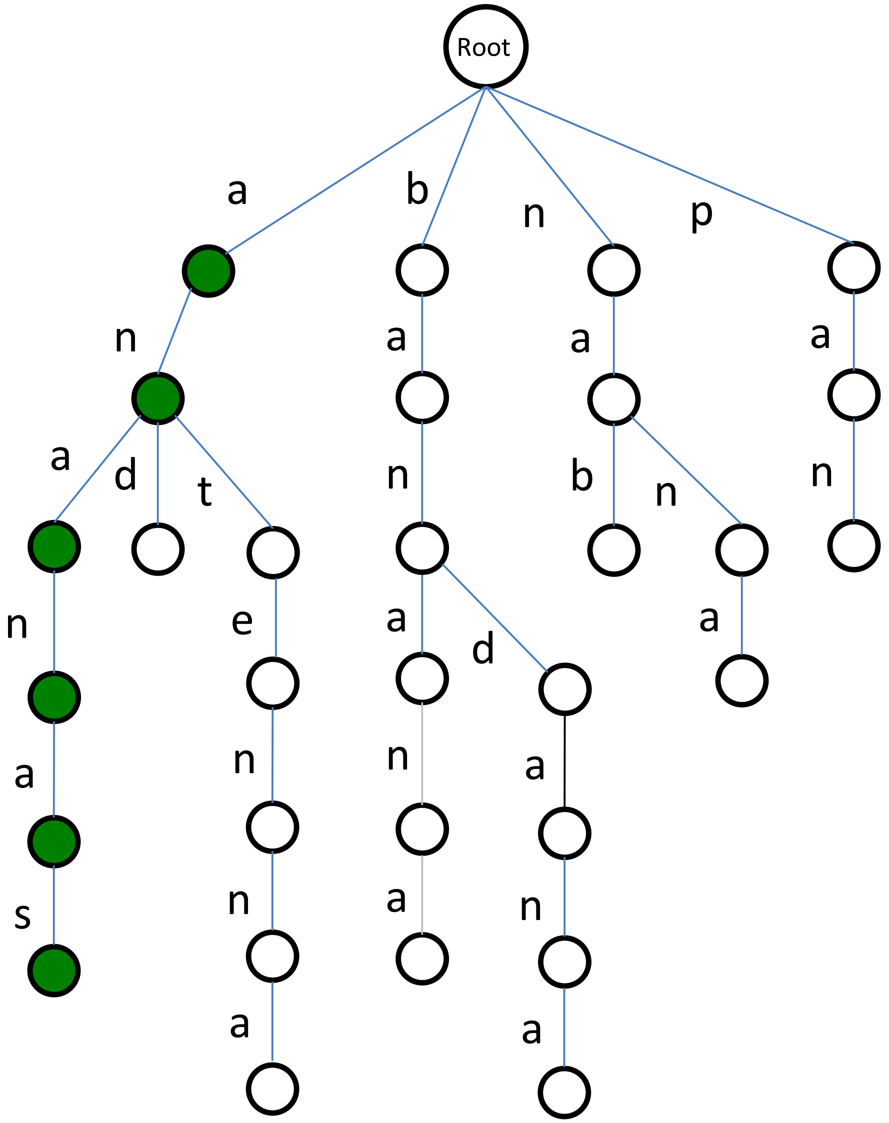
p a n a m a b **a n a n** a s



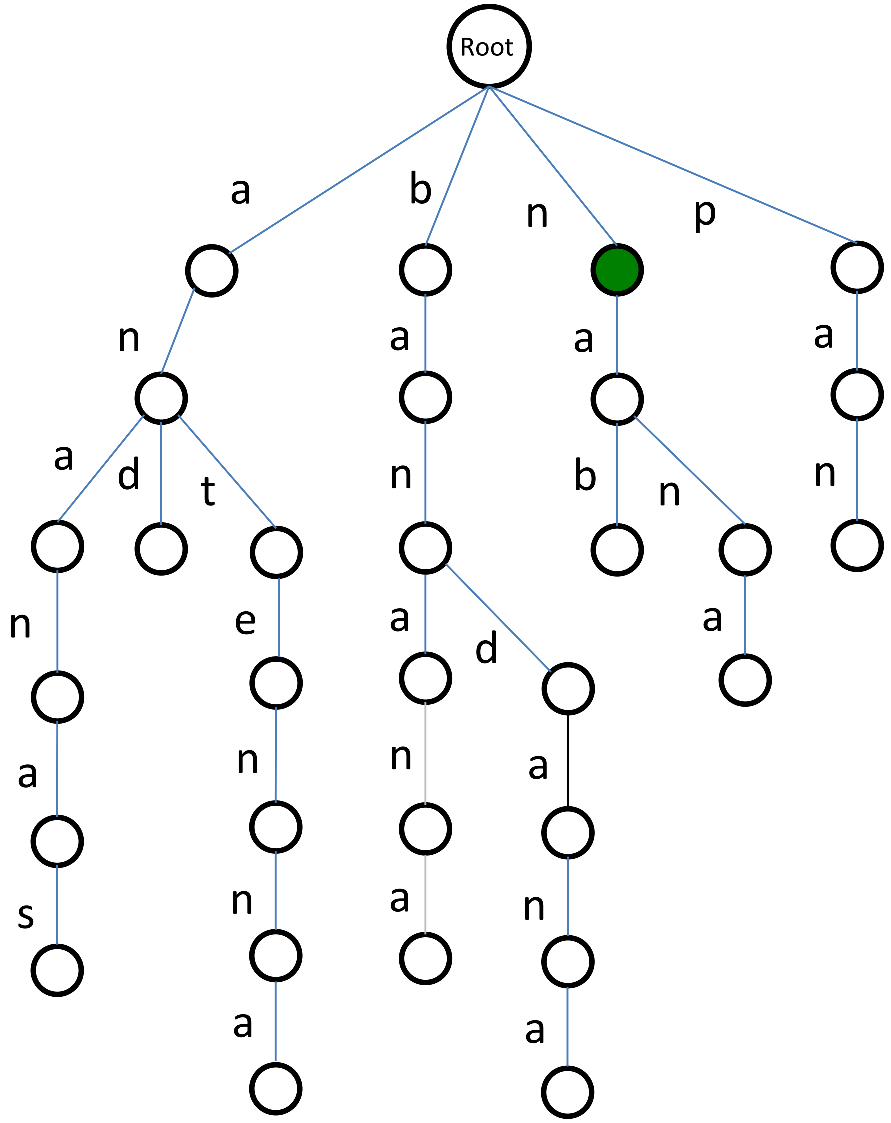
p a n a m a b **a n a n a** s



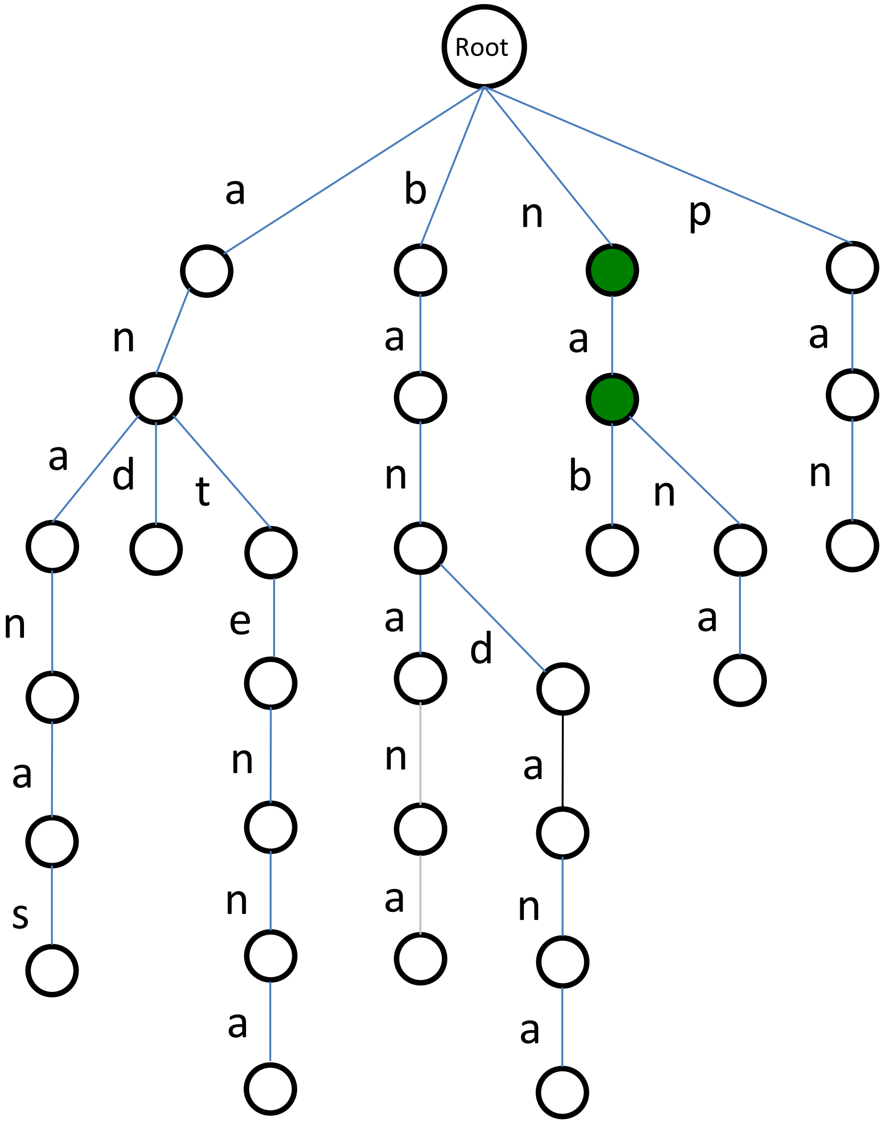
p a n a m a b **a n a n a s**



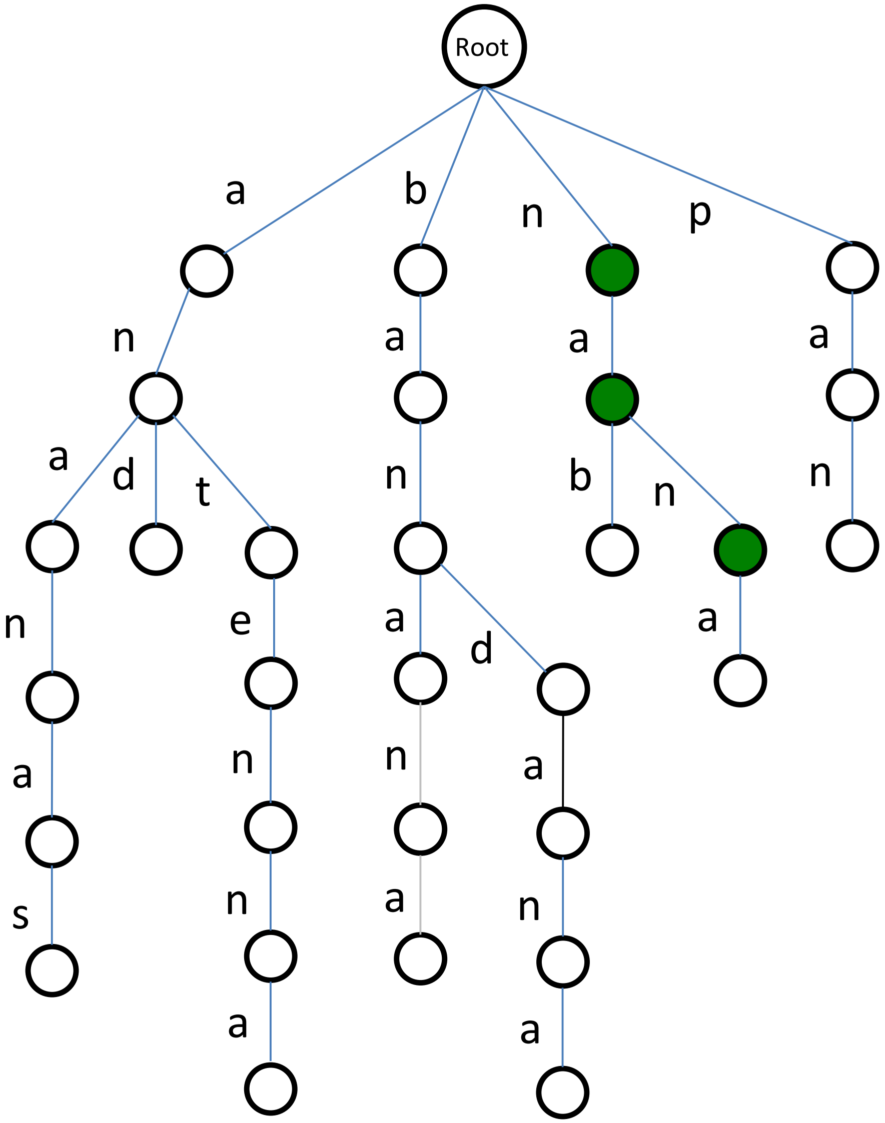
p a n a m a b a **n** a n a s



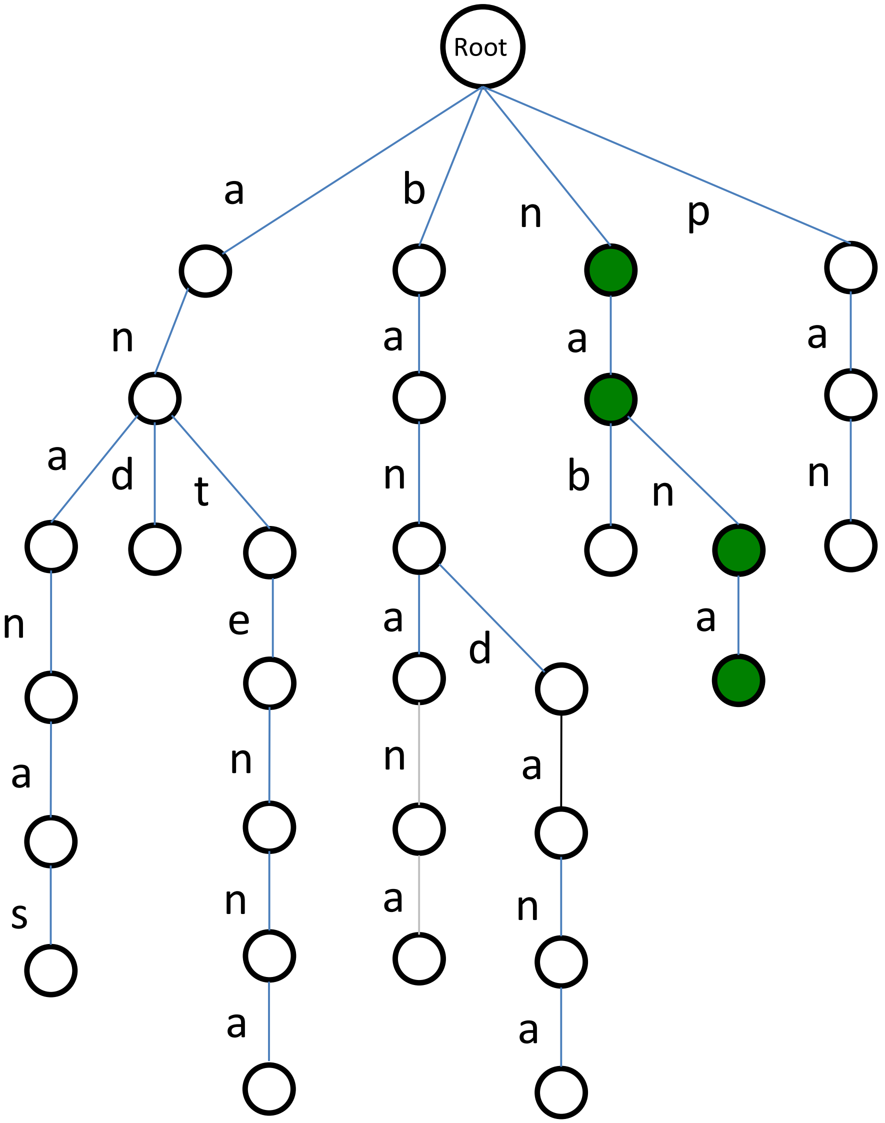
p a n a m a b a **n a** n a s



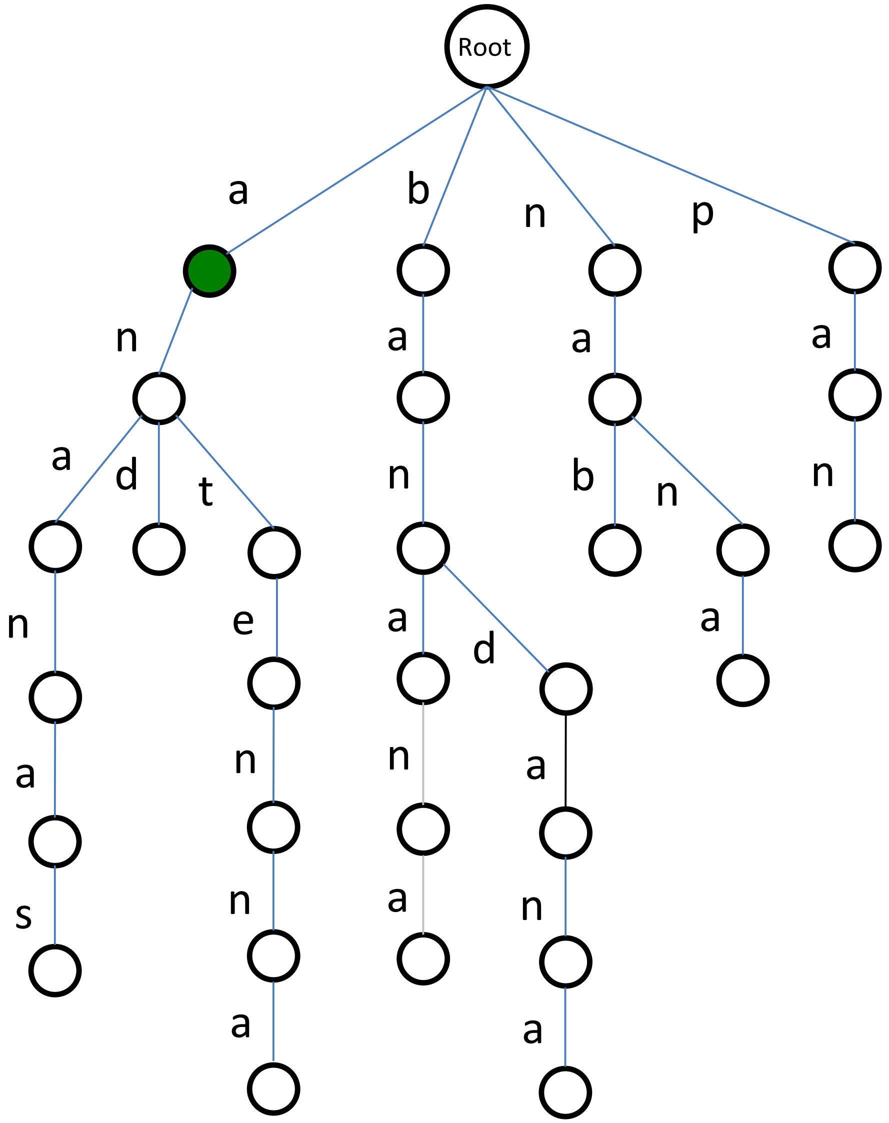
p a n a m a b a **n a n** a s



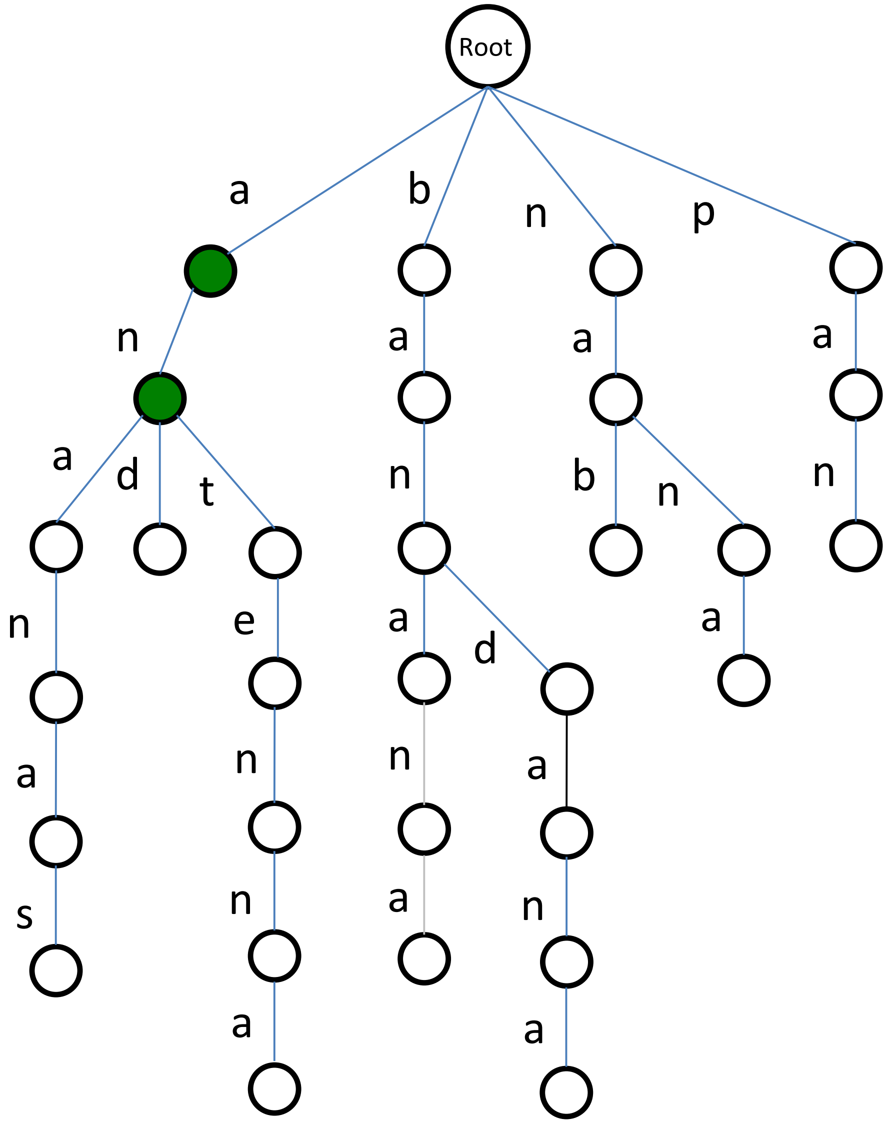
p a n a m a b a **n a n a** s



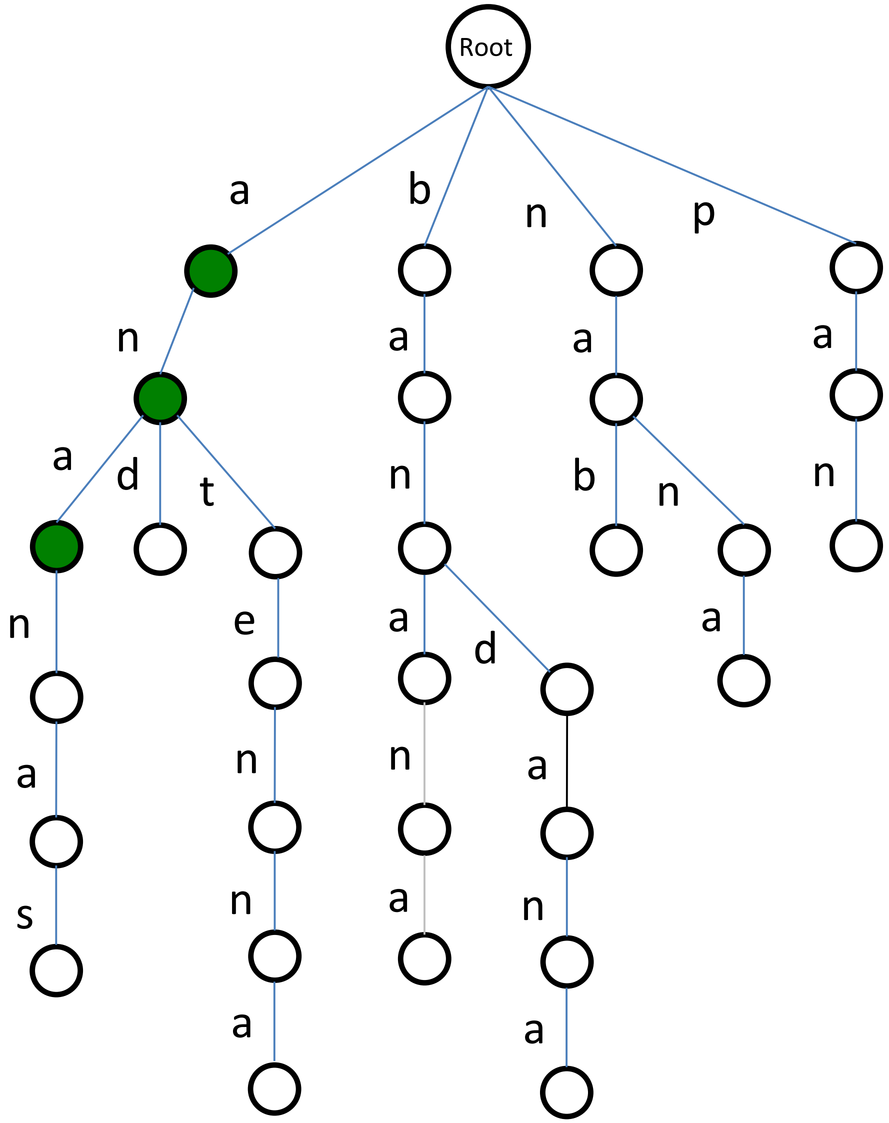
p a n a m a b a n a n a s



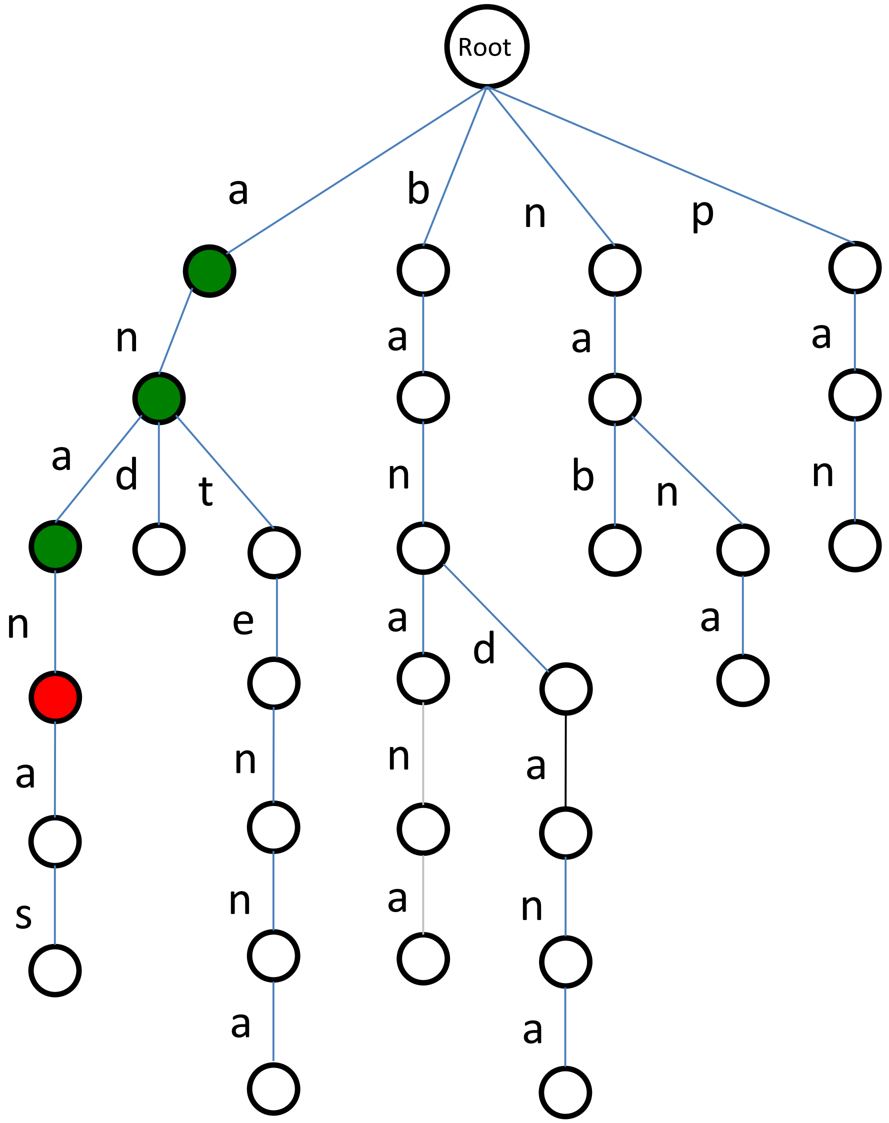
p a n a m a b a n **a n** a s



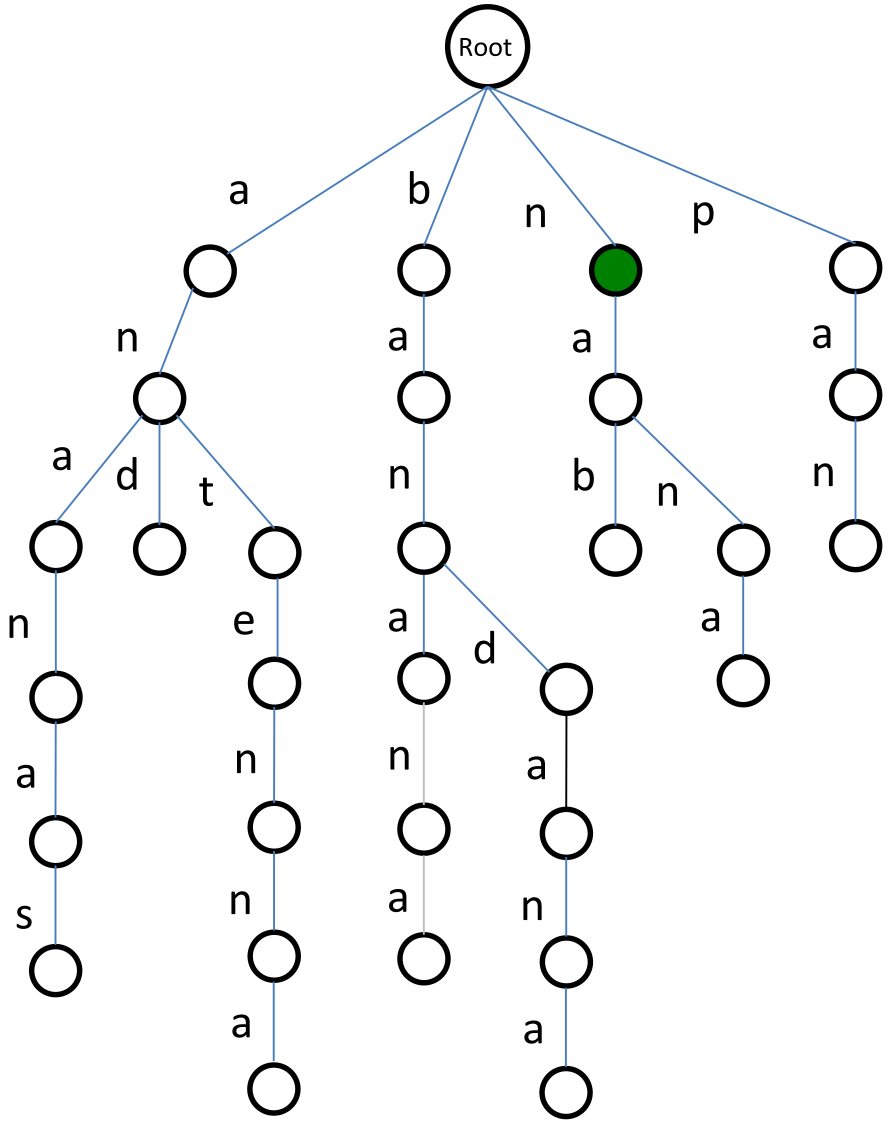
p a n a m a b a n **a n a** s



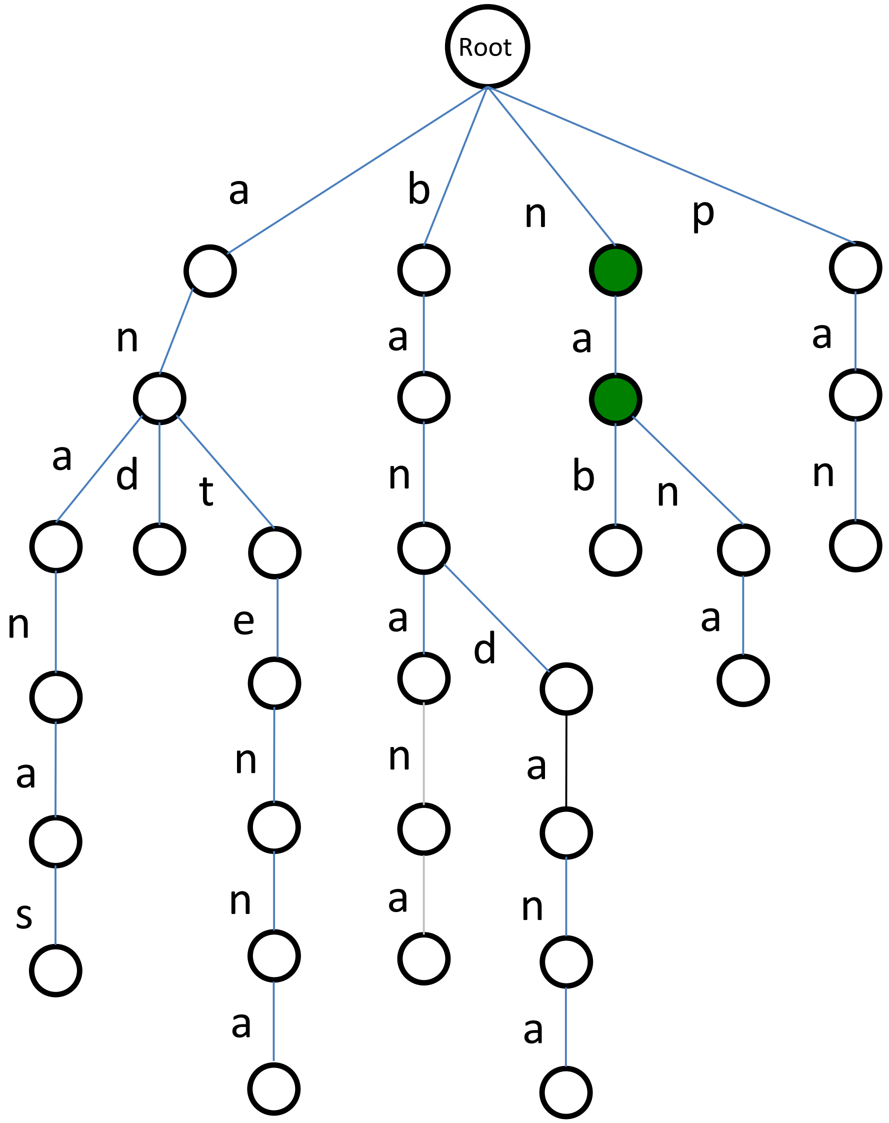
p a n a m a b a n **a n a s**



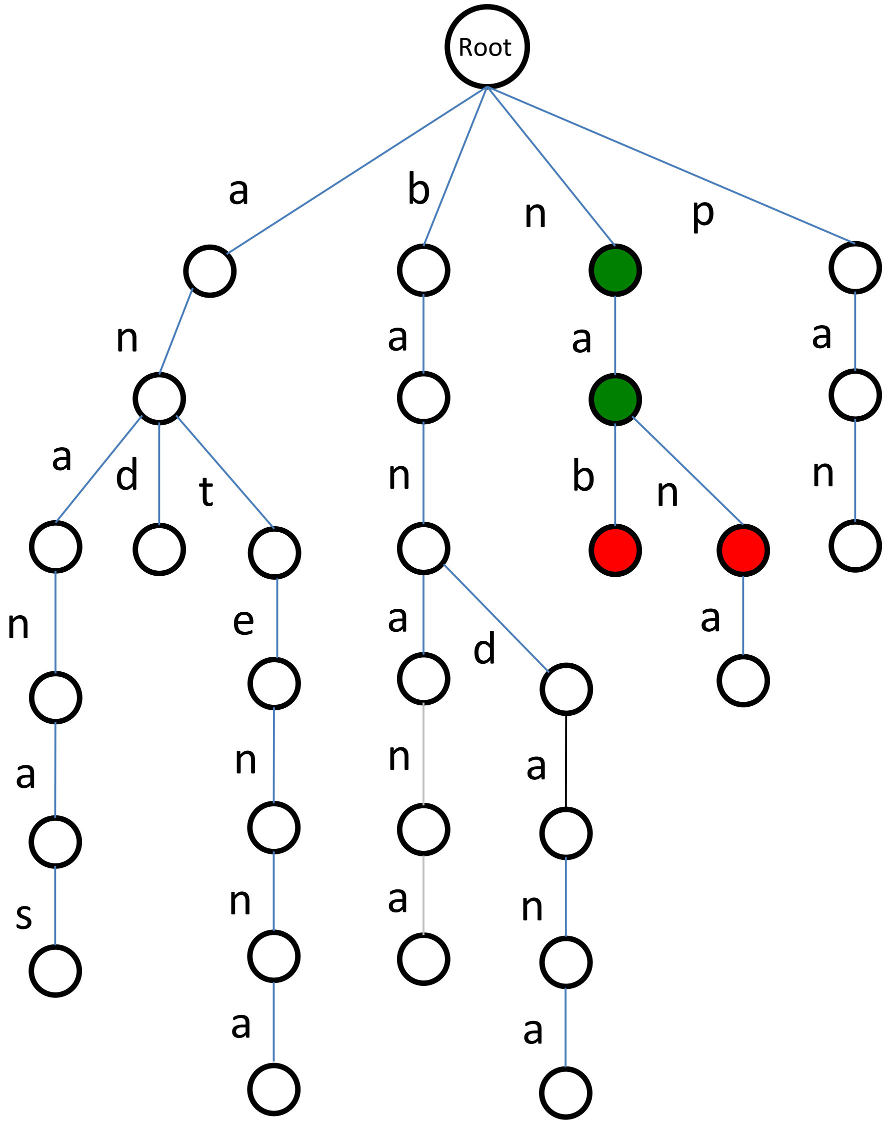
p a n a m a b a n a **n** a s



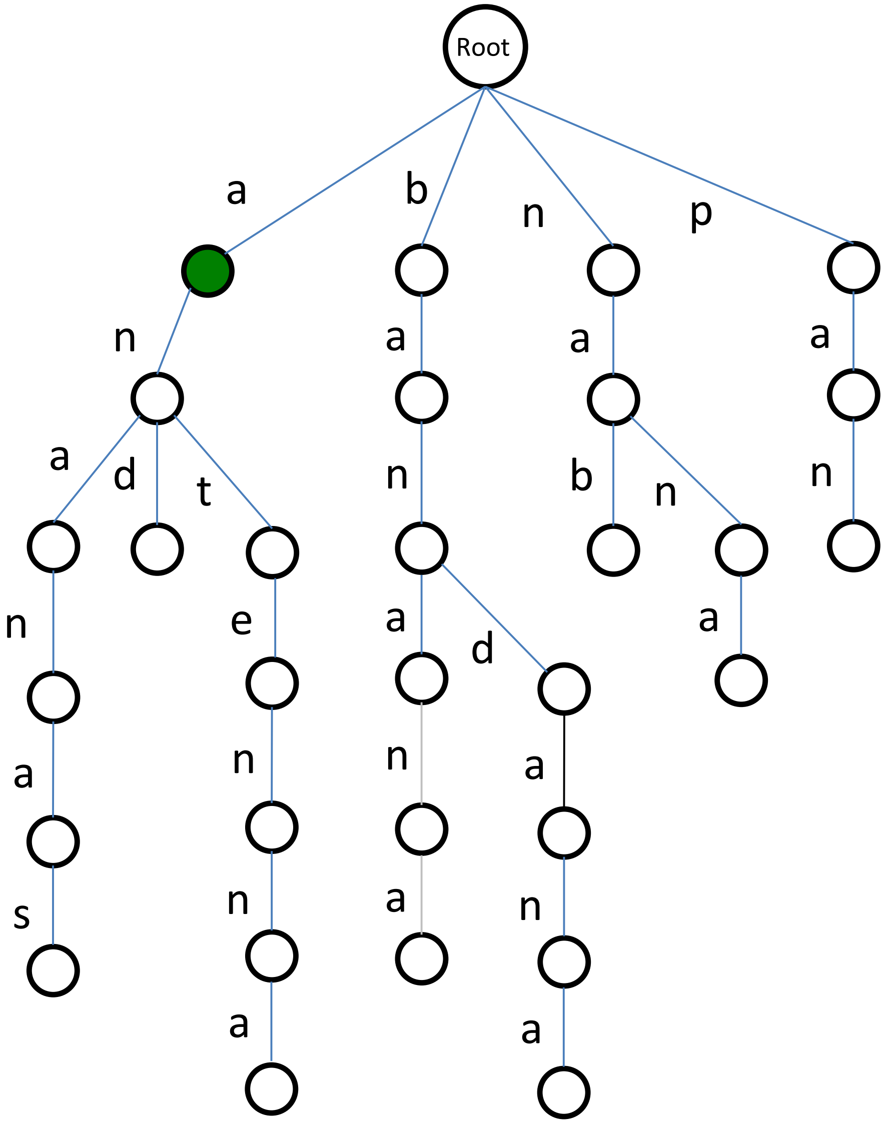
p a n a m a b a n a **n a** s



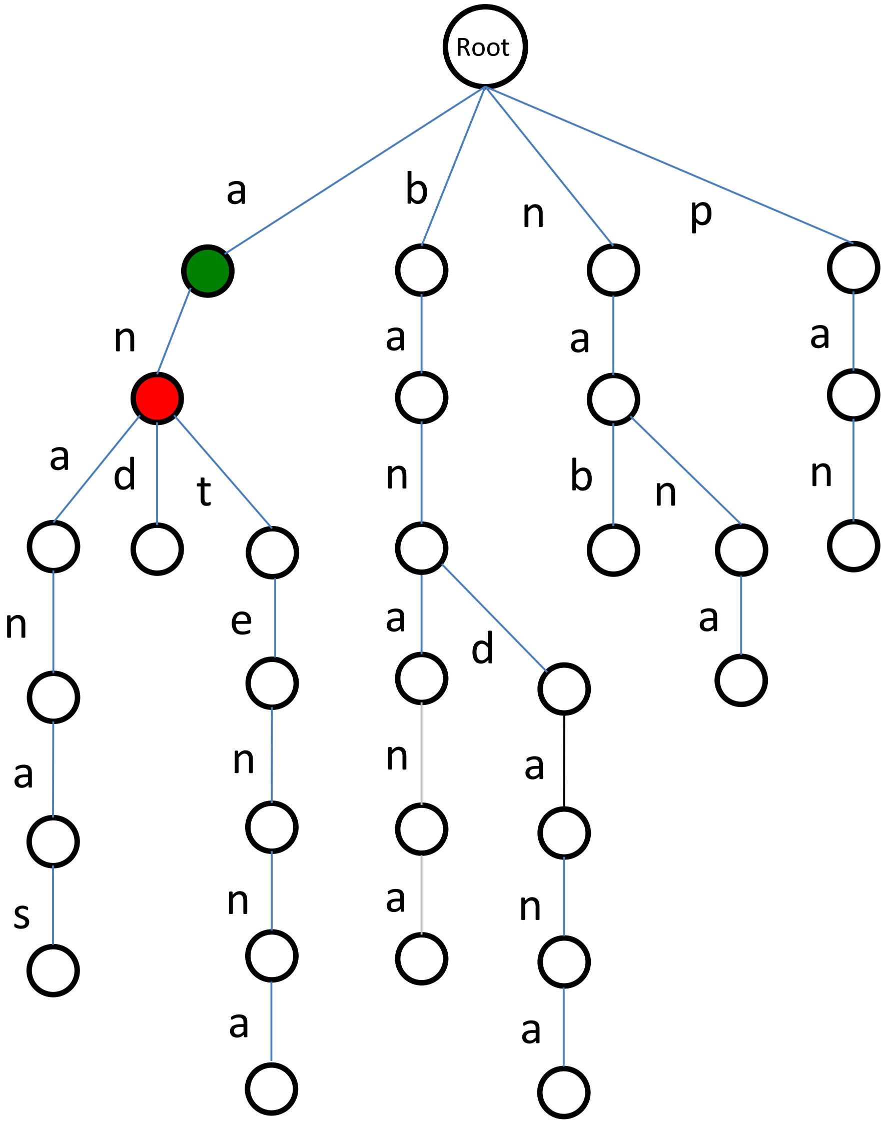
p a n a m a b a n a n a s



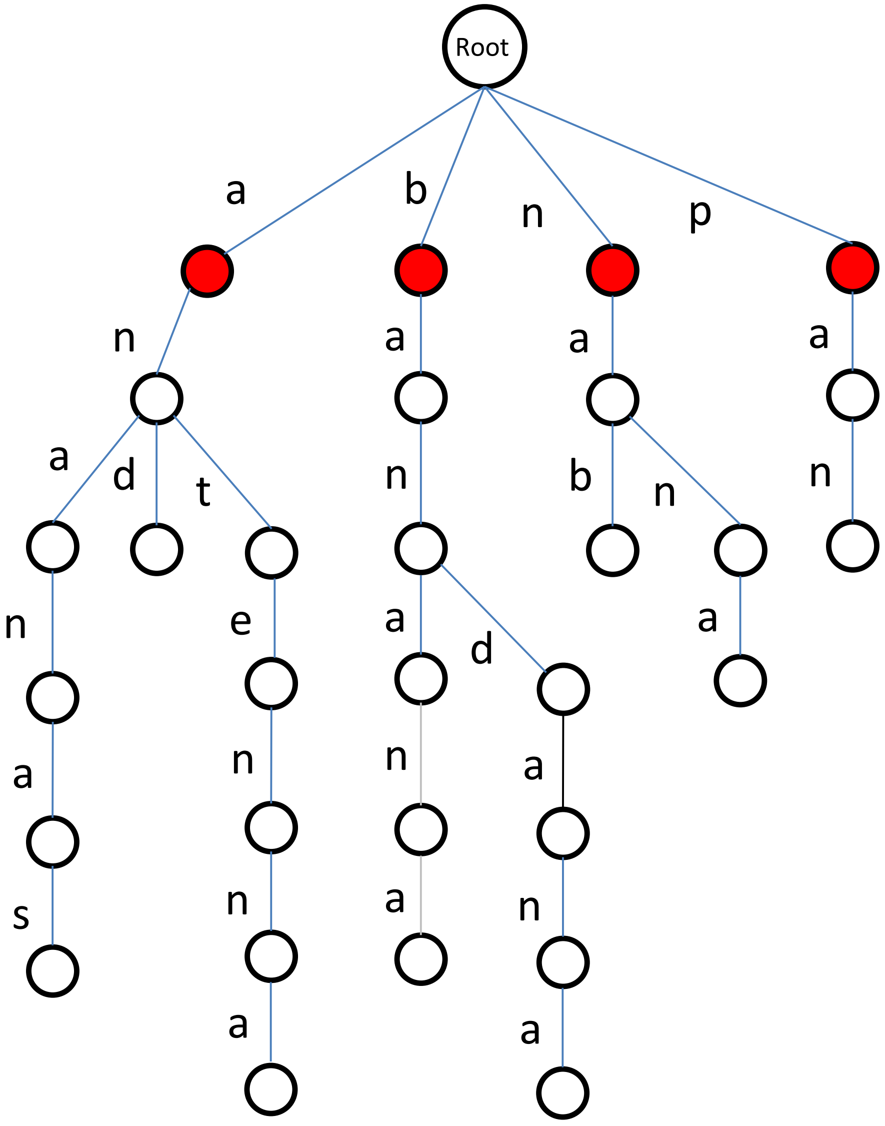
p a n a m a b a n a n a **a** s



p a n a m a b a n a n a s



p a n a m a b a n a n a **s**



Our Bus Is Fast!

- Runtime of brute force approach:
  - O(|*Text*| • |*Patterns*|)

- Runtime of **TrieMatching**:
  - O(|*Text*| * |*LongestPattern*|)

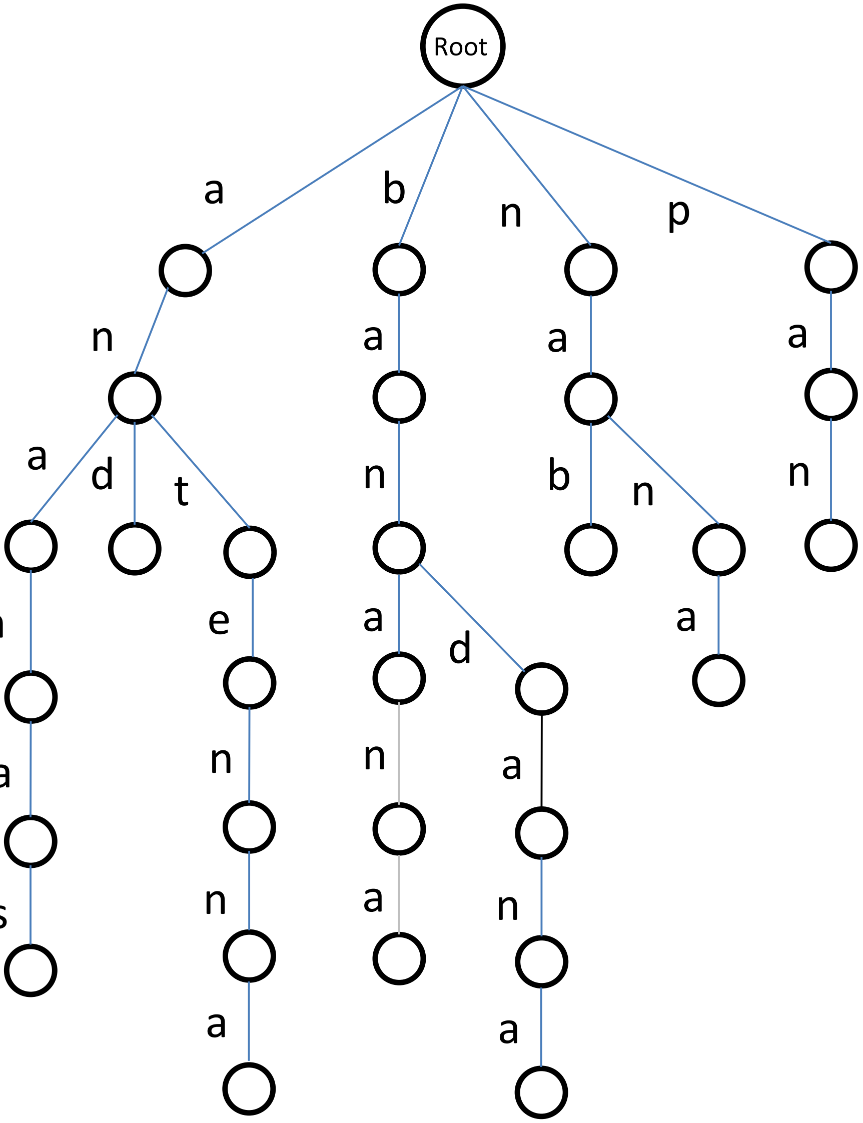
Trie construction takes O(|*Patterns*|) time

Memory Footprint of **TrieMatching**

- Our trie has 30 edges

- # edges = O(|*Patterns*|)!

- For human genome: |*Patterns*| ≈ 10^{12}

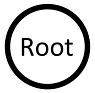


Outline

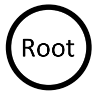
- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- **Herding Text into Suffix Trie**
- From Suffix Tries to Suffix Trees

New Idea: Packing *Text* onto a Bus

- Generate all suffixes of *Text*
- Form a trie out of these suffixes (**suffix trie**)
- For each *Pattern*, check if it can be spelled out from the root downward in the suffix trie

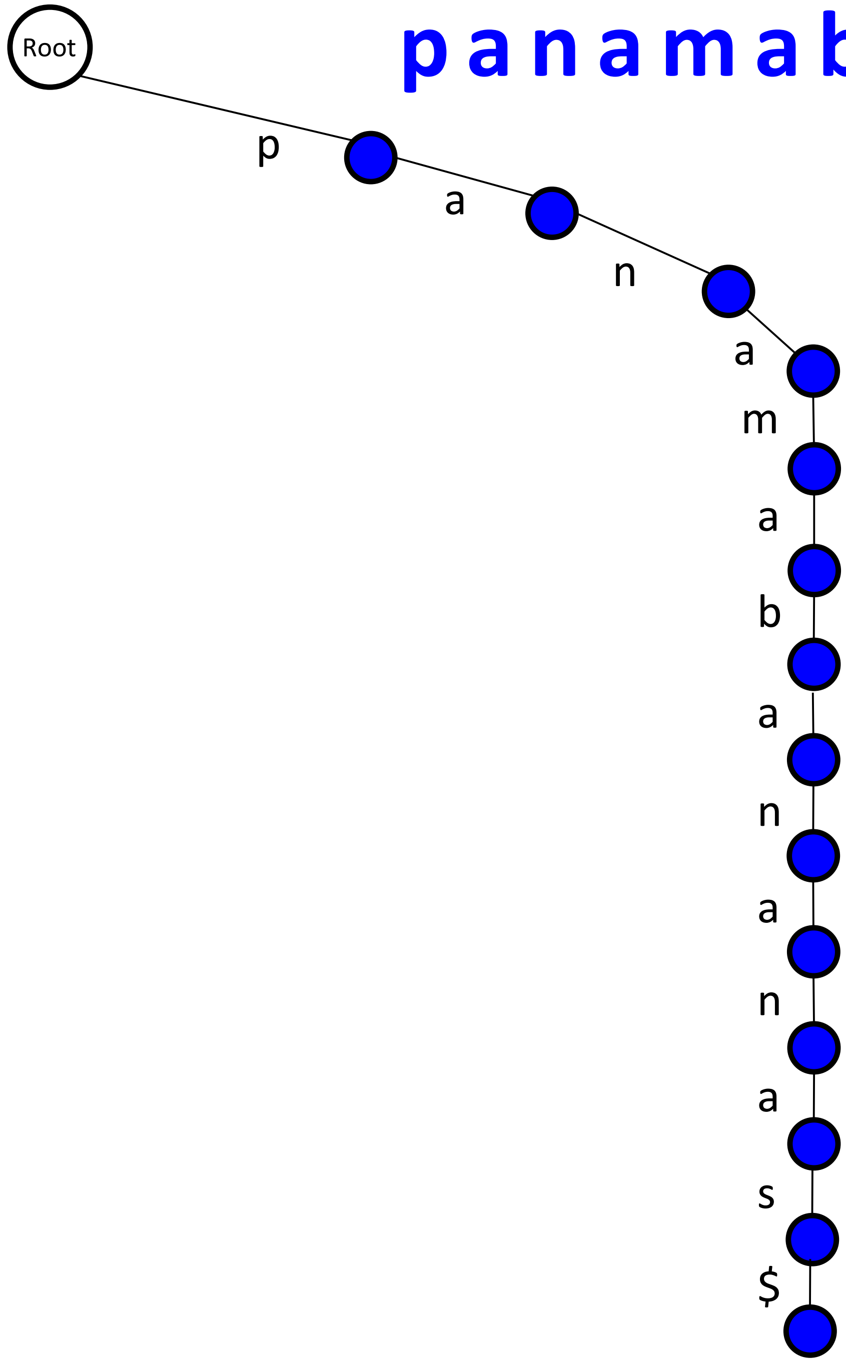


p a n a m a b a n a n a s



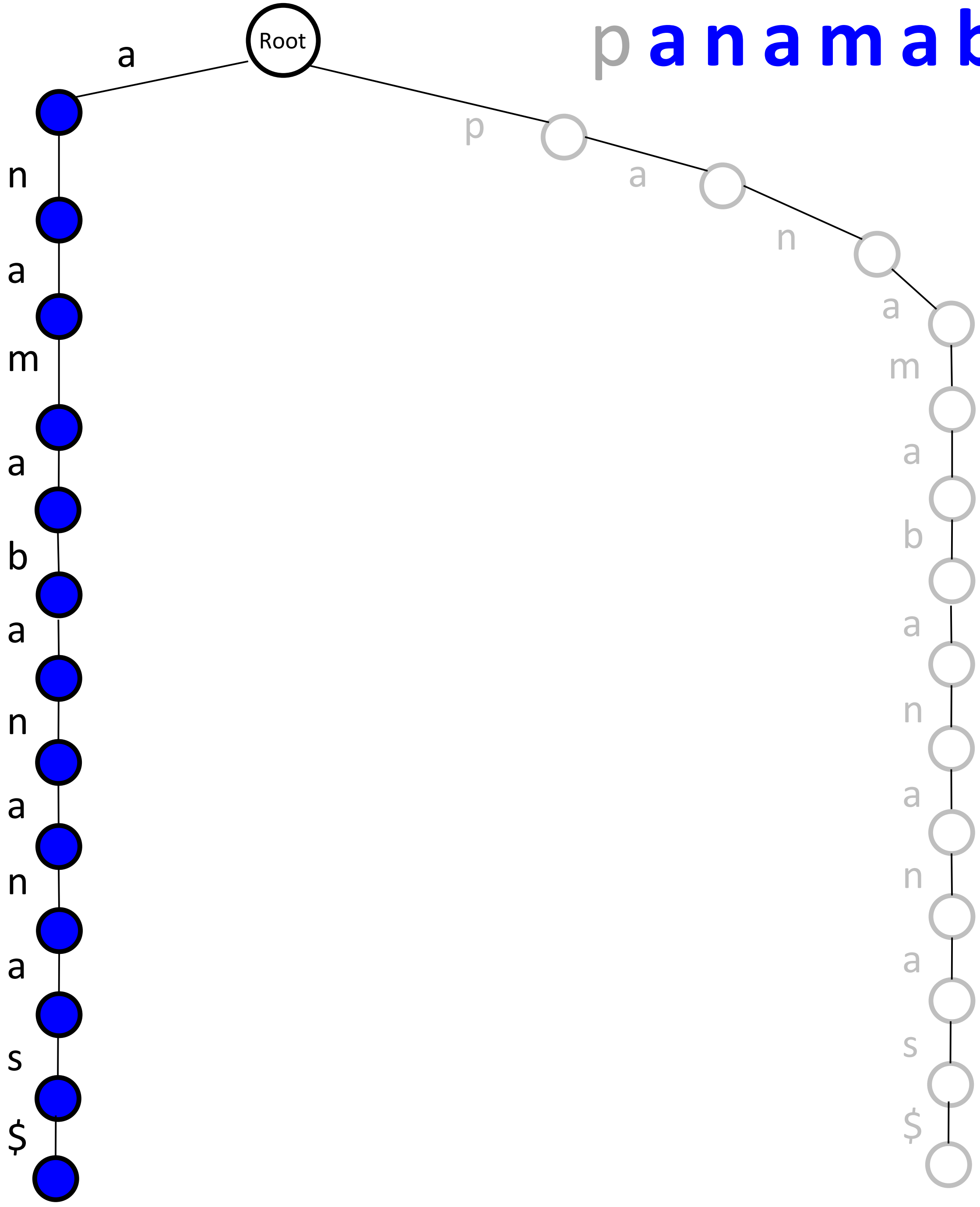
p a n a m a b a n a n a s $

Adding “$” sign in the end (we’ll explain later why)

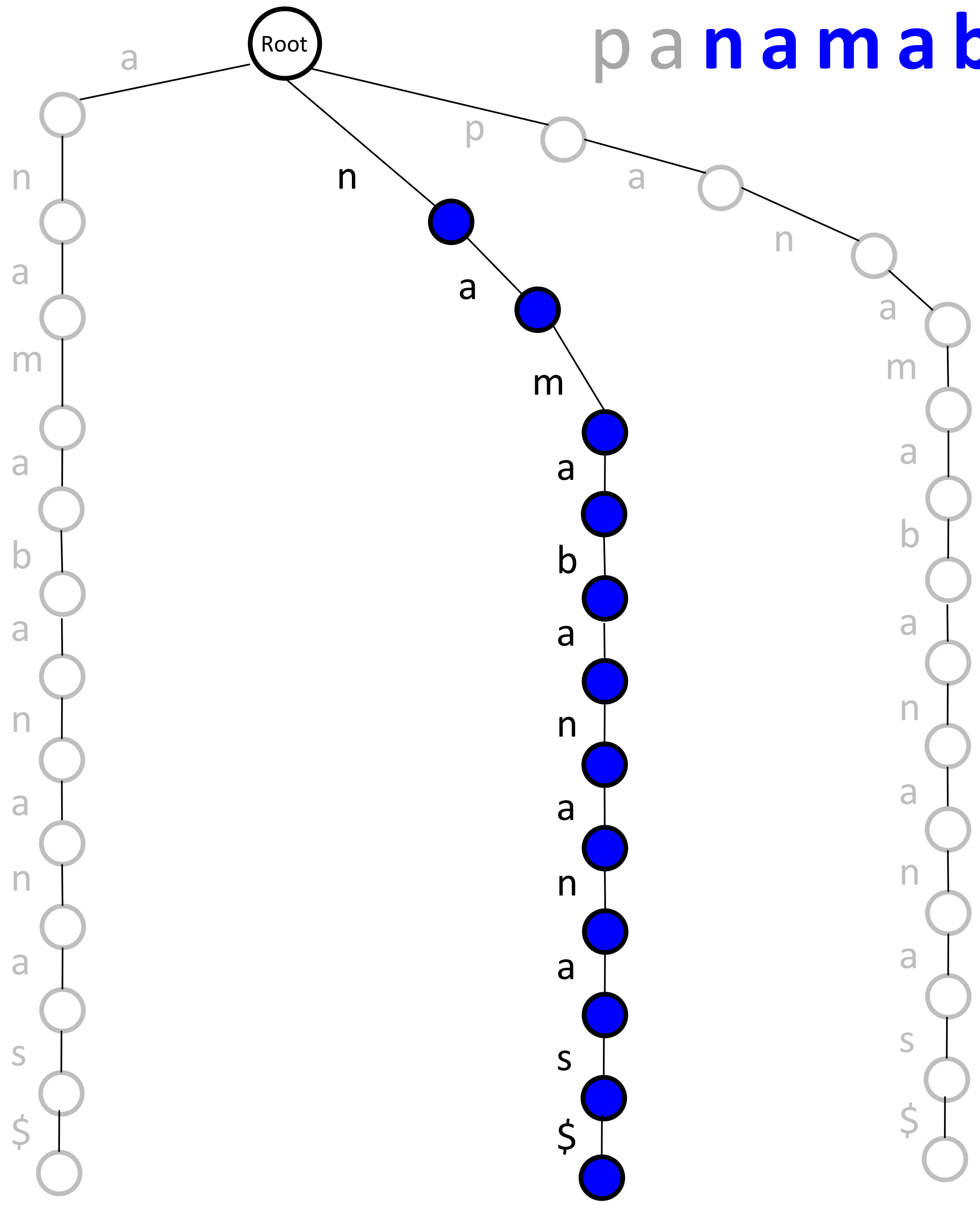


p a n a m a b a n a n a s $

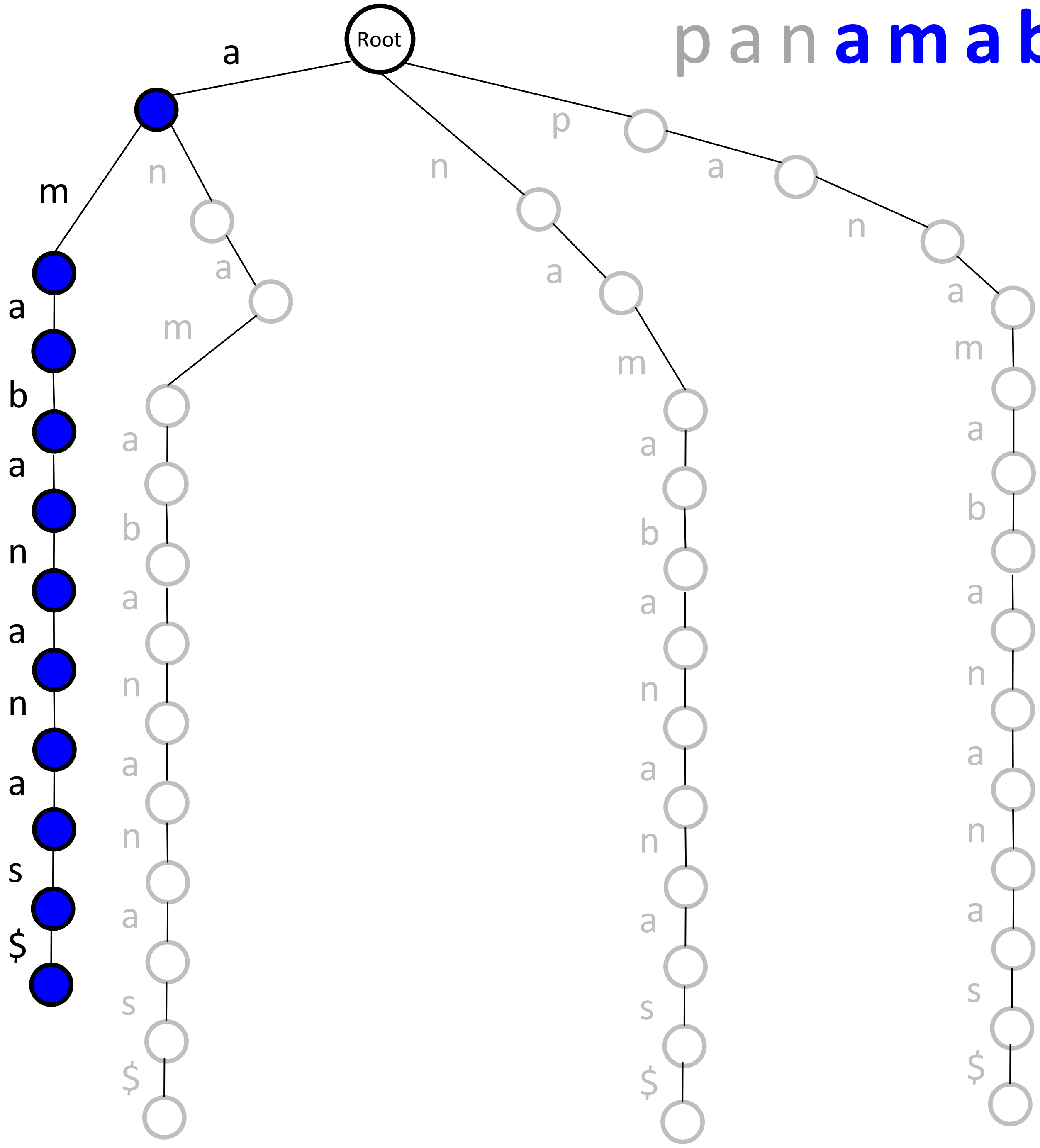
p a n a m a b a n a n a s $



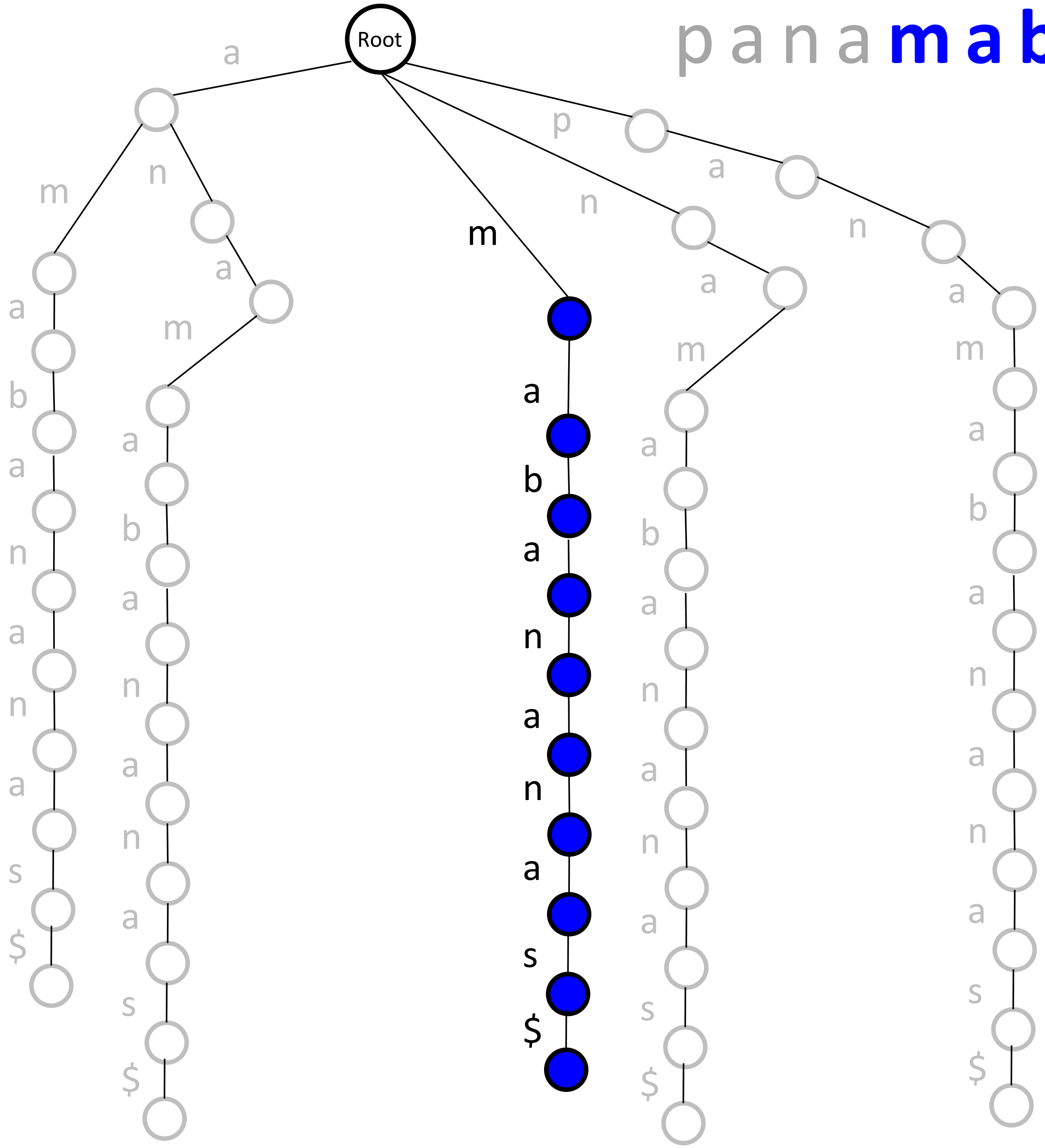
p a **n a m a b a n a n a s $**

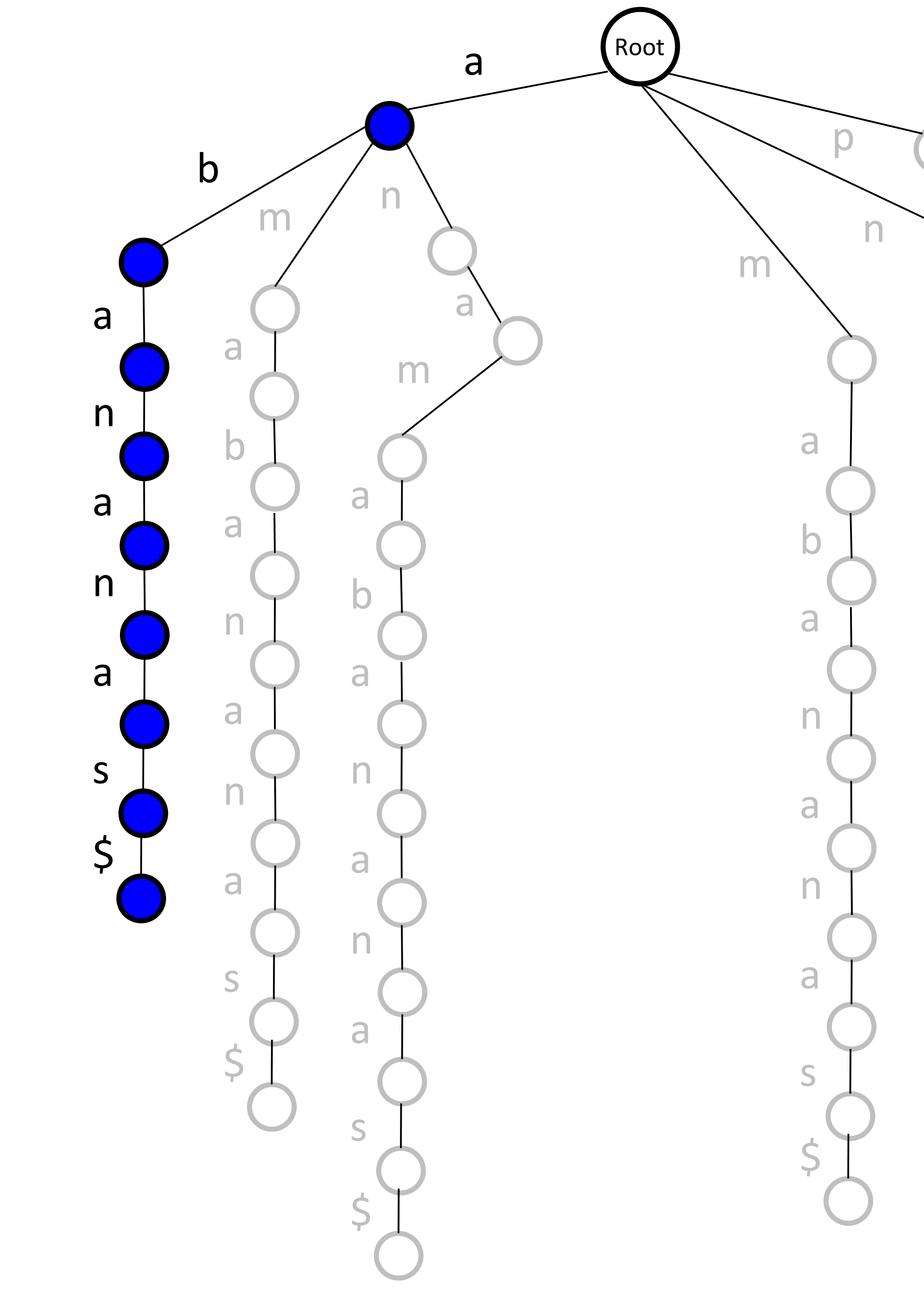


p a n **a m a b a n a n a s $**



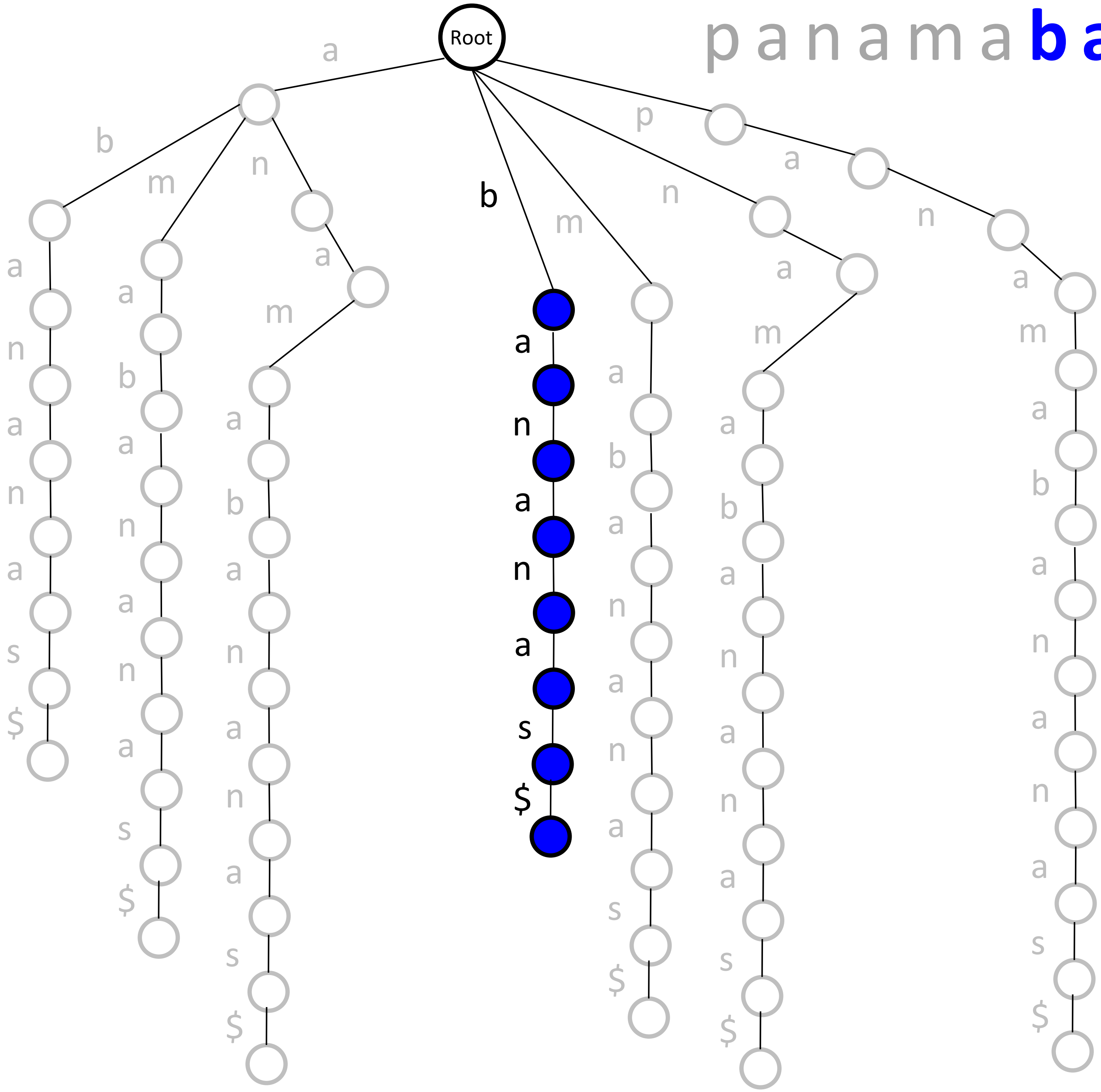
panamabananas$



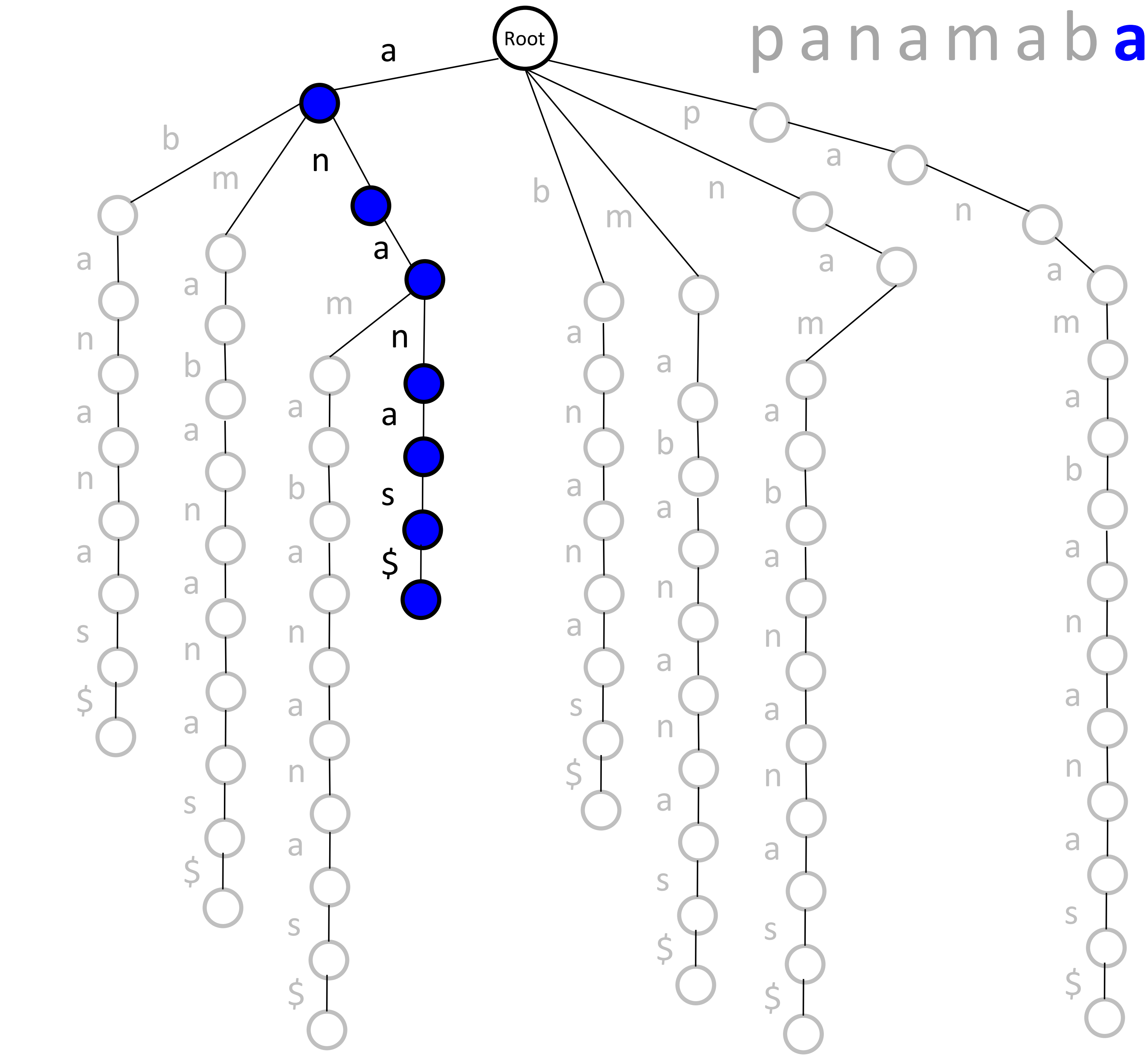


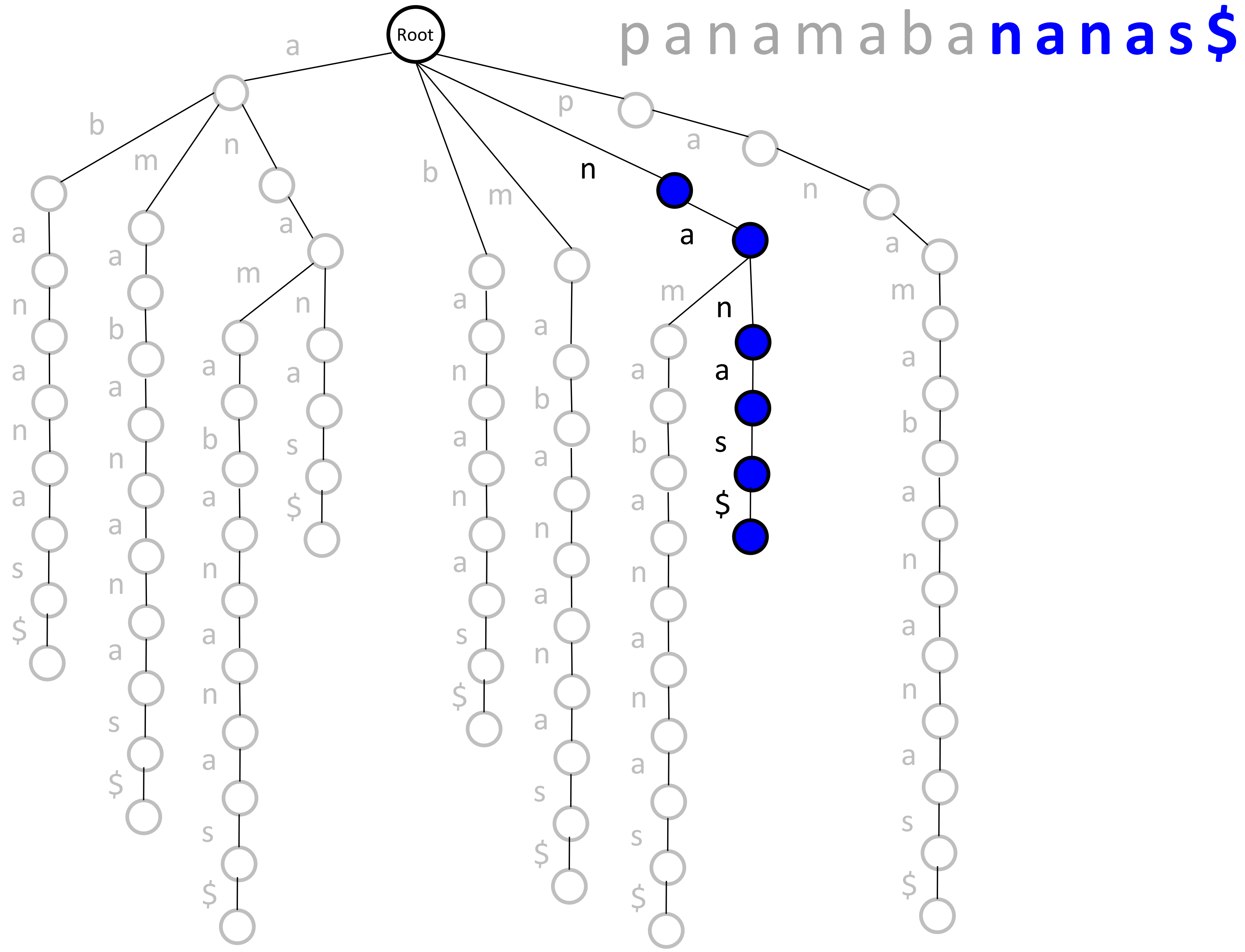
p a n a m **a b a n a n a s $**

p a n a m a **b a n a n a s $**

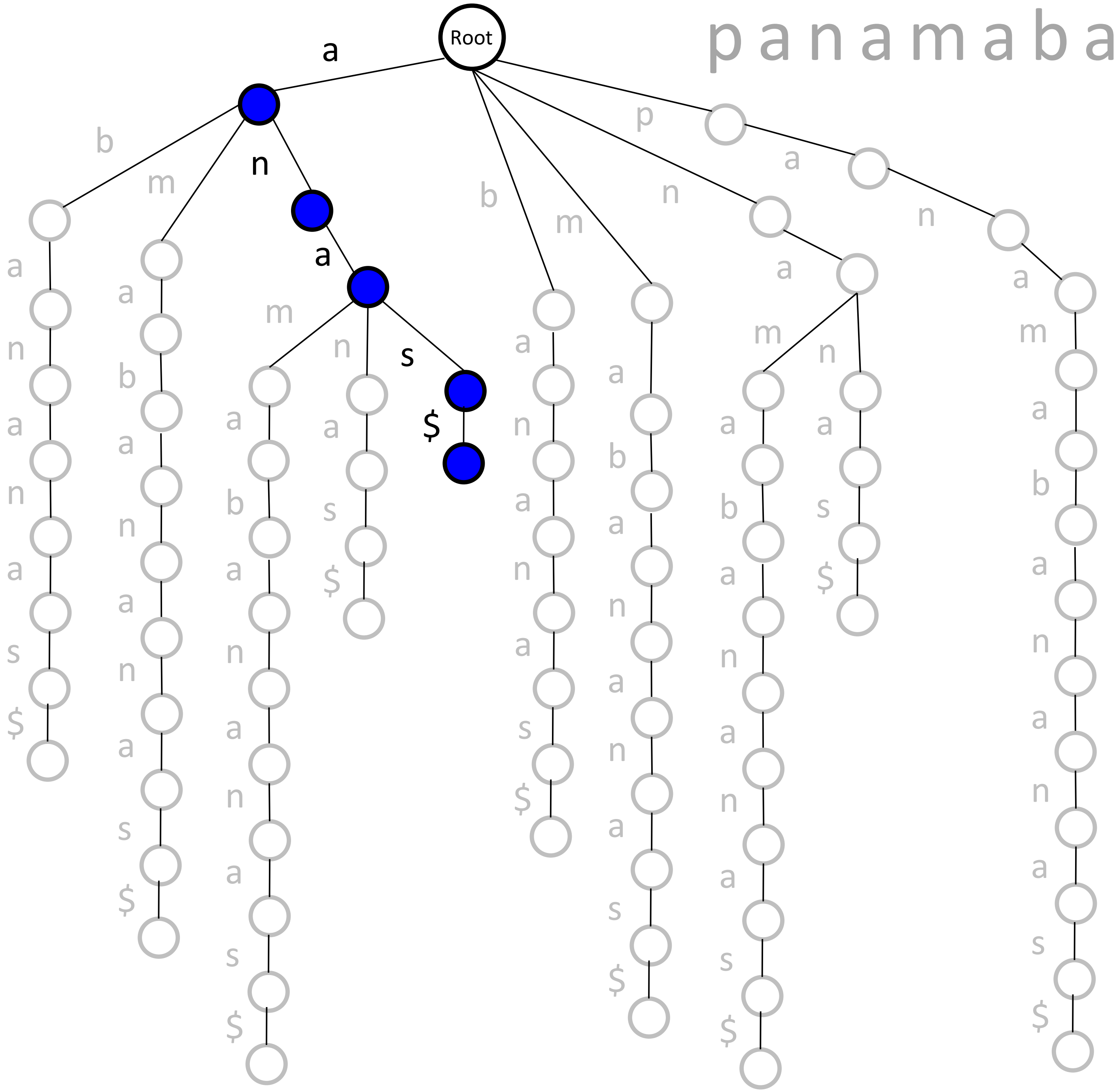


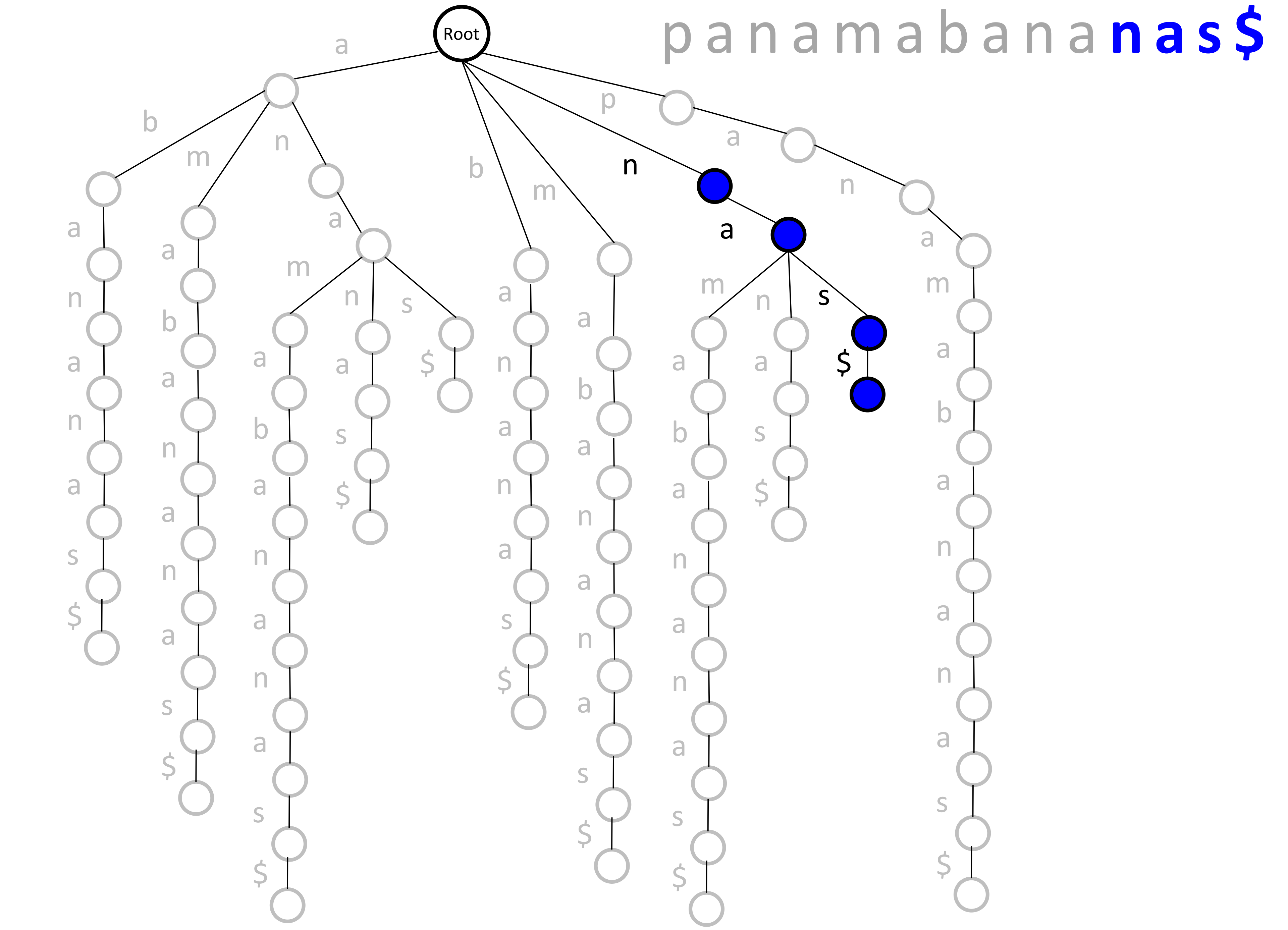
p a n a m a b **a n a n a s $**



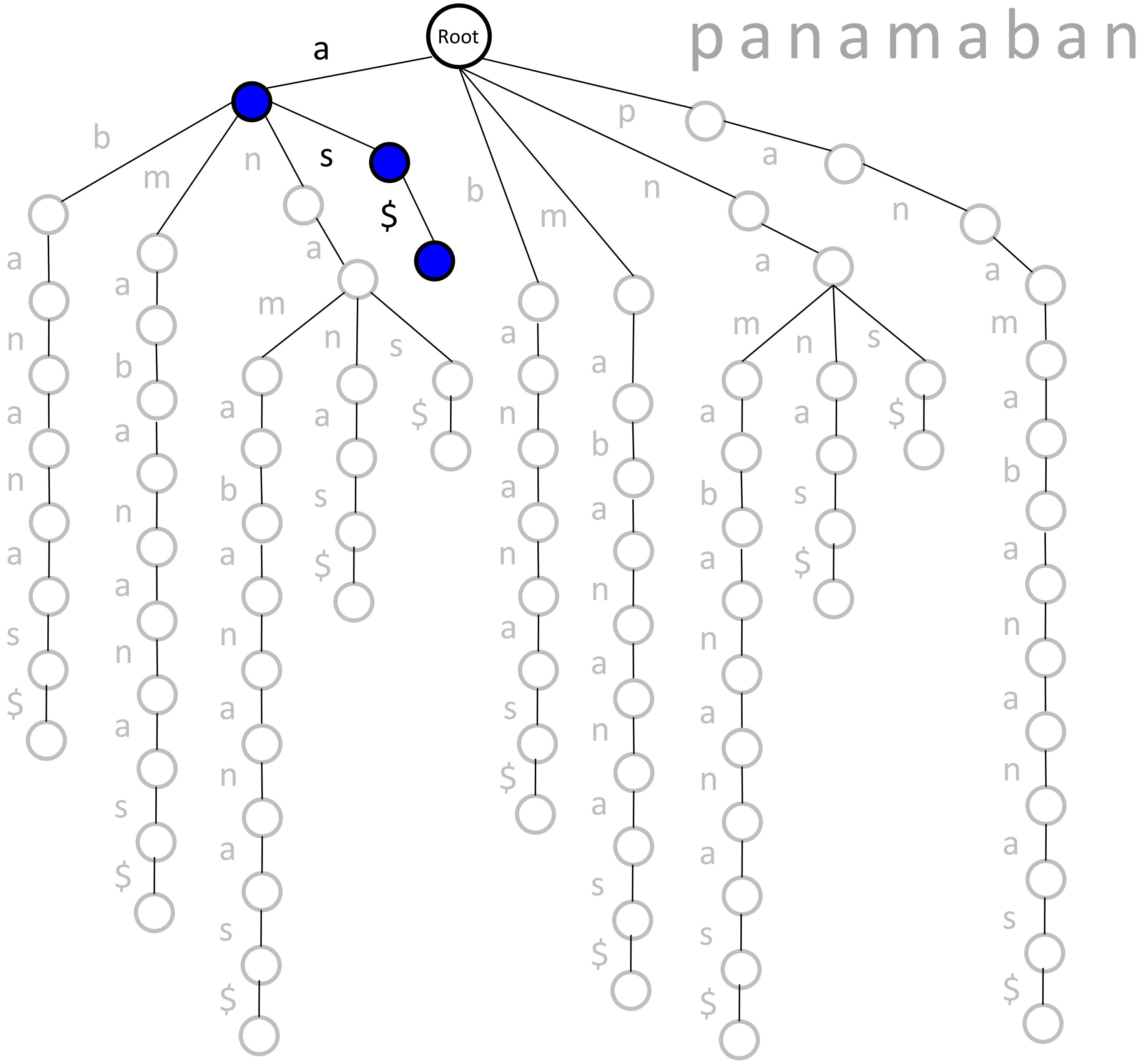

p a n a m a b a n a n a s $
Root
a
b
m
n
p
a
n
a
m
n
a
b
a
n
a
n
a
s
$

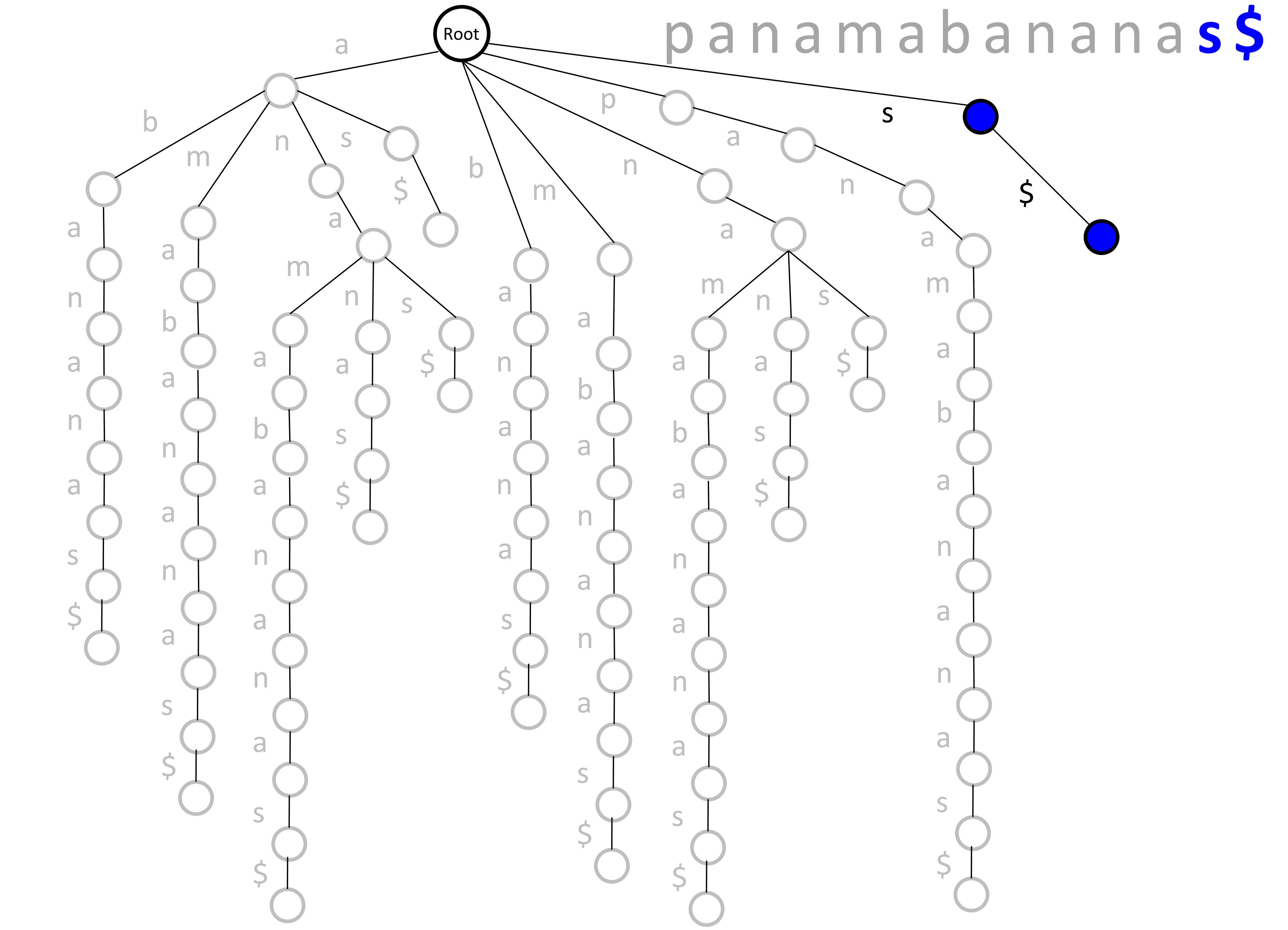
p a n a m a b a n **a n a s $**

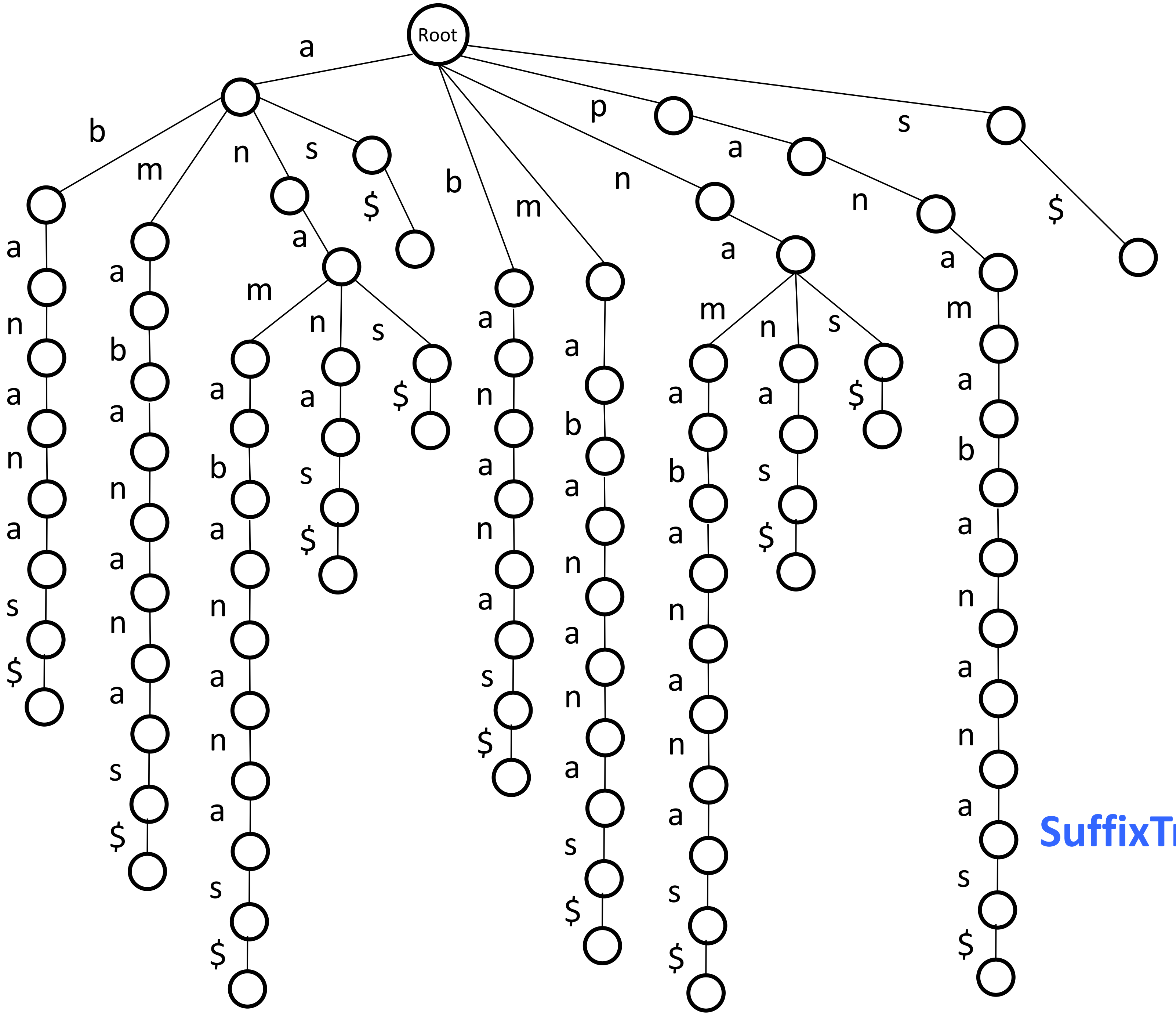



p a n a m a b a n a **n a s $**
Root
a
b
m
n
a
m
n
s
p
a
n
s
$

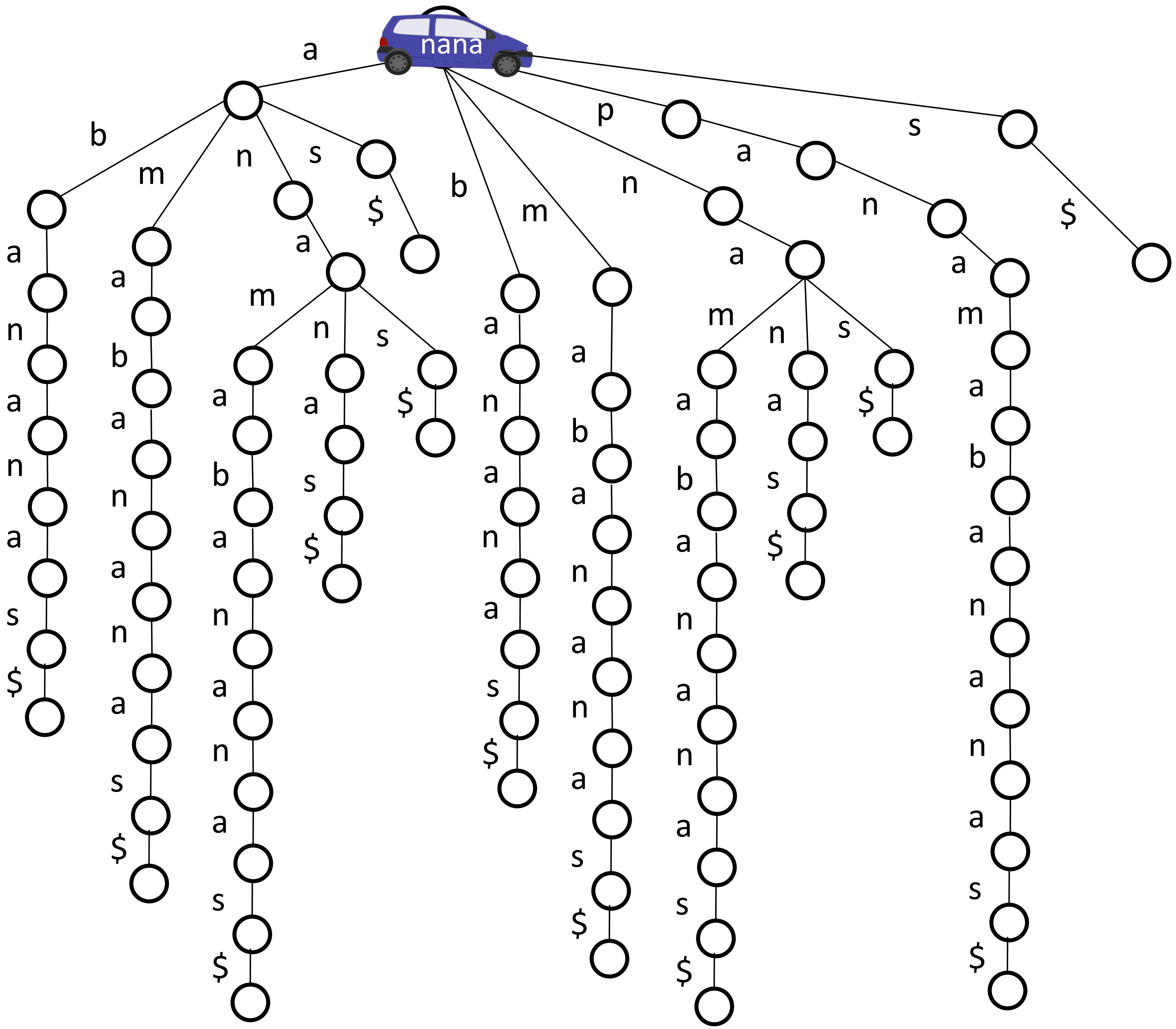
p a n a m a b a n a n **a s $**

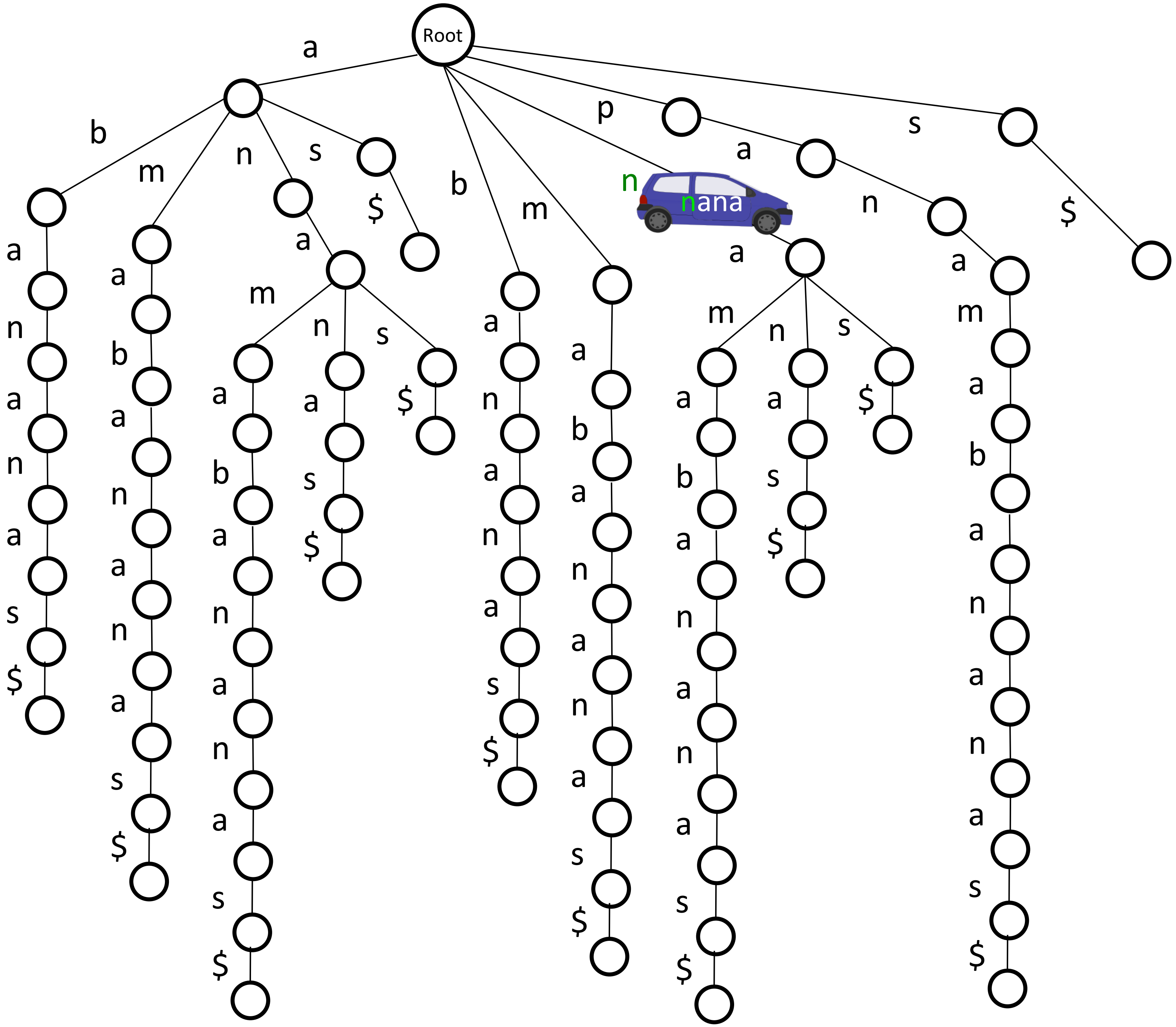


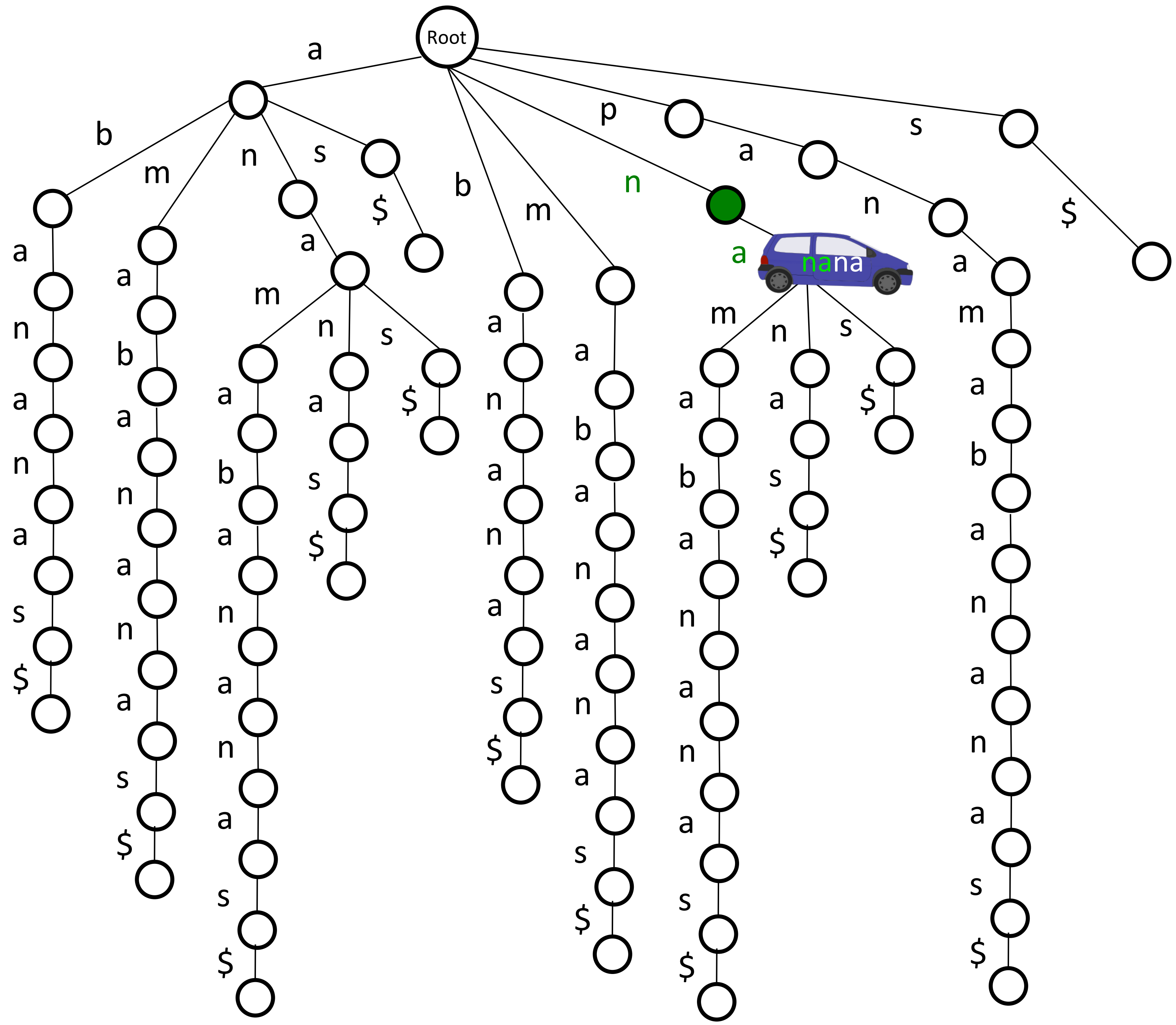

panamabananas$
Root
a
b
m
n
s
p
$

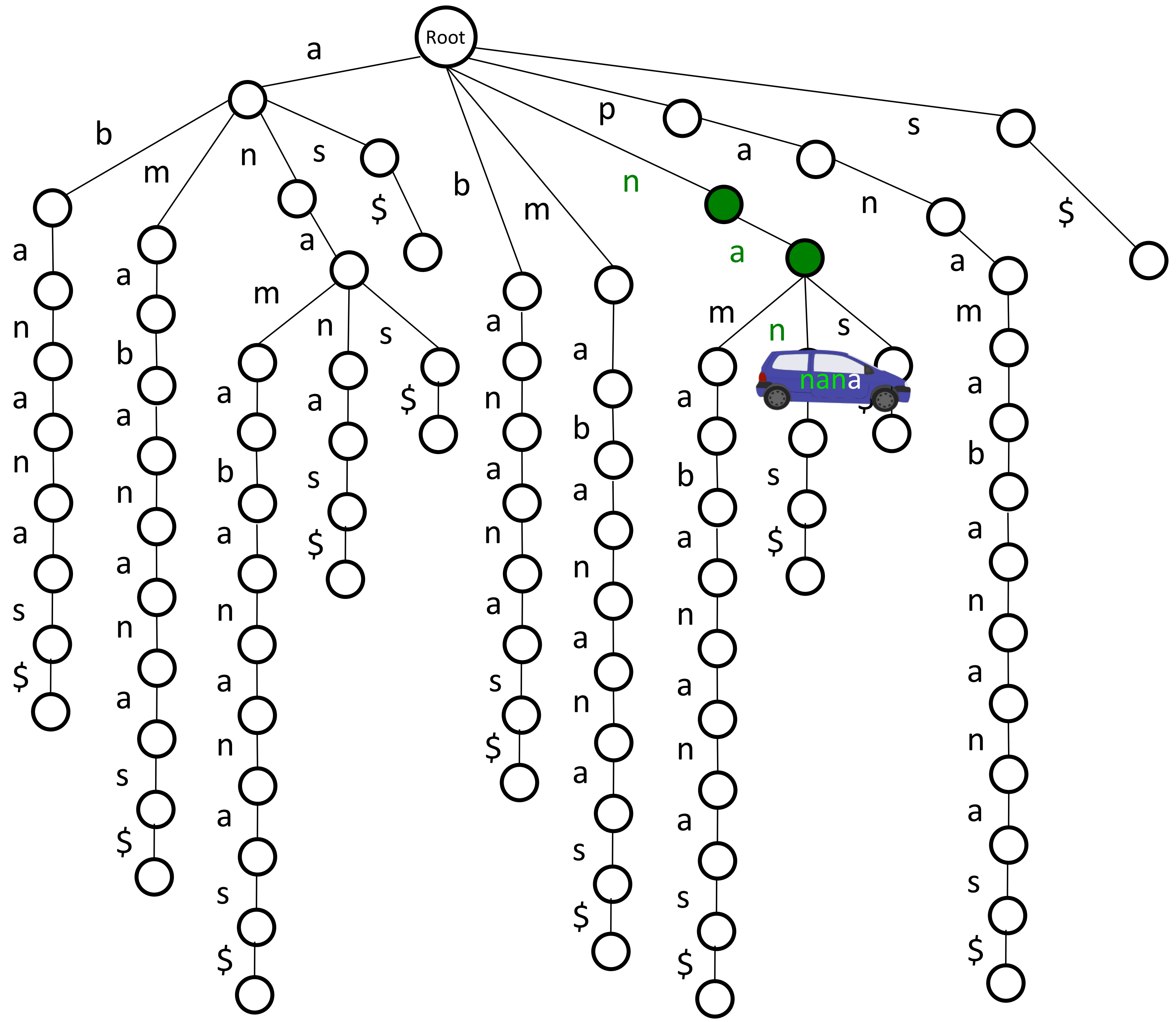


SuffixTrie(*Text*)

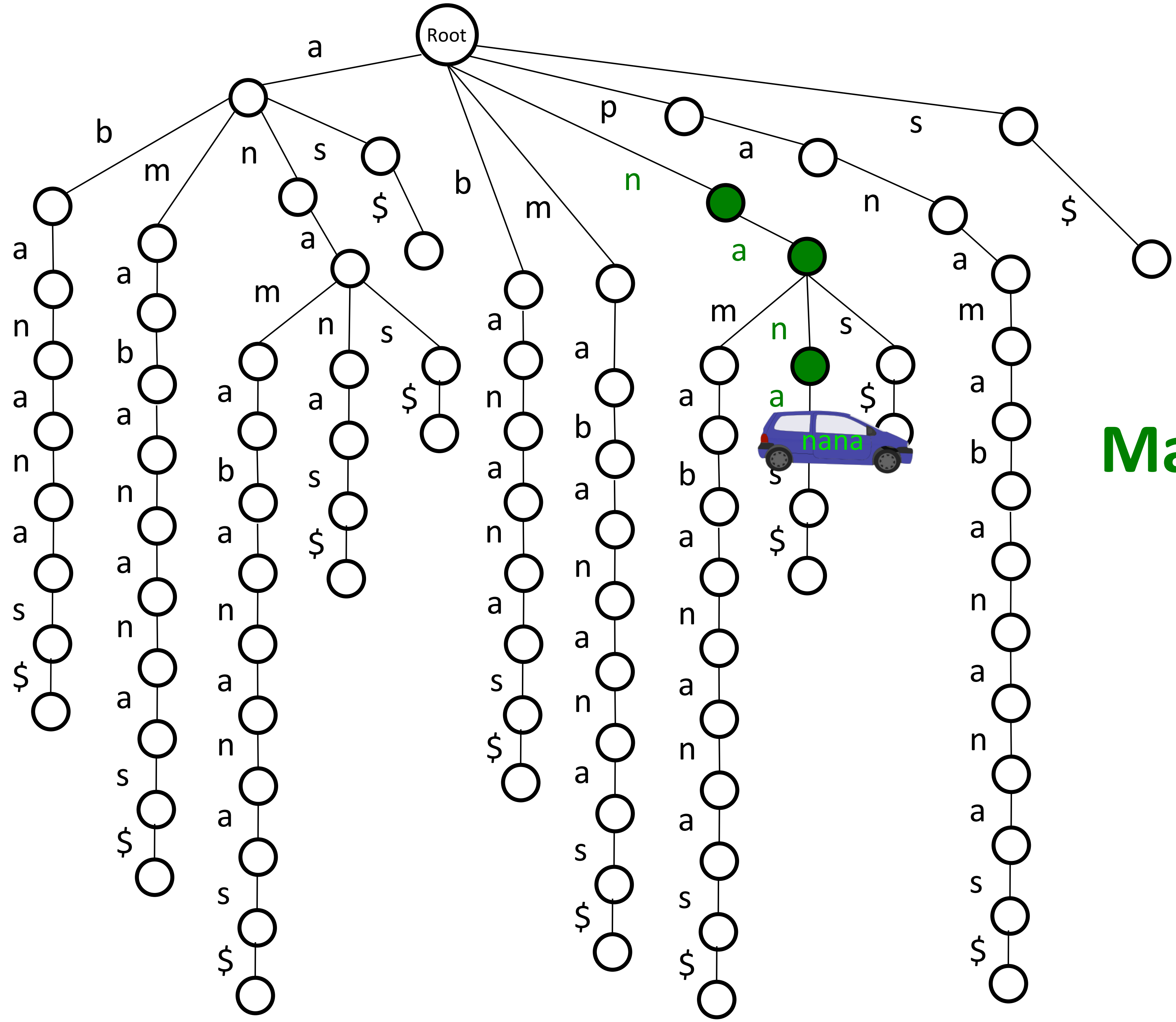

nana


Root
n
nana

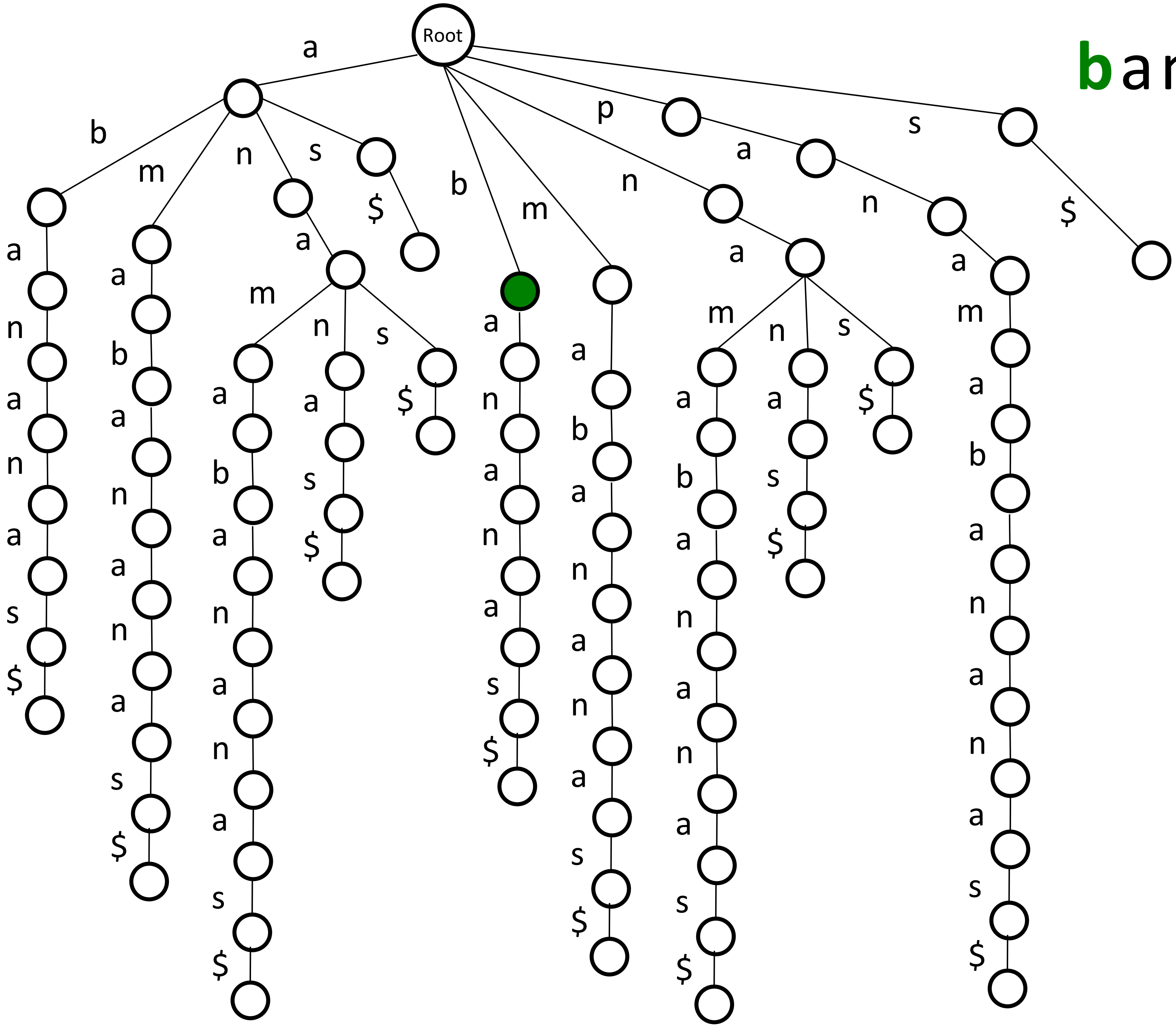

Root
nana



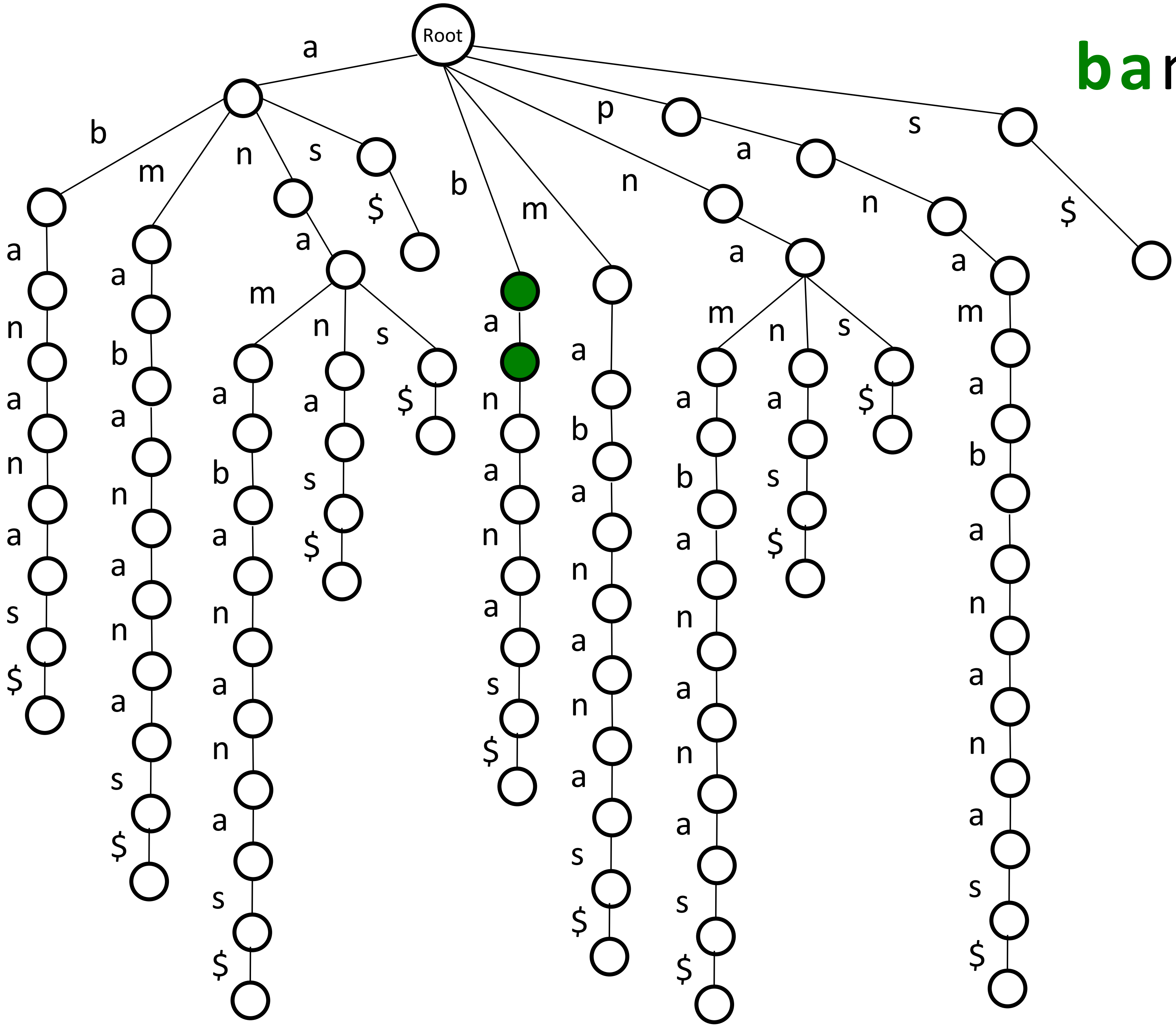
Root
nana



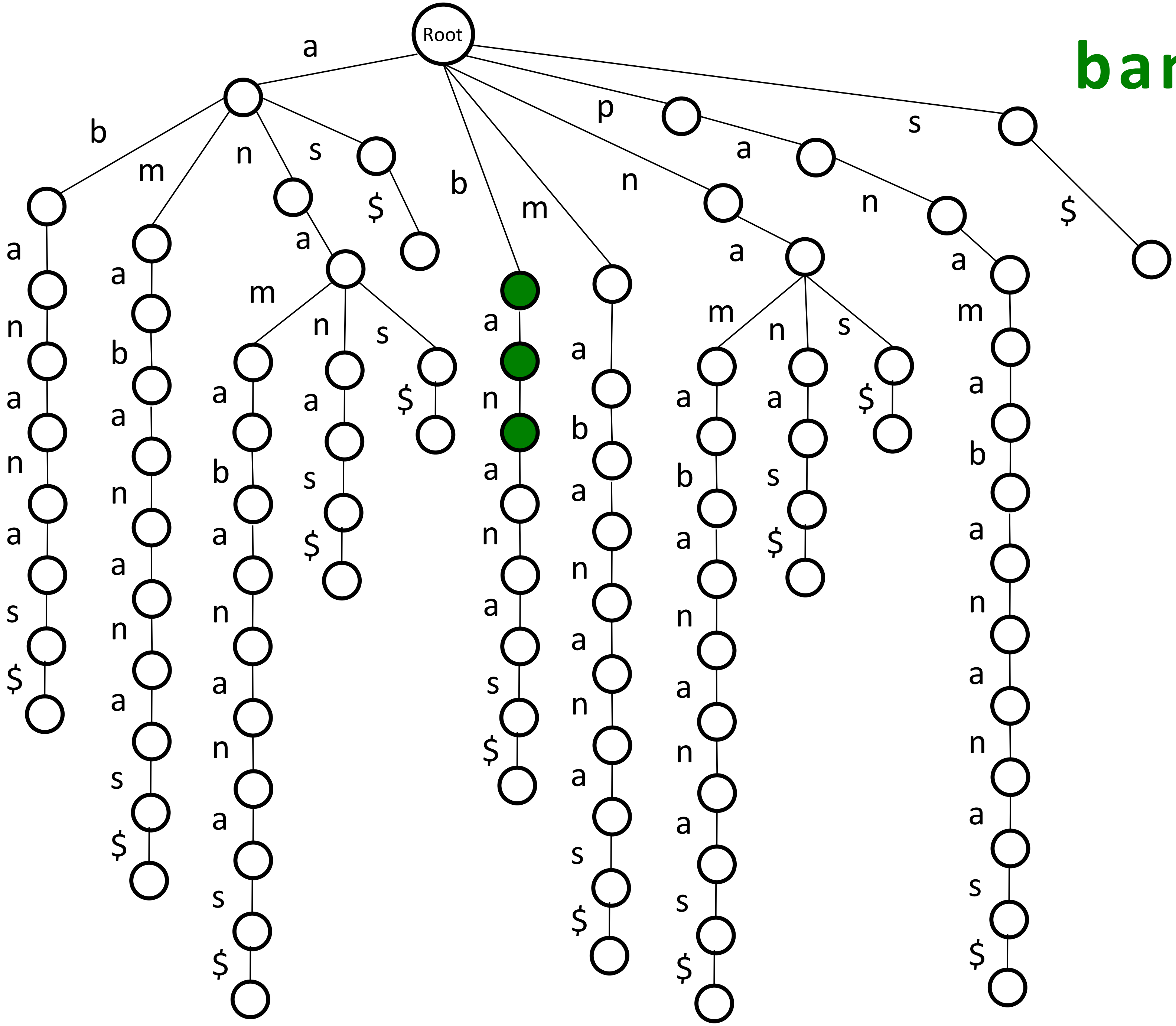
Match!



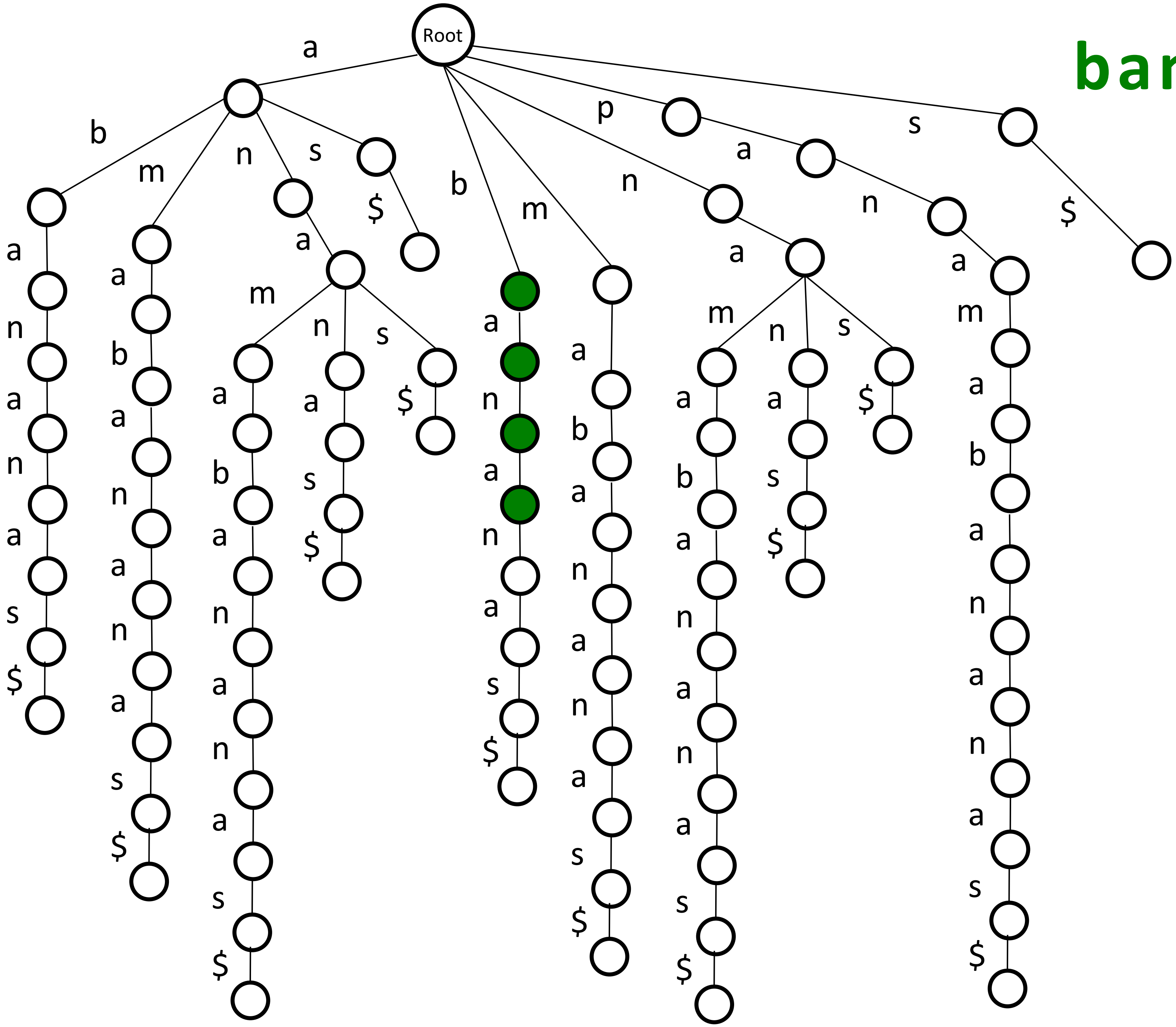
banana



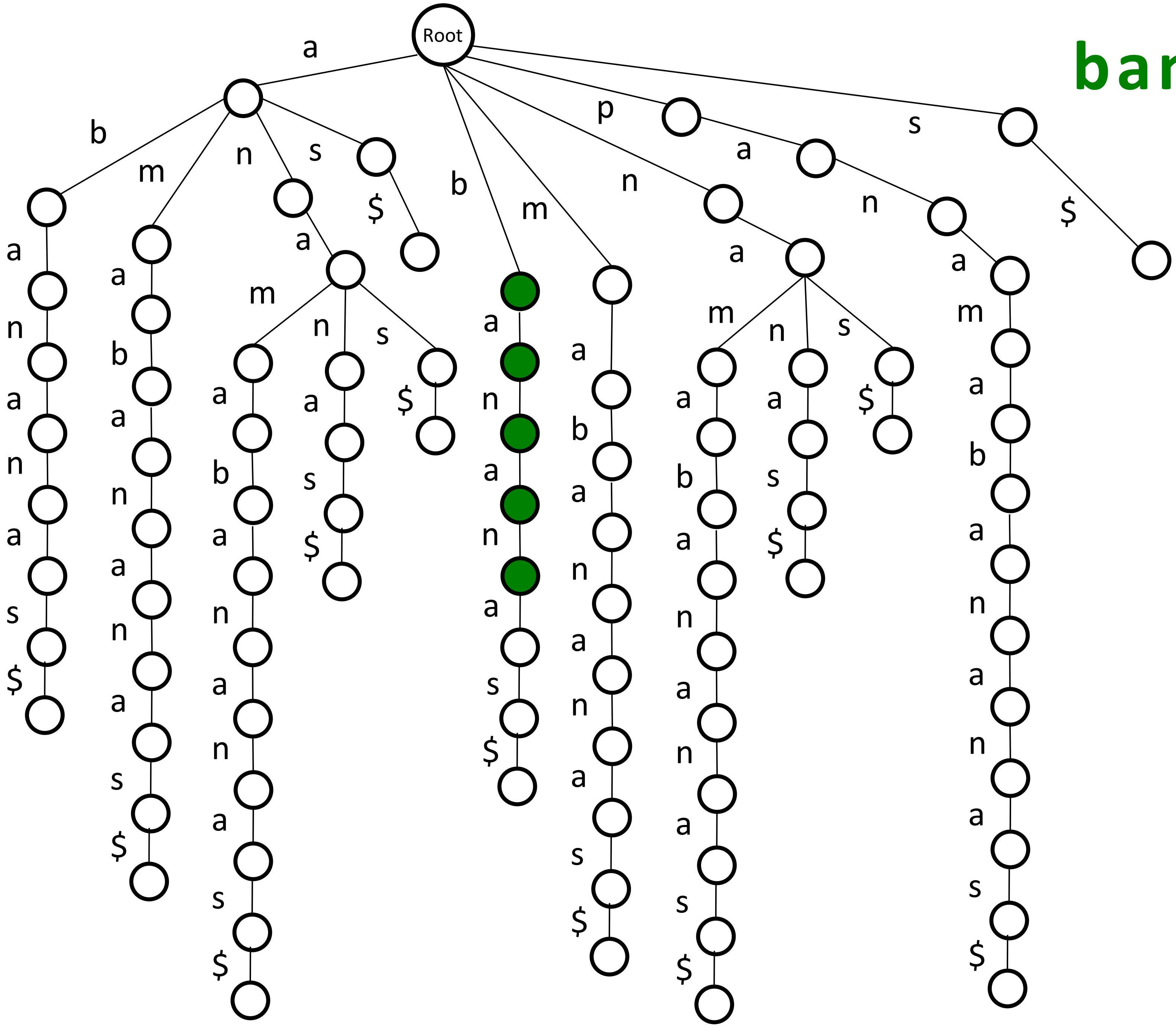
banana


Root
a b a n a n a s $
a m a b a n a n a s $
a n a m a b a n a n a s $
a n a n a s $
a n a s $
a s $
b a n a n a s $
m a b a n a n a s $
n a m a b a n a n a s $
n a n a s $
n a s $
p a n a m a b a n a n a s $
s $

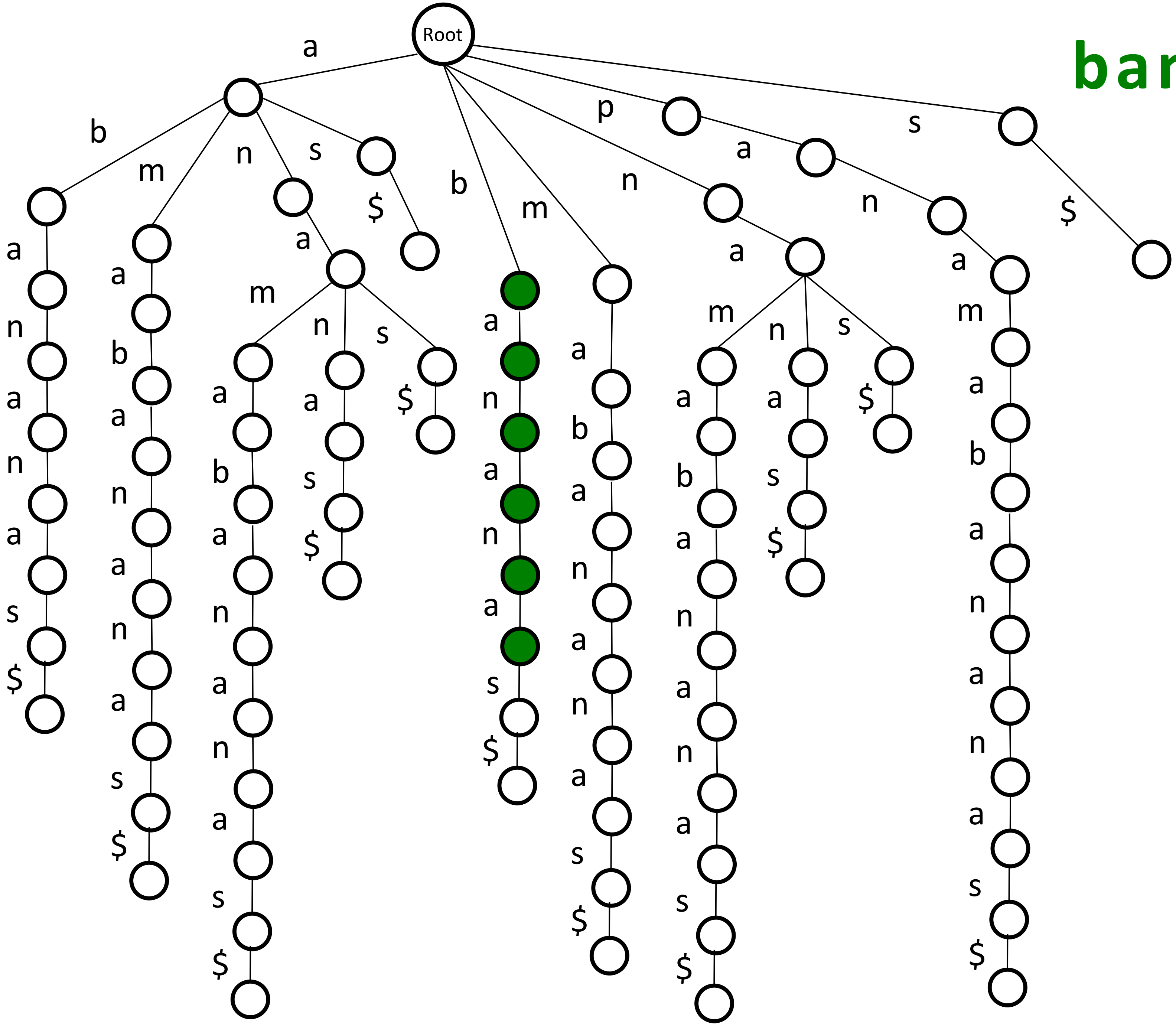
banana



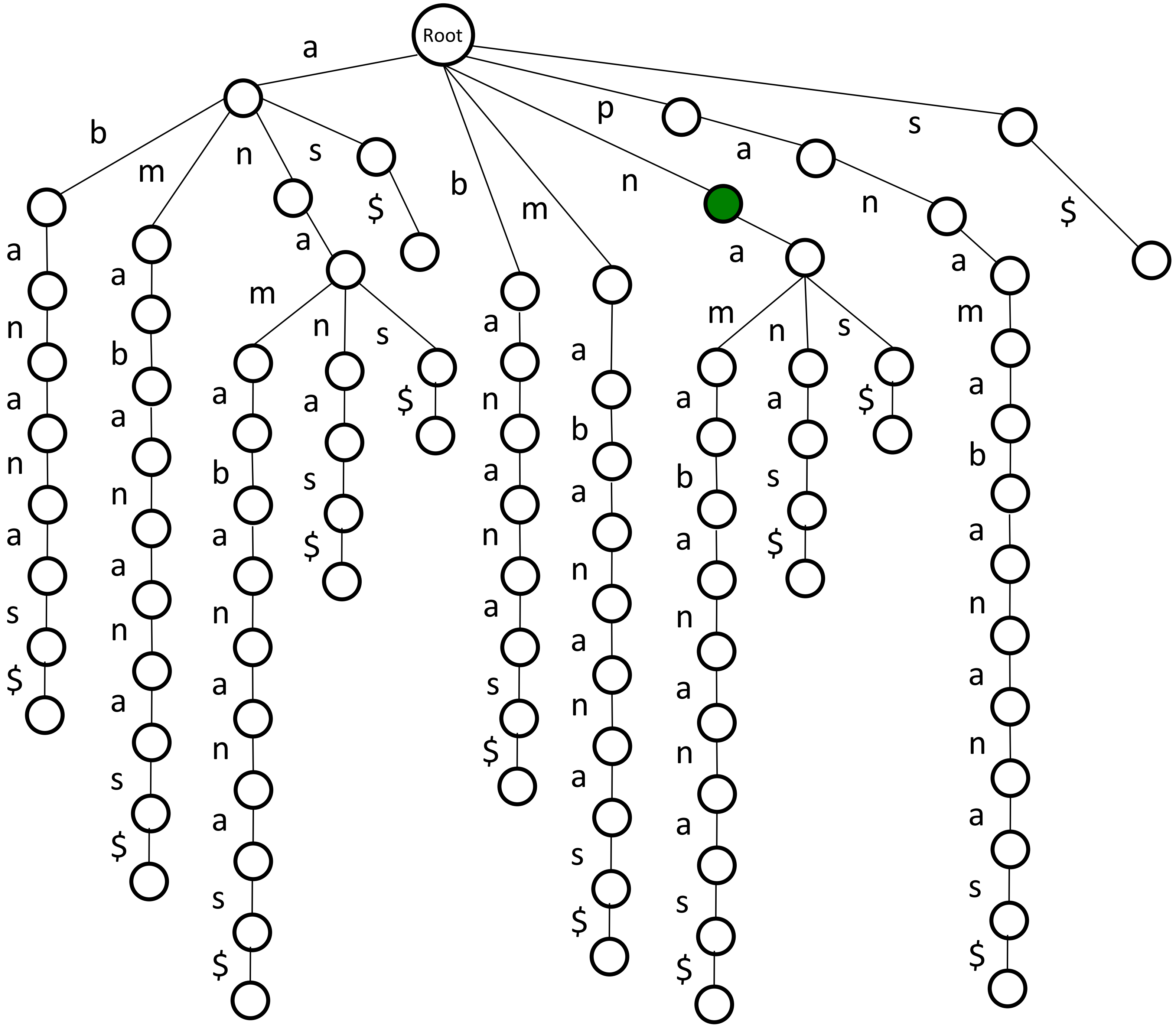
banana



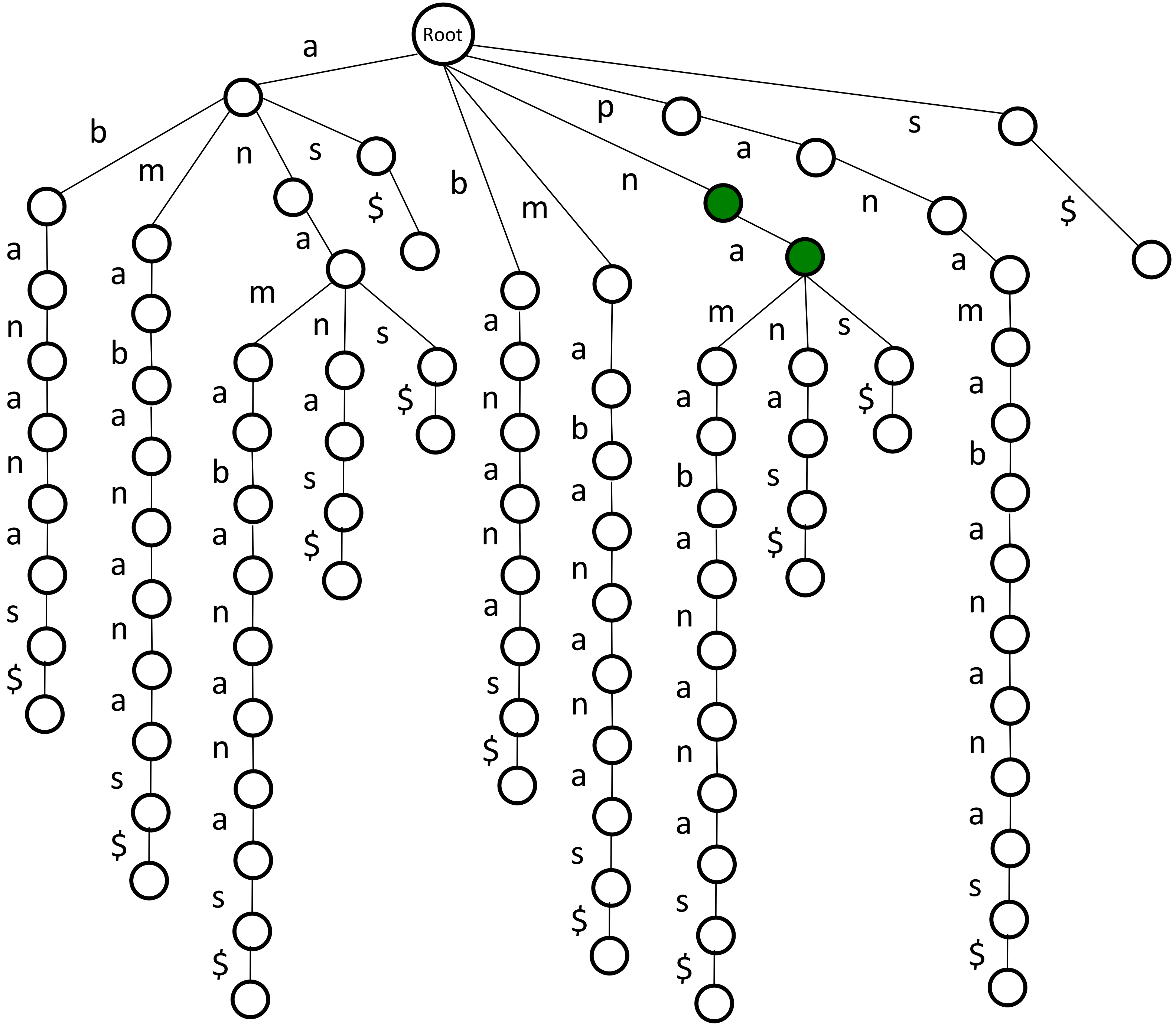
banana



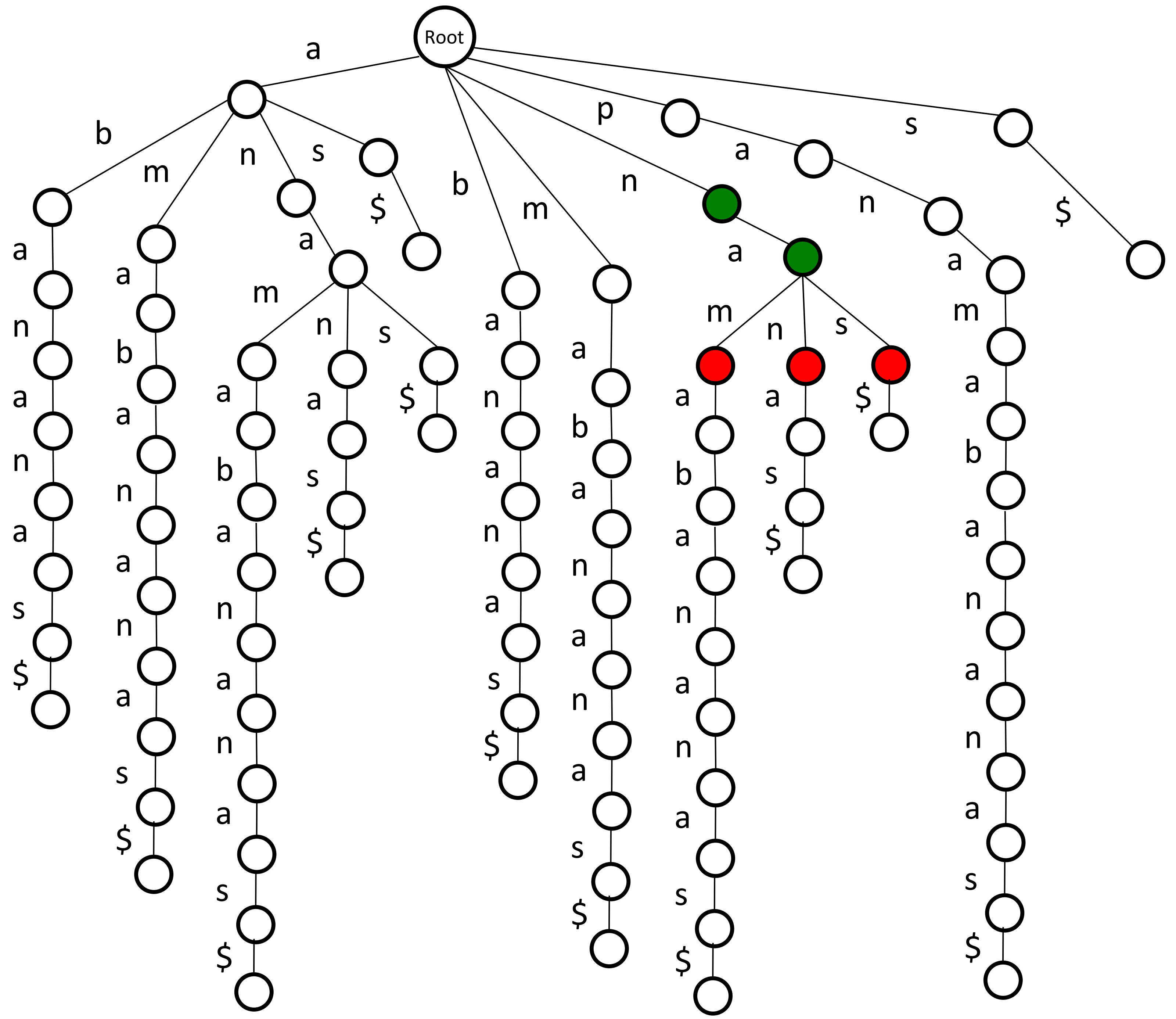
banana



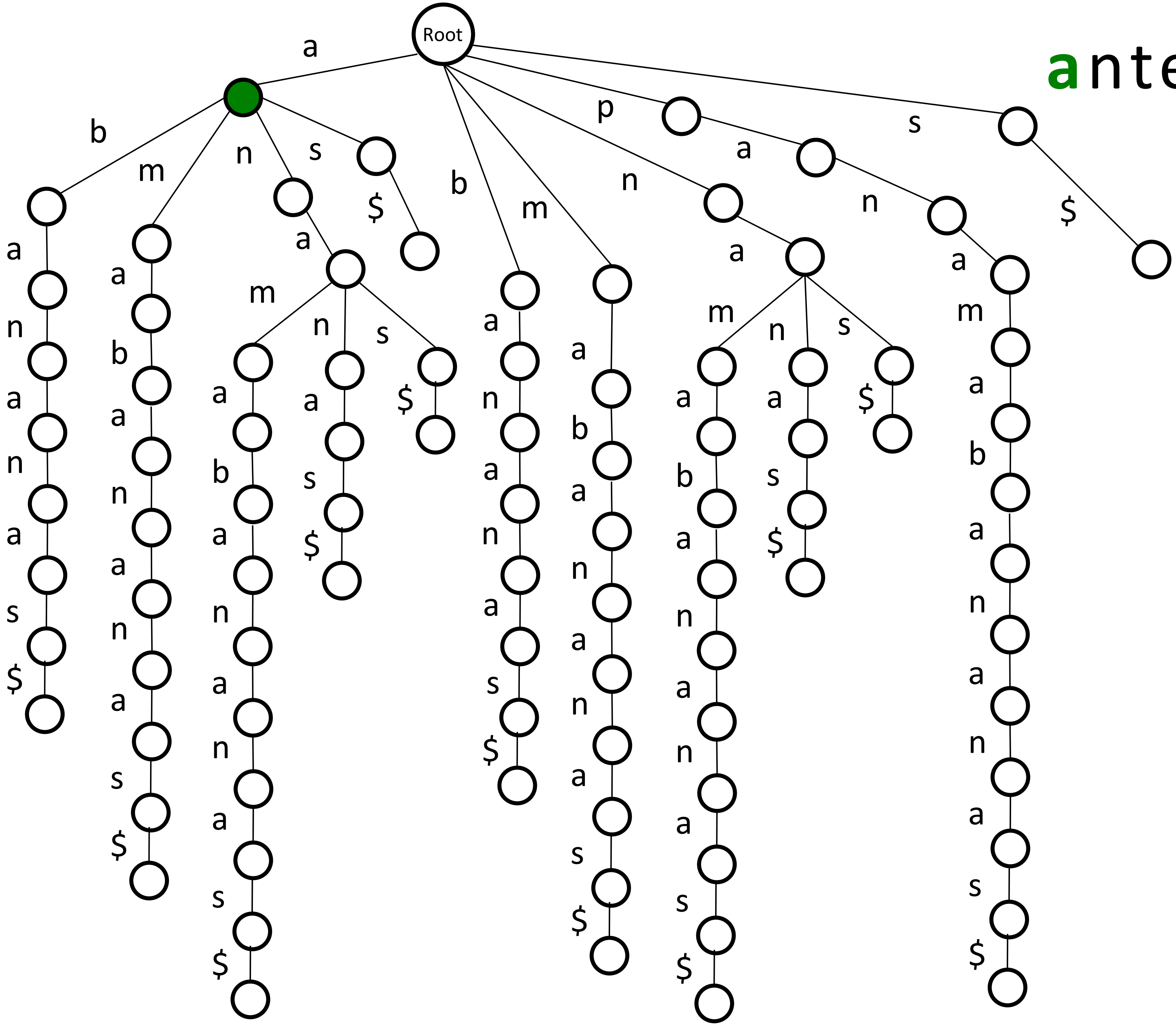
nab



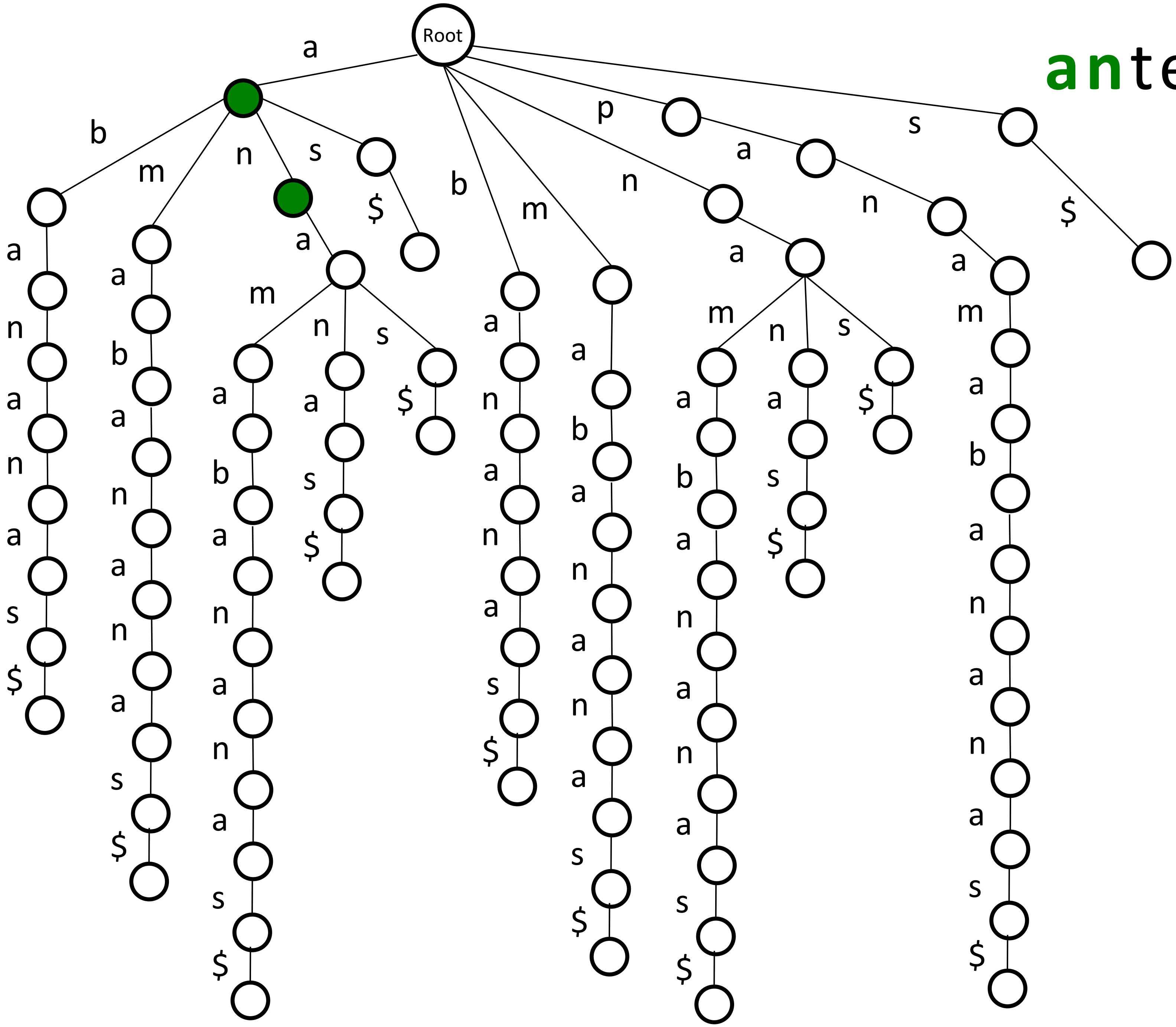
nab



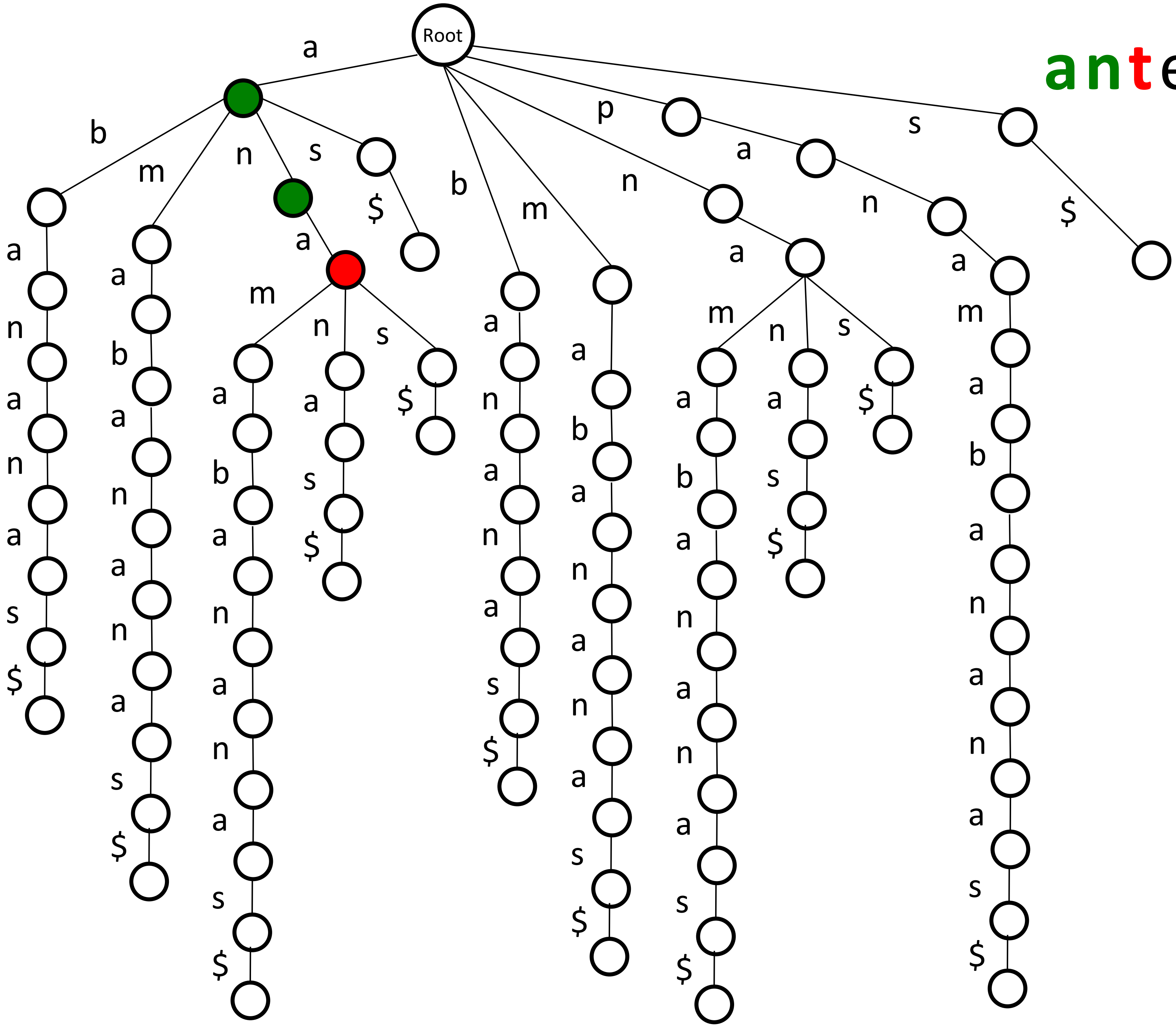
nab



antenna

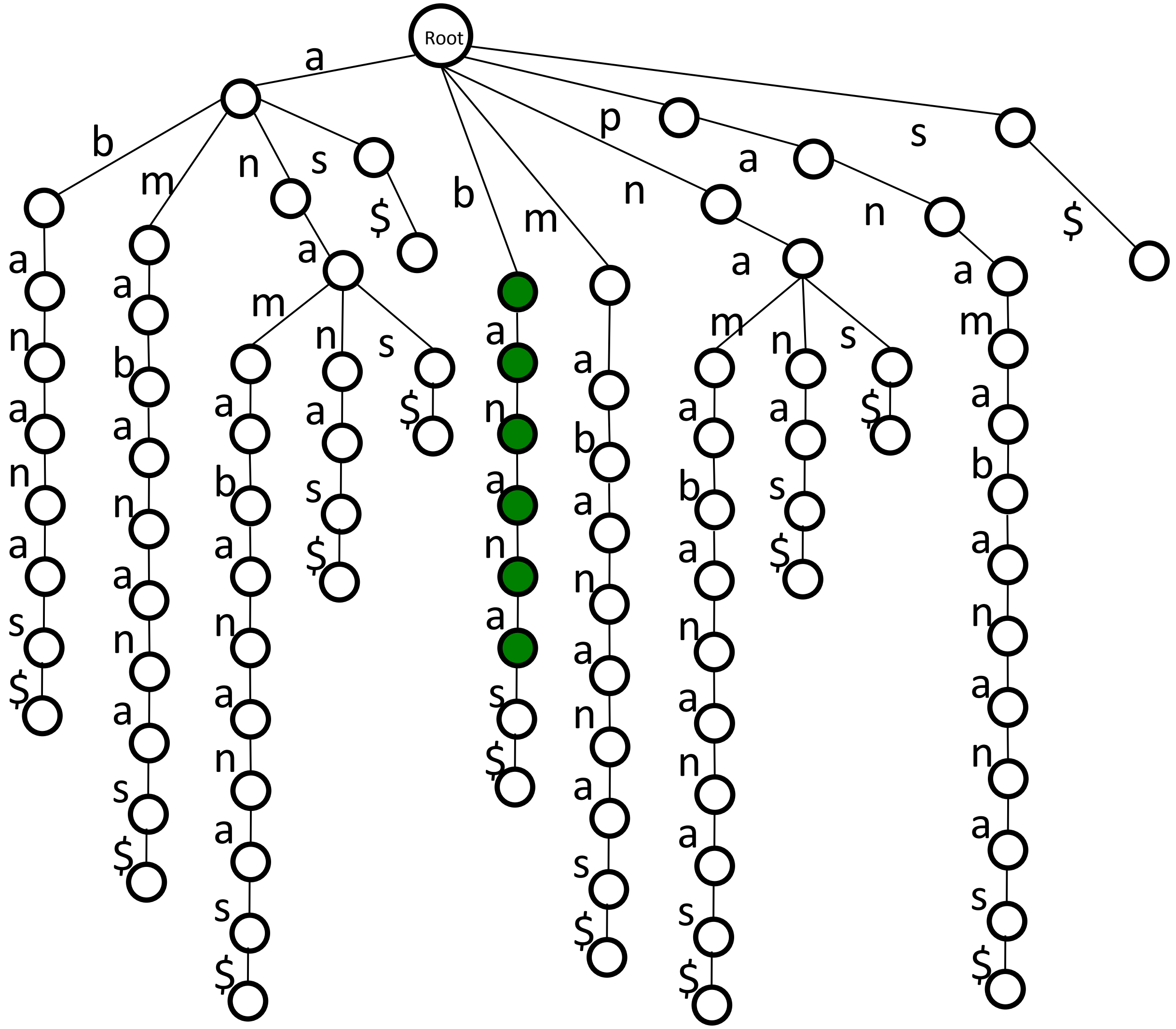


antenna



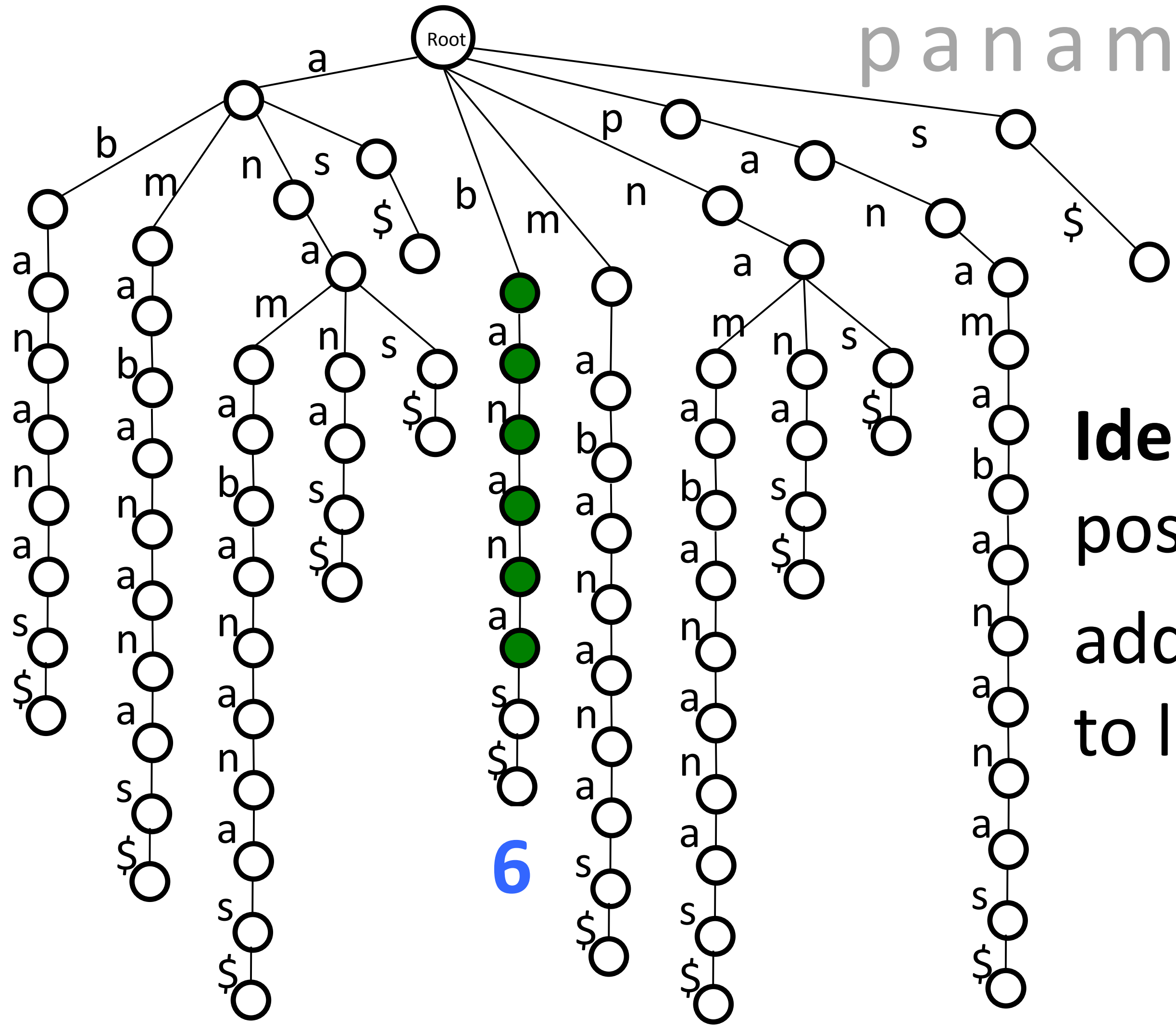
antenna

Where Are the Matches???



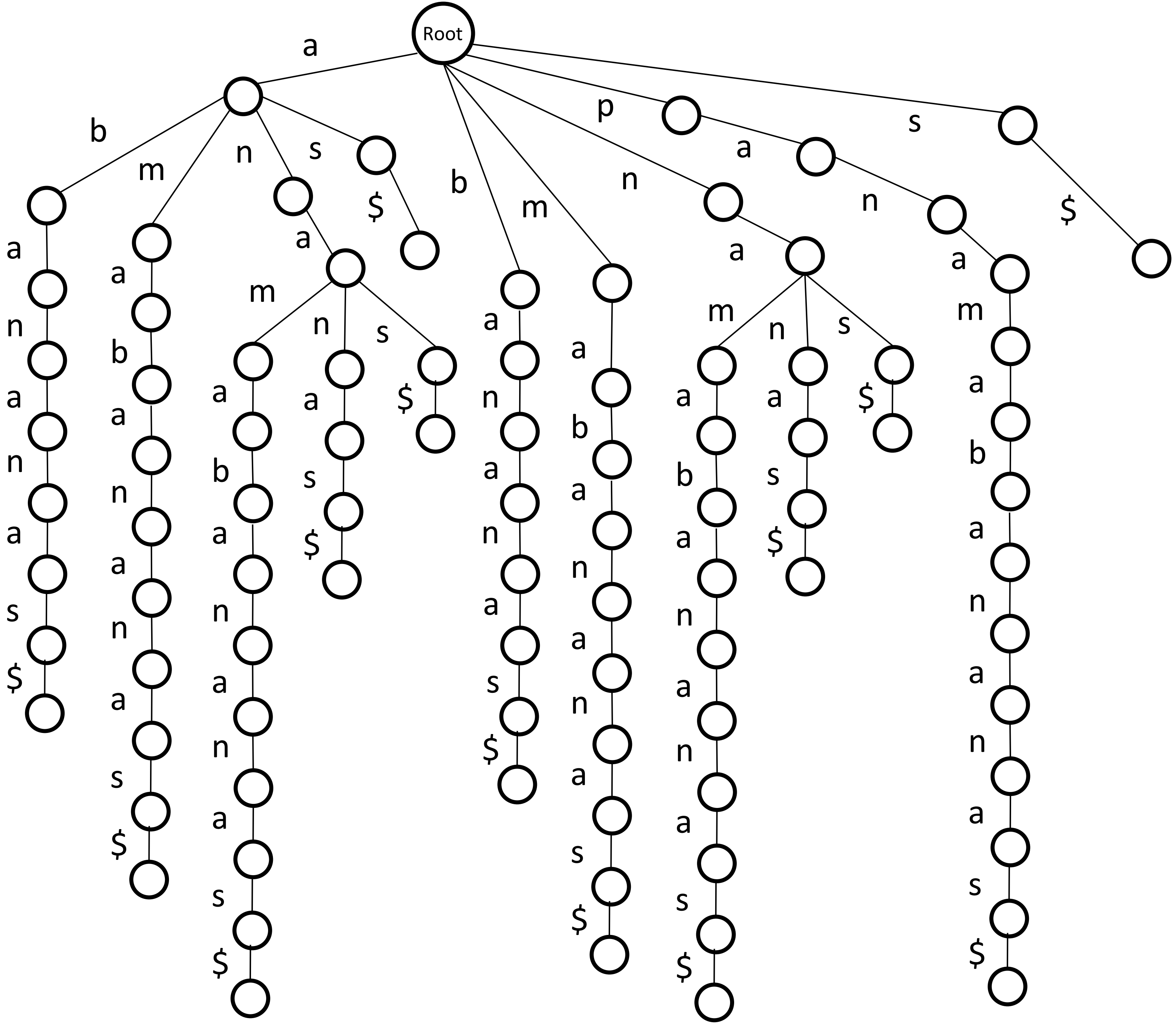
bananas$

Where Are the Matches???

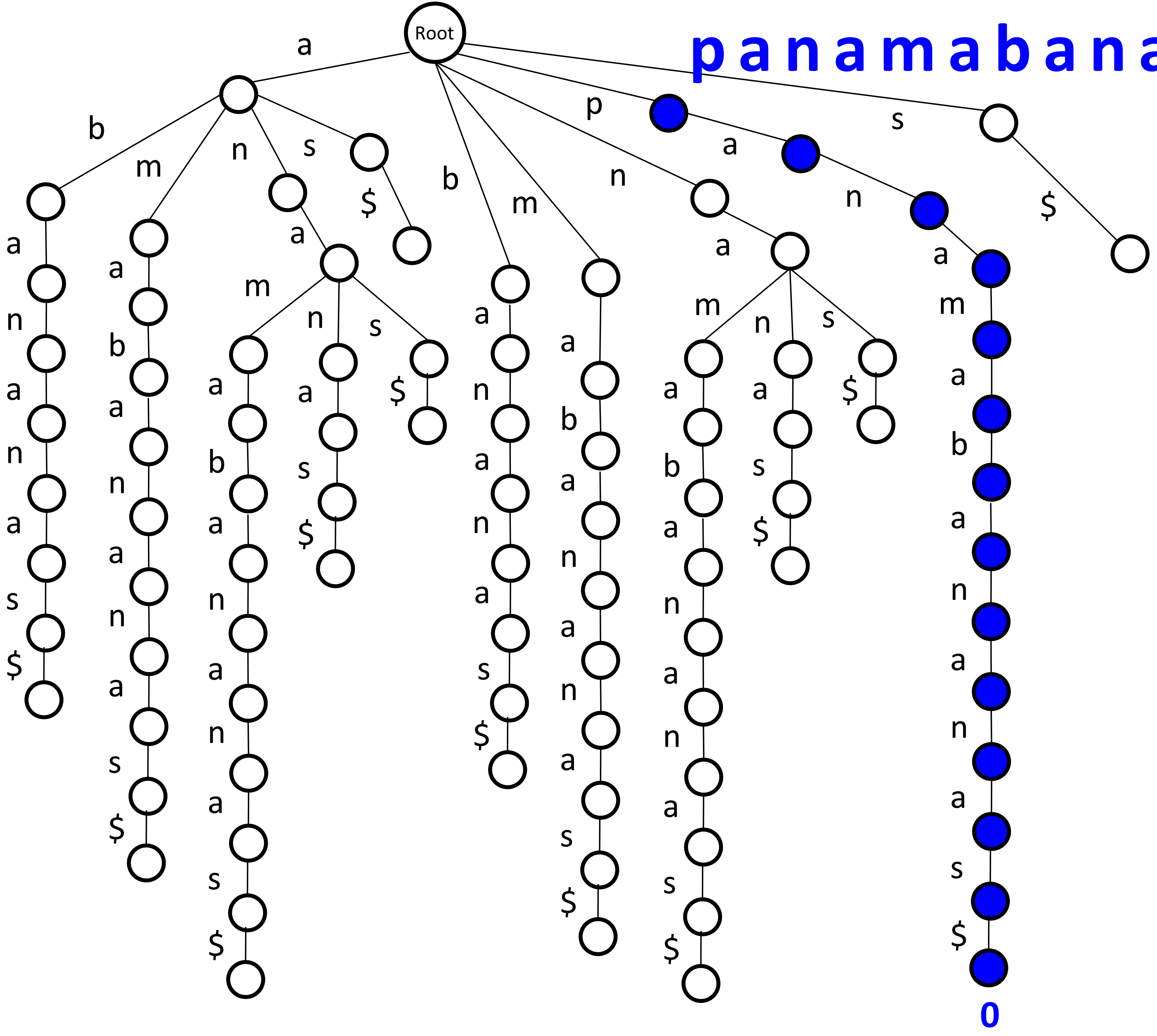


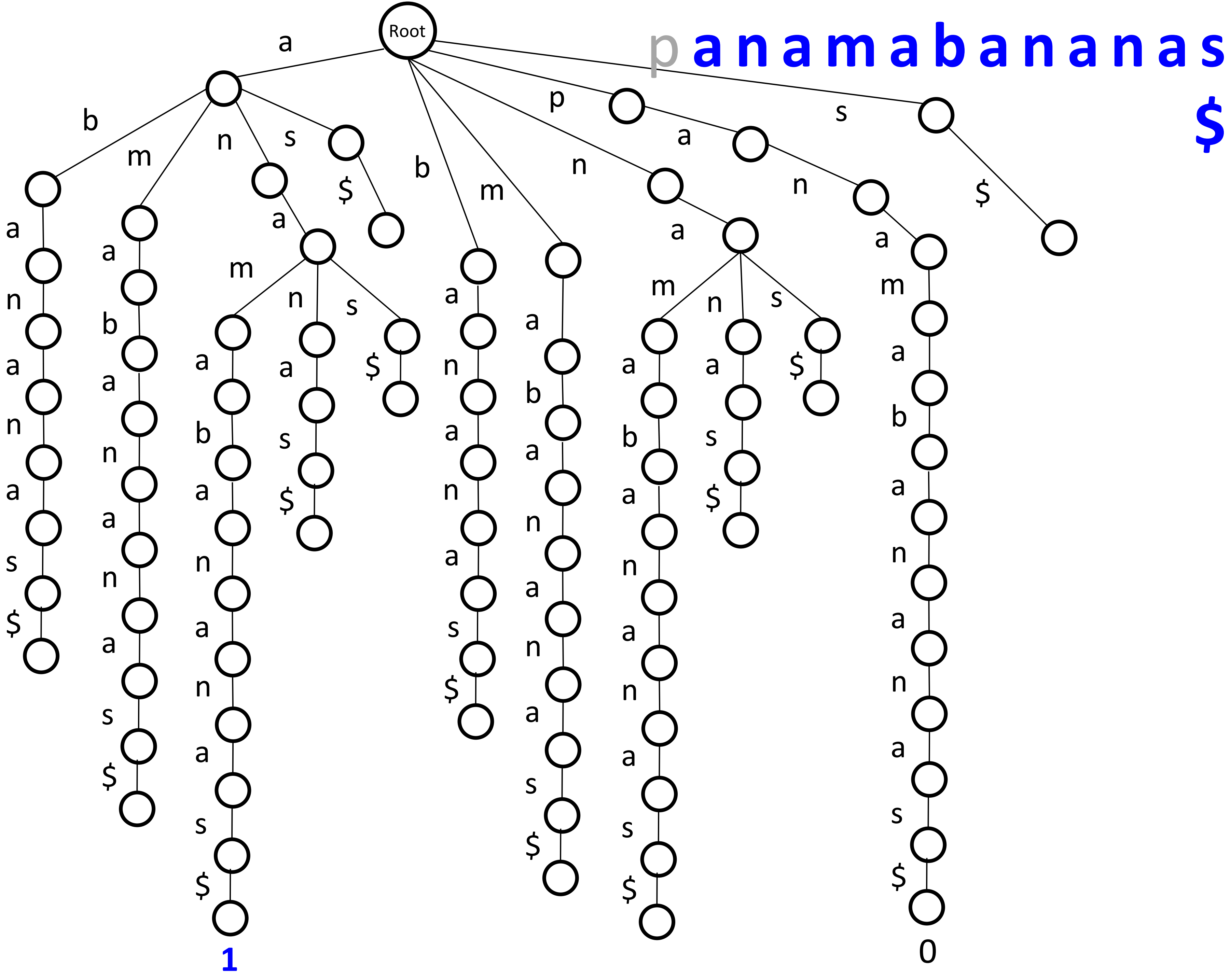
p a n a m a **b a n a n a s $**

Idea: to find the positions of matches, add some information to leaves

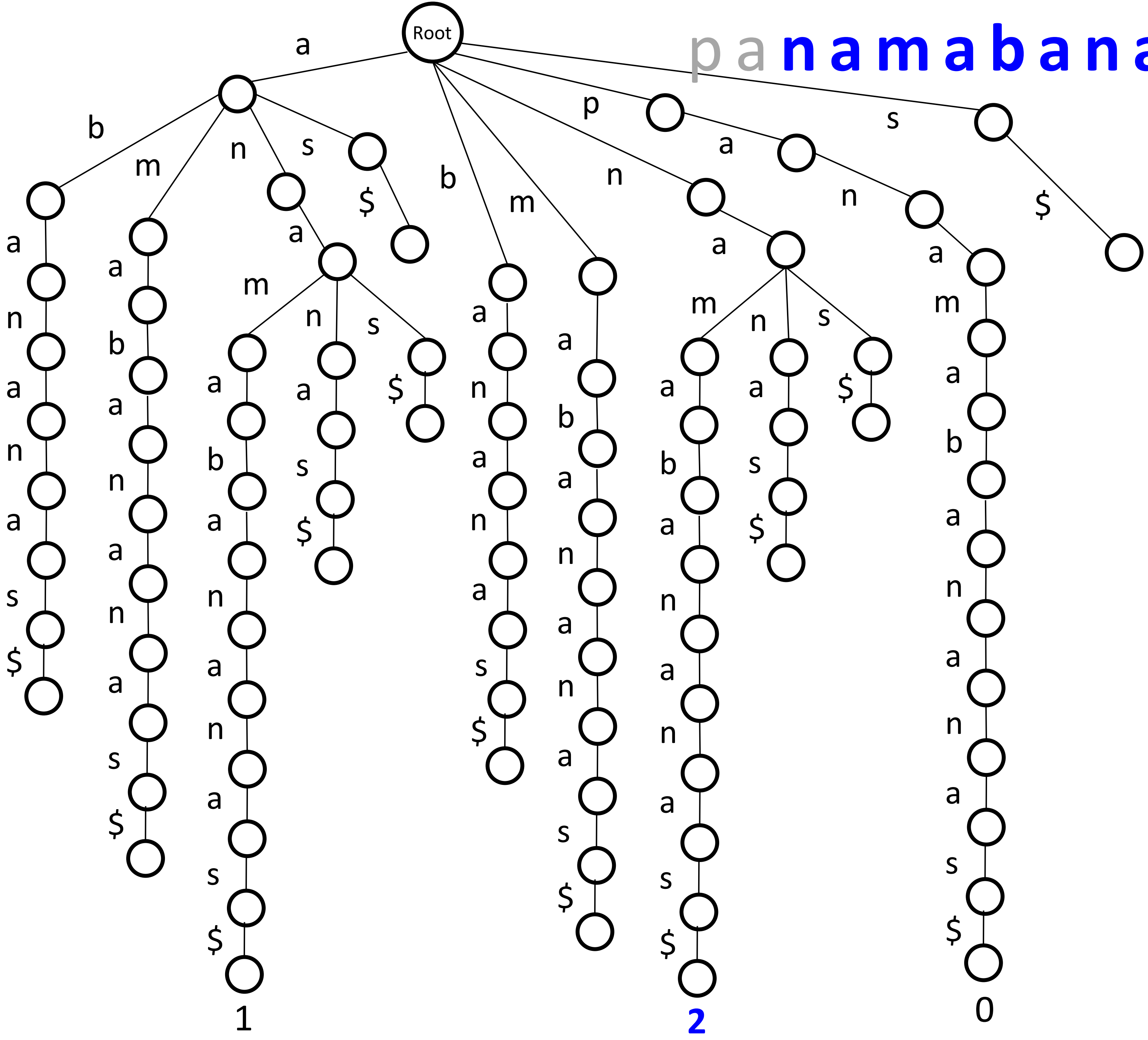

Root

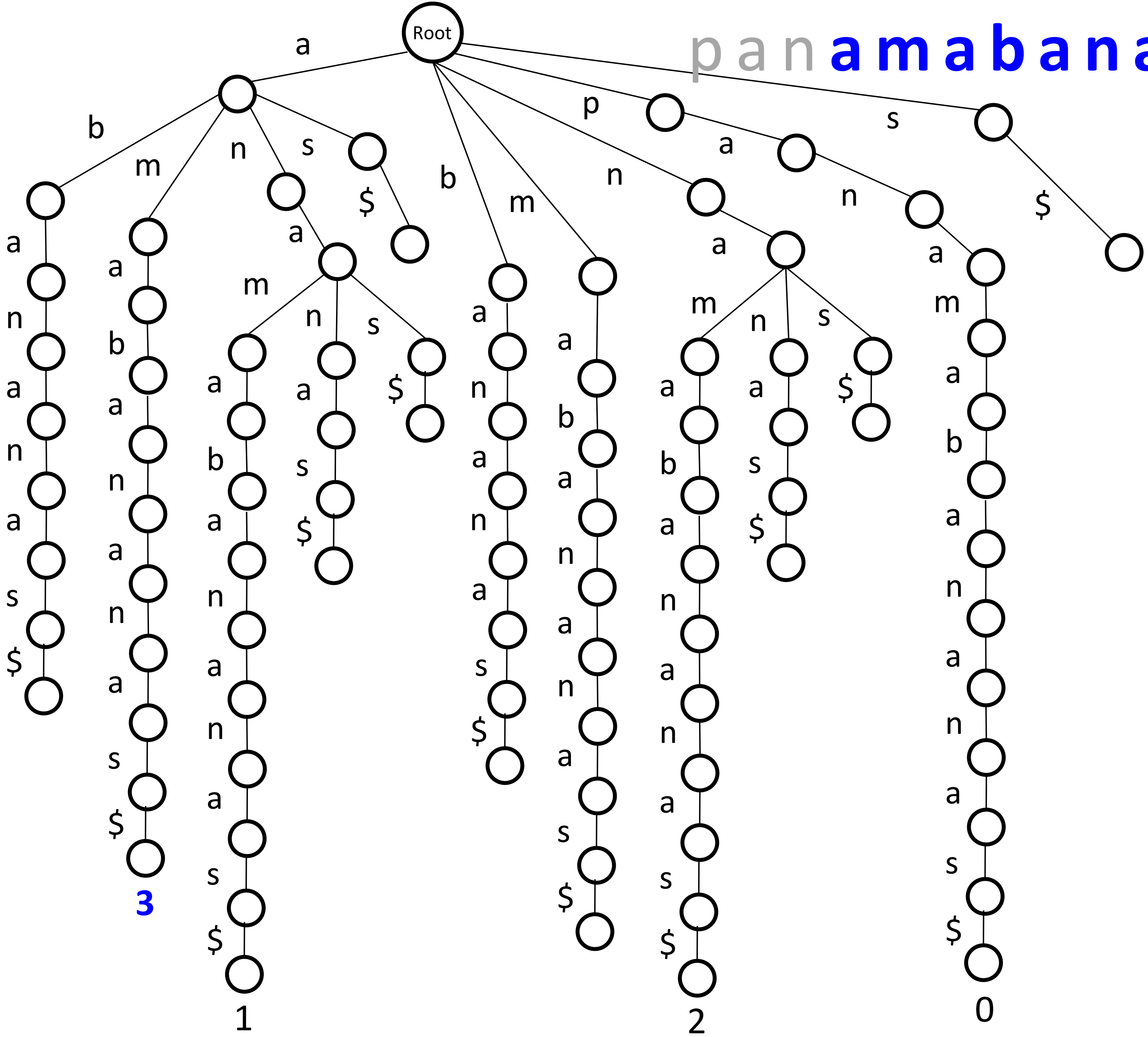
panamabananas$




Root
p a n a m a b a n a n a s
$
a
b
m
n
s
p
1
0

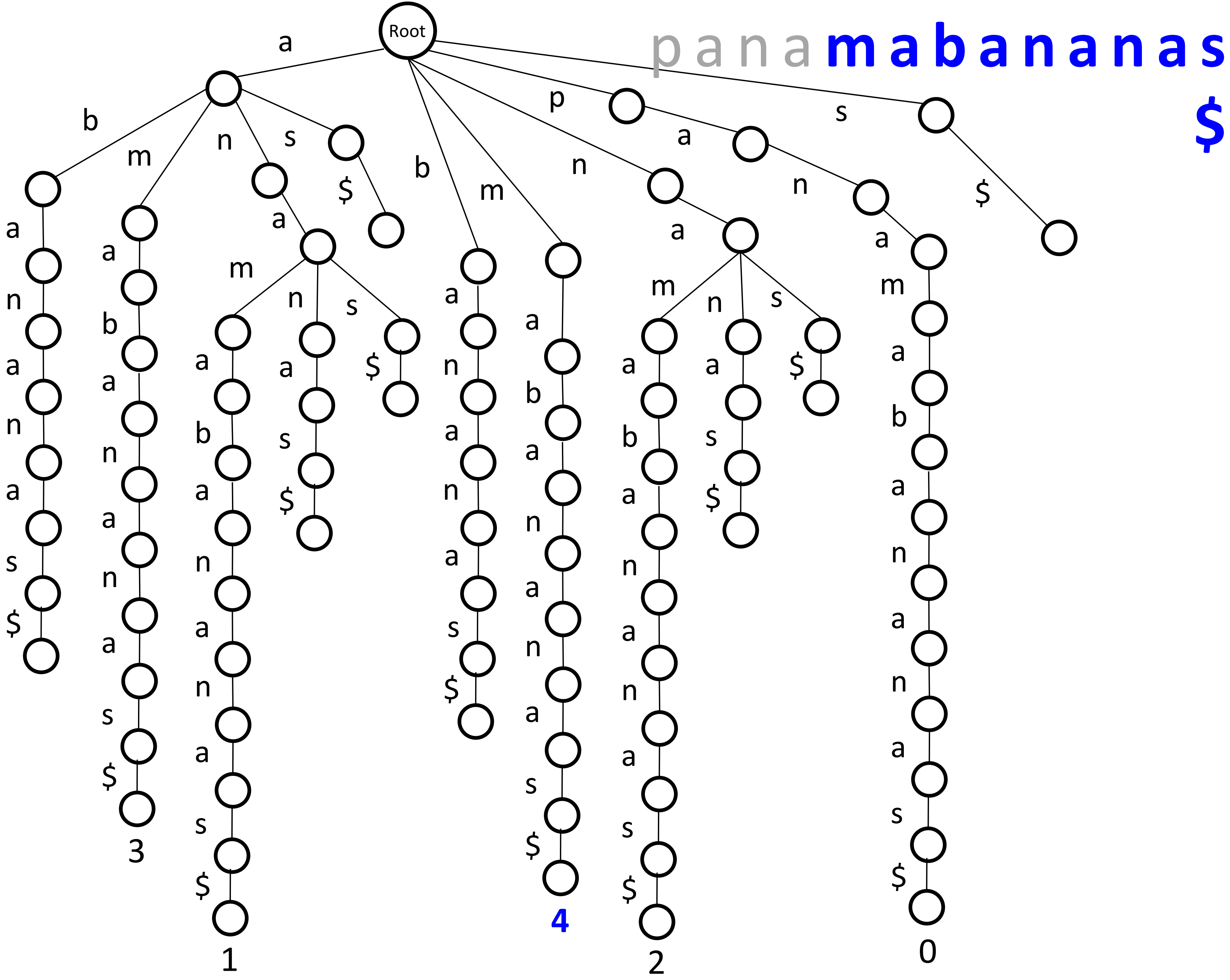
panamabananas
$

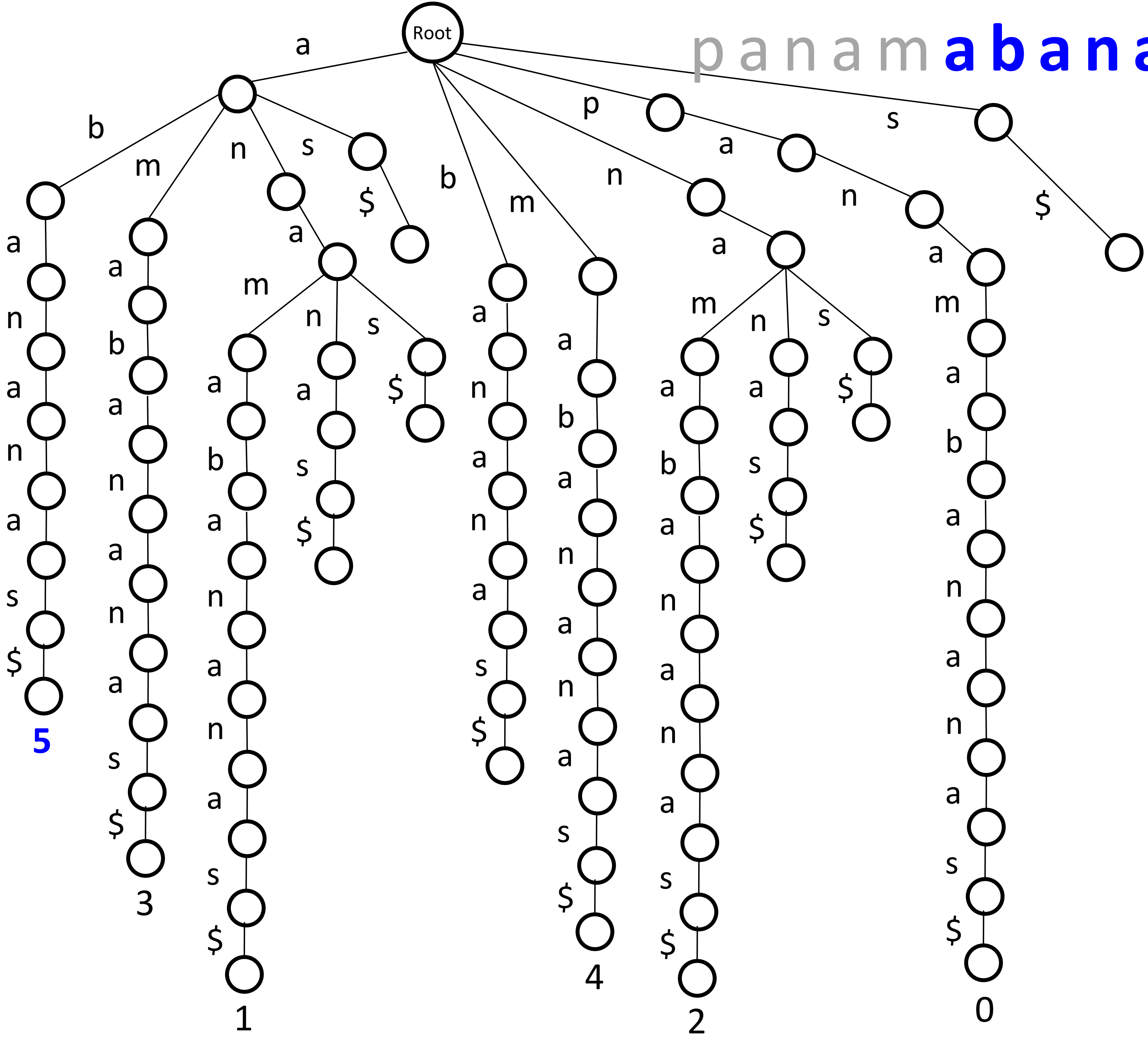




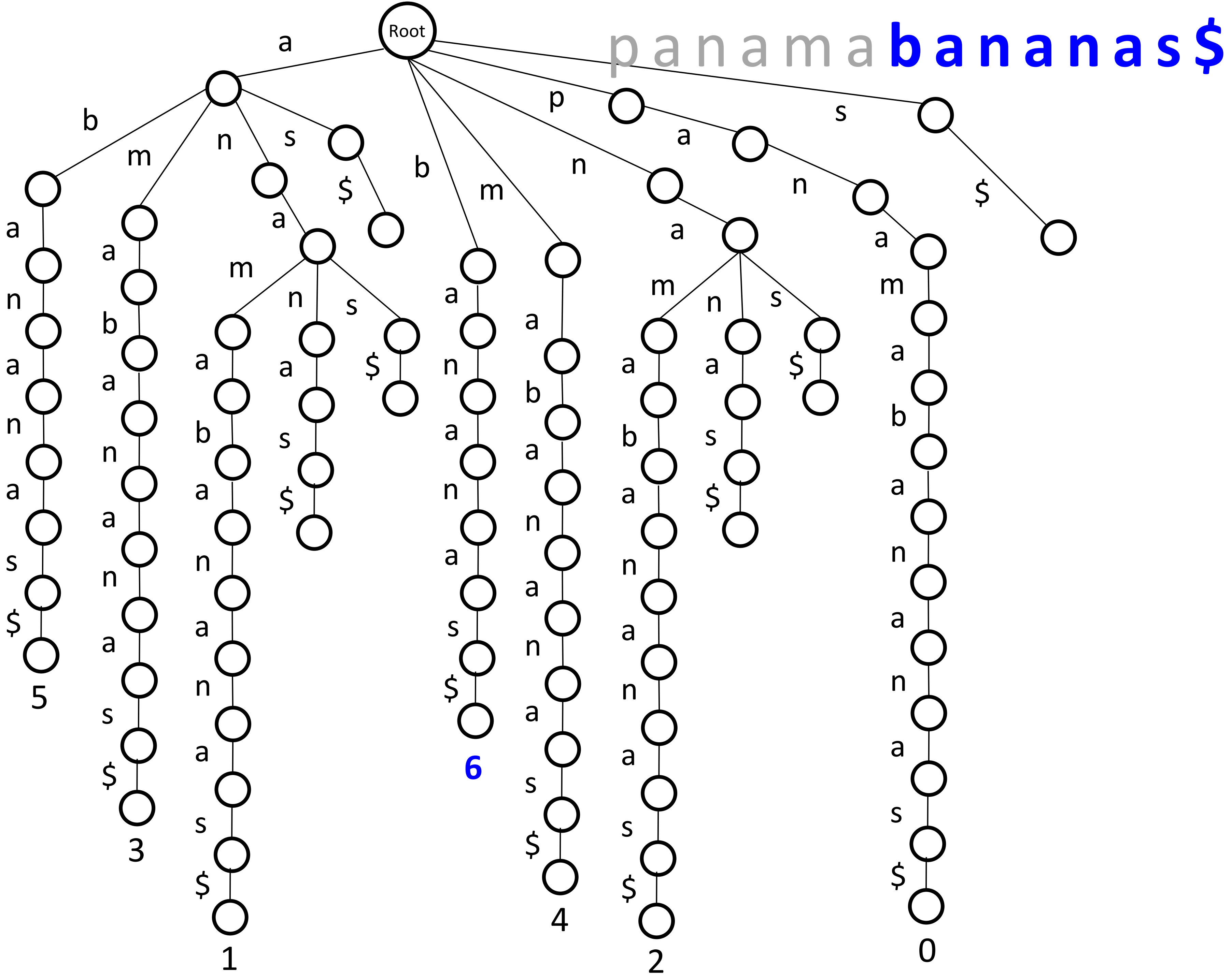
p a n **a m a b a n a n a s**

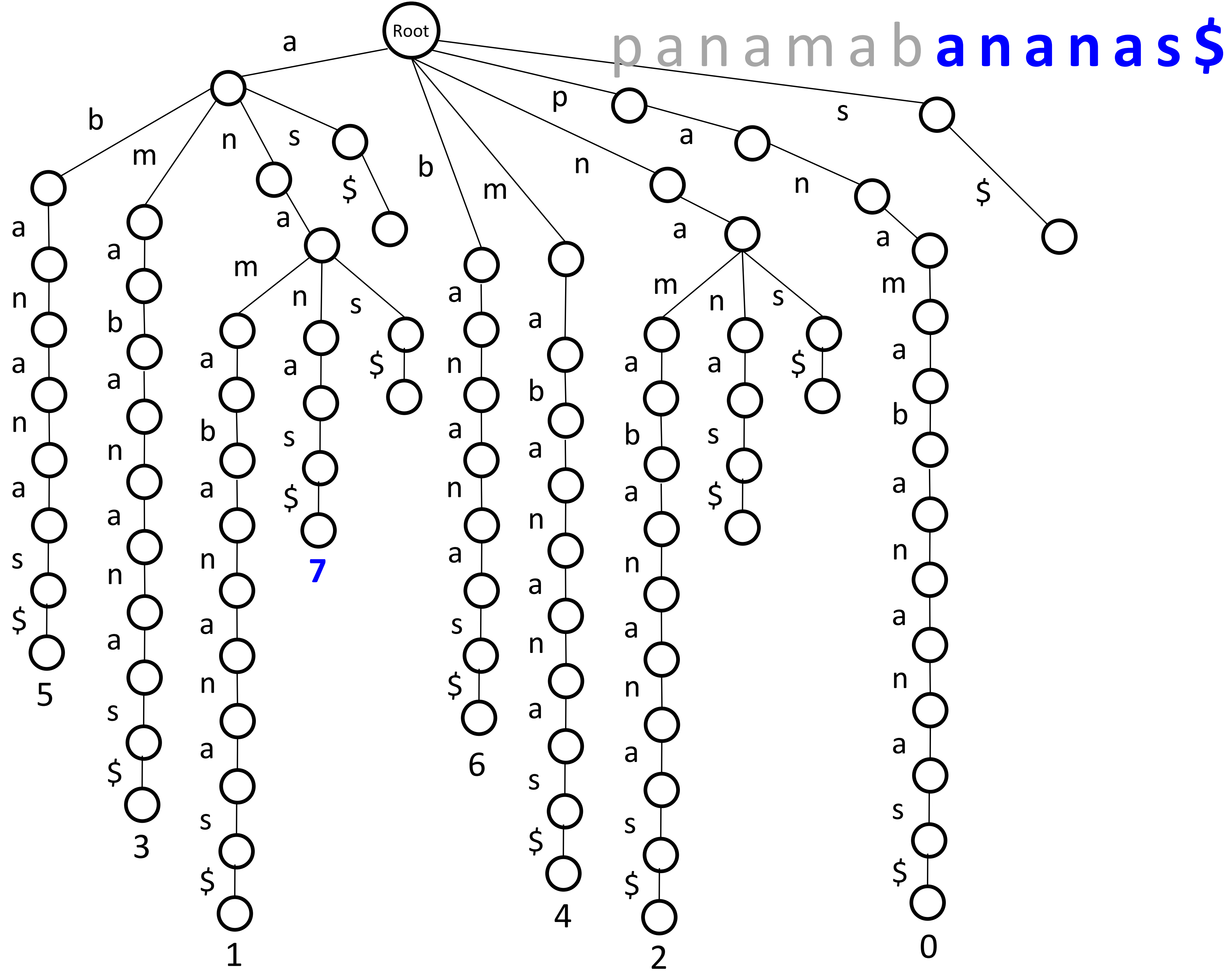
$


panamabananas
$
Root
3
1
4
2
0

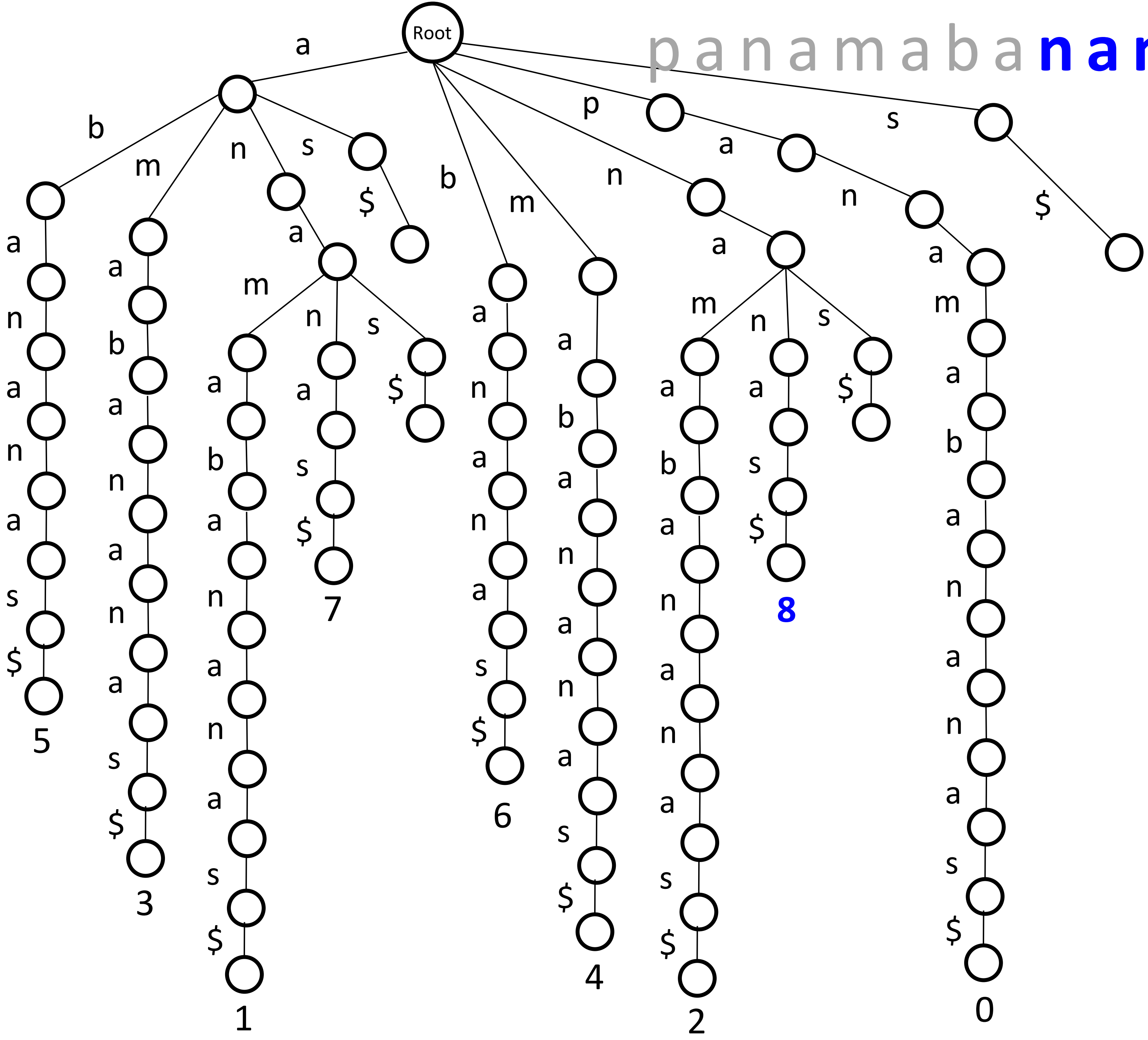


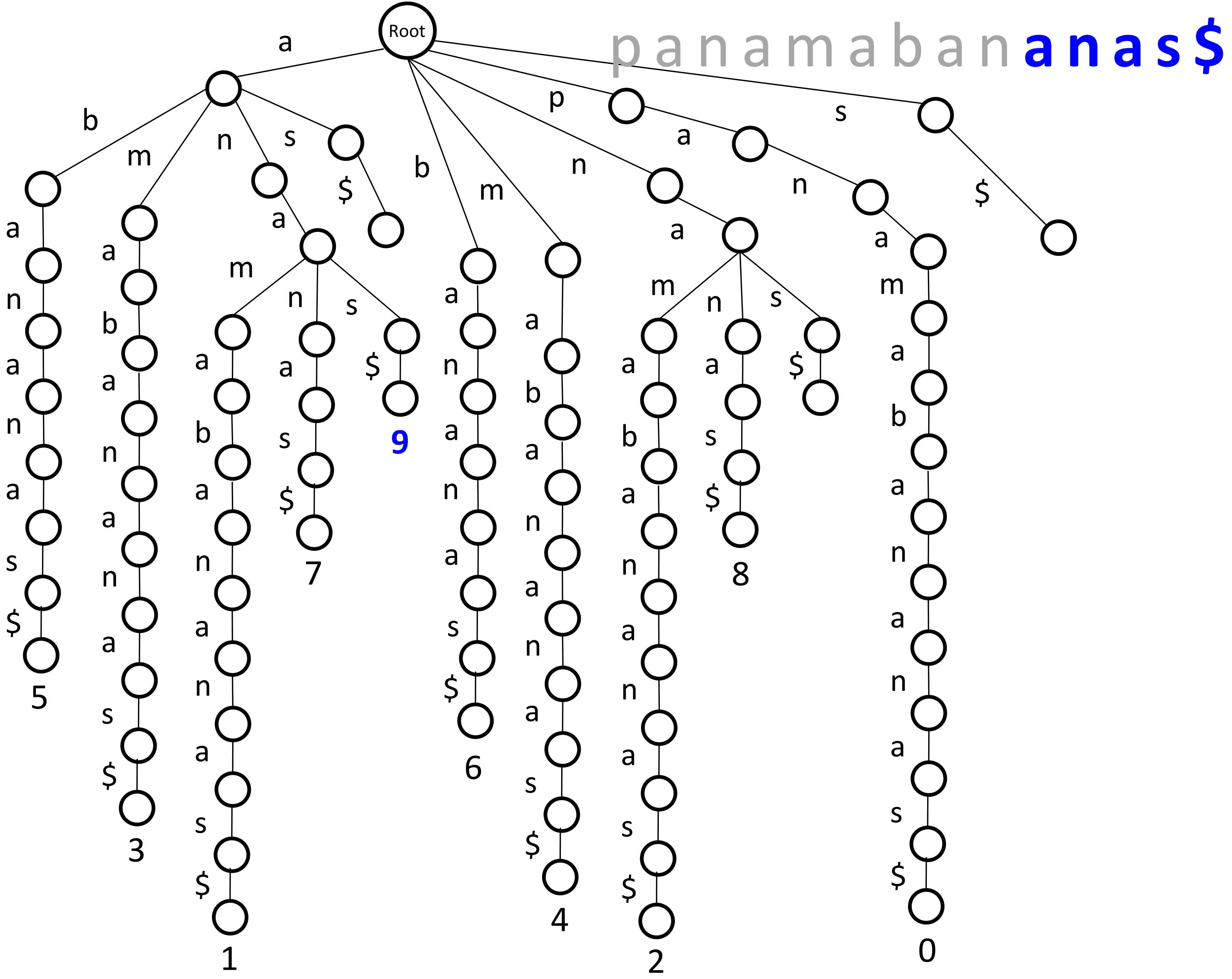
panamabananas$
Root
5
3
1
4
2
0

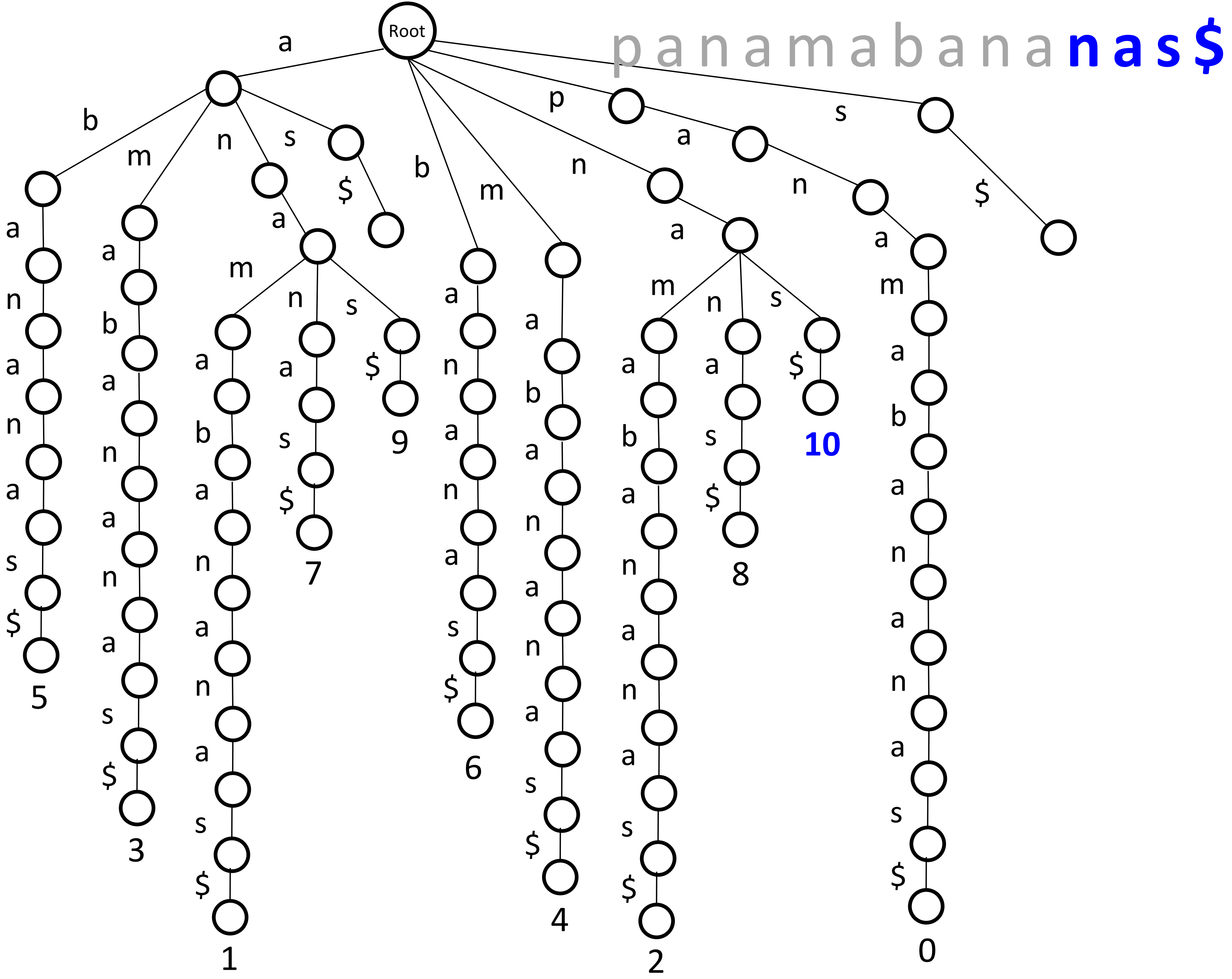

panamabananas$
Root
a
b
m
n
s
p
$
5
3
1
6
4
2
0

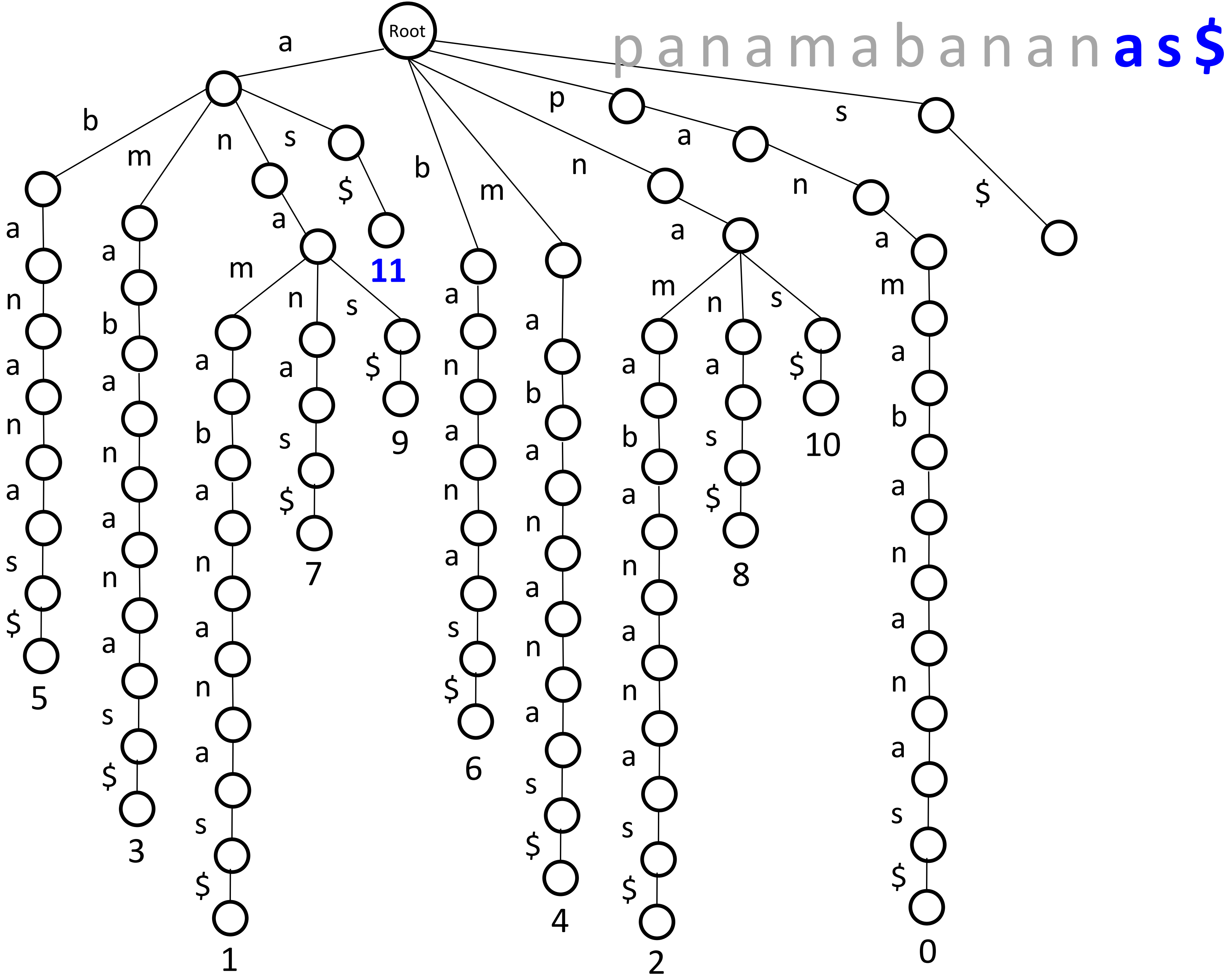

panamabananas$
Root
5
3
1
7
6
4
2
0

panamaba**nanas$**

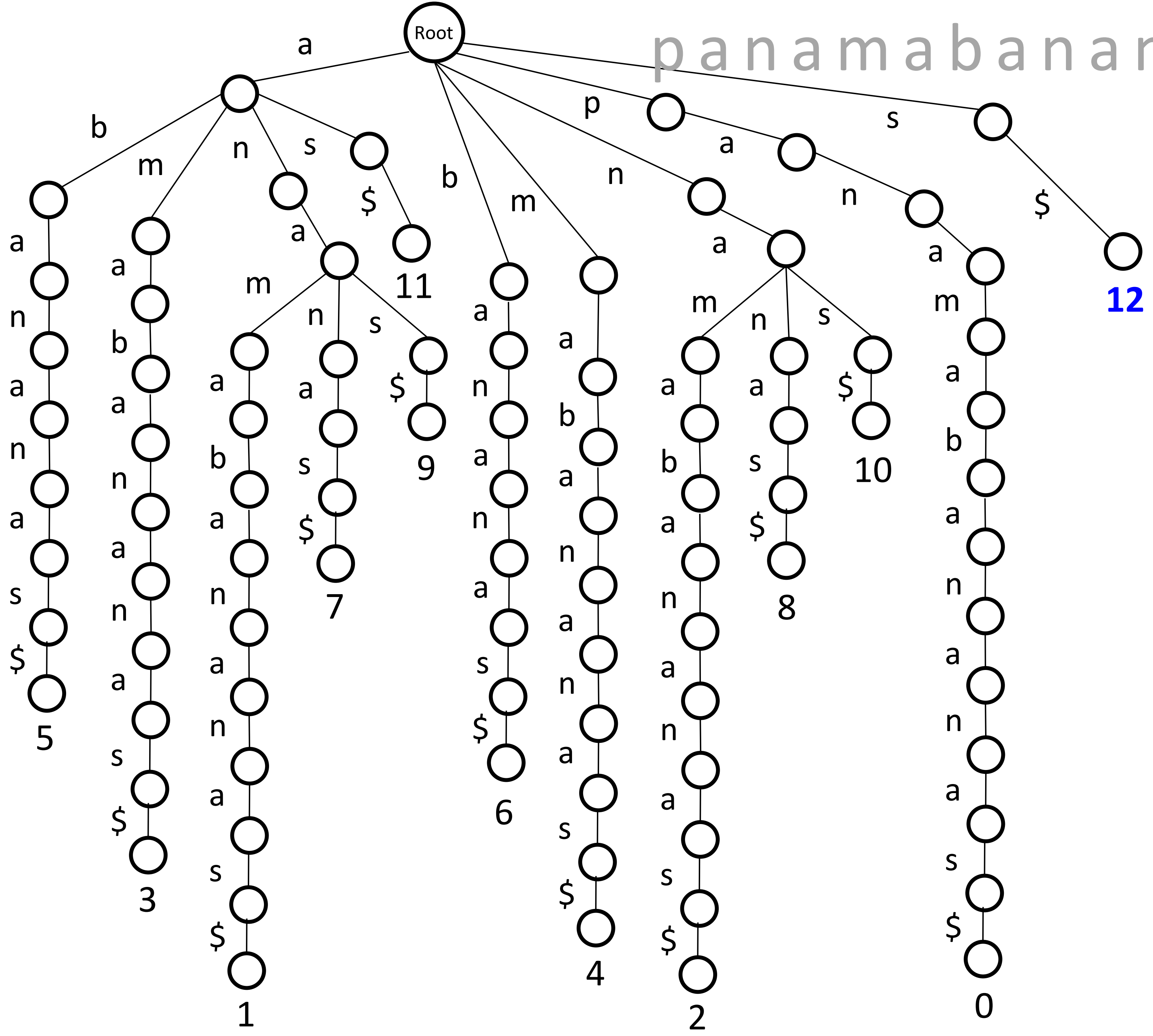


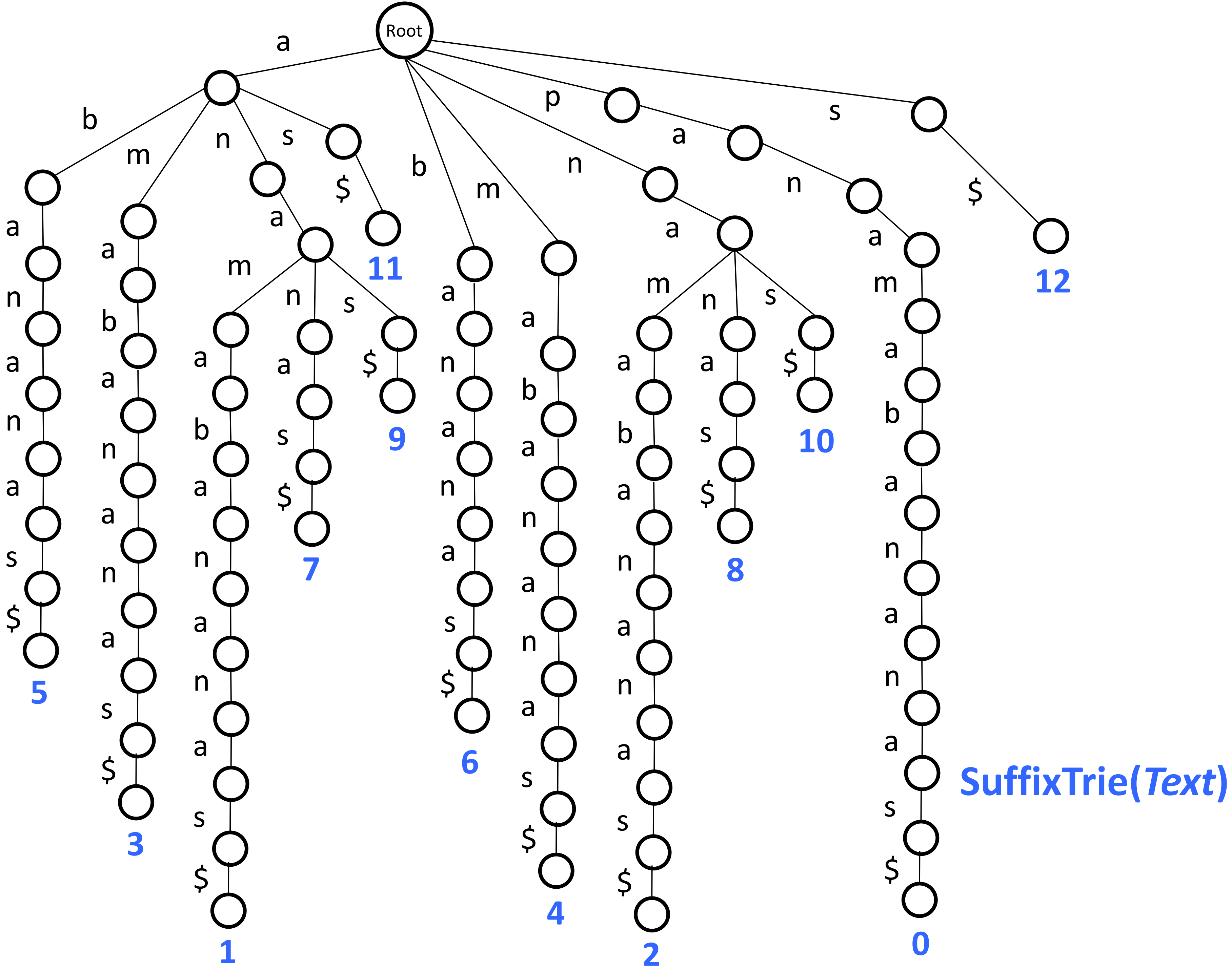

panamaban**anas$**
Root
5
3
1
7
9
6
4
2
8
0


panamabananas$
Root
5
3
1
7
9
6
4
2
8
10
0


Root
panamabananas$
a
b
a
n
a
n
a
s
$
5
m
a
b
a
n
a
n
a
s
$
3
n
a
m
a
b
a
n
a
n
a
s
$
1
n
a
s
$
7
s
$
9
s
$
11
b
a
n
a
n
a
s
$
6
m
a
b
a
n
a
n
a
s
$
4
n
a
m
a
b
a
n
a
n
a
s
$
2
n
a
s
$
8
s
$
10
p
a
n
a
m
a
b
a
n
a
n
a
s
$
0
s
$

p a n a m a b a n a n a **s $**

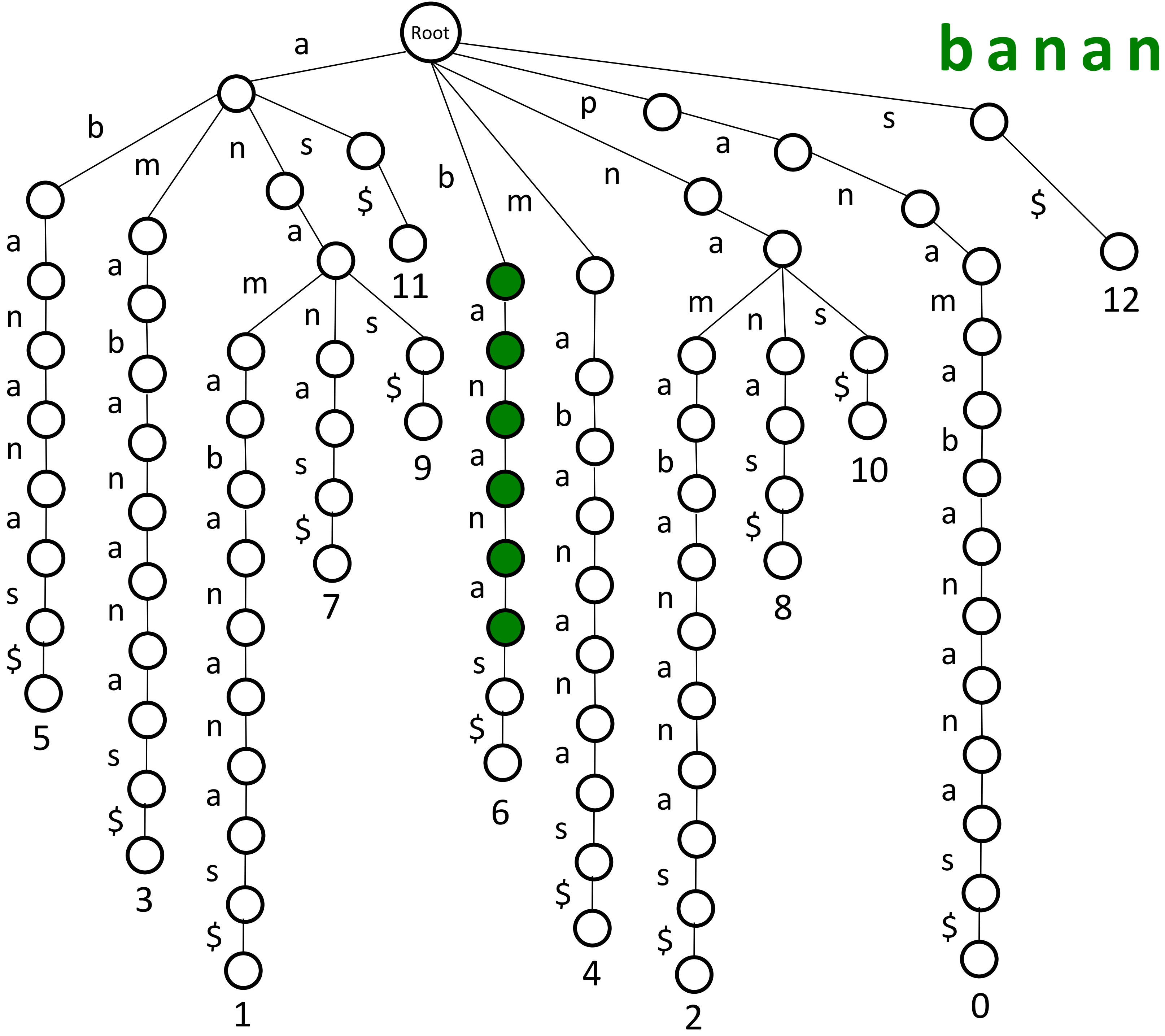




SuffixTrie(*Text*)

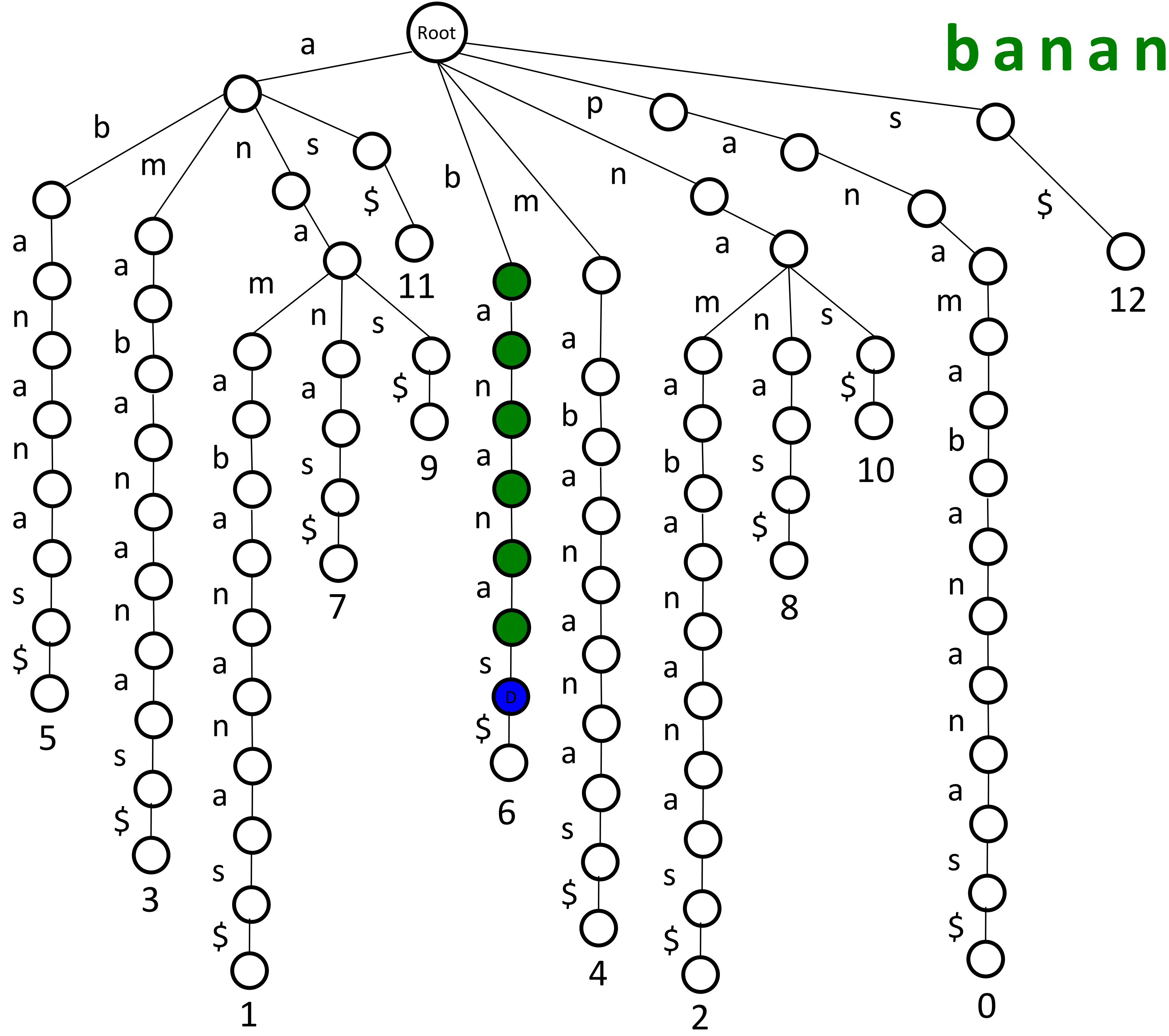
Walking Down to the Leaves to Find Matches

- Once we find a match, we “walk down” to the leaf (or leaves) in order to find the starting position of the match.

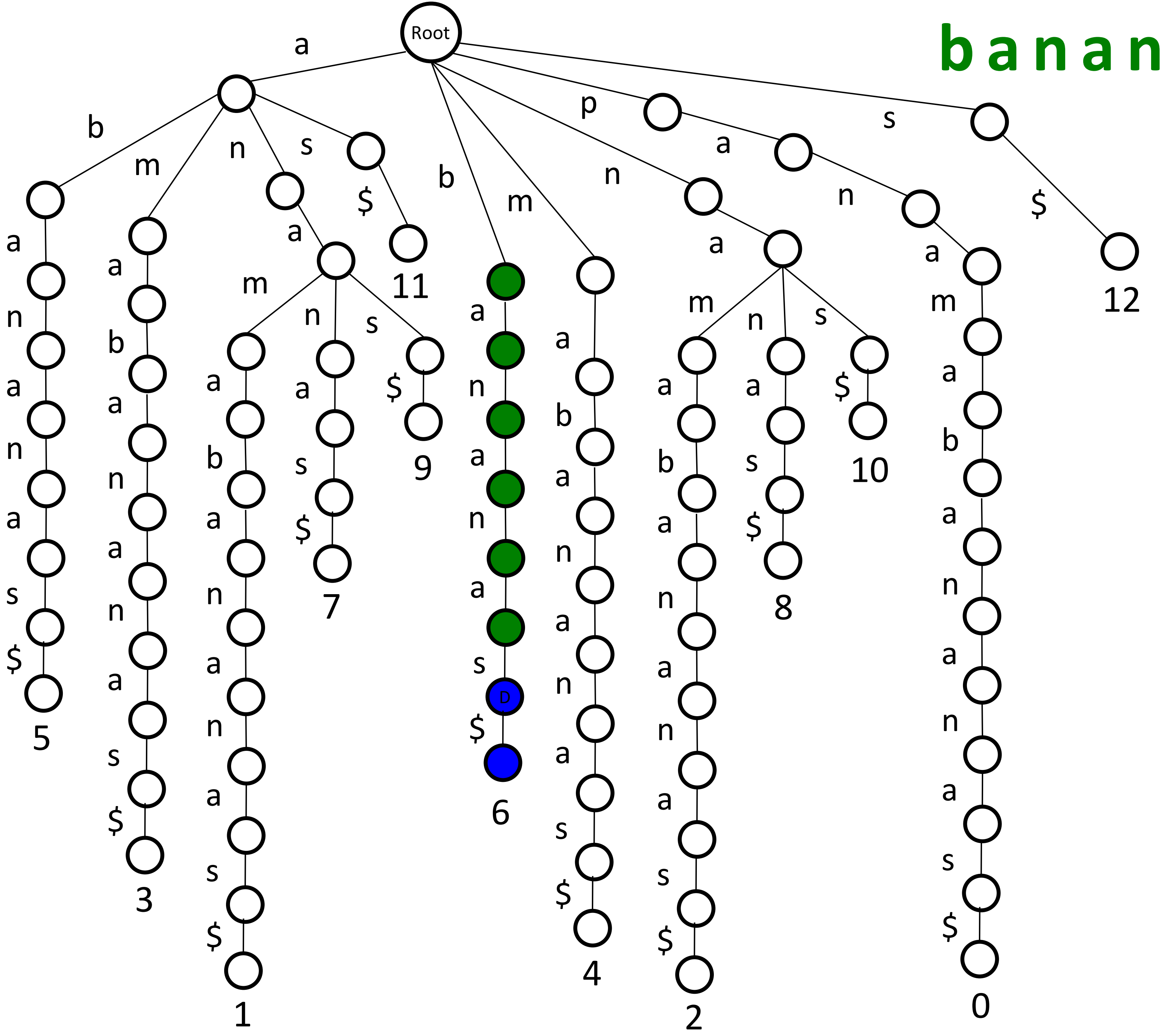


Root
banana
a
b
m
n
s
p
$
5
3
1
7
9
11
6
4
2
8
10
0
12

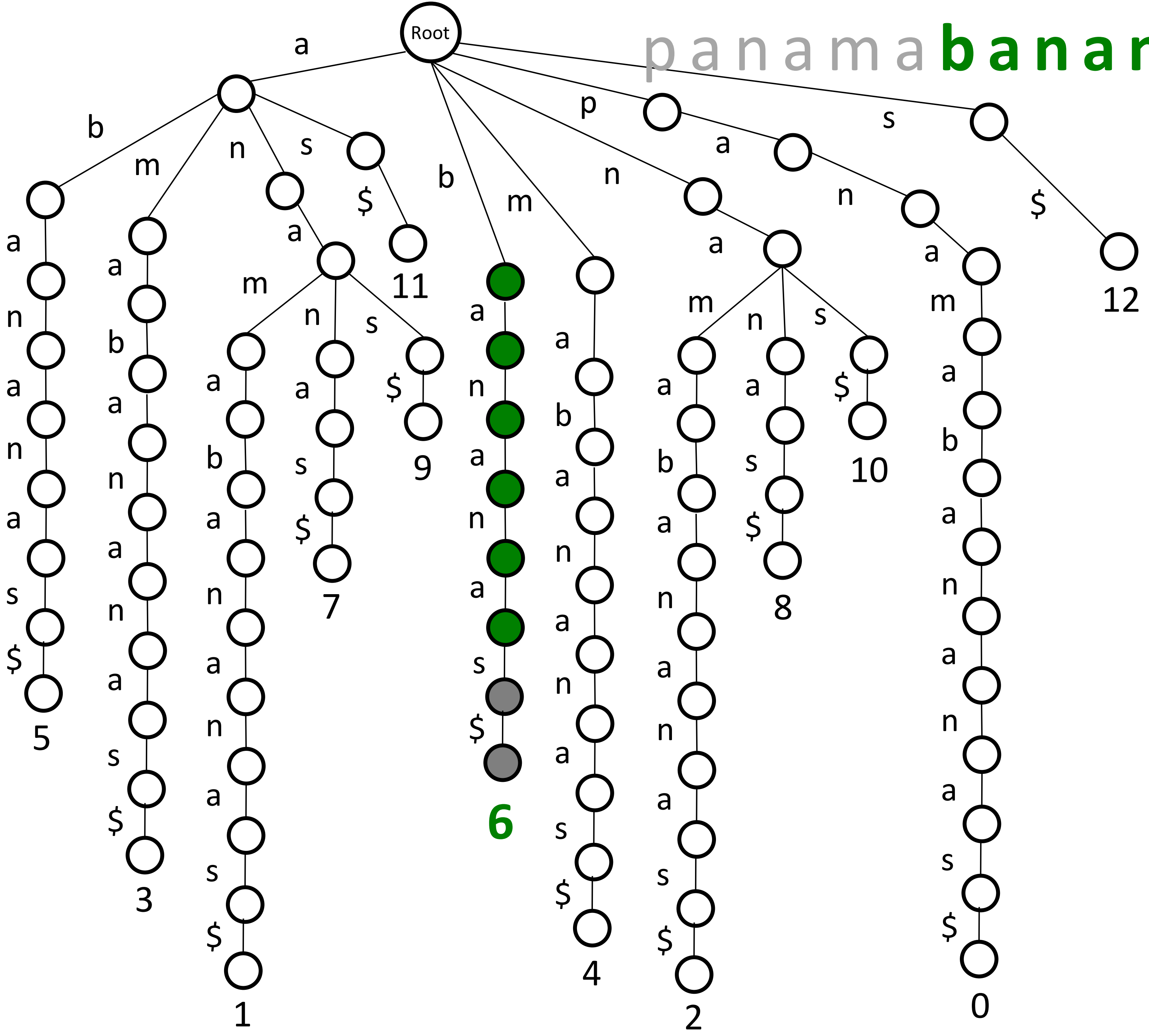
b a n a n a

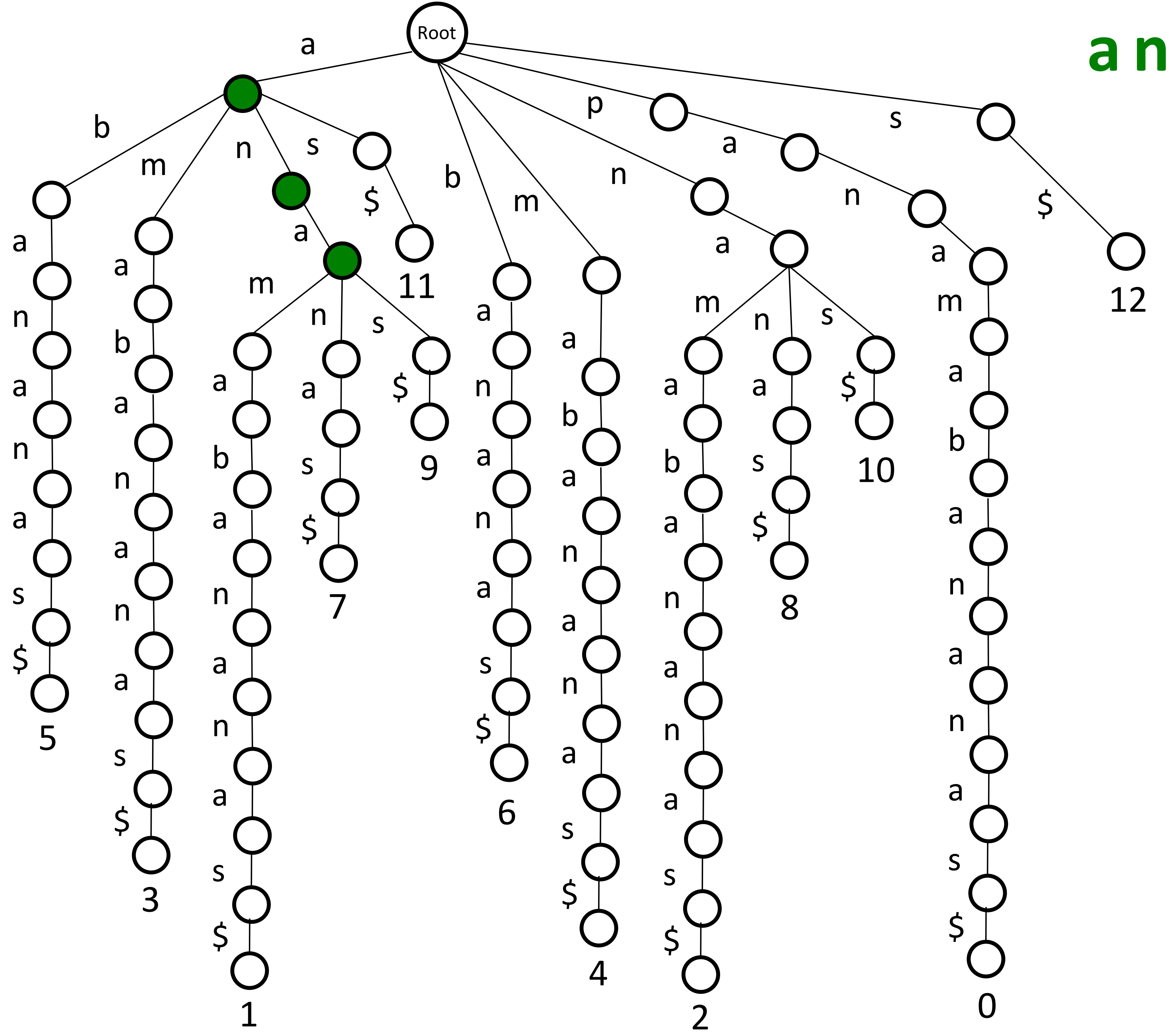


banana



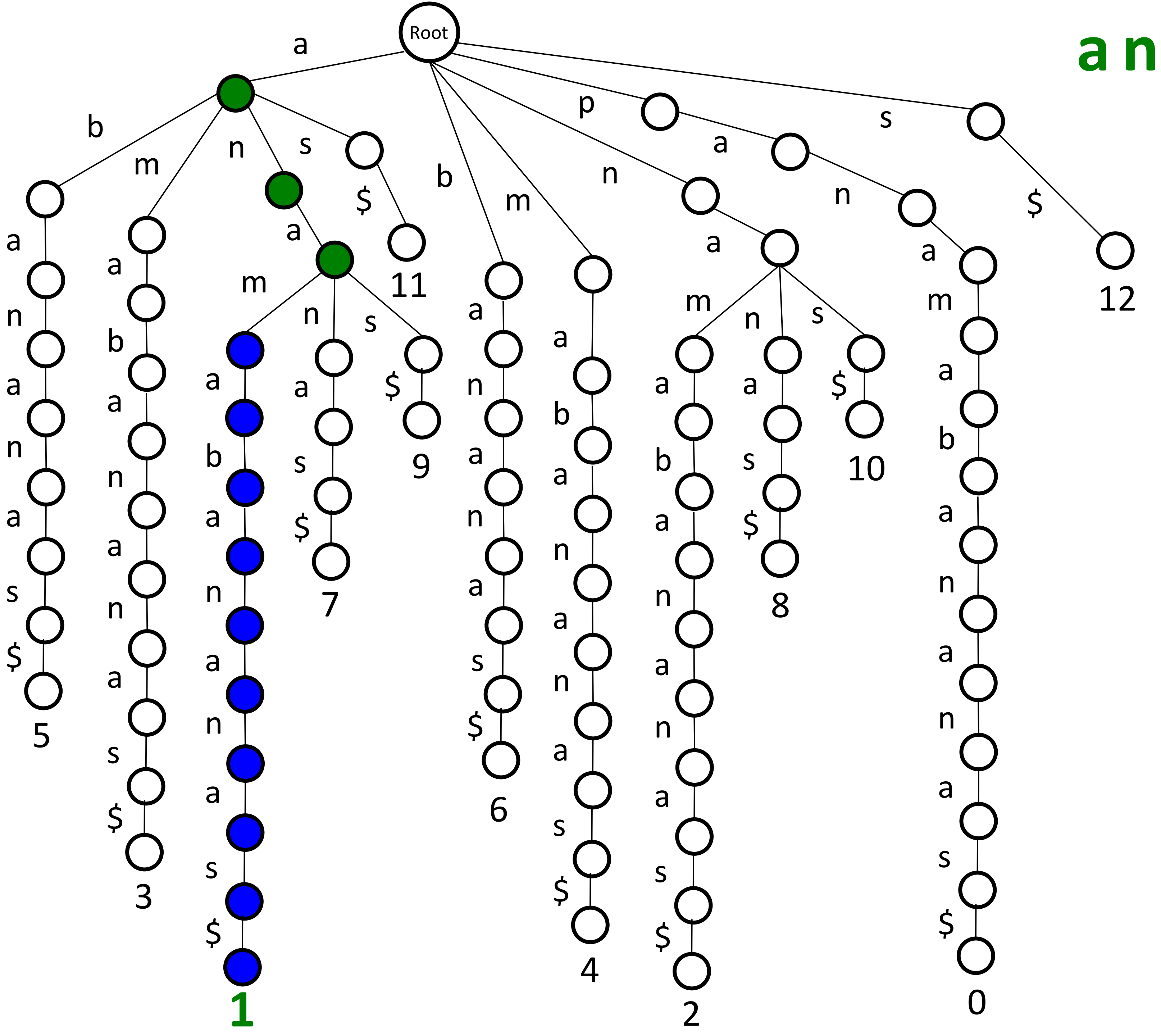
p a n a m a **b a n a n a** s $





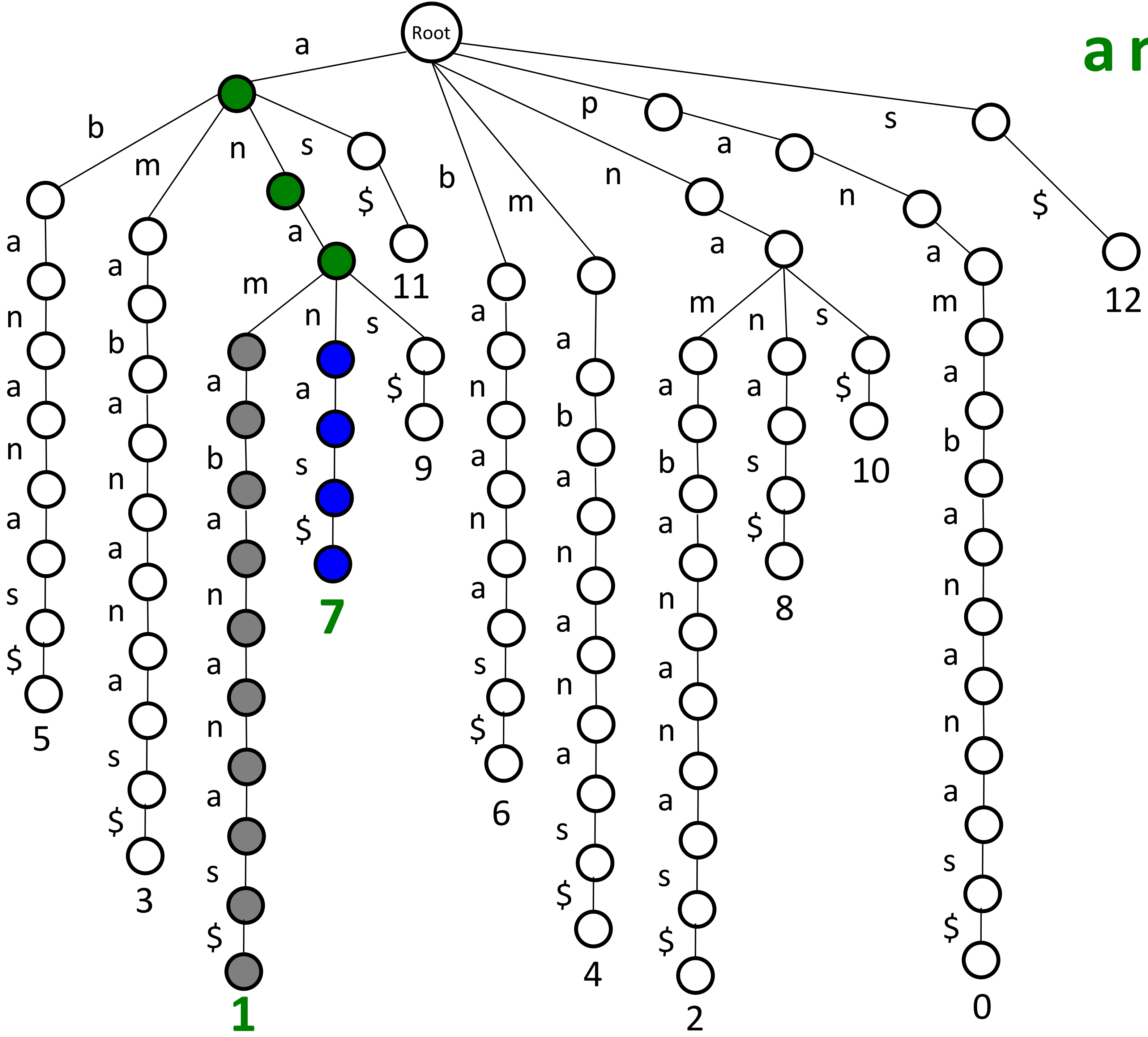
a n a

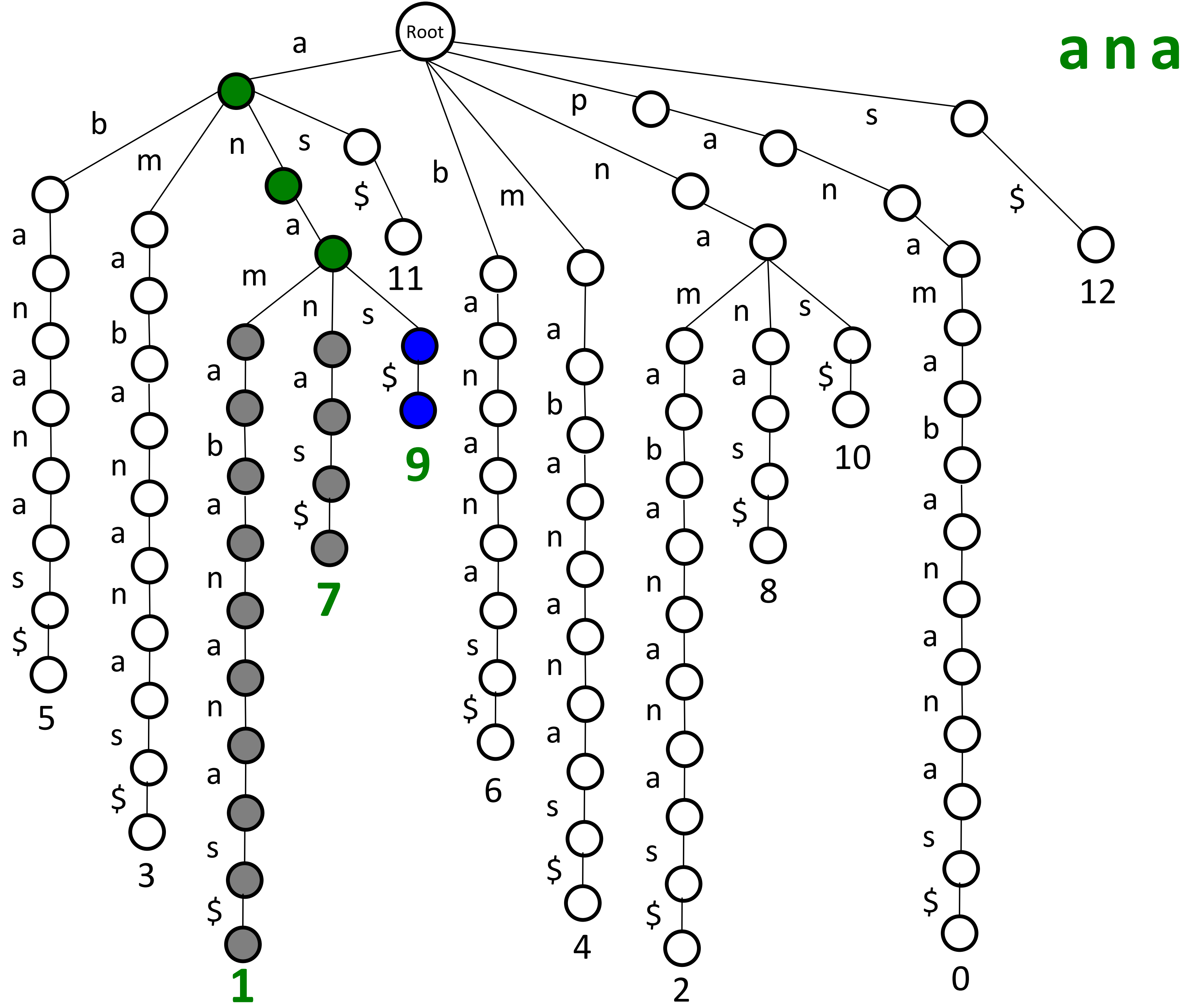
a n a



Root
a
b
m
n
s
a
$
11
m
n
s
a
a
$
9
b
s
a
$
n
7
a
n
a
s
$
1
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
b
a
n
a
n
a
s
$
6
m
a
b
a
n
a
n
a
s
$
4
n
a
m
n
s
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
10
p
a
n
a
m
a
b
a
n
a
n
a
s
$
0
s
$
12

ana

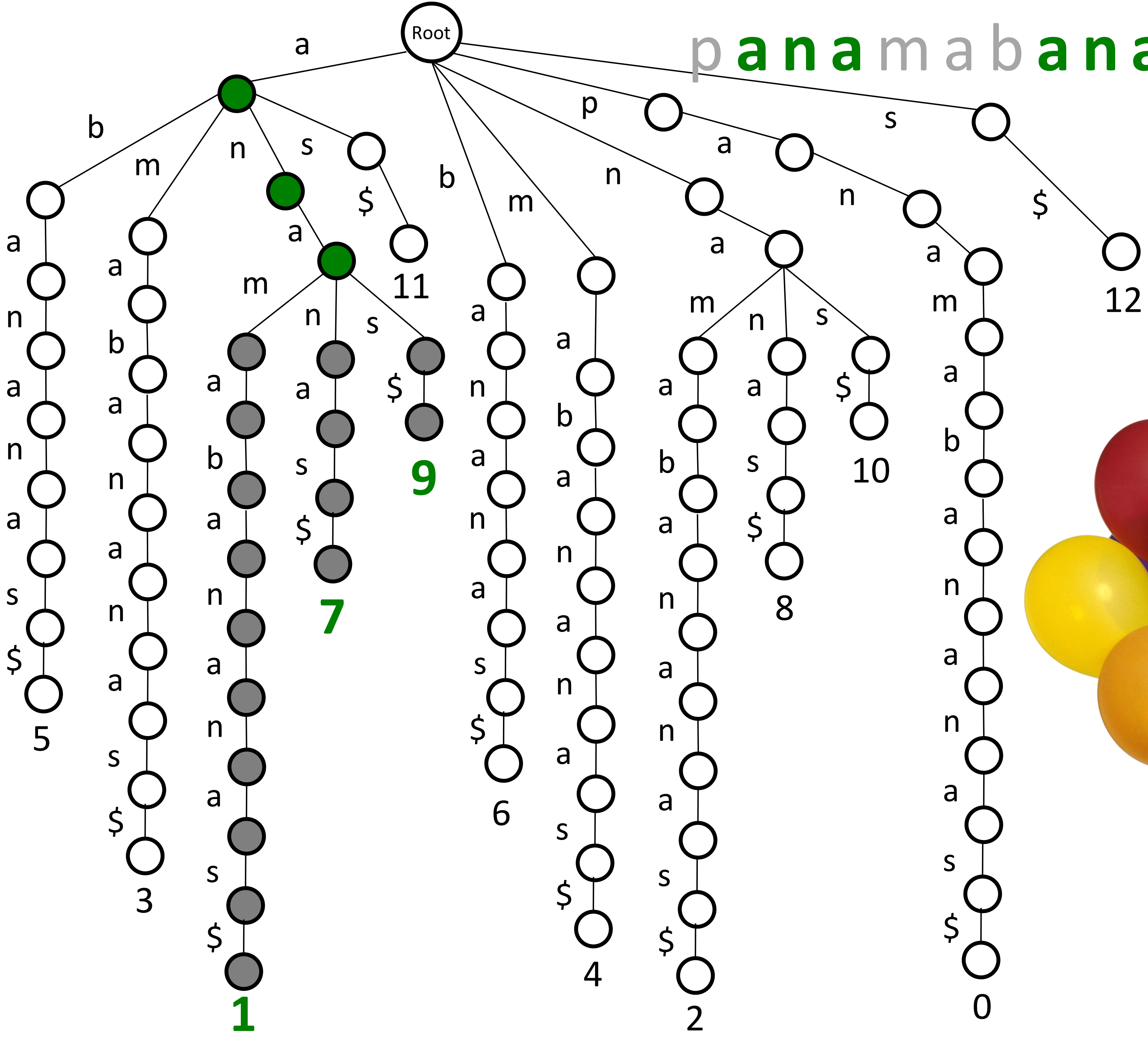




a n a
Root
5
3
1
7
9
11
6
4
2
8
10
0
12

p**ana**mab**ananas**s

$



Memory Footprint of Suffix Trie

The suffix trie is formed from |*Text*| suffixes with total length:

$$|Text| * (|Text| - 1)/2$$

For human genome:

- $|Text| \approx 3*10^9$

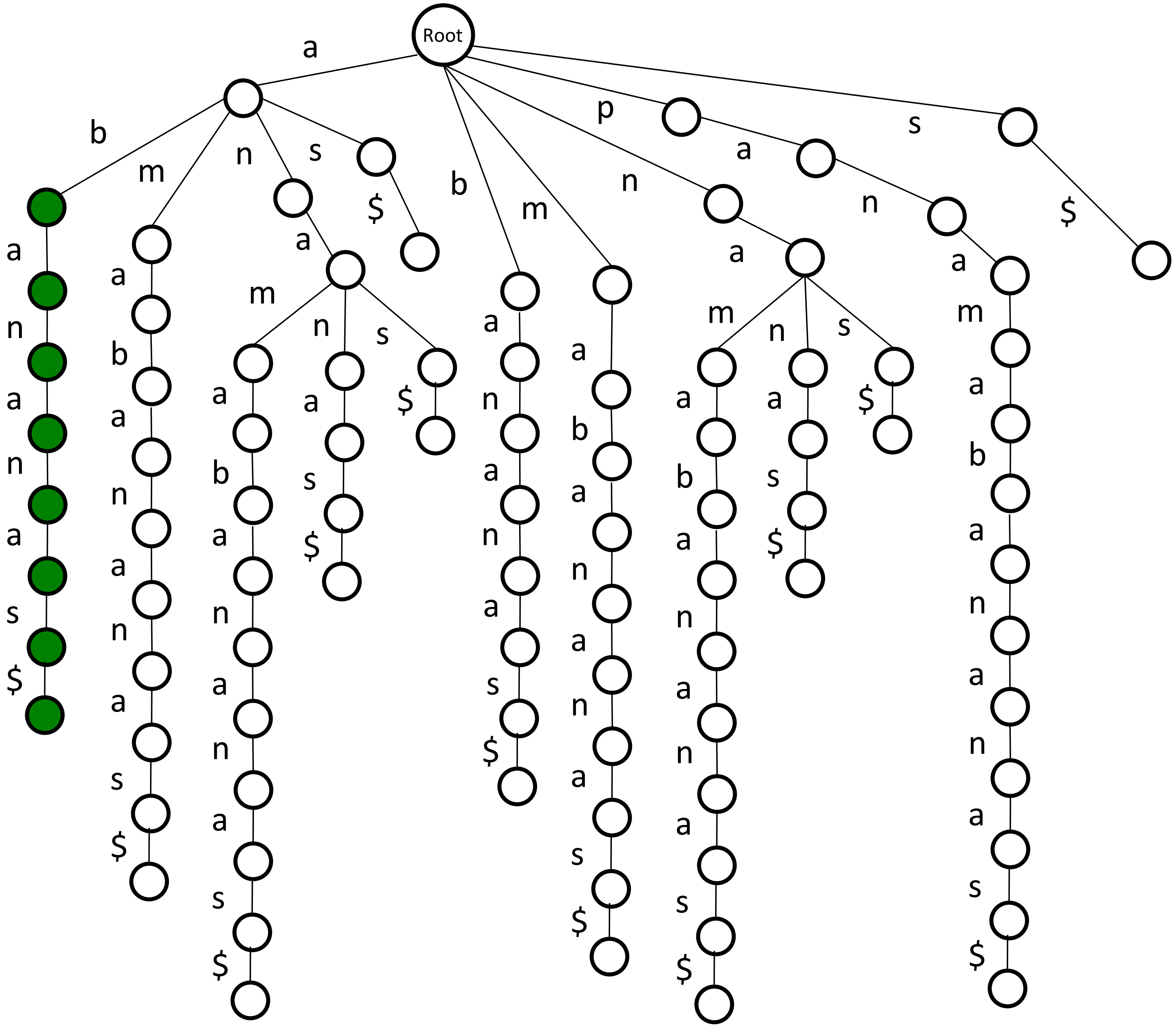
|*Text*| symbols

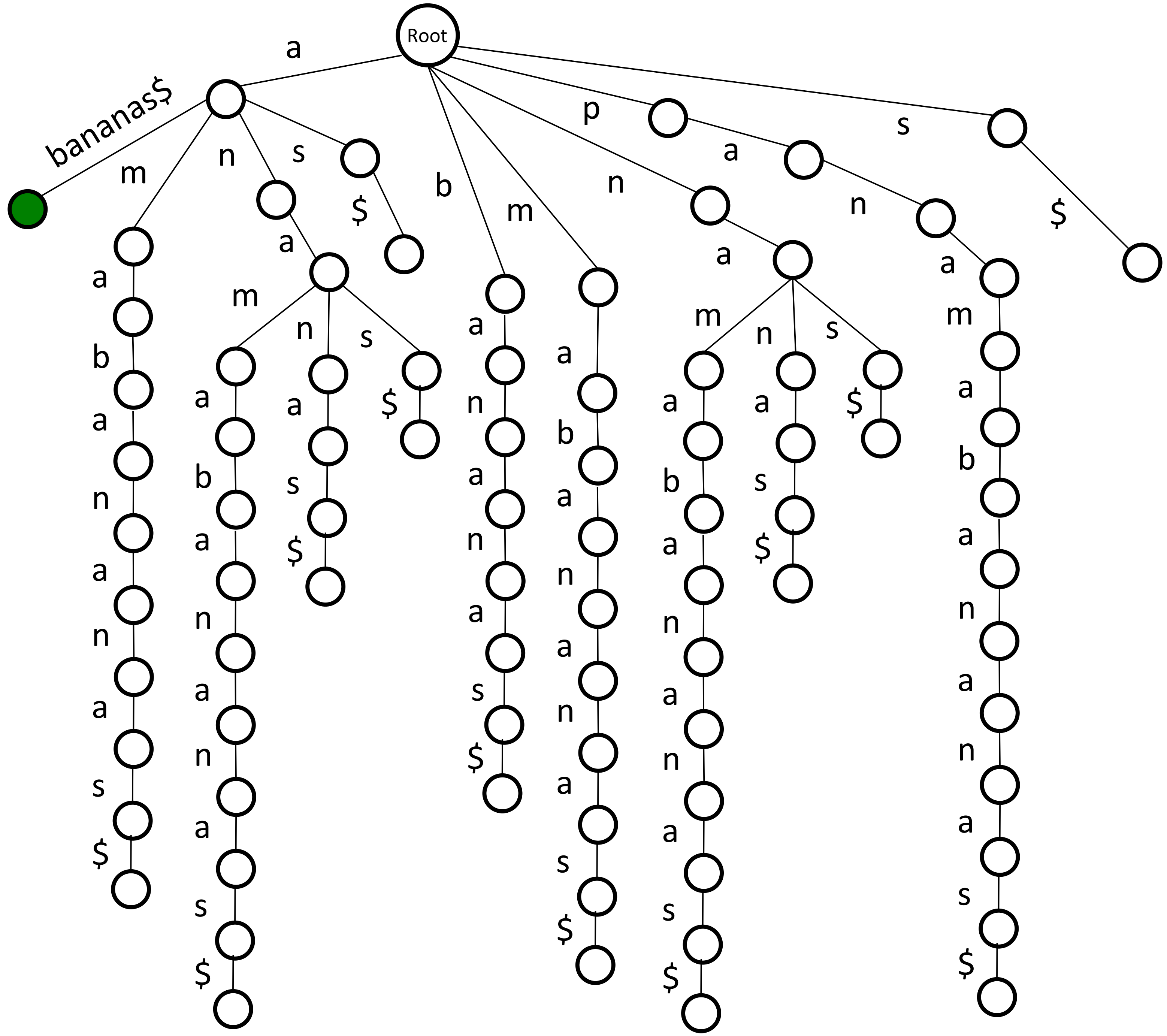
```
p a n a m a b a n a n a s $
  a n a m a b a n a n a s $
    n a m a b a n a n a s $
      a m a b a n a n a s $
        m a b a n a n a s $
          a b a n a n a s $
            b a n a n a s $
              a n a n a s $
                n a n a s $
                  a n a s $
                    n a s $
                      a s $
                        s $
                          $
```

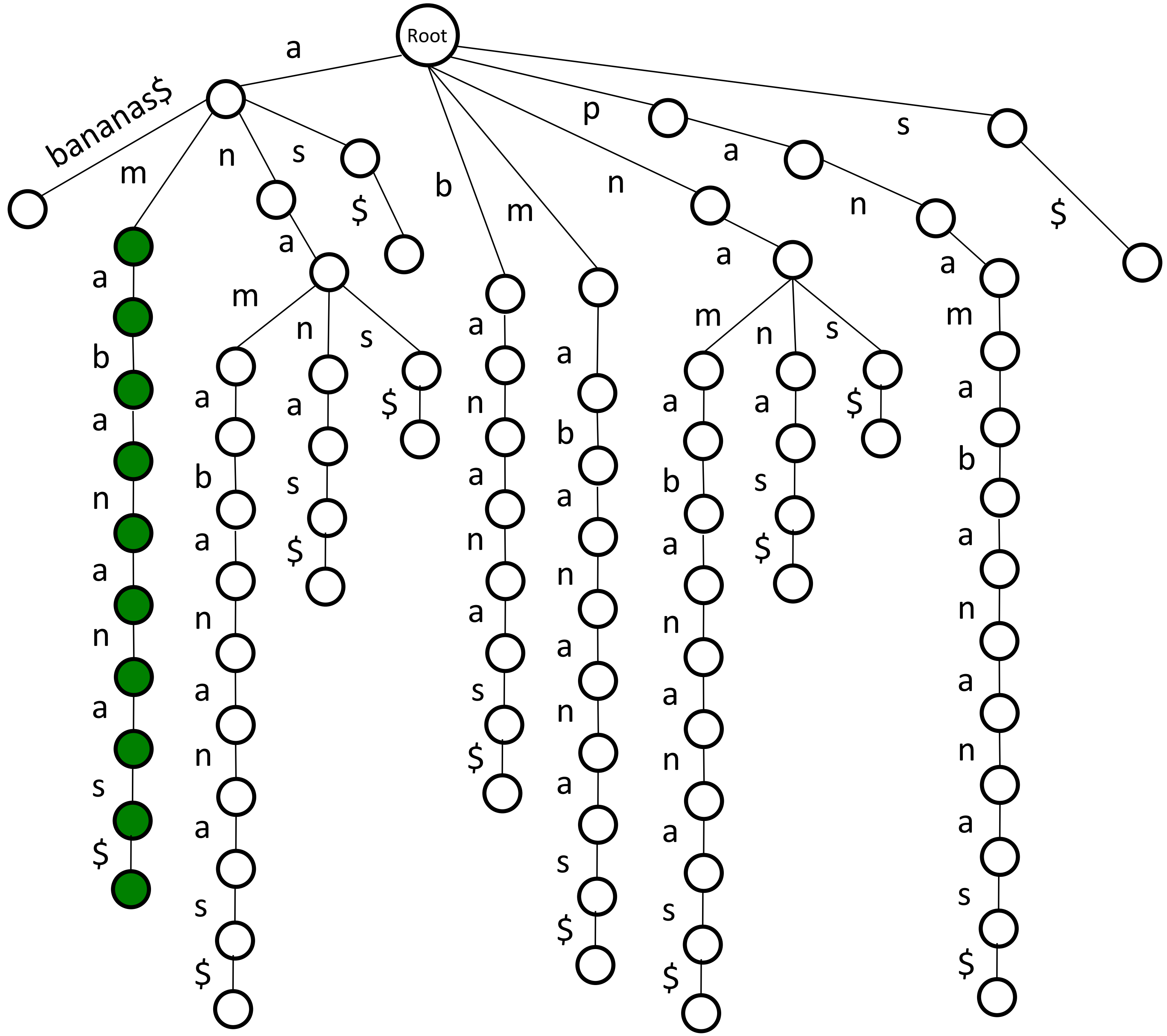
|*Text*| suffixes

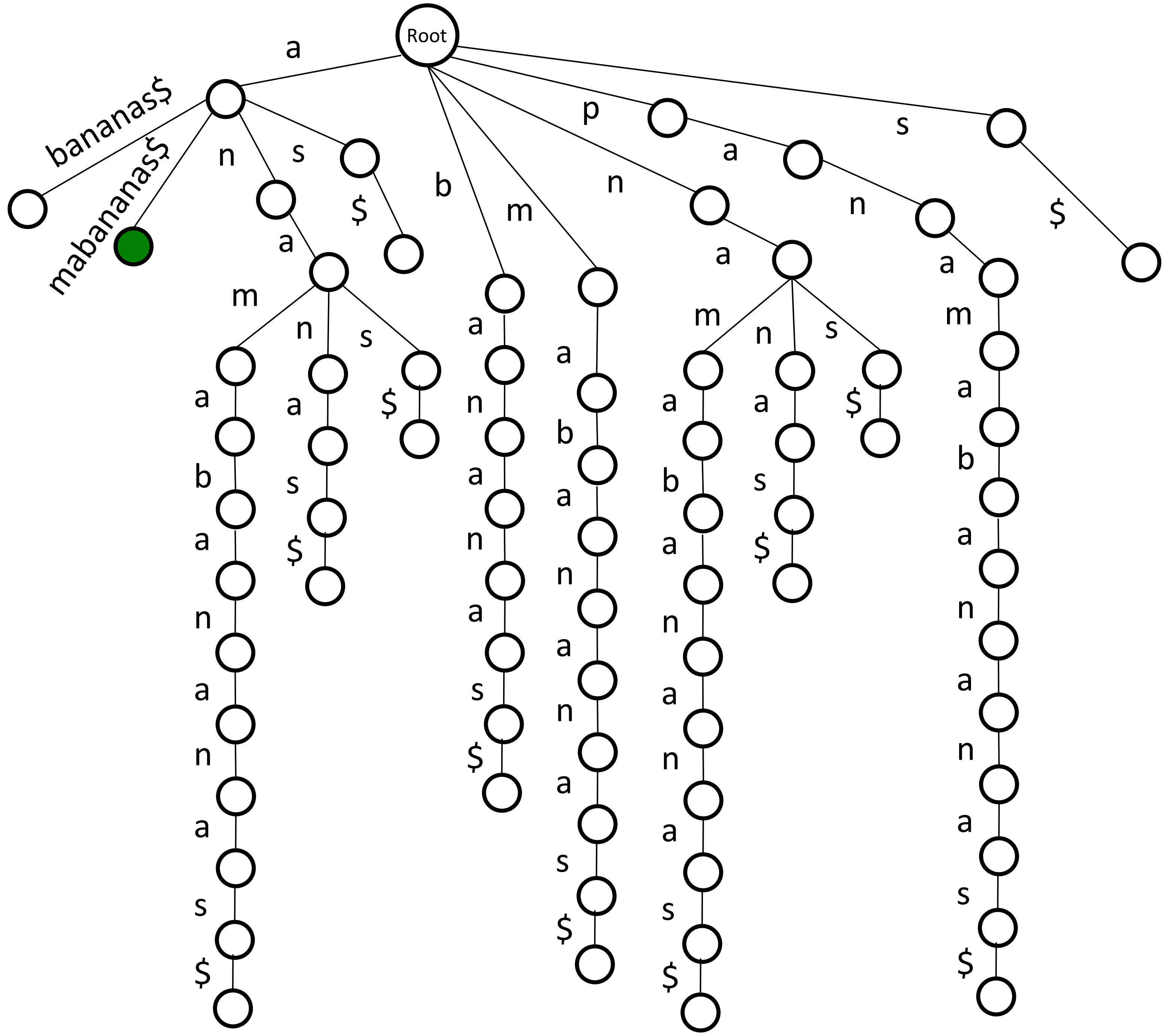
Outline

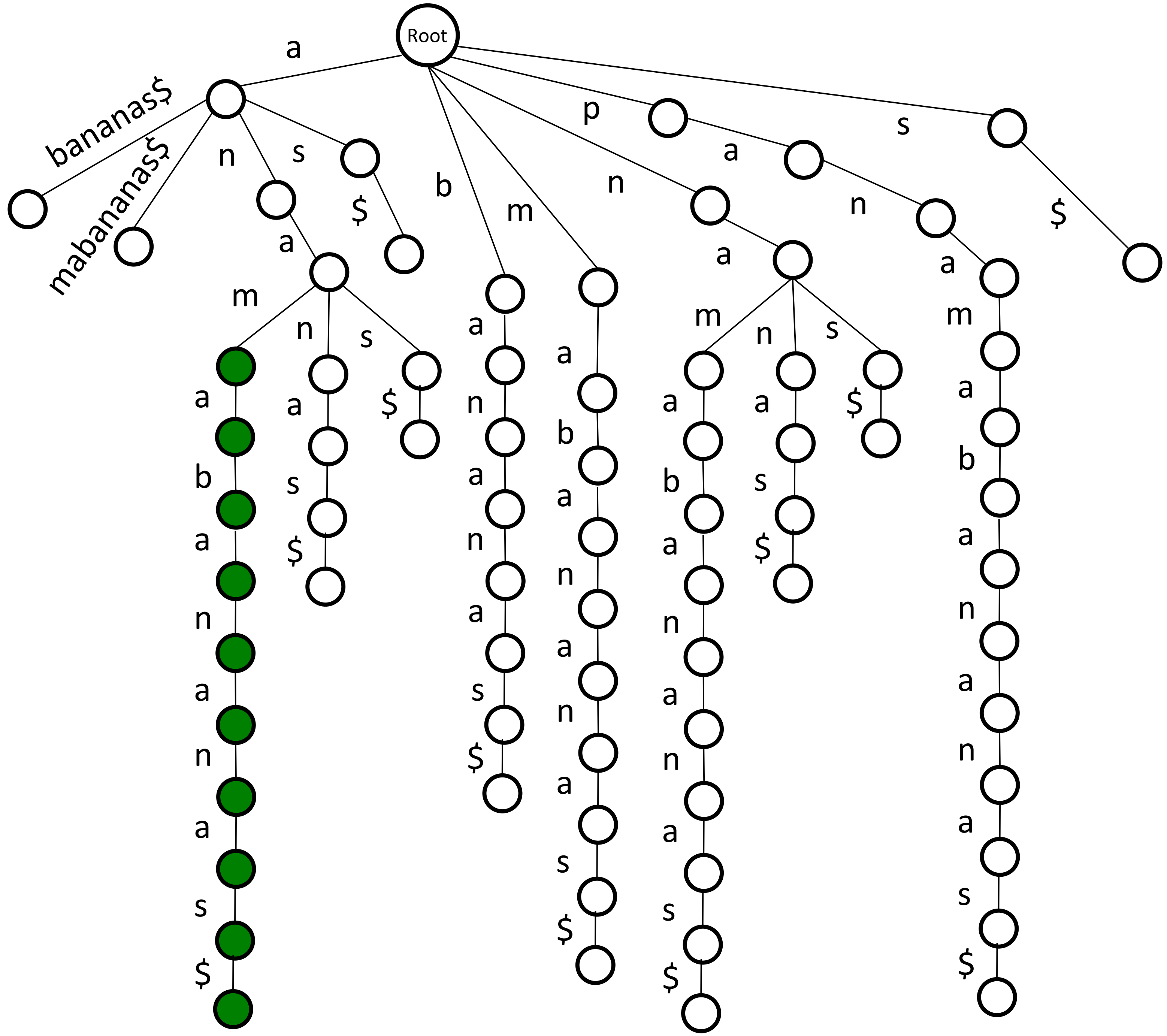
- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- **From Suffix Tries to Suffix Trees**


Root
a
b
m
n
s
$
p

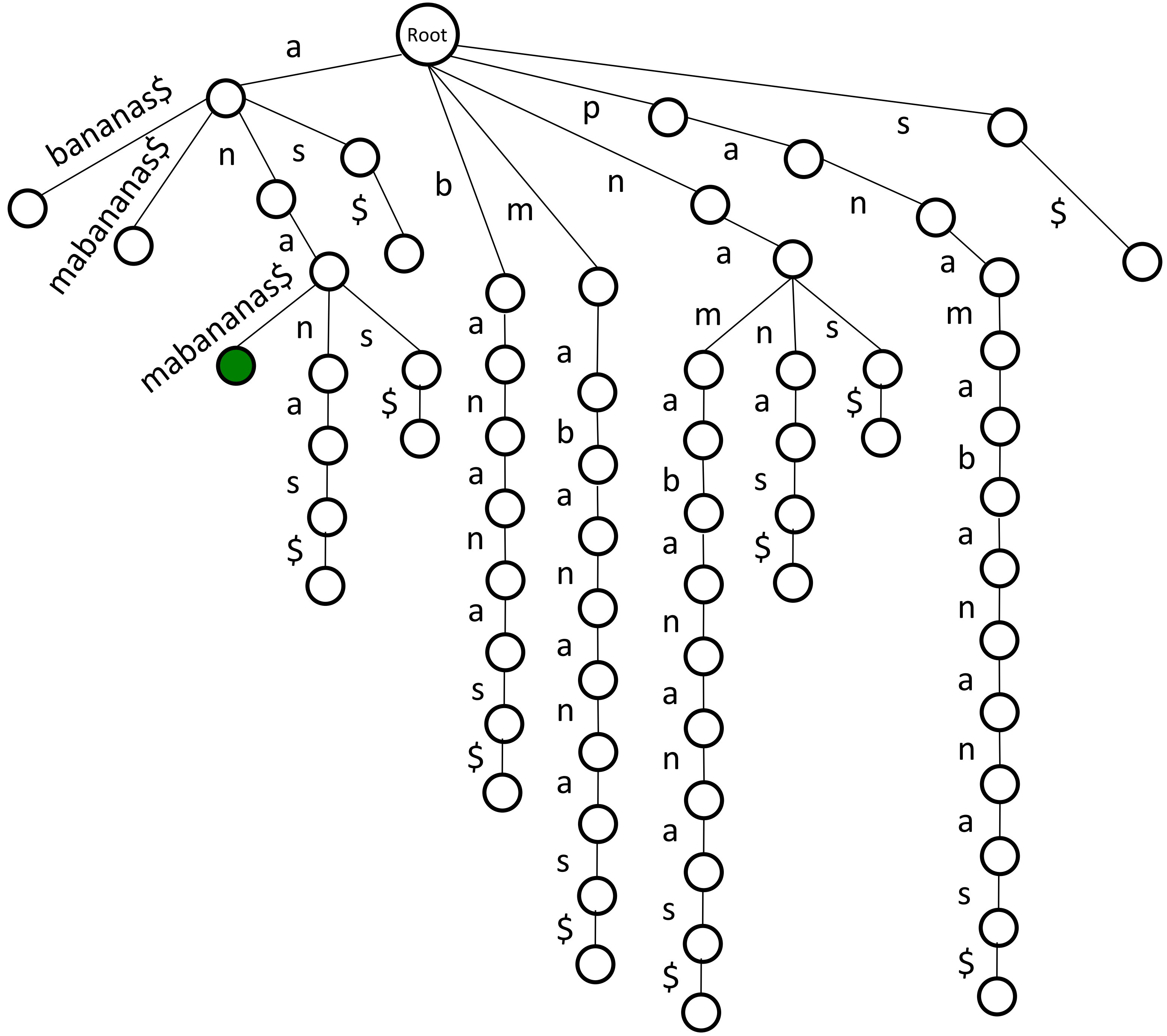

Root
a
bananas$
m
n
s
b
m
n
p
a
n
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
$
a
b
a
n
a
n
a
s
$
a
s
$
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
$
a
m
a
b
a
n
a
n
a
s
$

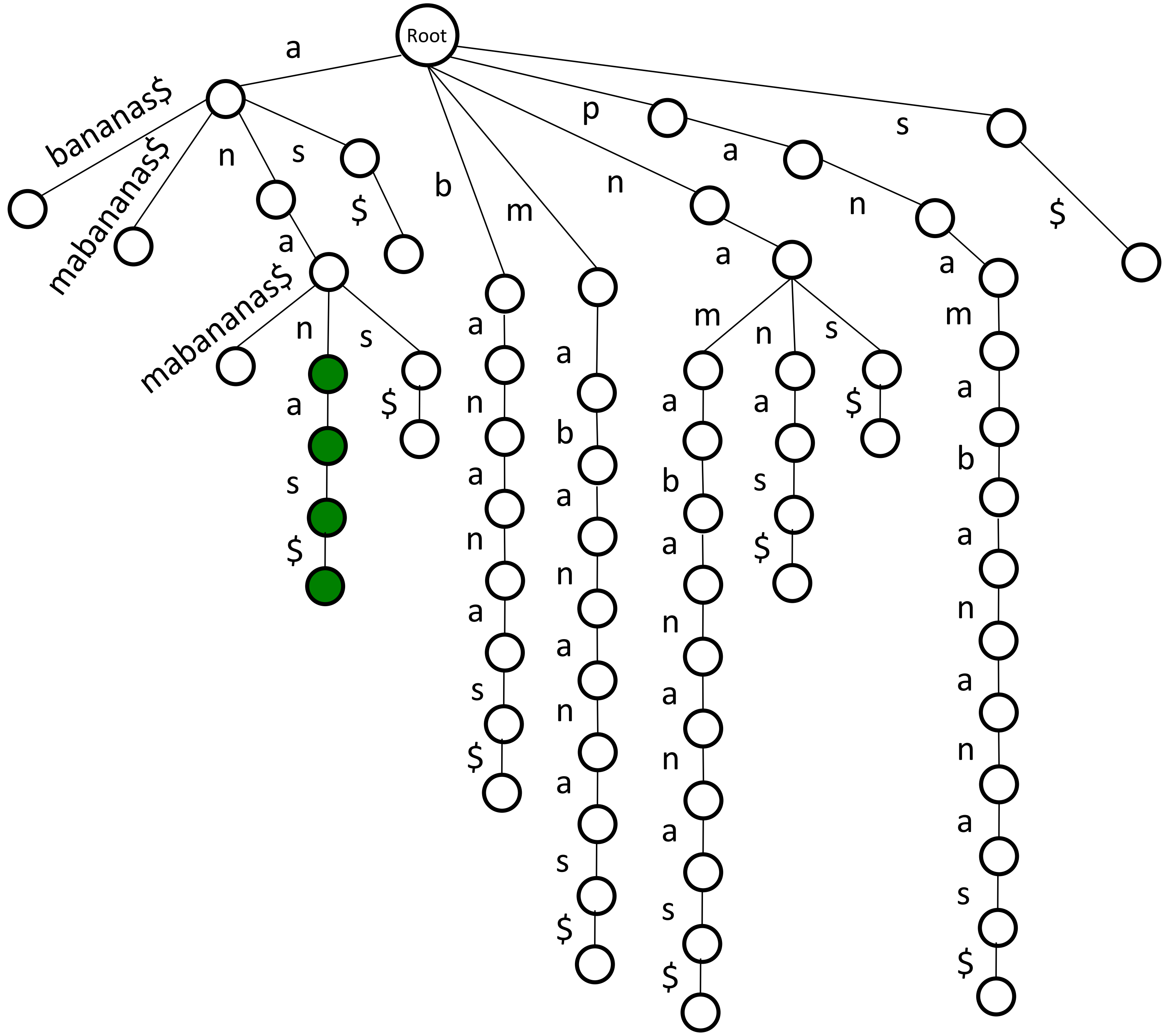

Root
a
bananas$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
s
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

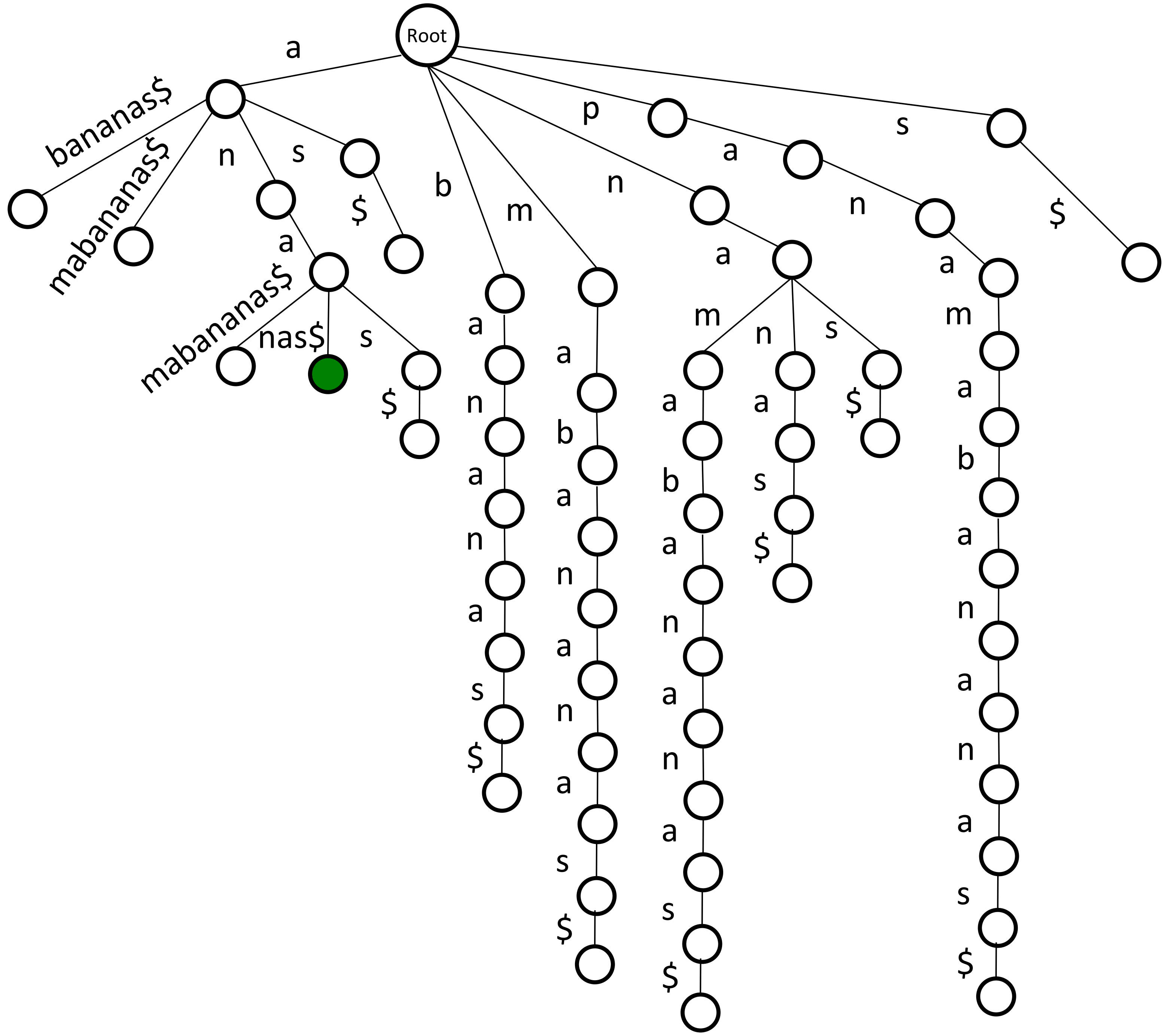

Root
a
bananas$
mabananas$
n
s
$
b
m
p

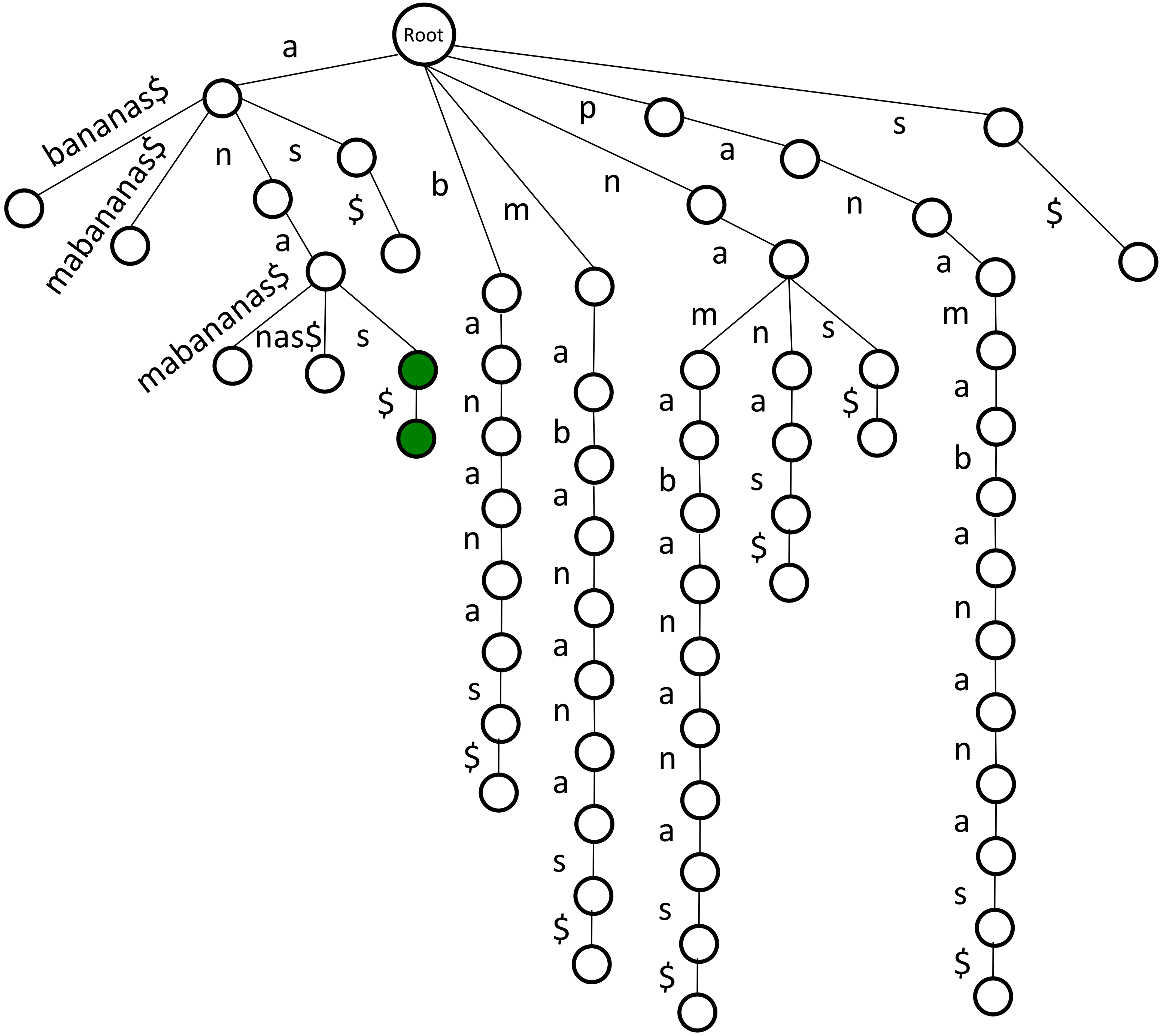


Root
a
bananas$
mabananas$
n
s
$
a
m
n
s
b
m
n
p
a
n
a
m
s
$
a
b
a
n
a
n
a
s
$

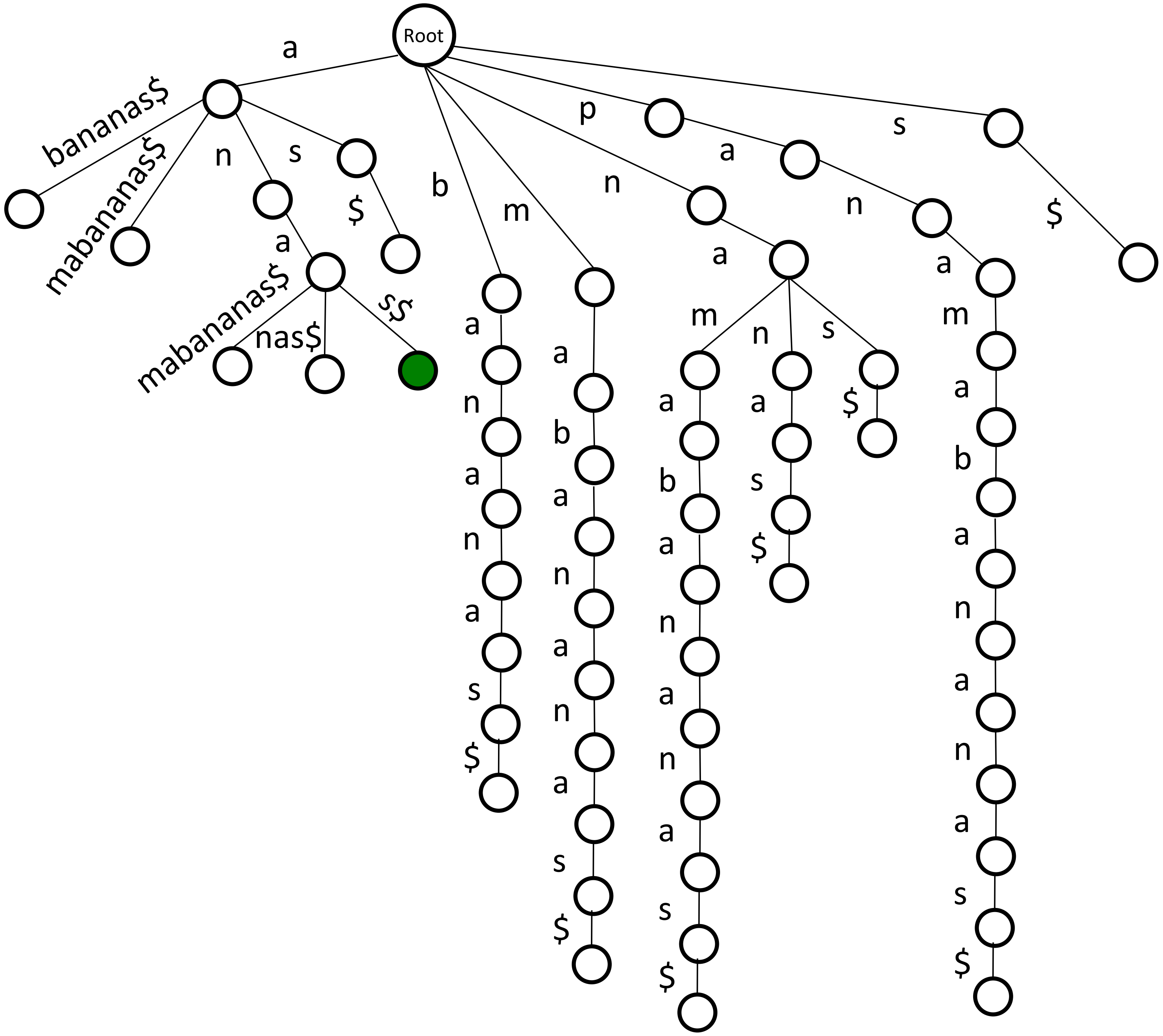

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
n
s
a
s
$
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$


Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
n
s
a
s
$
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

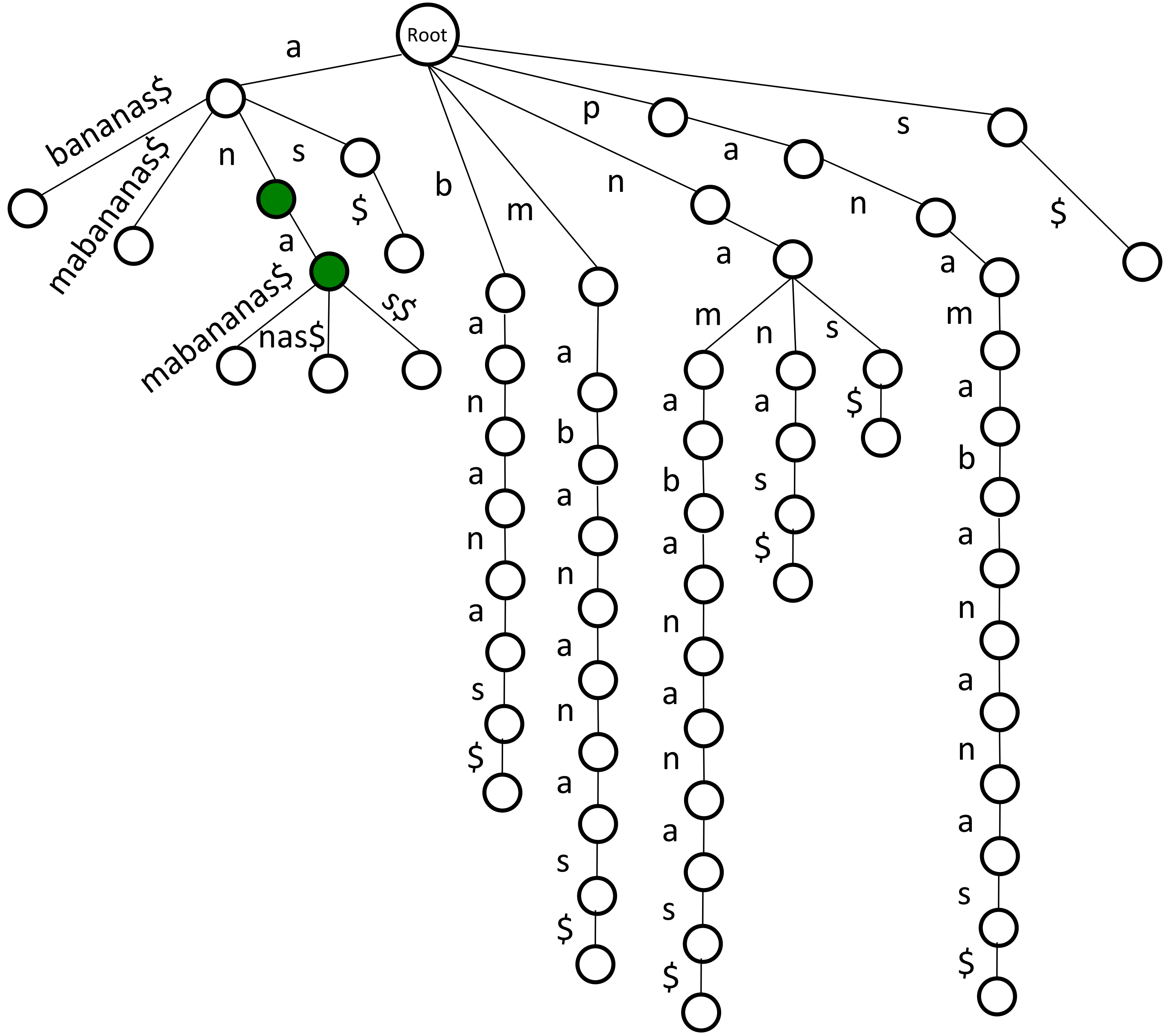

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

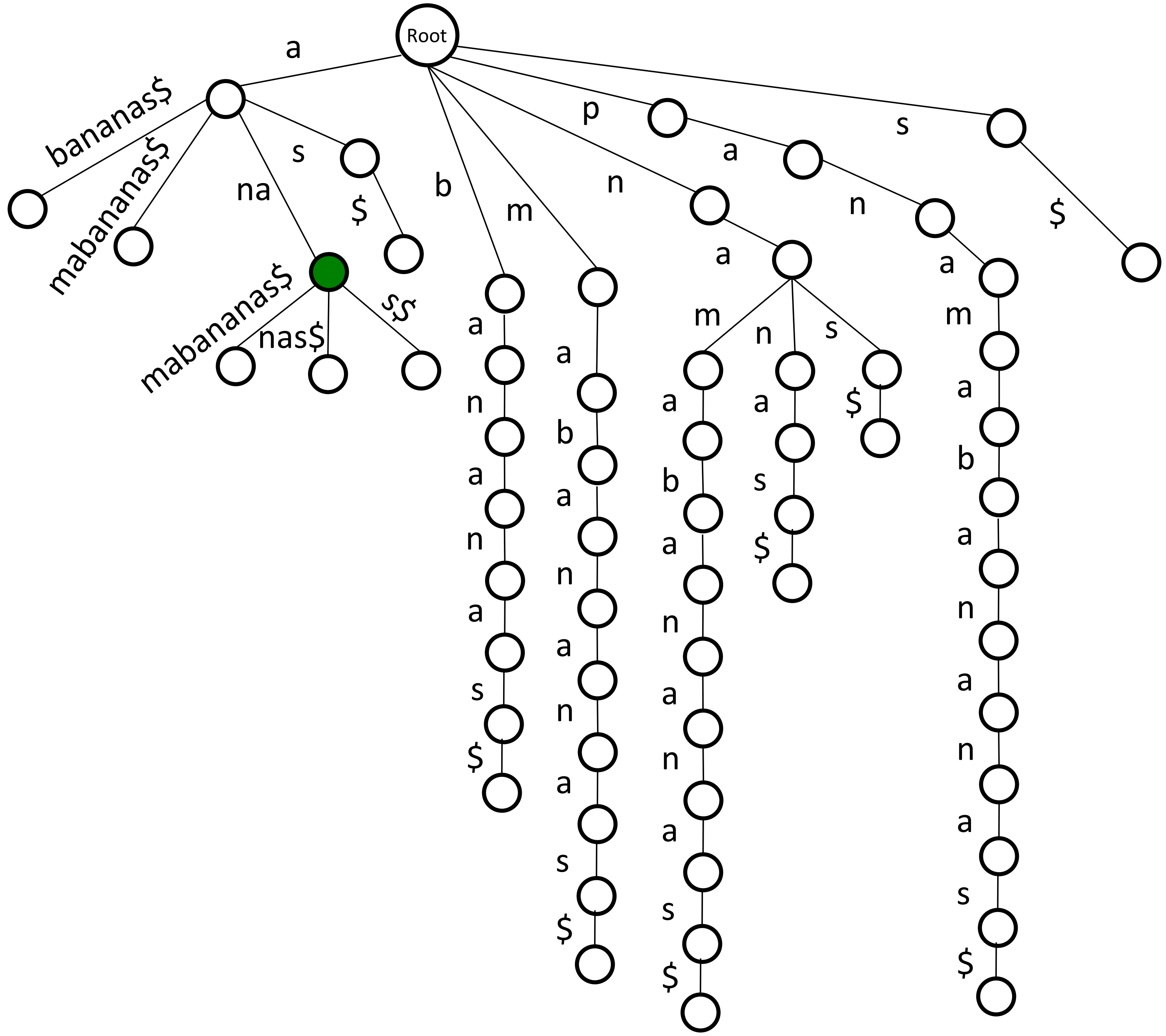


Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

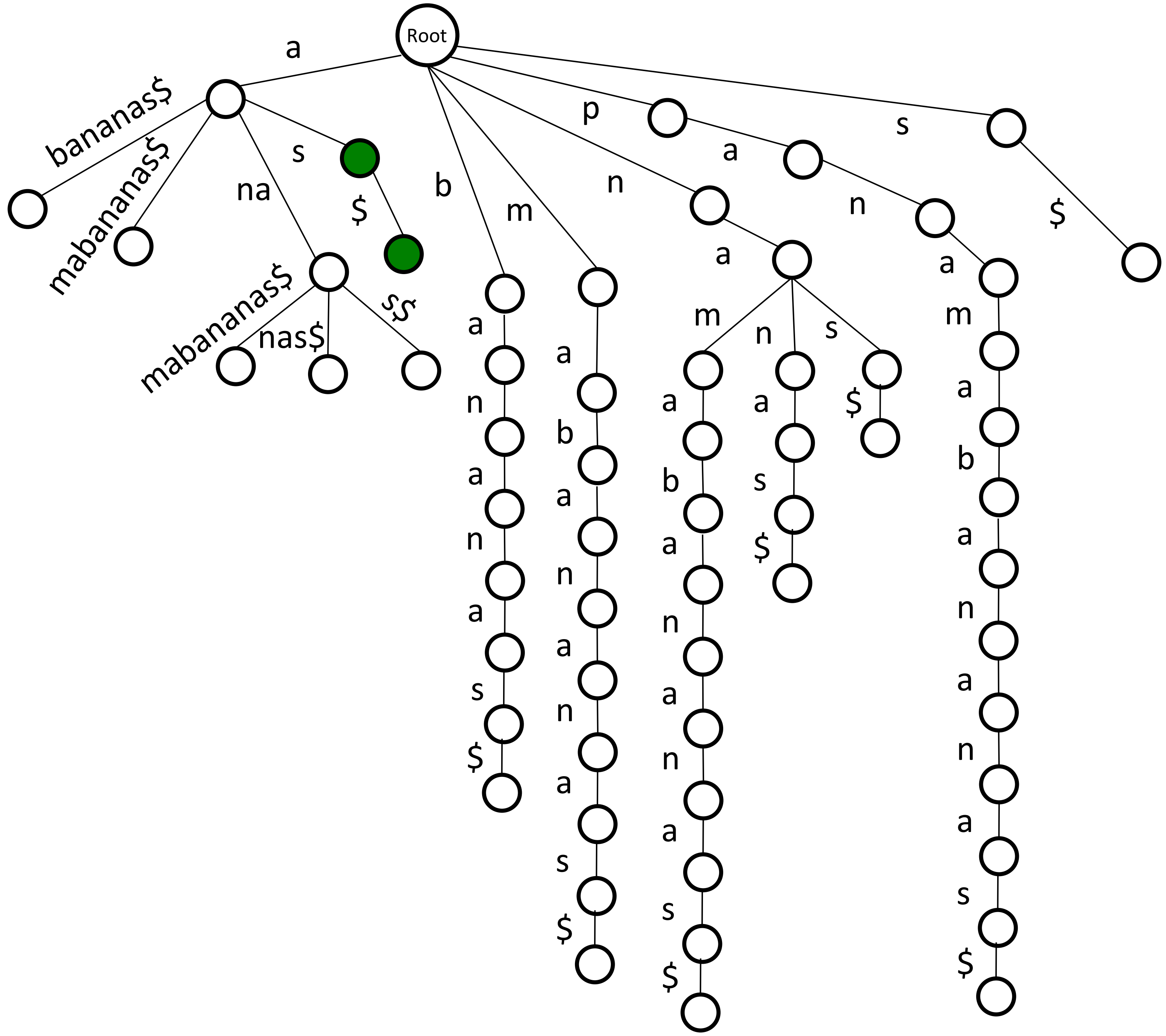


Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s$
b
m
n
p
a
n
s
$

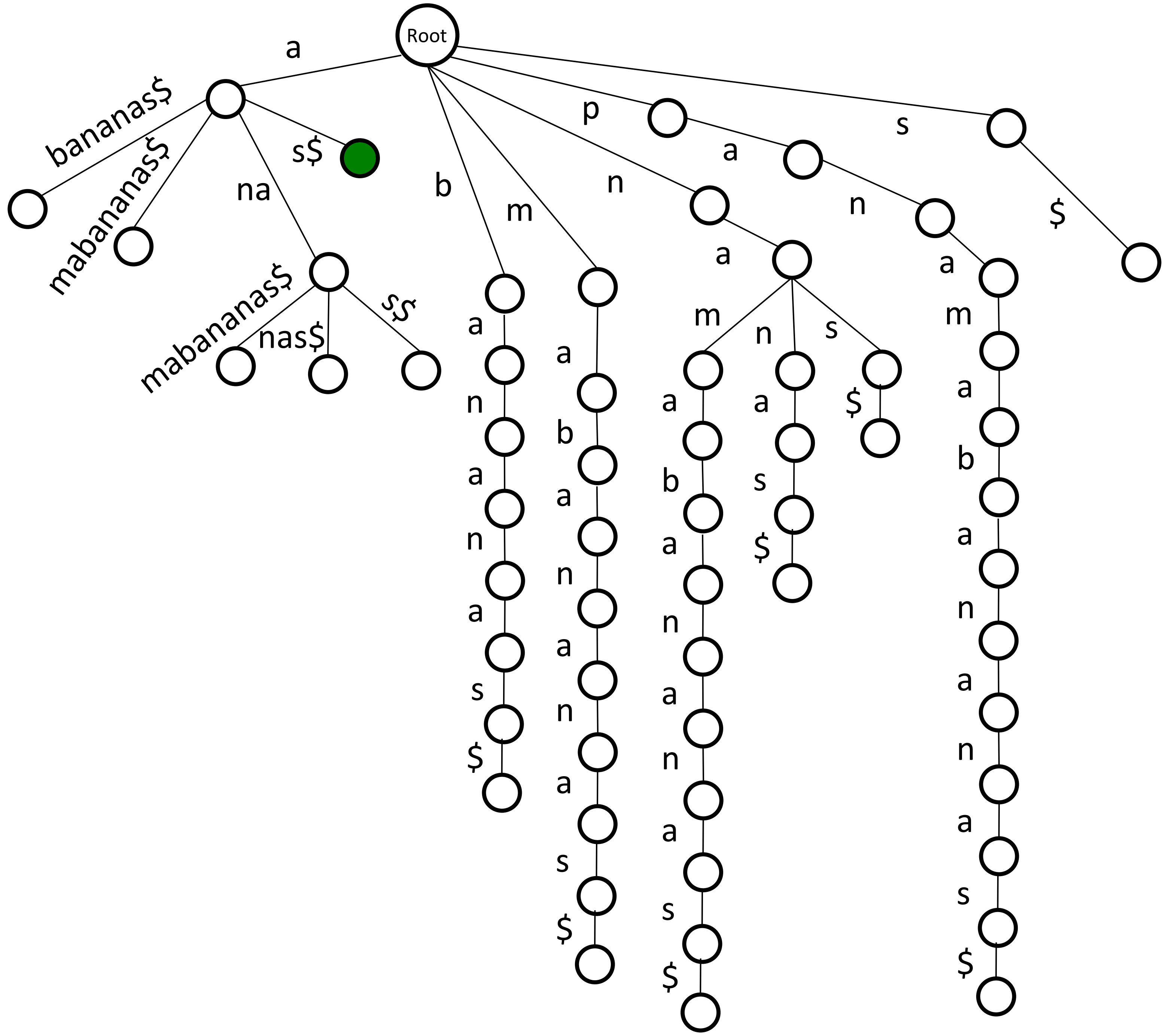

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$



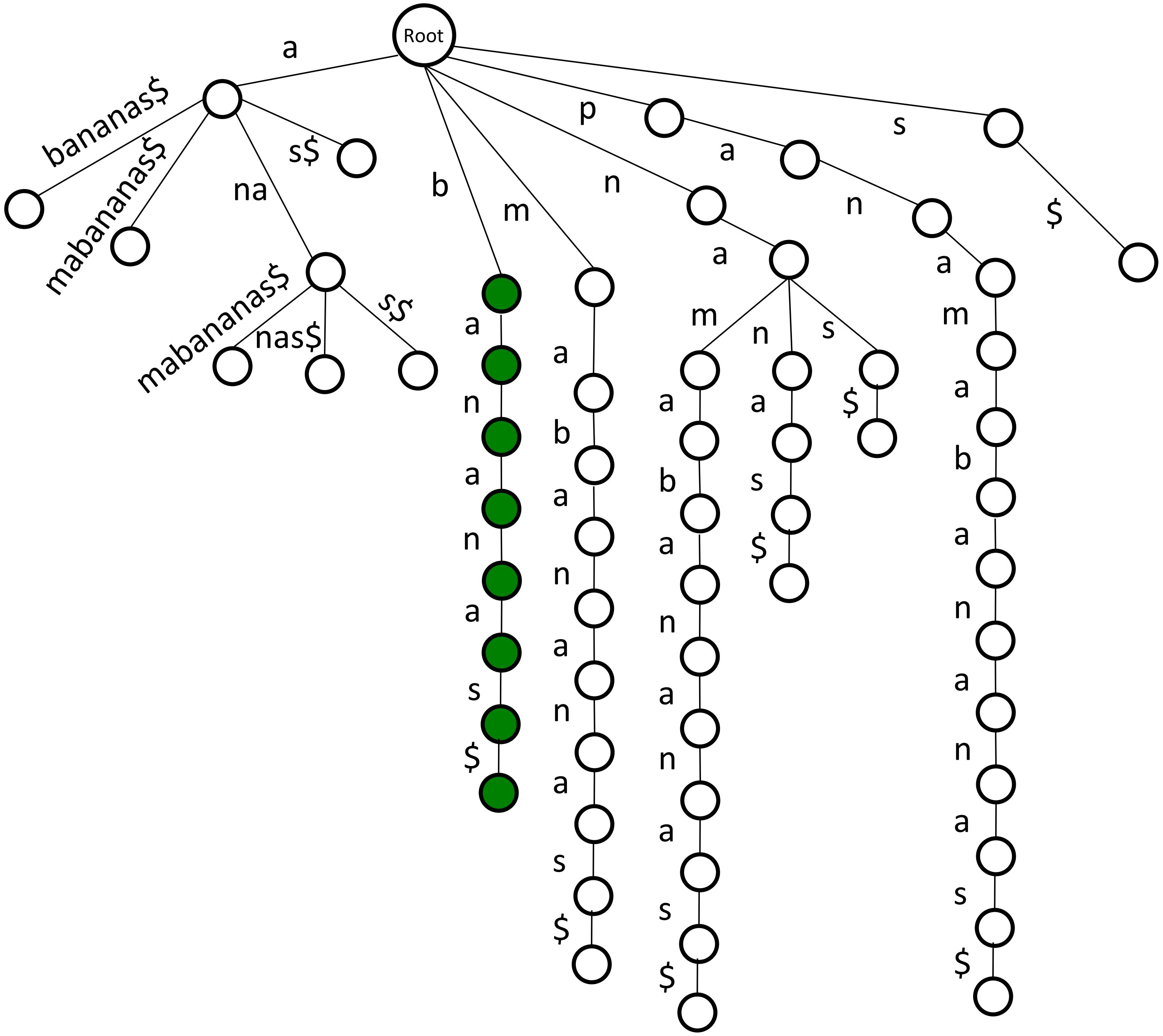
Root
a
bananas$
mabananas$
na
s
$
mabananas$
nas$
s$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$



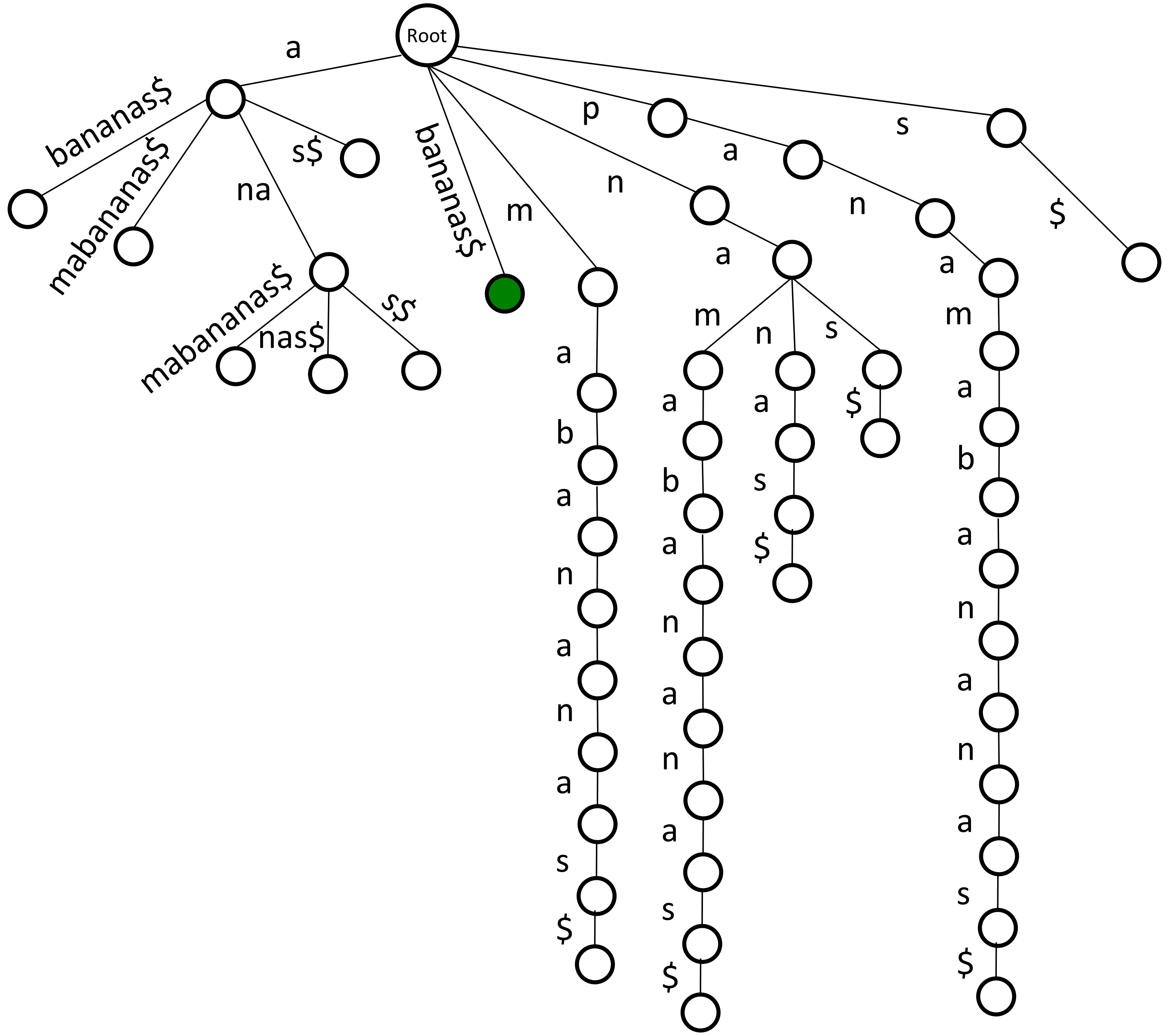
Root
a
bananas$
mabananas$
na
s
$
mabananas$
nas$
s$
b
a n a n a s $
m
a b a n a n a s $
n
a
m a b a n a n a s $
n a s $
s $
p
a
n
a m a b a n a n a s $
s
$

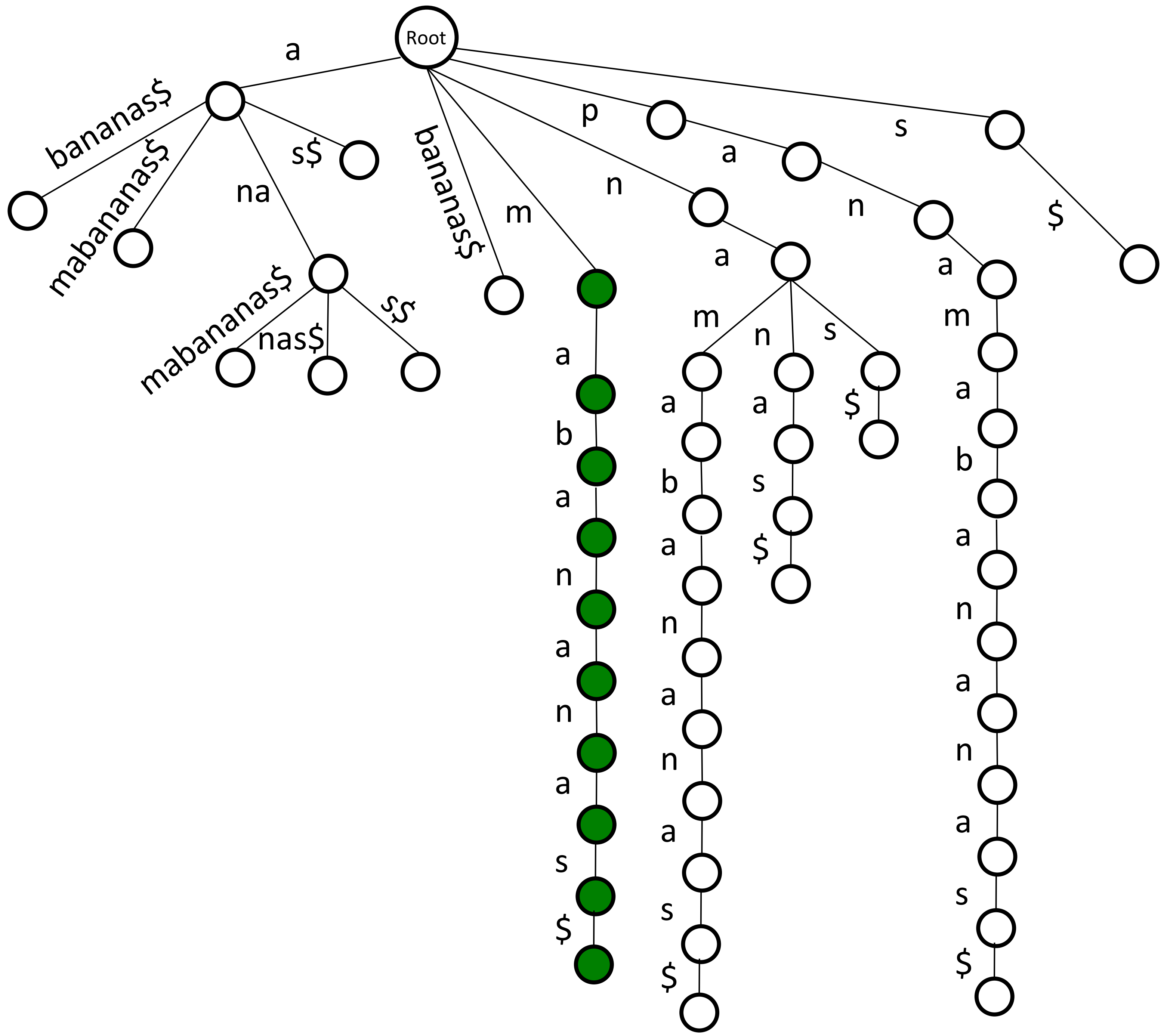


Root
a
bananas$
mabananas$
s$
na
mabananas$
nas$
s$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

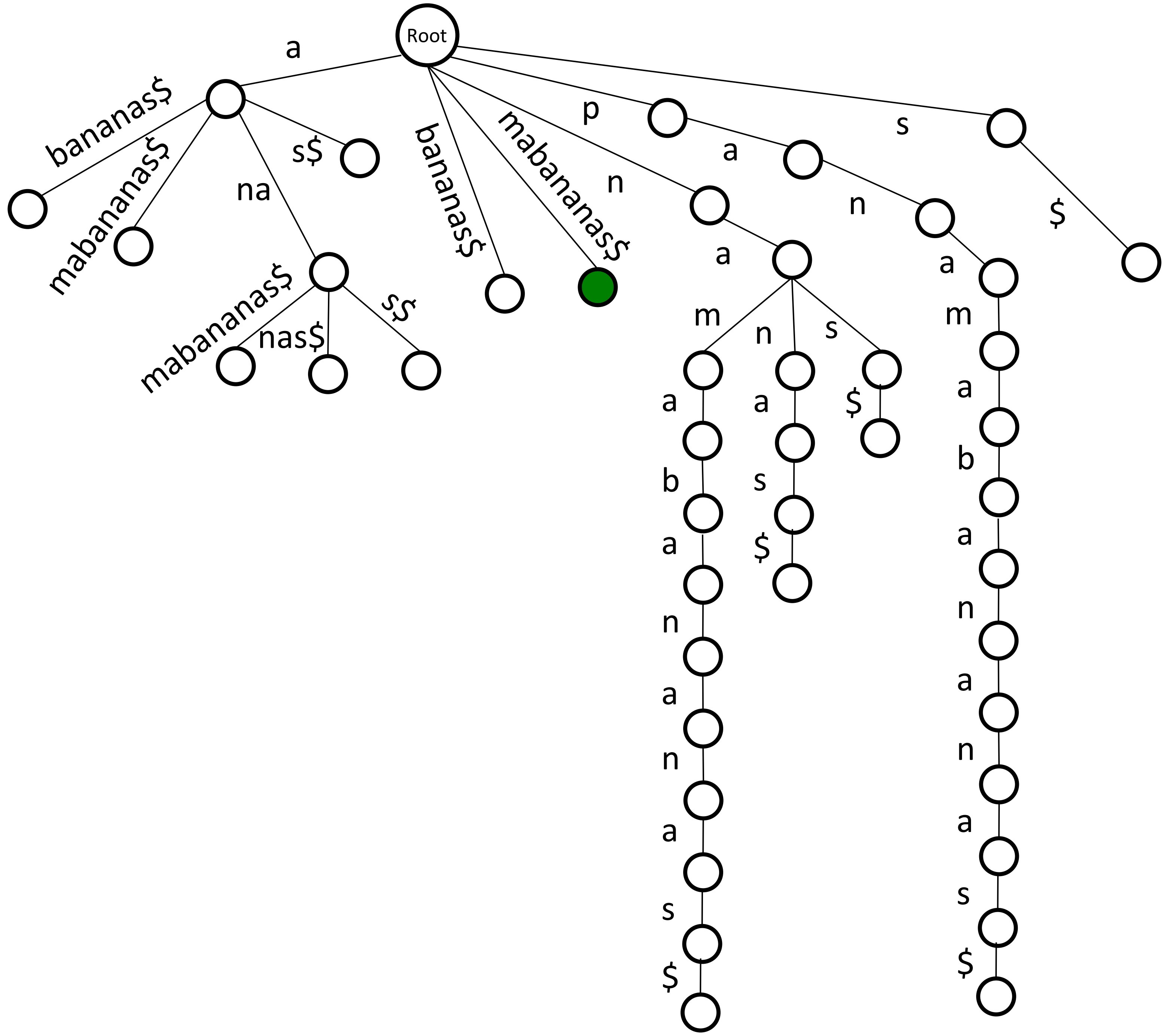


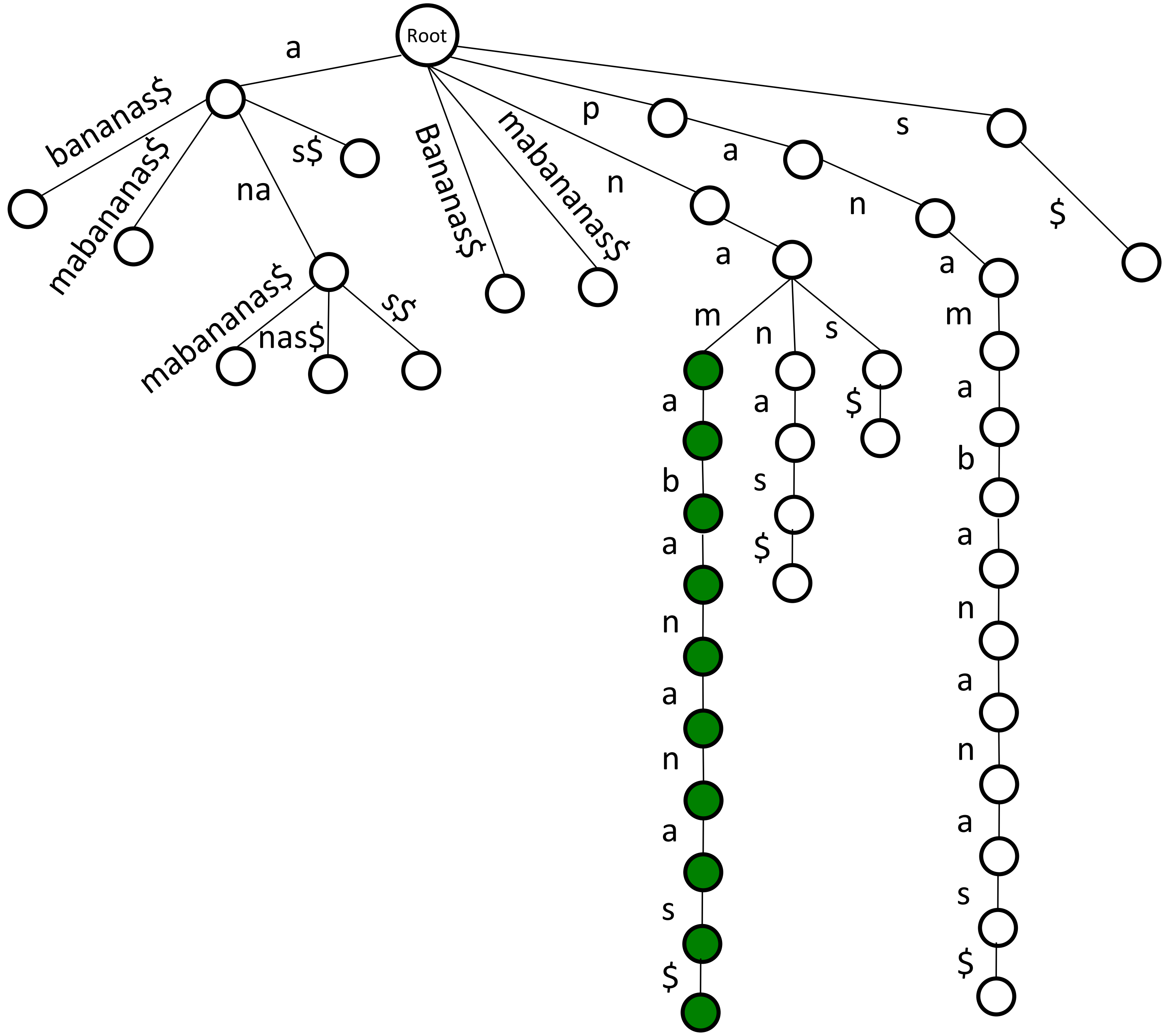
Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$


Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
m
a b a n a n a s $
n
a
m a b a n a n a s $
n a s $
s $
p
a
n
a
m a b a n a n a s $
s
$

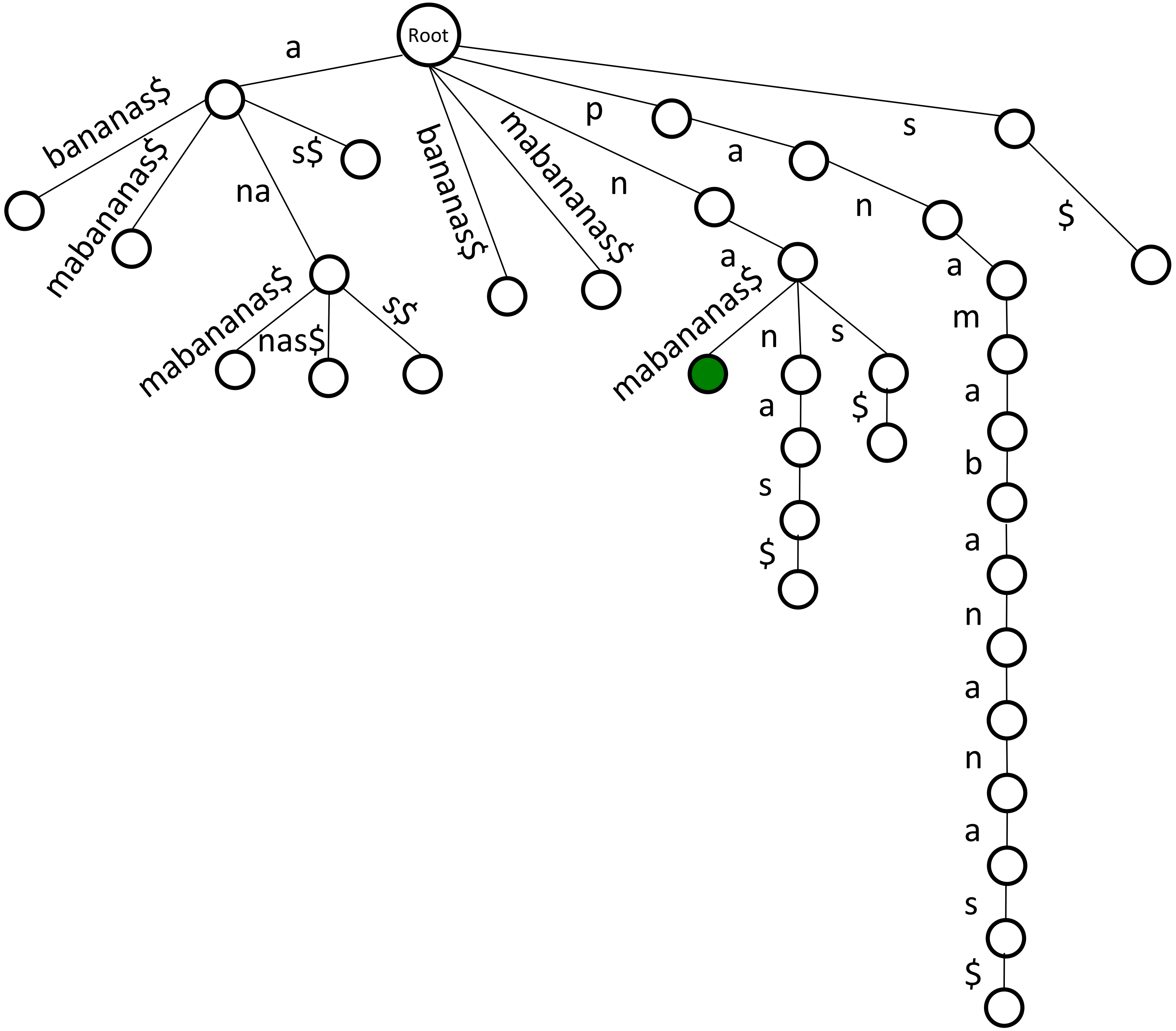


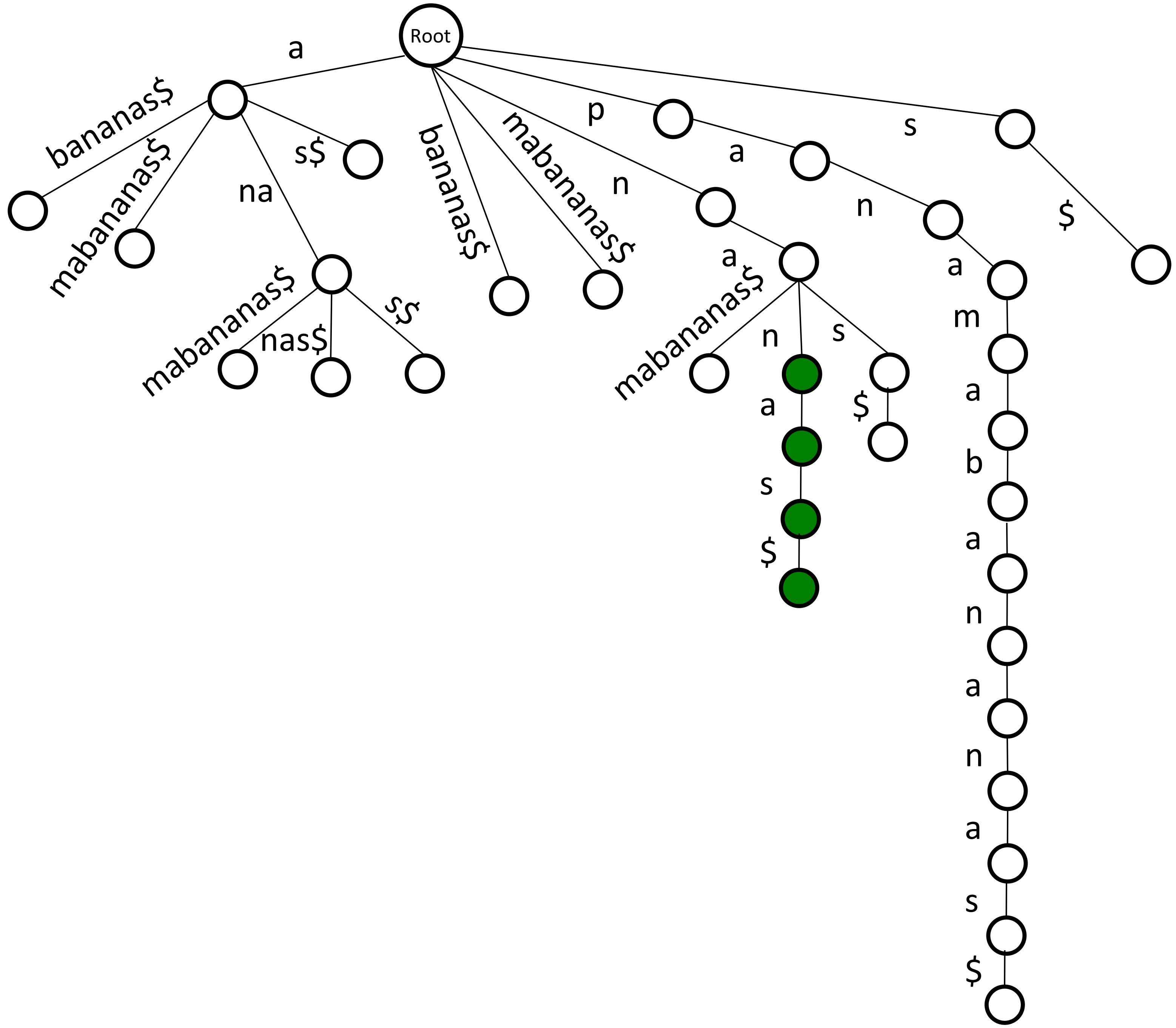
Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

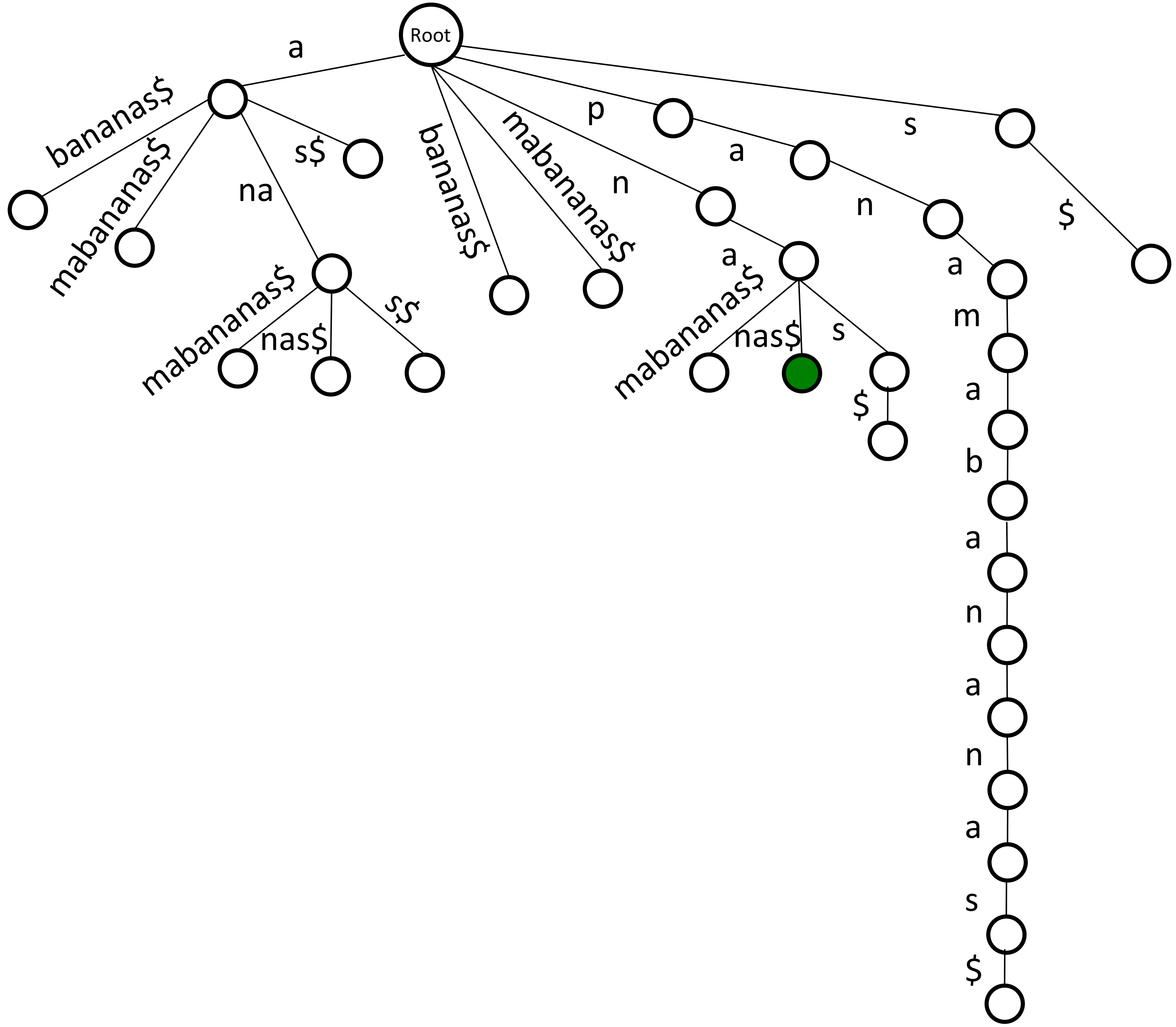

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
s
$

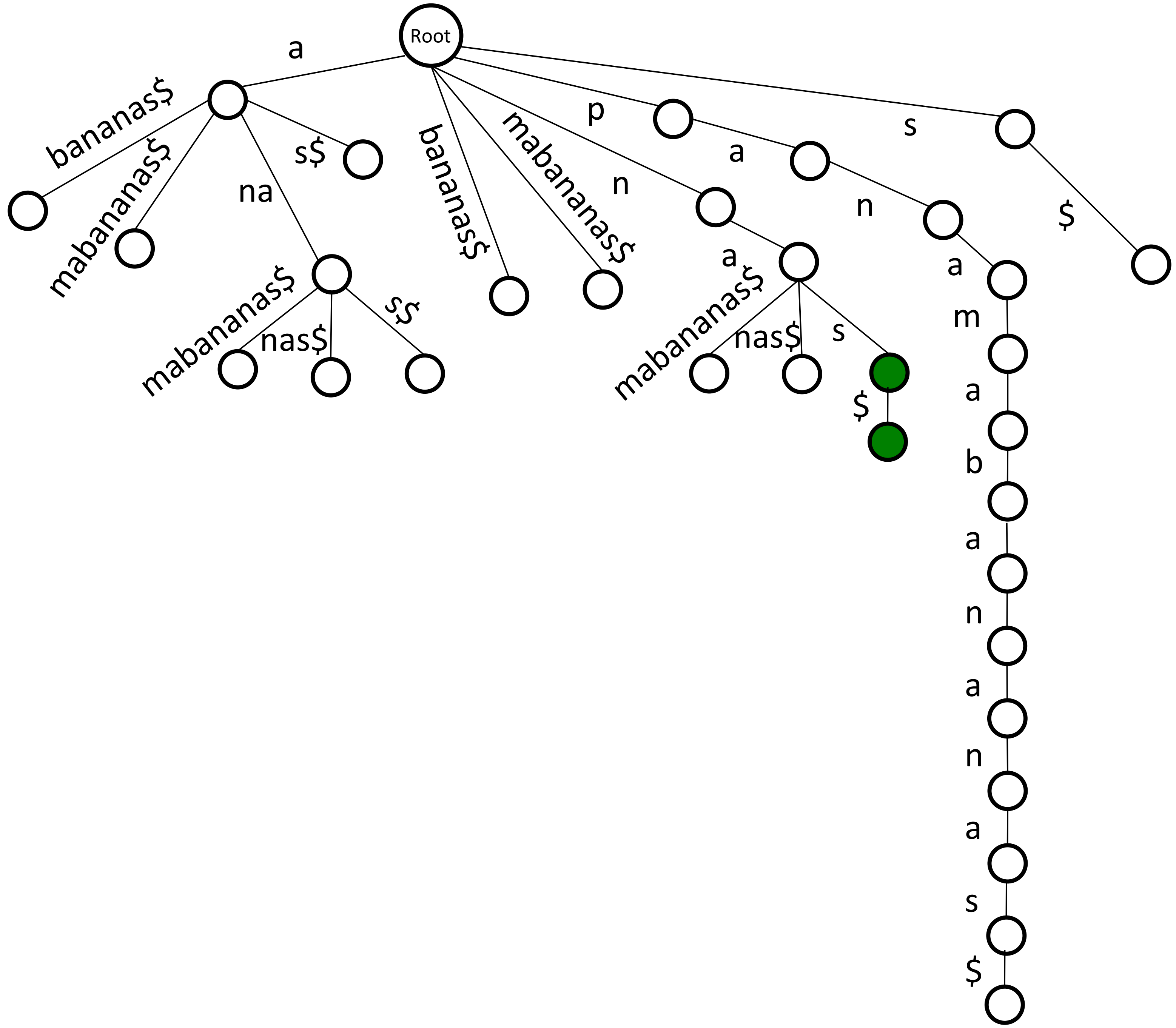


Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
Bananas$
mabananas$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

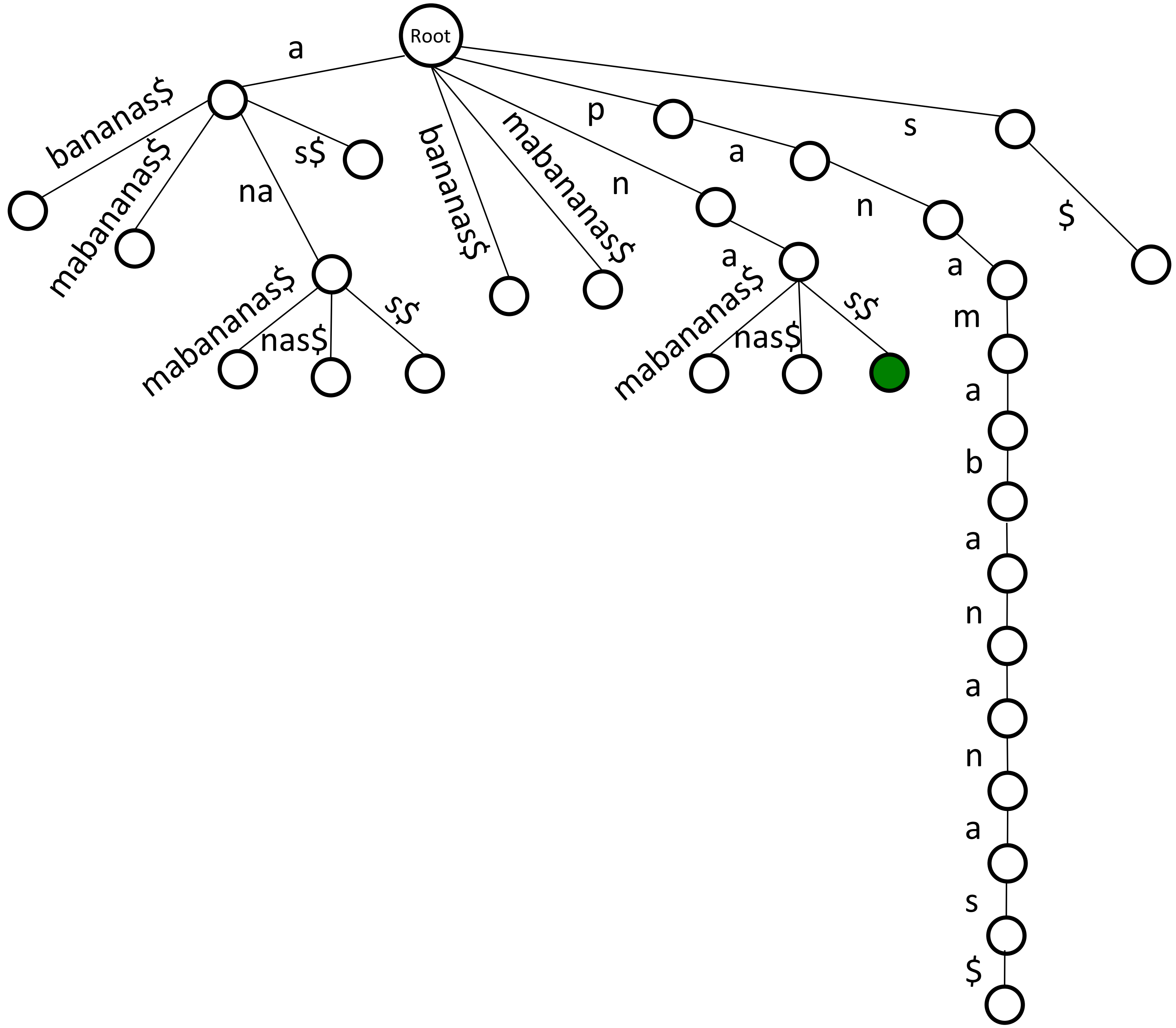

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$


Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
n
a
s
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

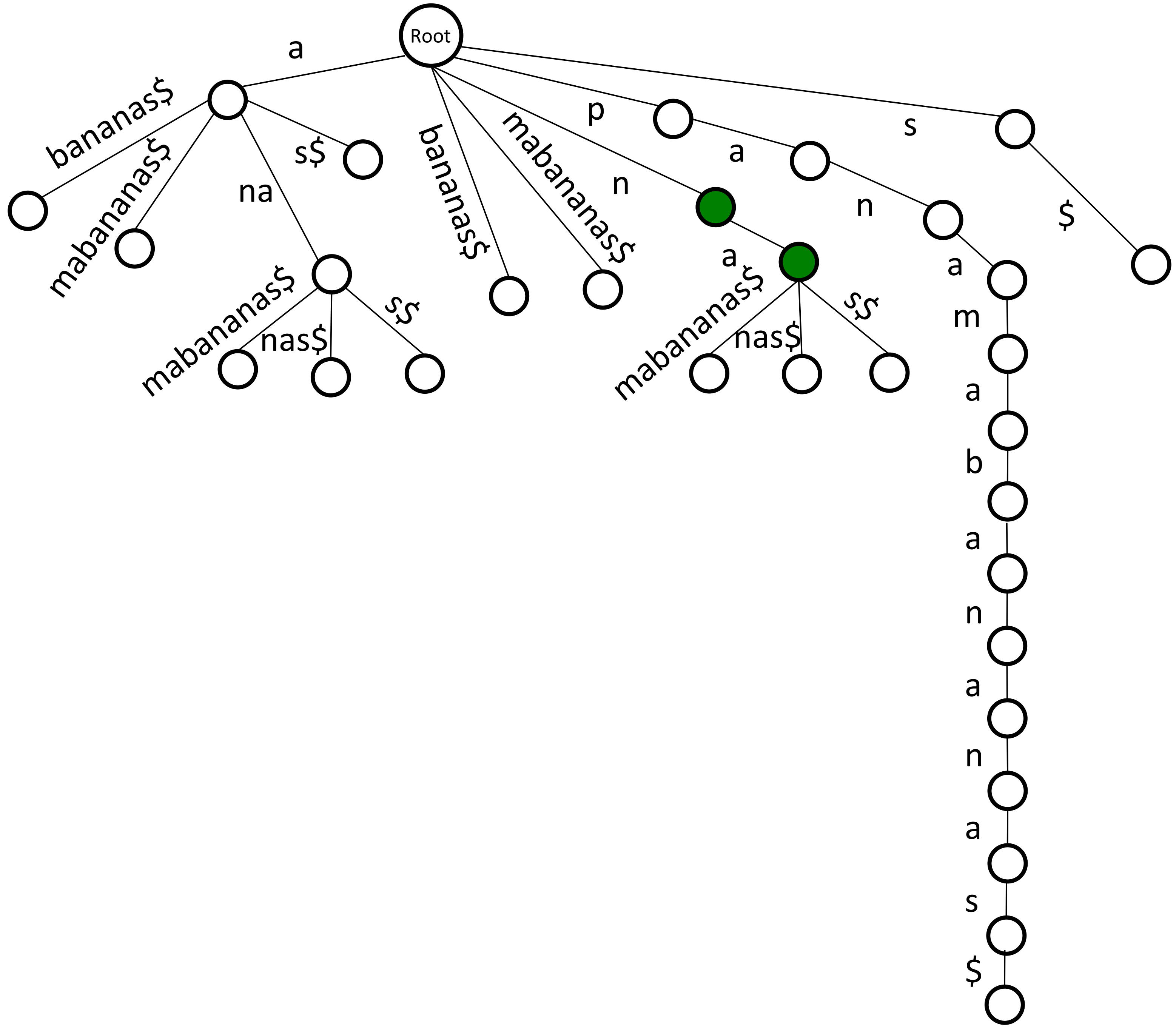

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

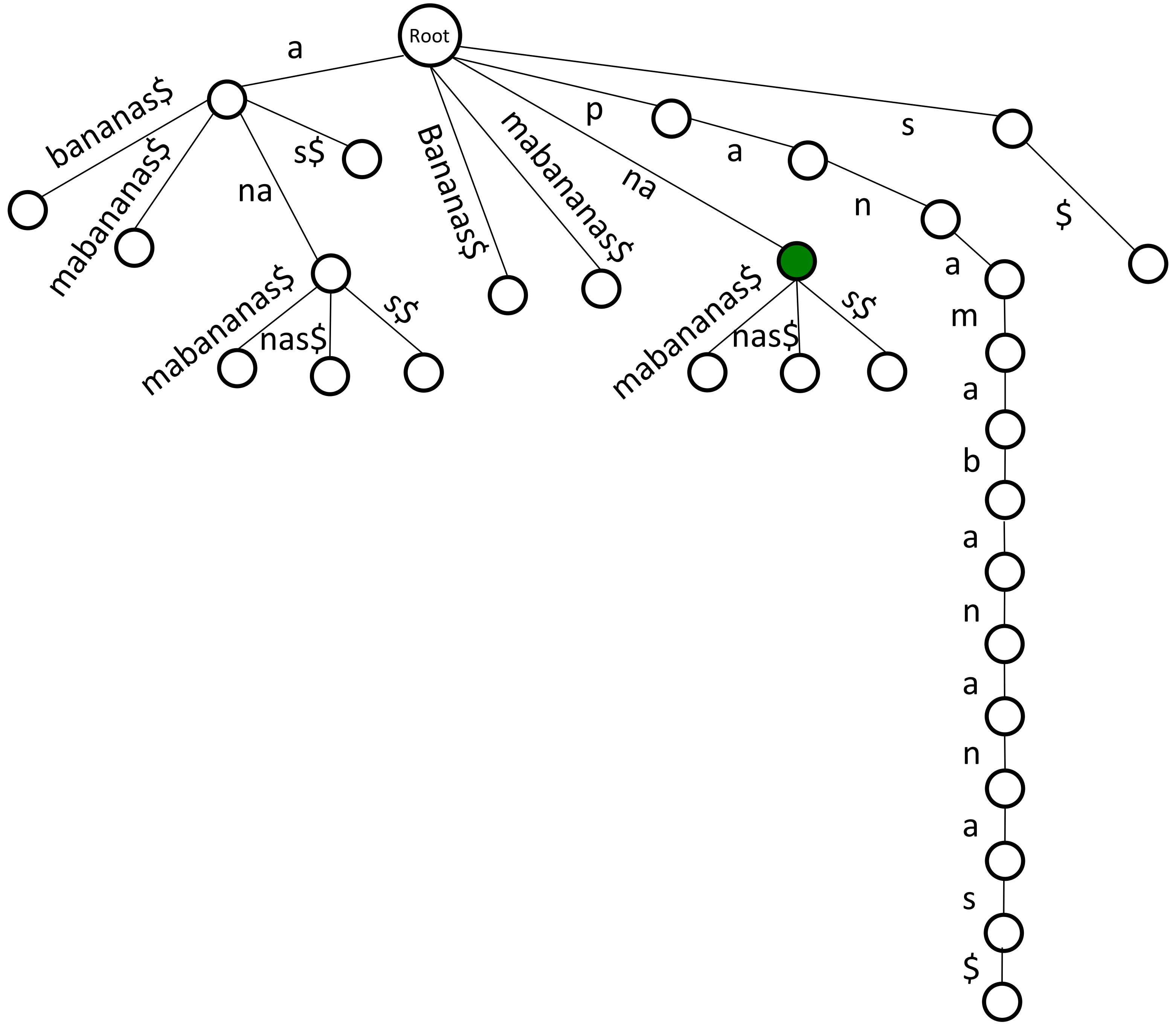


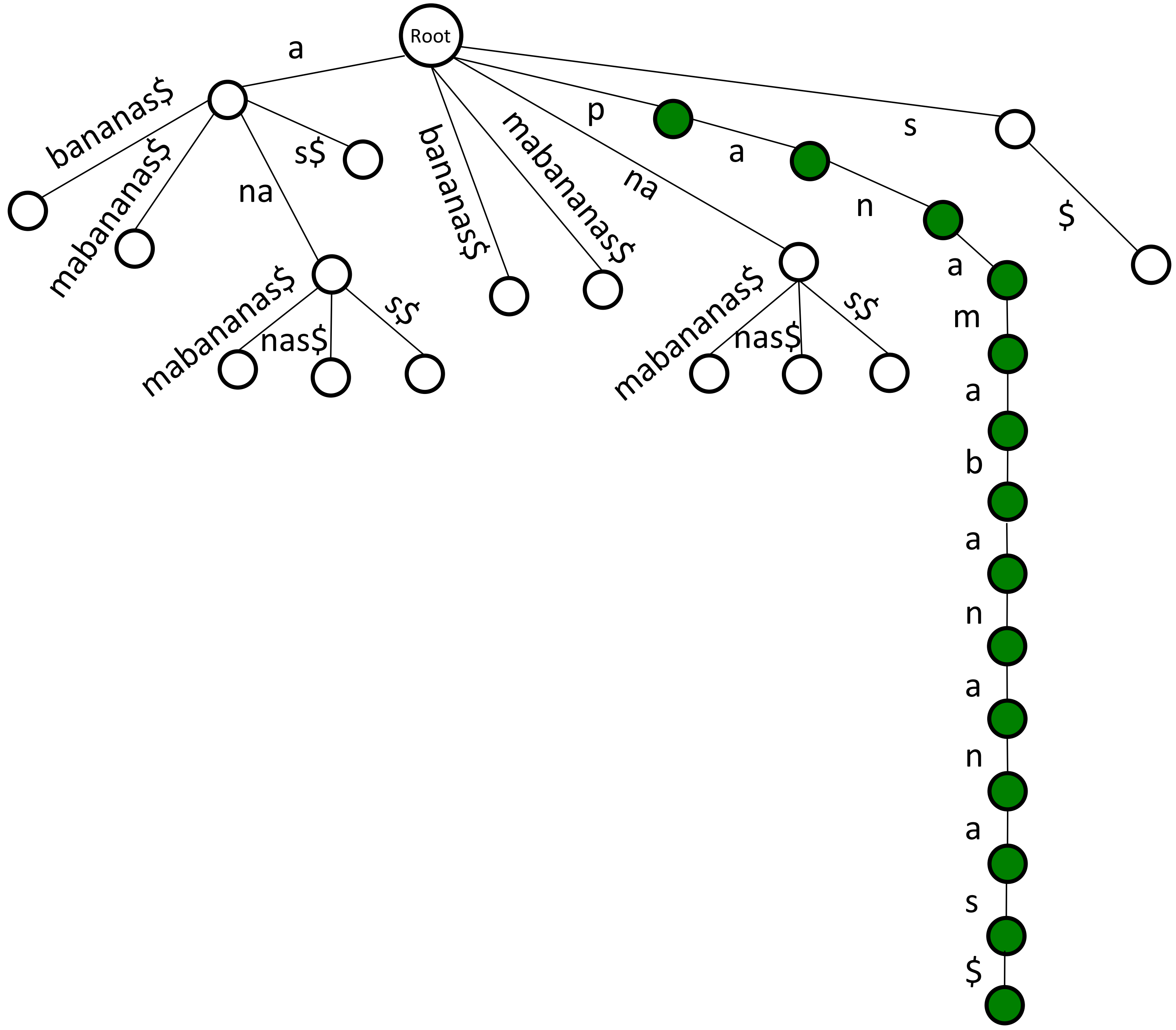
Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$



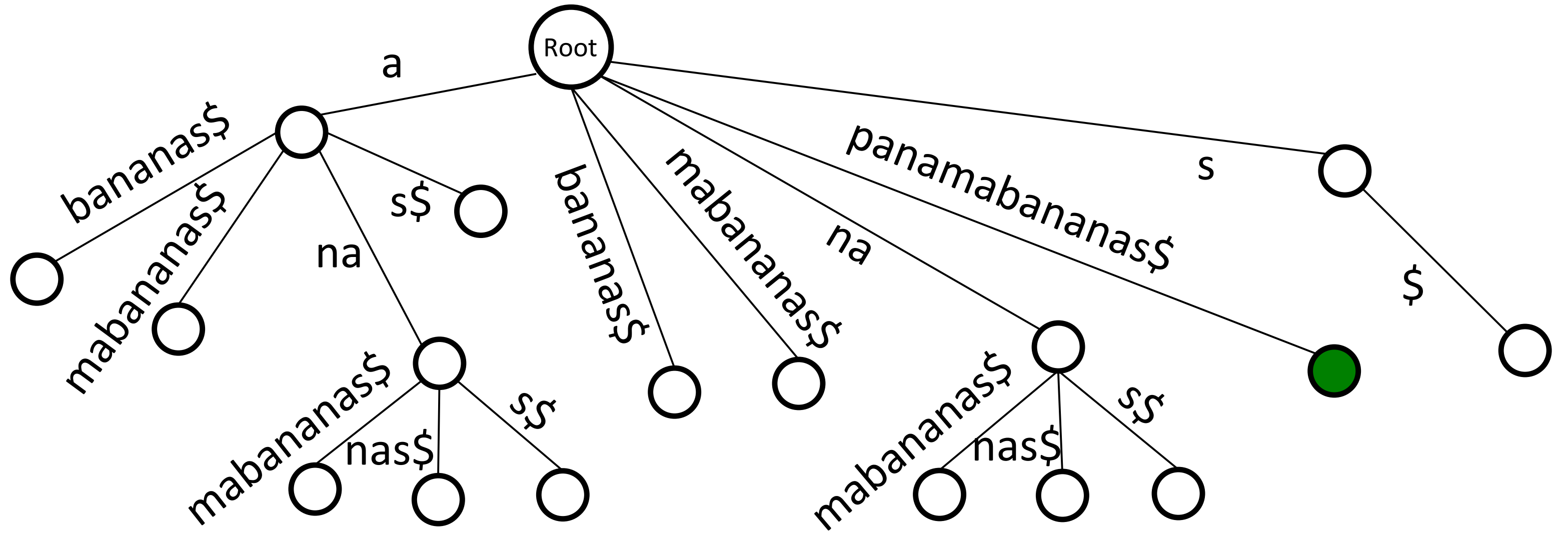
Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

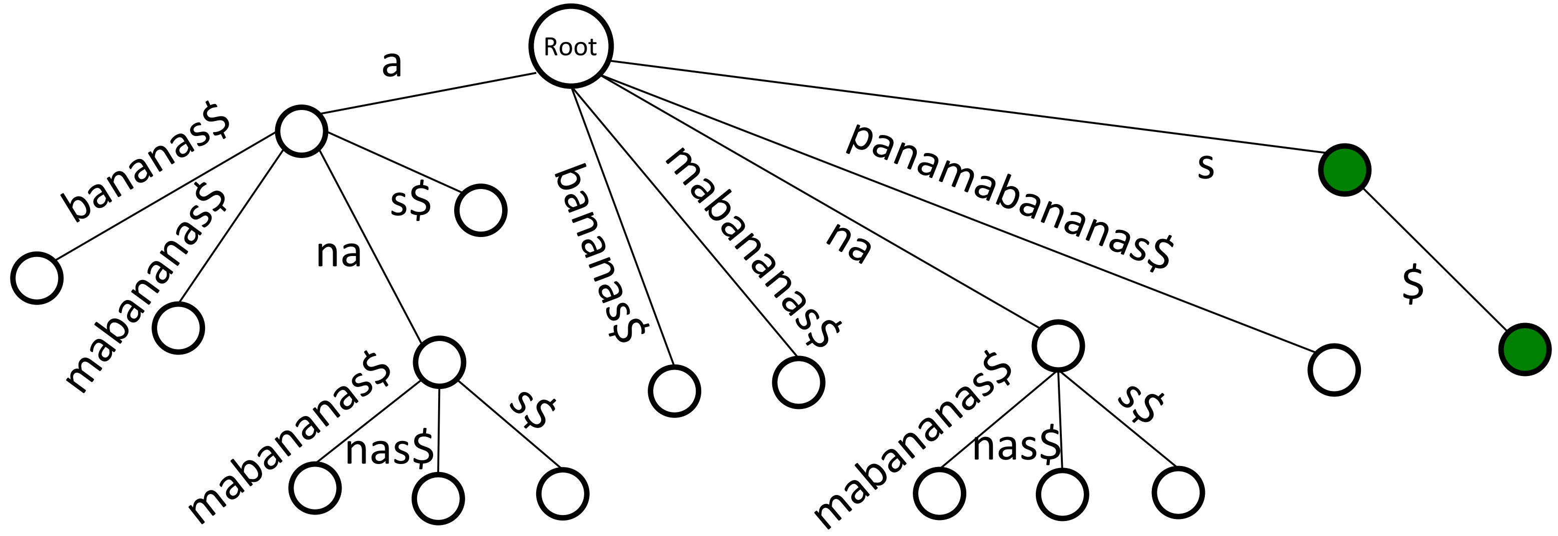

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

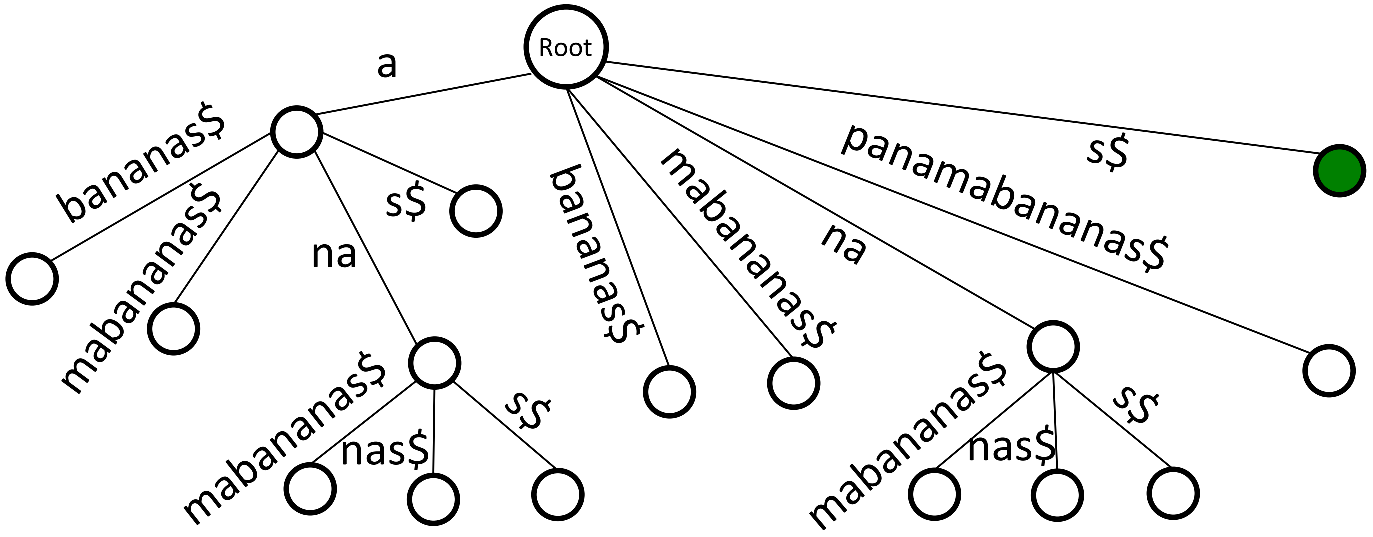

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
Bananas$
mabananas$
na
mabananas$
nas$
s$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$



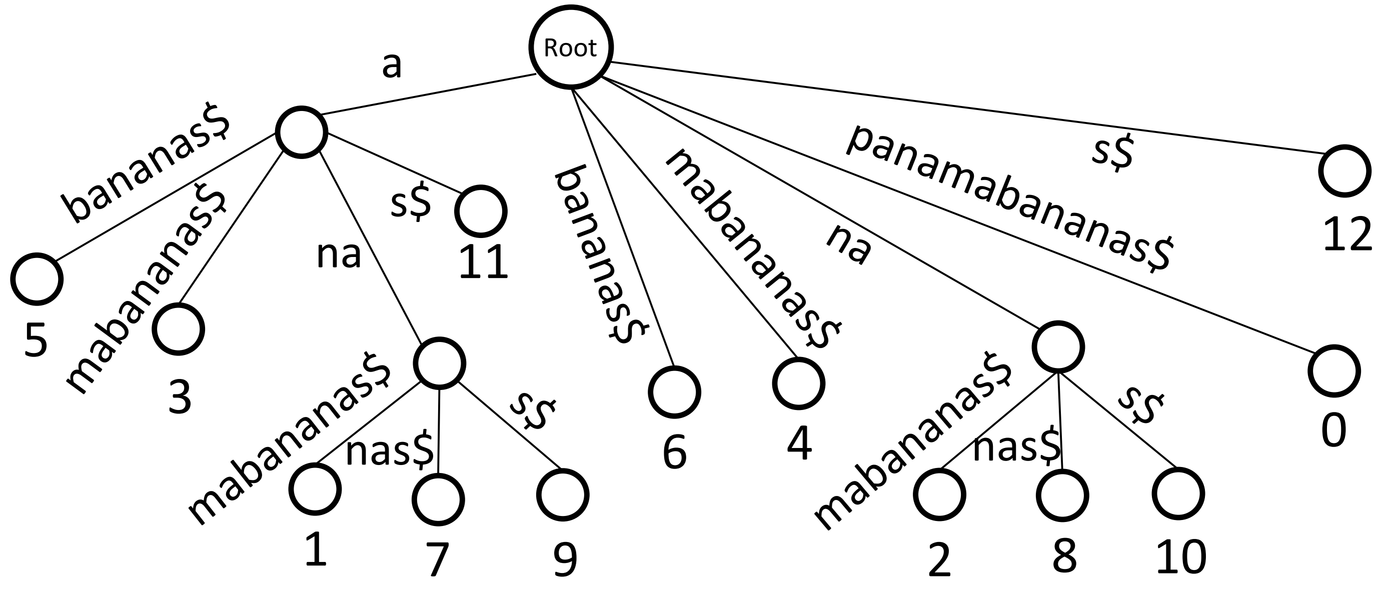
Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
na
mabananas$
nas$
s$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$


Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
na
mabananas$
nas$
s$
panamabananas$
s
$


Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
na
mabananas$
nas$
s$
panamabananas$
s
$

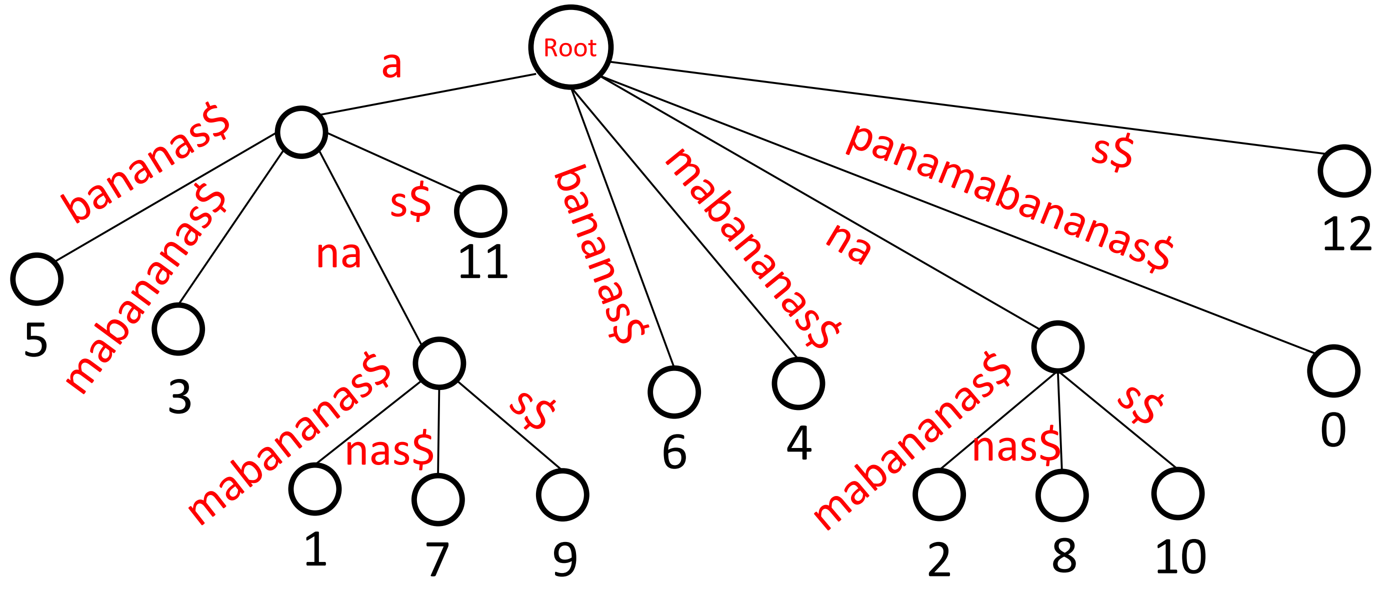


SuffixTree(*Text*)



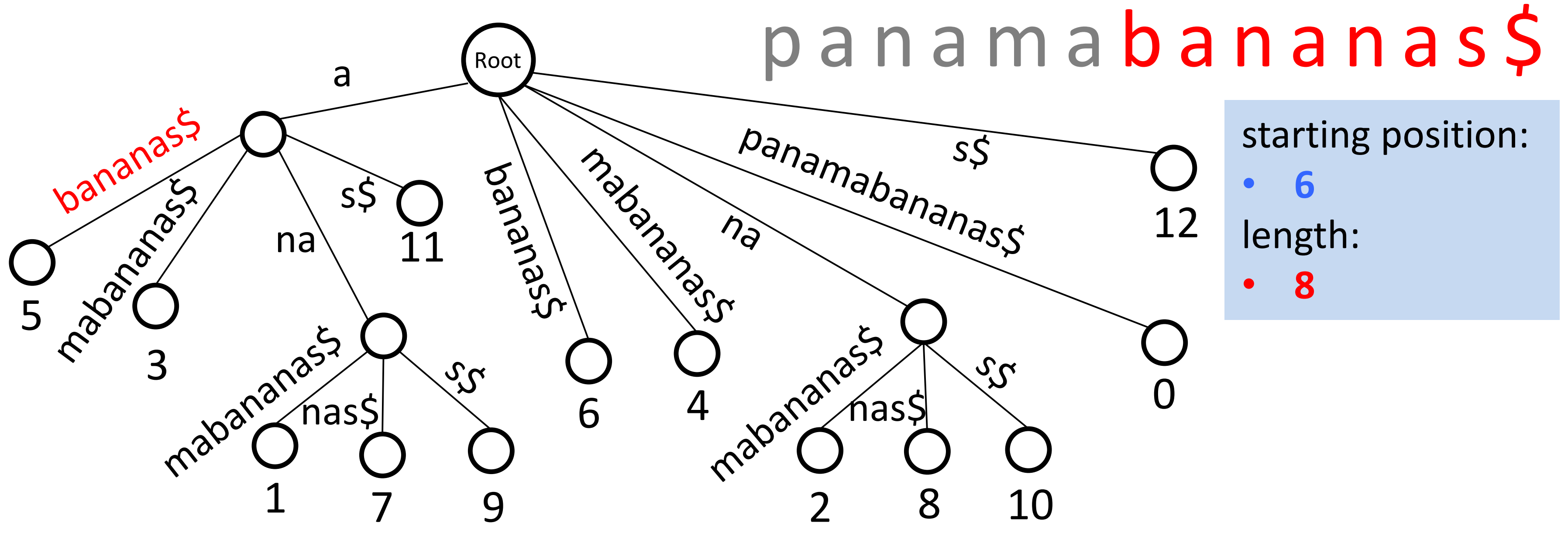
Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)



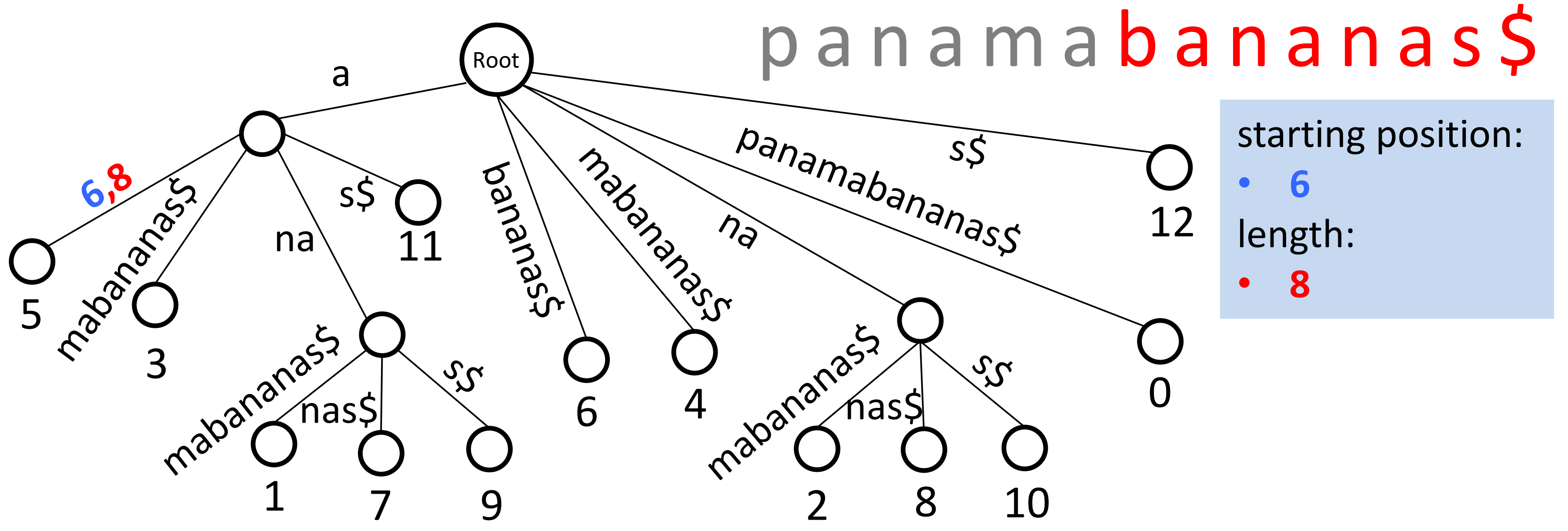
Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- **Cheating!!!** - how do we store all edge labels?



Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- **Cheating!!!** - how do we store all edge labels?

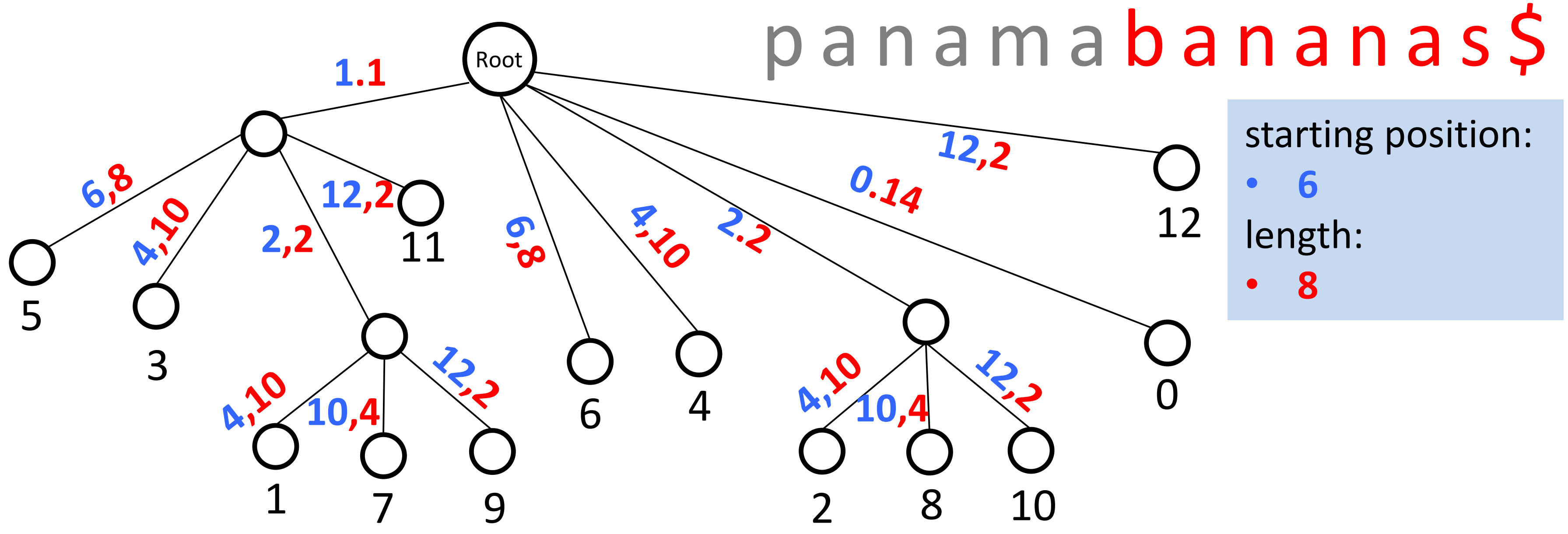


starting position:
- 6

length:
- 8

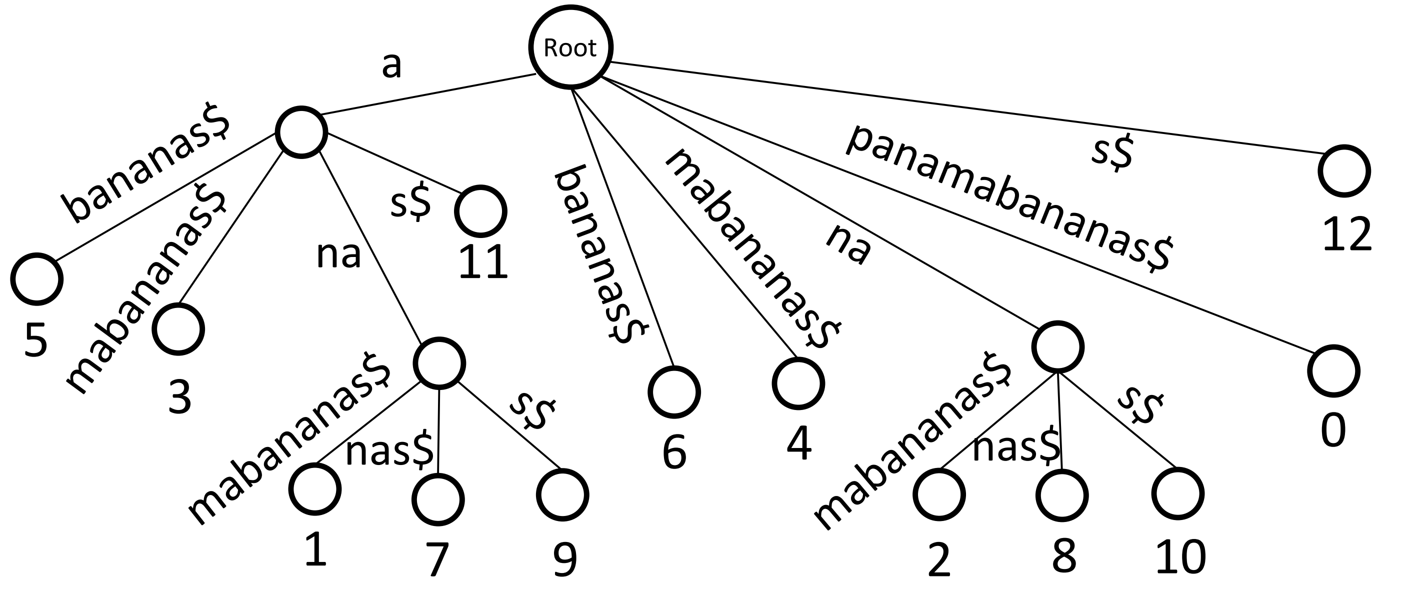
Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- storing edge labels



Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- storing edge labels



Why did we bother to add “**$**” to “panamabananas”?

- to make sure that each suffix corresponds to a leaf

Why do we want to make sure that each suffix correspond to a leaf?

- construct suffix tree for “papa”(without adding “$”) and compare it with the suffix tree for “papa$”

Constructing Suffix Tree: **Naive Approach**

- **Quadratic** runtime:
  - $O(|\textit{Text}|^2)$

$O(|\textit{Genome}| + |\textit{Patterns}|)$ to find pattern matches

Constructing Suffix Tree: **Linear-Time Algorithm**

- **Linear** runtime (for a constant-size alphabet):
  - O(|*Text*|)

Linear-time algorithm (Weiner, 1973) was simplified by Ukkonen (1995) but it is still too complex to cover in this course

Big Secret of the Big O Notation

- Suffix trees enable fast **Exact** Multiple Pattern Matching:
  - Runtime: O(|*Text*| + |*Patterns*|)
  - Memory: O(|*Text*|)
- However, big-O notation hides constants!
  - suffix tree algorithms has large memory footprint ~ 20•|*Text*| for long texts like human genome
- We want to find mutations!
  - it is unclear how to develop fast **Approximate** Multiple Pattern Matching using suffix trees