Alignment of Short Reads: Suffix Trees

Ahmet Sacan

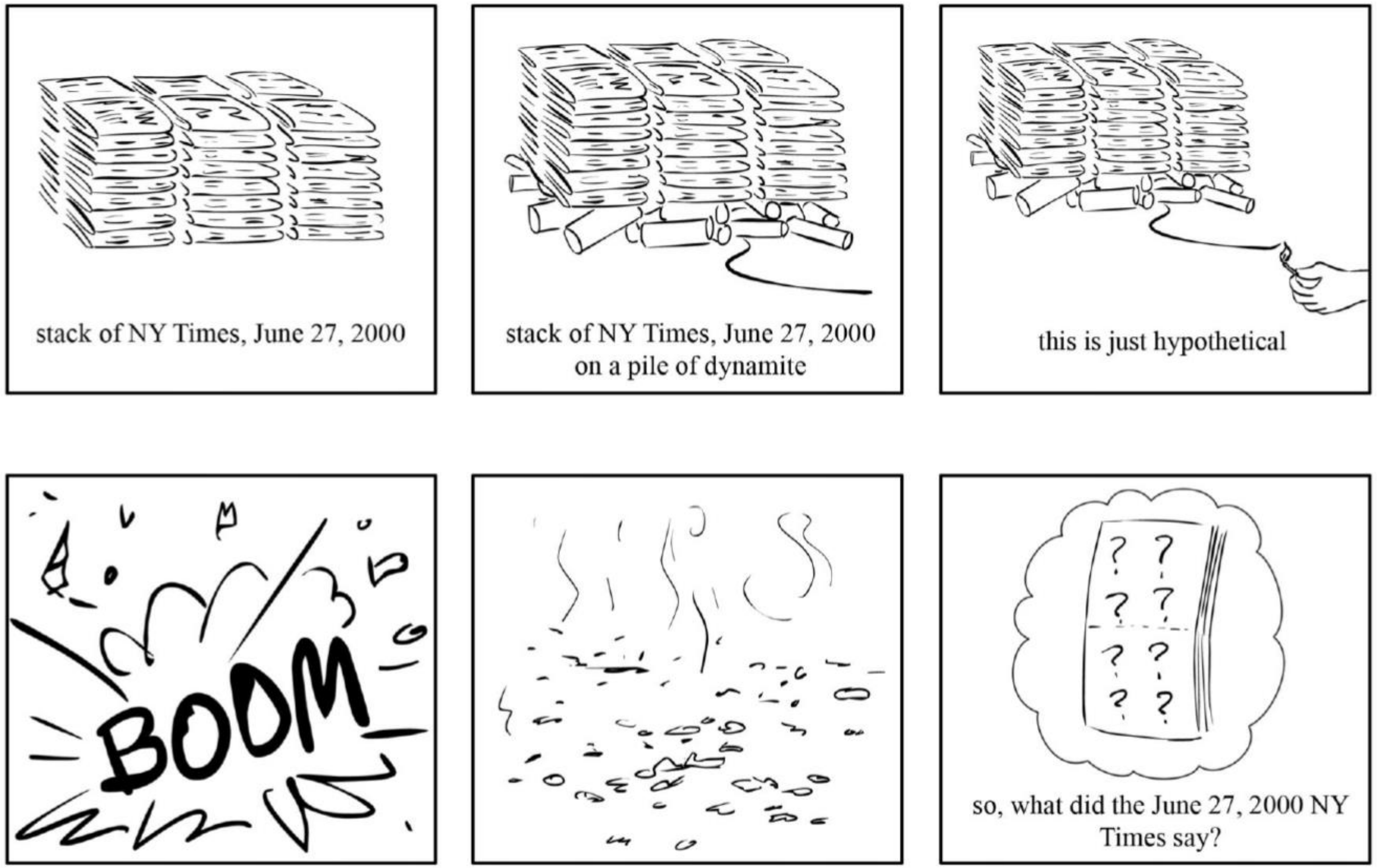
Slides adapted from:

https://www.coursera.org/learn/algorithms-on-strings

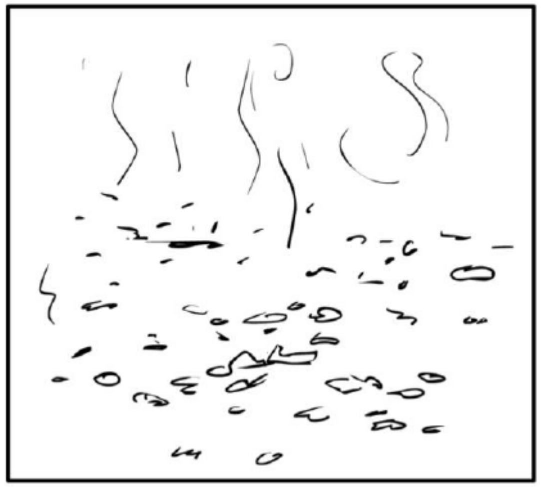
https://ocw.mit.edu/courses/biology/7-91j-foundations-of-computational-and-systems-biology-spring-2014/video-lectures/lecture-5-library-complexity-and-short-read-alignment-mapping/

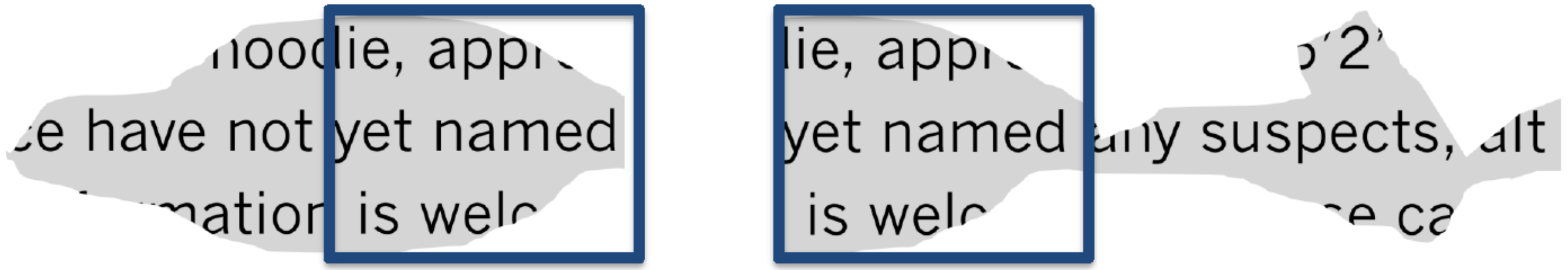
http://www.cs.utsa.edu/~jruan/teaching/cs5263_spring_2015/slides/slide5_shortreadmapping.ppt

The Newspaper Problem

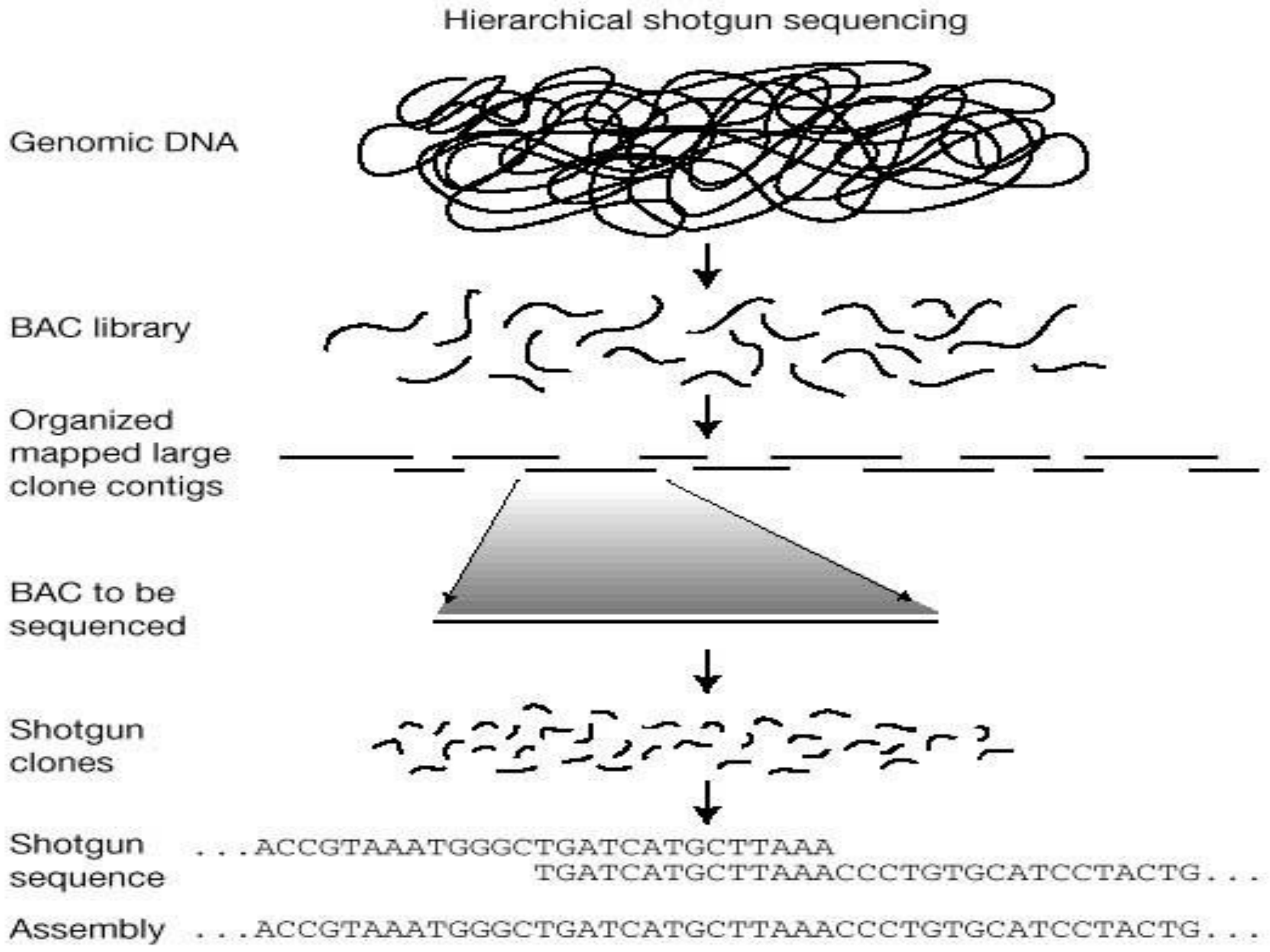

stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM
so, what did the June 27, 2000 NY Times say?

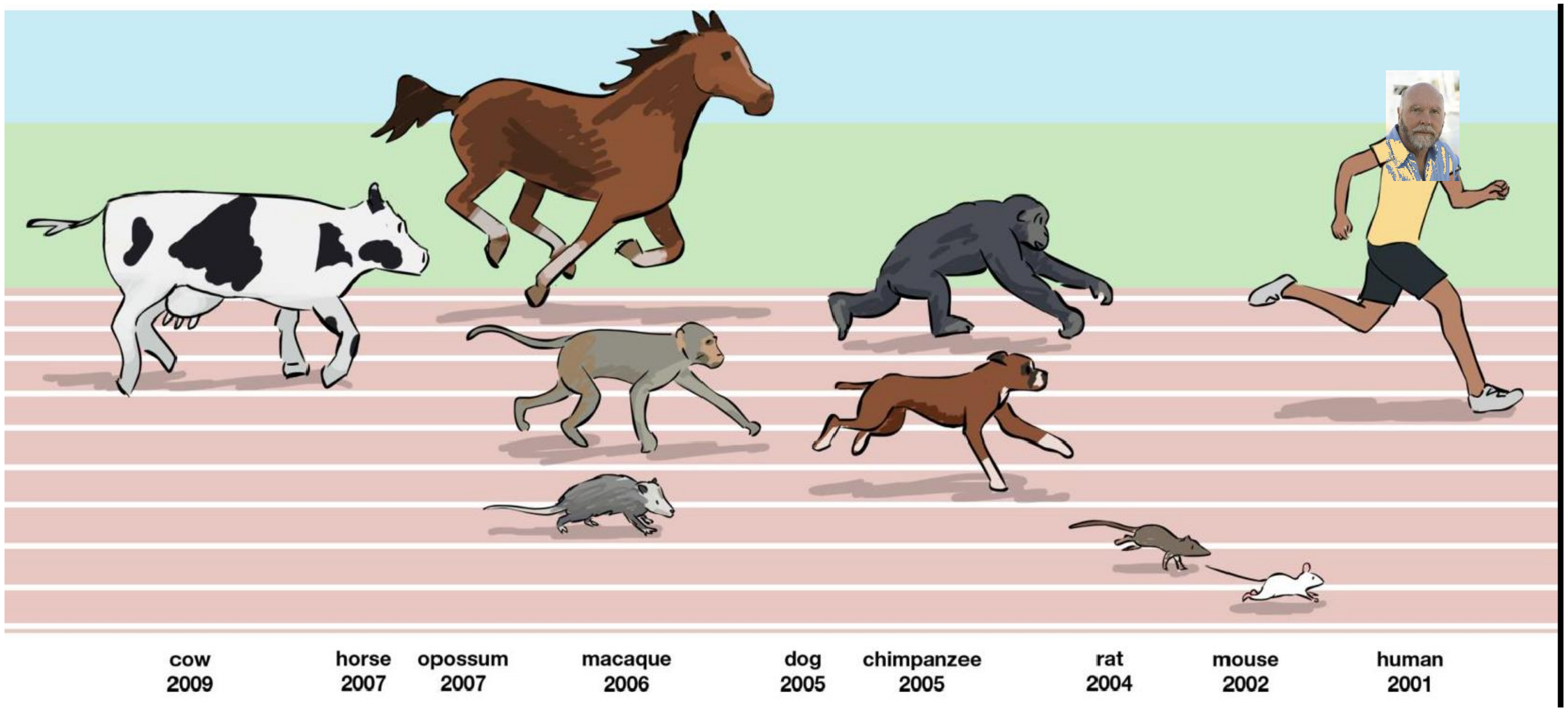
The Newspaper Problem as an Overlapping Puzzle





Whole-Genome Sequencing


Hierarchical shotgun sequencing
Genomic DNA
BAC library
Organized mapped large clone contigs
BAC to be sequenced
Shotgun clones
Shotgun sequence
...ACCGTAAATGGGCTGATCATGCTTAAA
TGATCATGCTTAAACCCTGTGCATCCTACTG...
Assembly
...ACCGTAAATGGGCTGATCATGCTTAAACCCTGTGCATCCTACTG...


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

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
```

Reference-based sequencing

- Map short reads to the reference genome.
- Exact "pattern matching"
  - Locate the occurrence of short read exactly.
- Approximate "pattern matching"
  - Allow mismatches, insertions, deletions.

Searching short read (pattern) in genome (text)

Pattern drives along *Text*

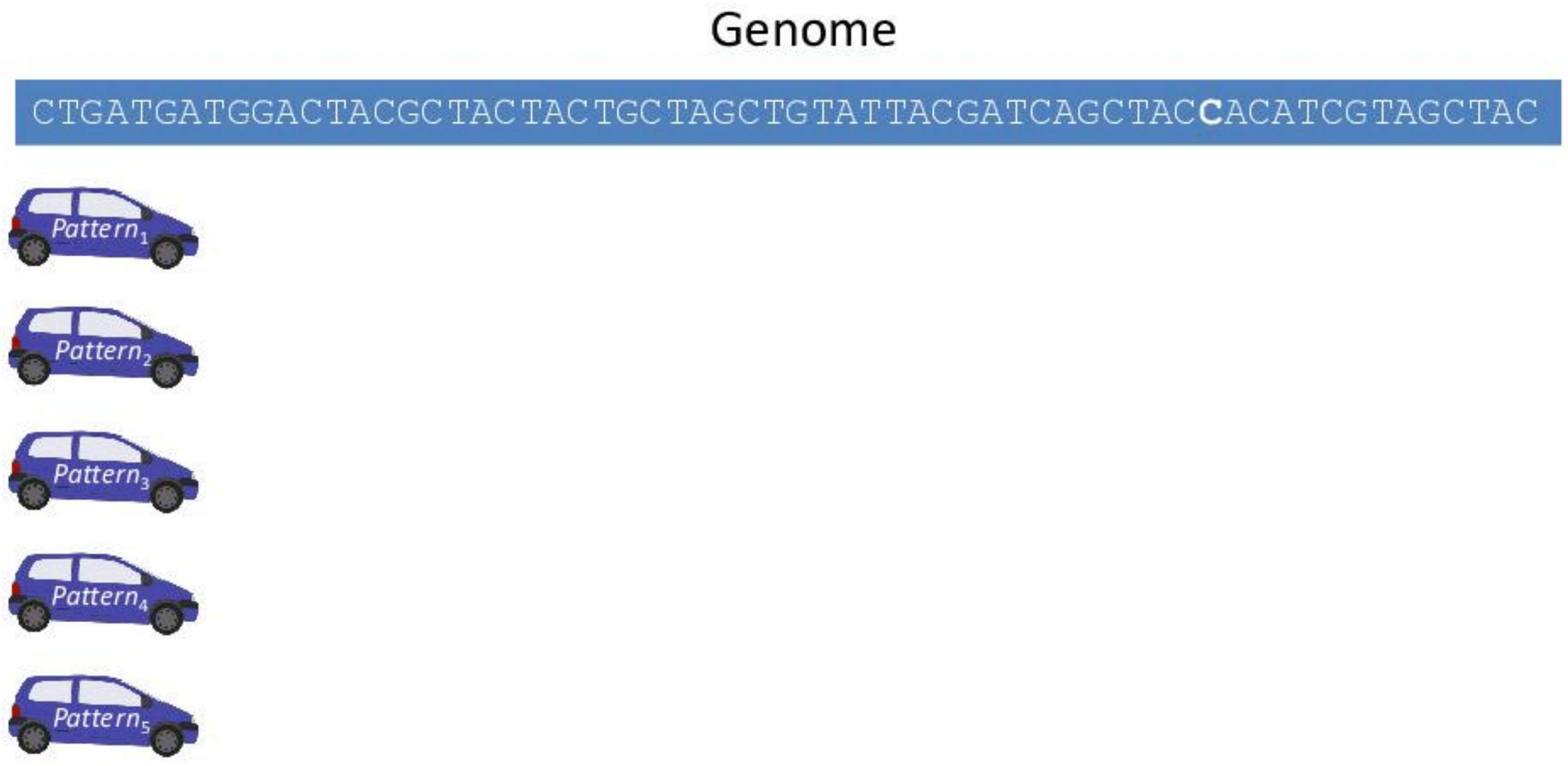
```
panamabananas      Text
nana               Pattern
```

Searching short read (pattern) in genome (text)

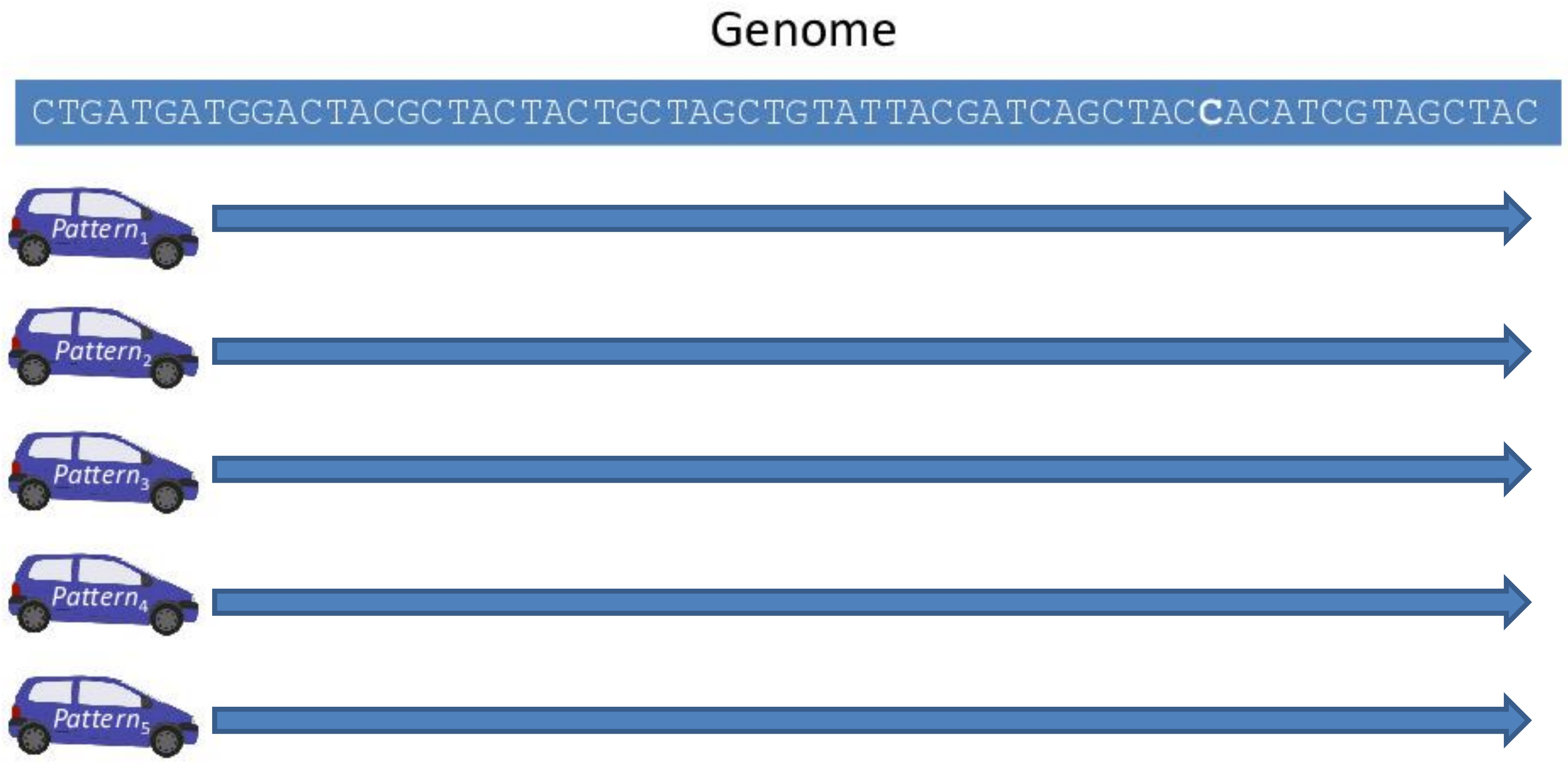
Pattern drives along *Text*

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

Searching for multiple patterns


Genome
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTAC
Pattern1
Pattern2
Pattern3
Pattern4
Pattern5

Searching for multiple patterns


Genome
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTAC
Pattern$_1$
Pattern$_2$
Pattern$_3$
Pattern$_4$
Pattern$_5$

Computational Complexity

- How long does it take to find a single pattern?

  O(|text| * |pattern|)
- Multiple patterns?

  O(|pattern1| * |text|) + O(|pattern2| * |text|) + ...

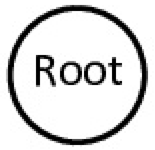
  = O(|text| * |pattern**s**|)
- Human genome:
  - |Text| ≈ 10^9
  - |Patterns| ≈ 10^12

Pack Patterns onto a bus

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC

Patterns

banana
pan
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

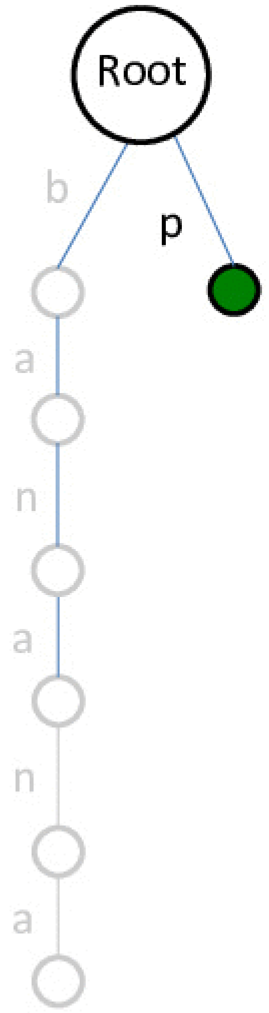
nab

antenna

bandana

ananas

nana



Patterns

banana

pan

and

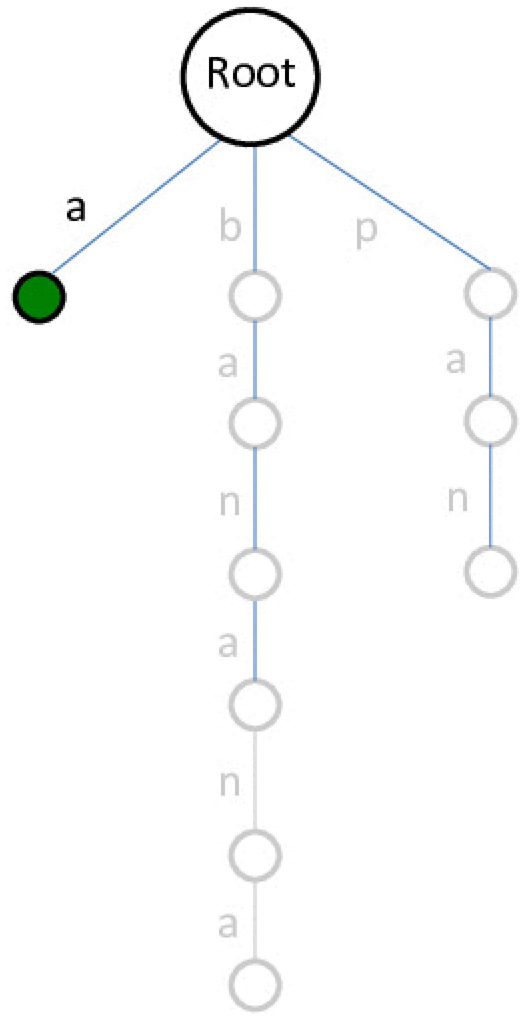
nab

antenna

bandana

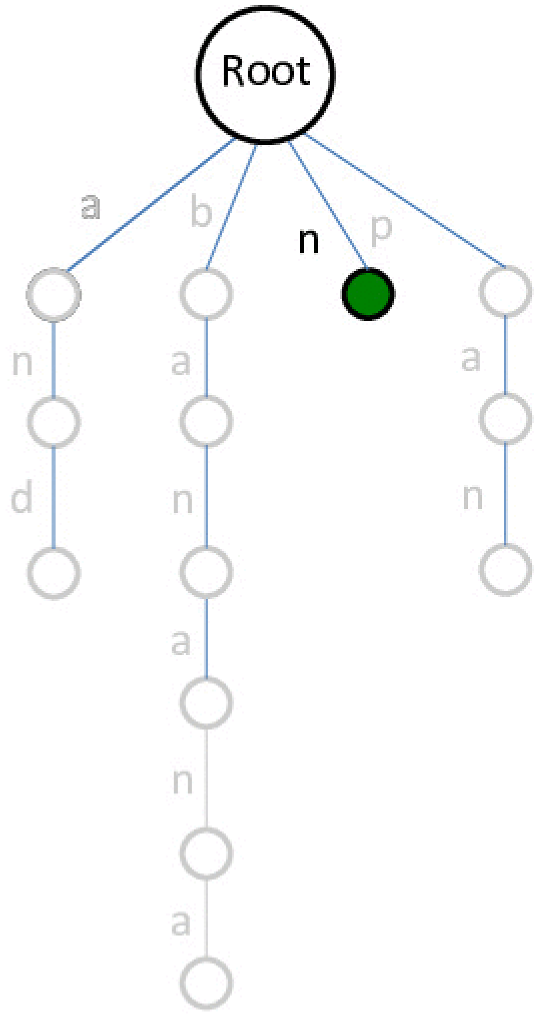
ananas

nana



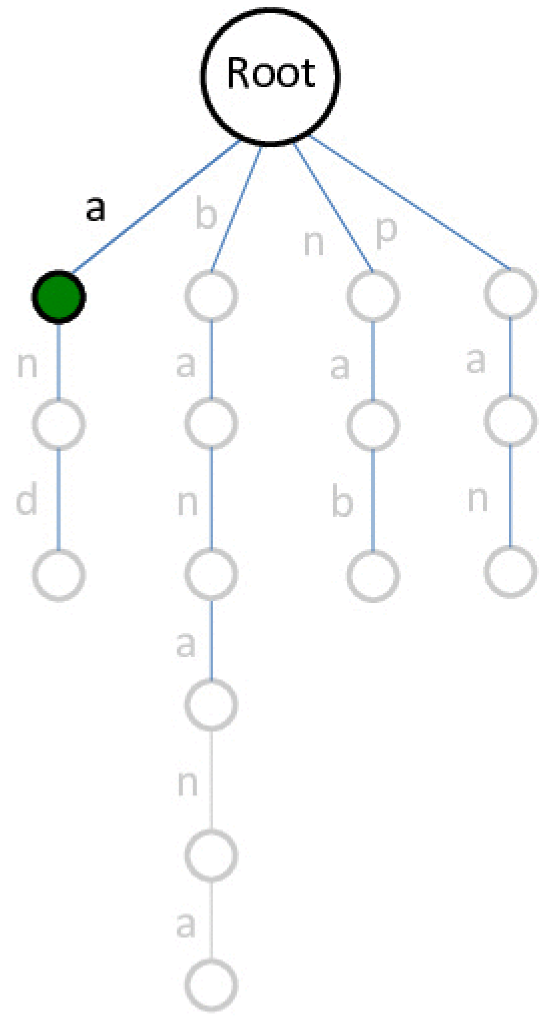
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



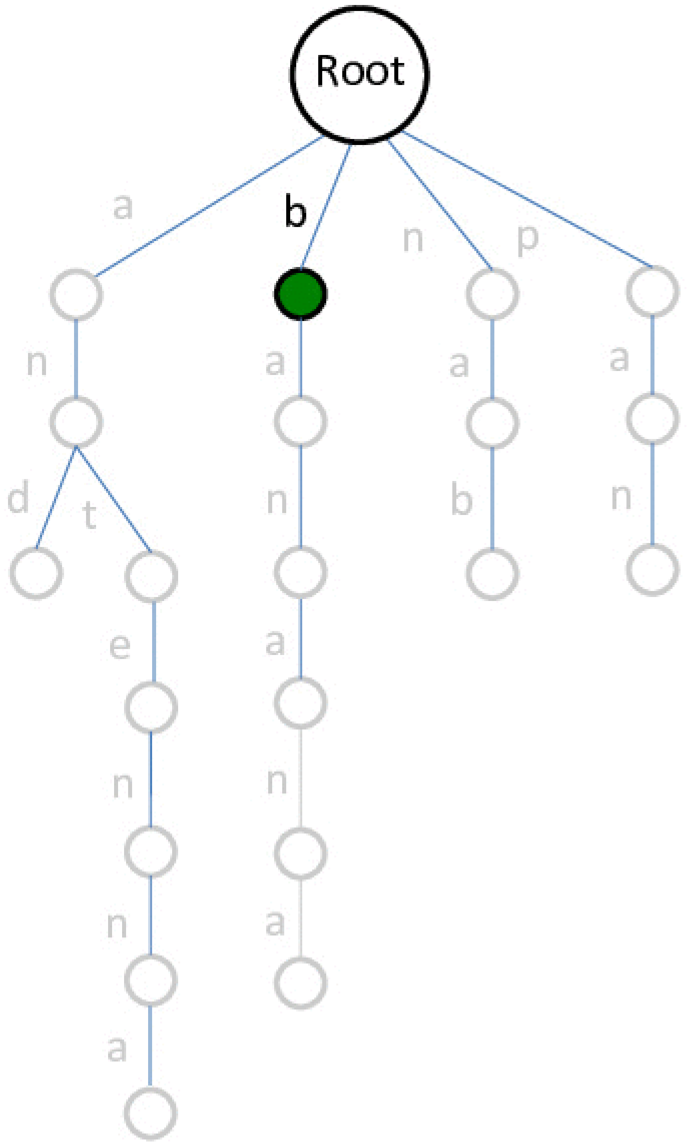
Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana



Patterns

banana

pan

and

nab

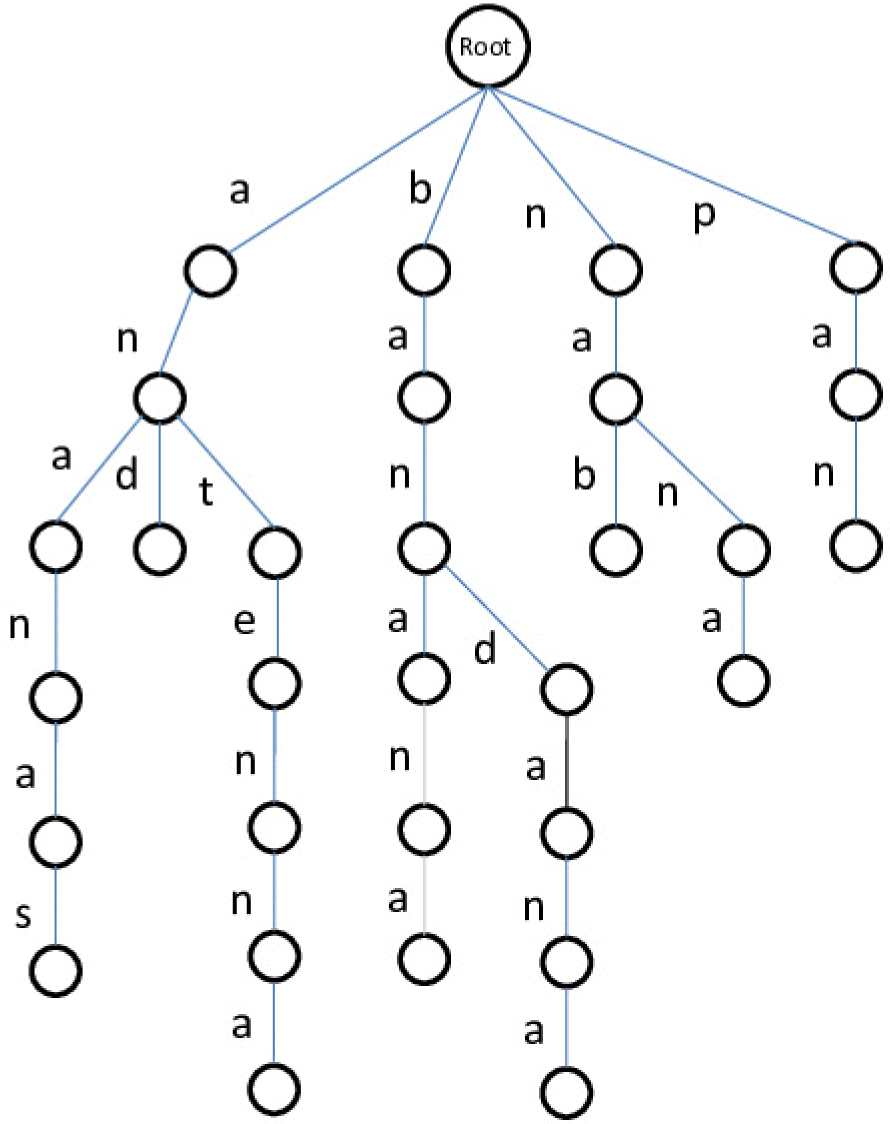
antenna

bandana

ananas

nana

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

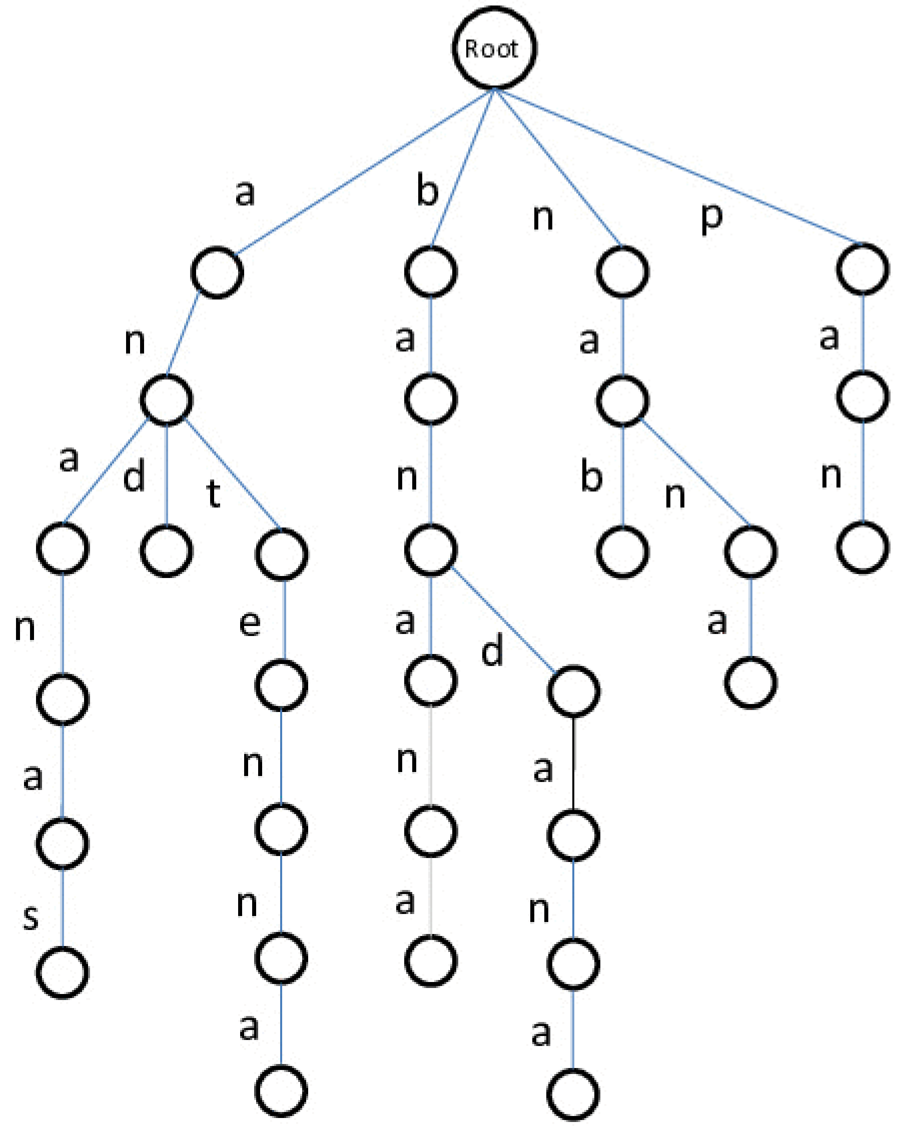


TrieMatching(*Text, Patterns*): drive Trie(*Patterns*) along *Text* at each position of *Text*

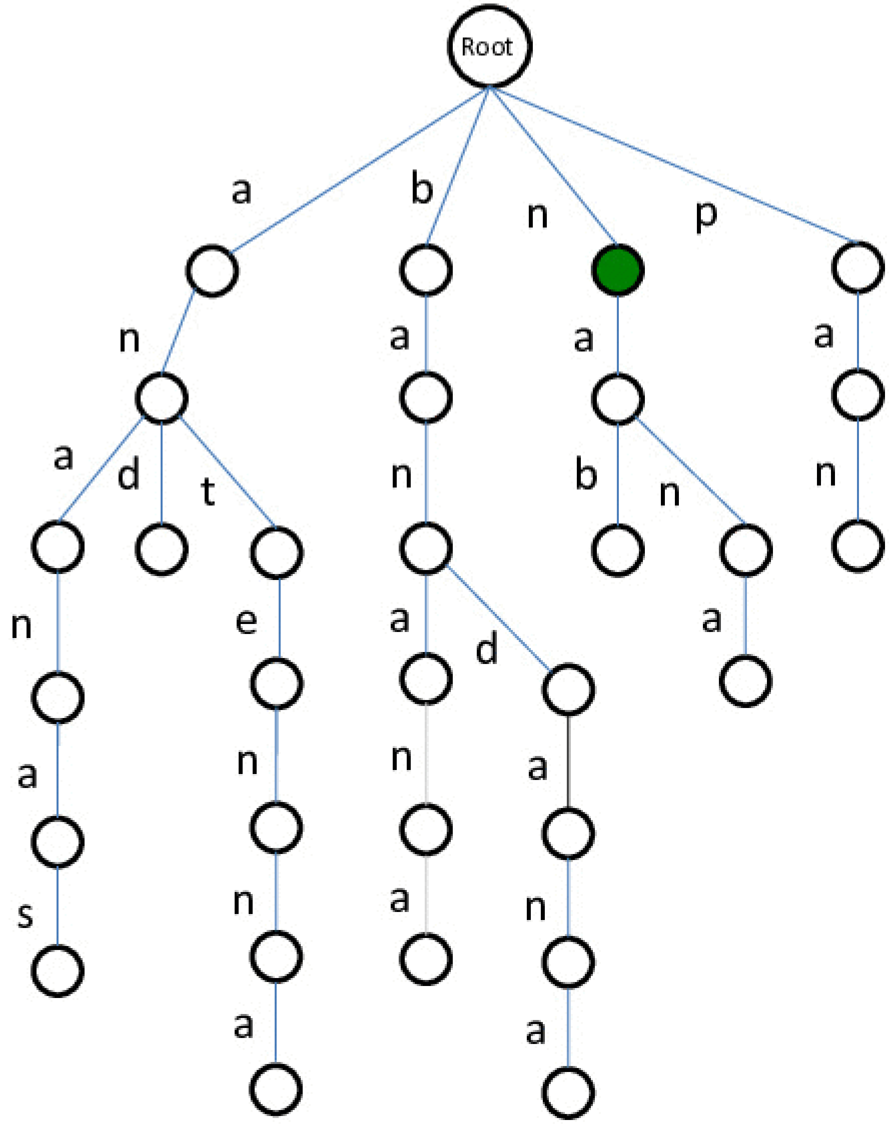
- walk down Trie(*Patterns*) by spelling symbols of *Text*
- a pattern from *Patterns* matches *Text* each time you reach a leaf!

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

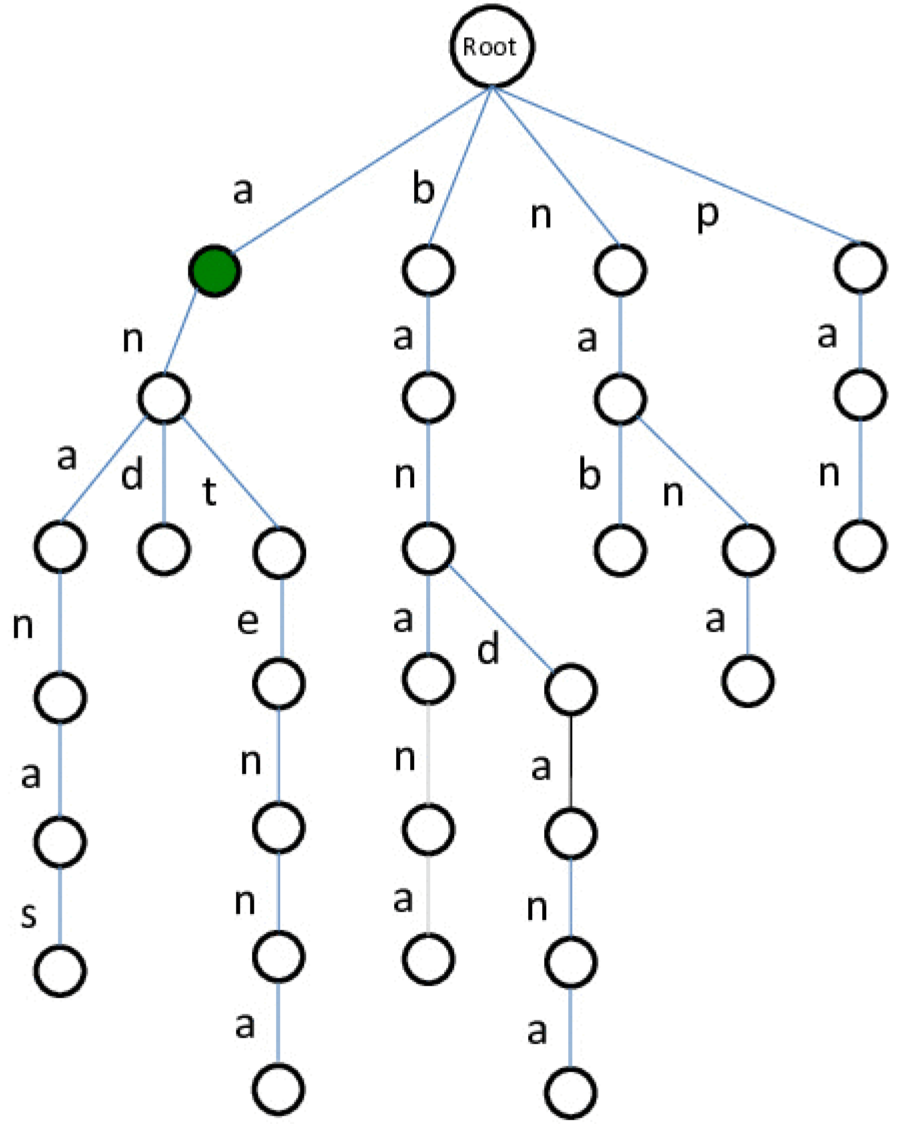
p a n a m a b a n a n a s



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



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



Computational Complexity

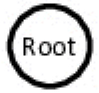
- How long does it take to find multiple patterns?
  - Brute force: O(|text| * |patterns|)

  - Pattern trie: O(|text| * |LongestPattern|)
- Space complexity
  - Number of edges in pattern trie: O(|Patterns|)
  - Human genome: |Patterns| ≈ 10^12

Alternative: Pack *Text* onto a bus

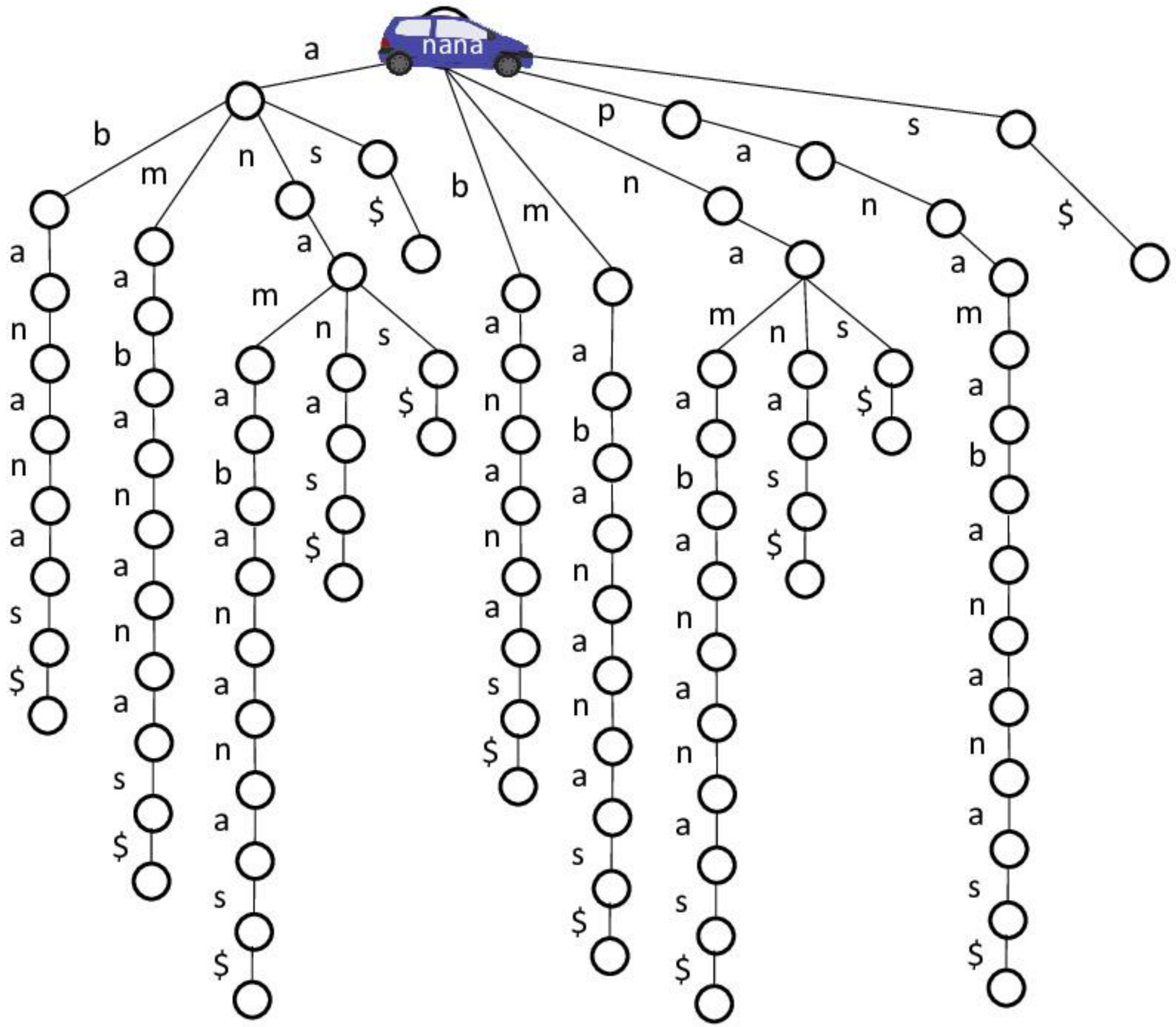
- Generate all suffixes of Text
  - The suffixes represent all possible places a pattern can match
- Form a trie from these suffixes (**suffix trie**)
- For each pattern, walk down from the root of the trie to see if there is a match.

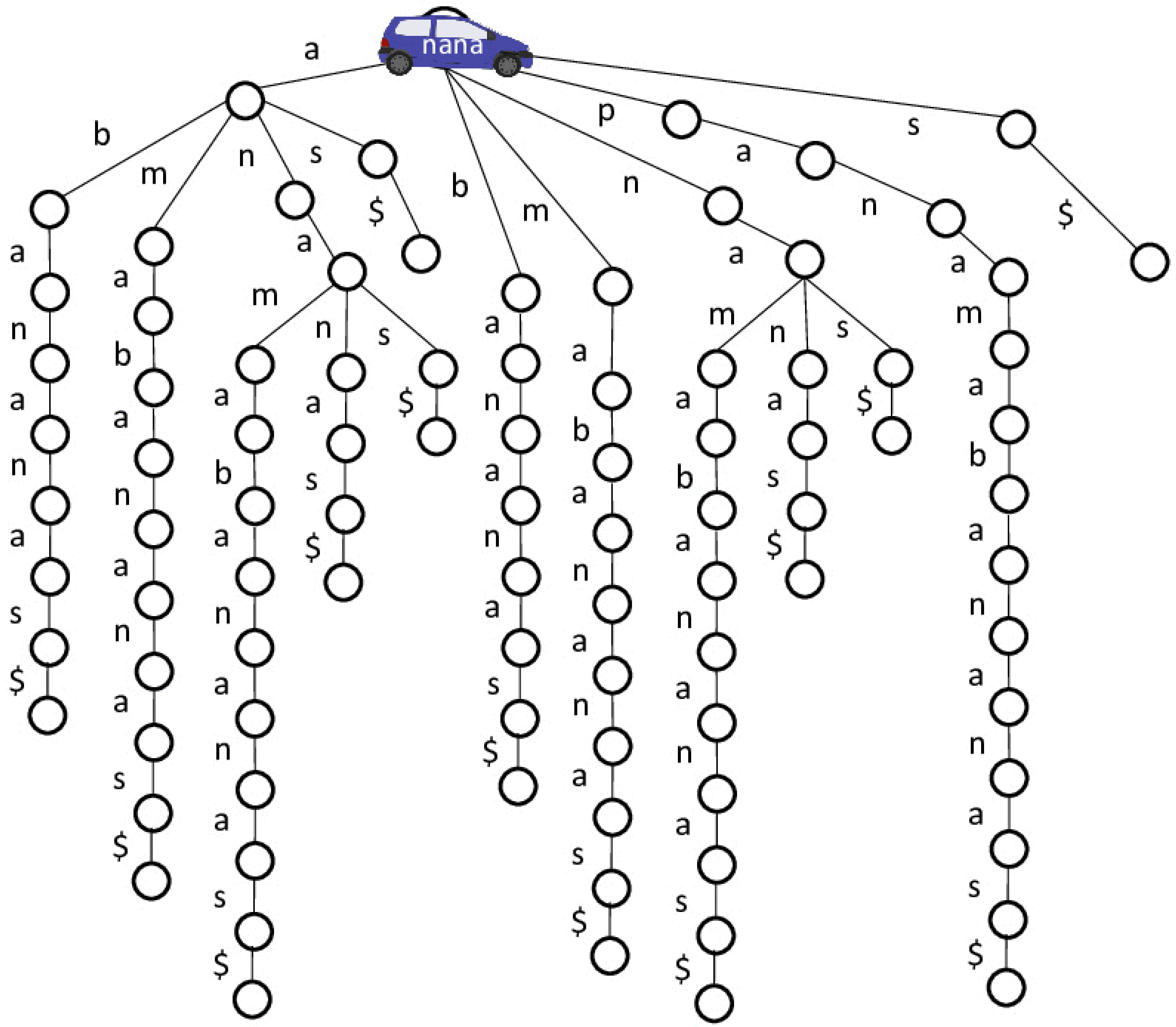
p a n a m a b a n a n a s

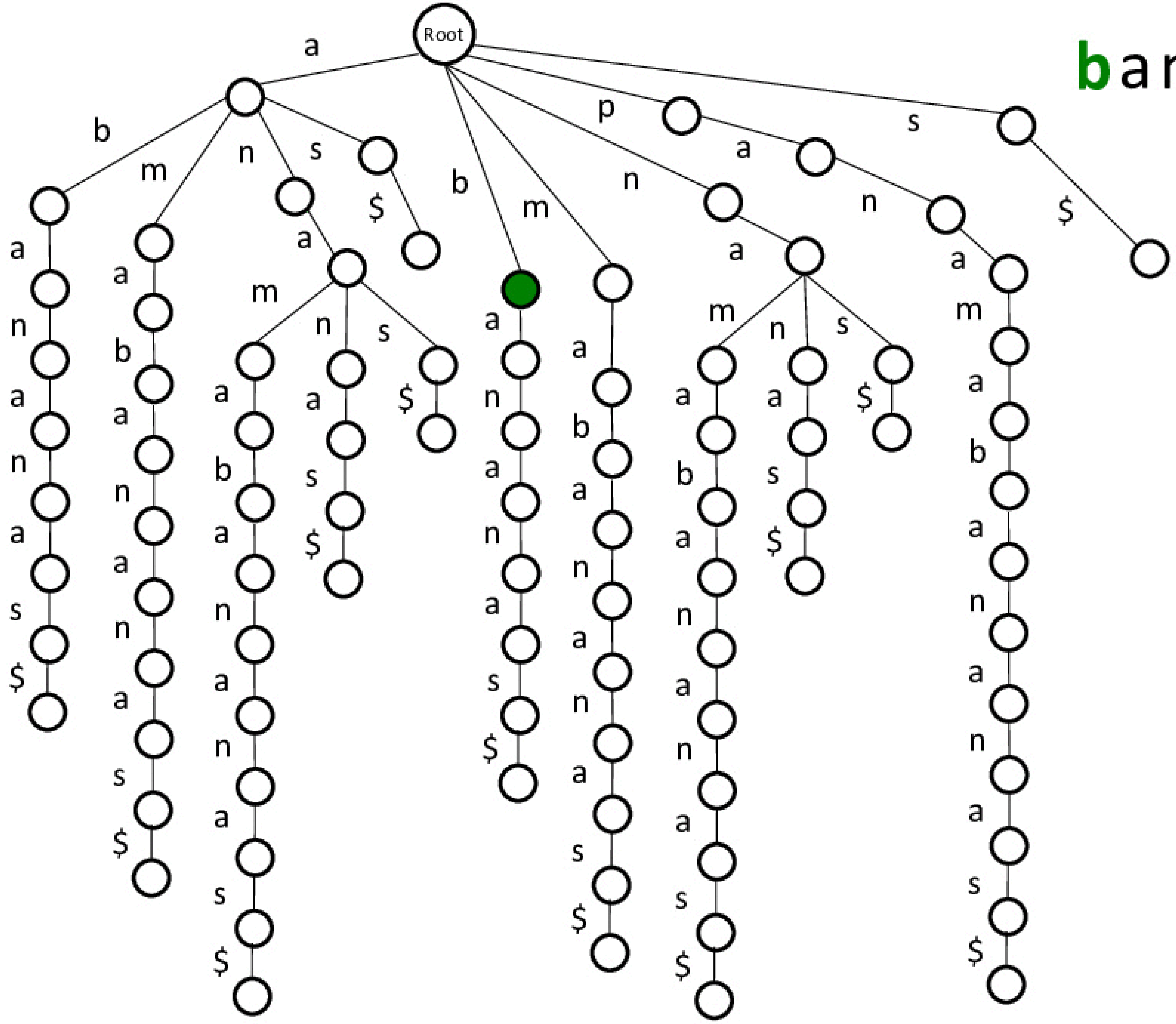


p a n a m a b a n a n a s $

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

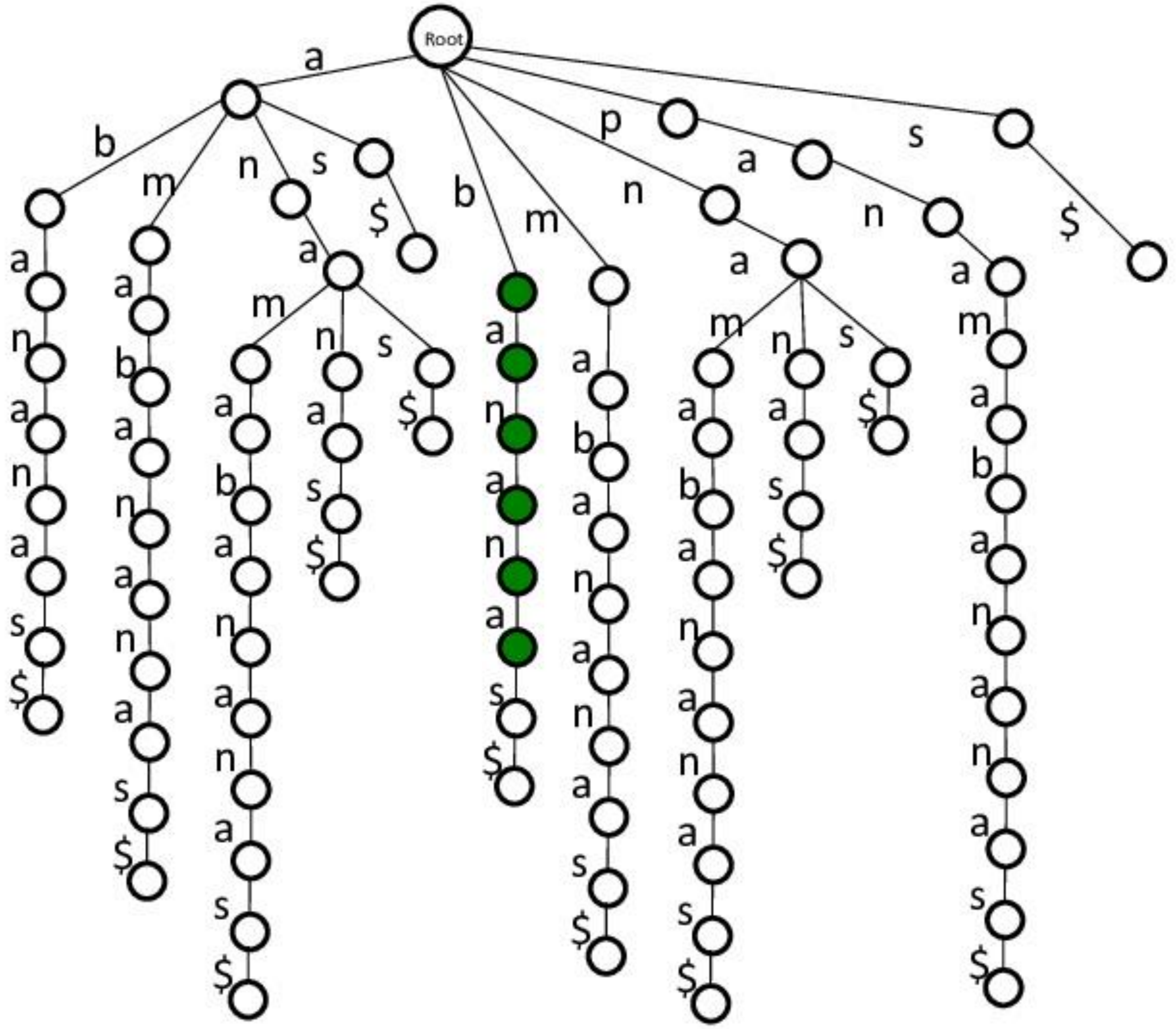

nana
a
b
m
n
s
$
p


nana



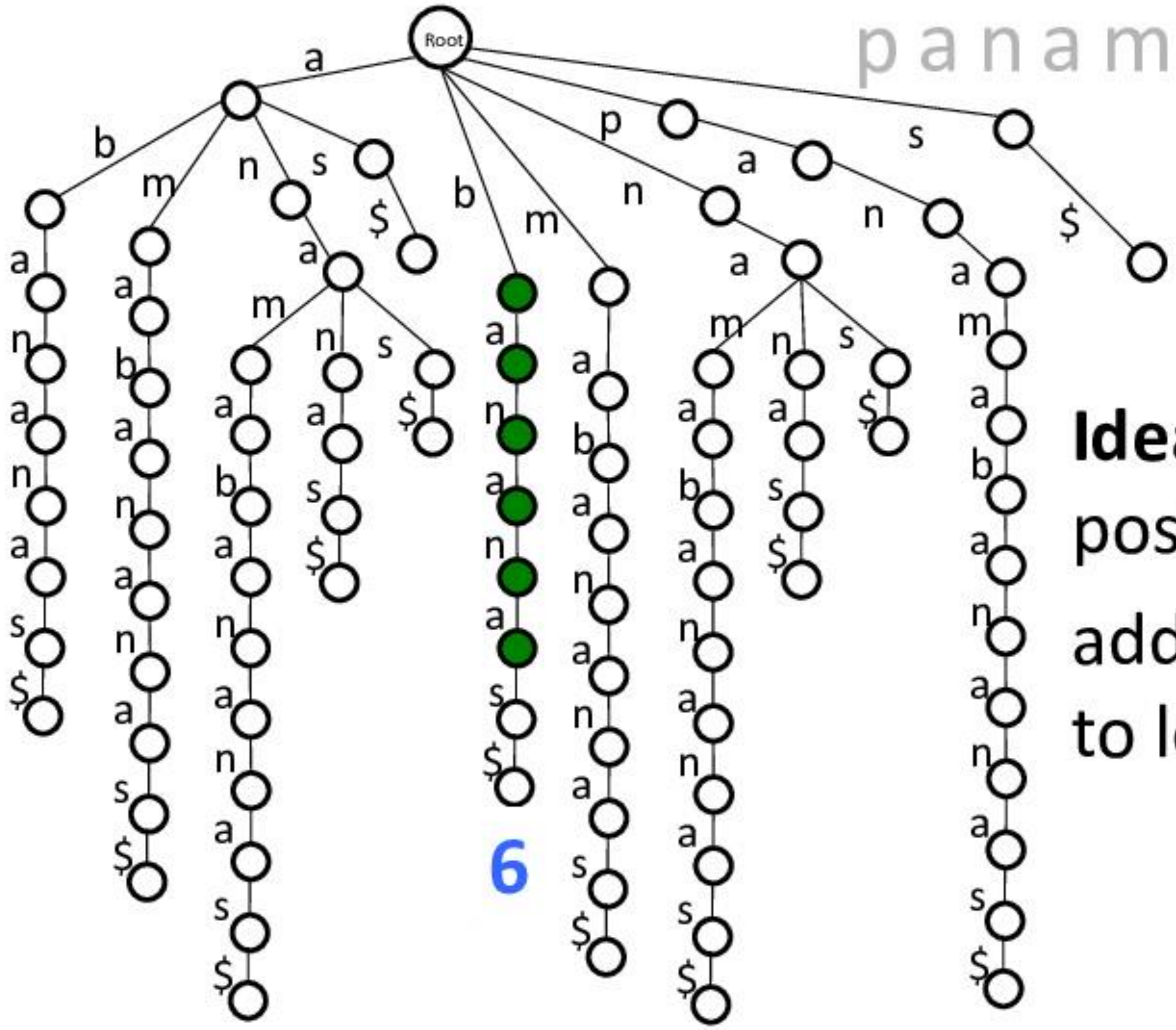
banana

Where Are the Matches???



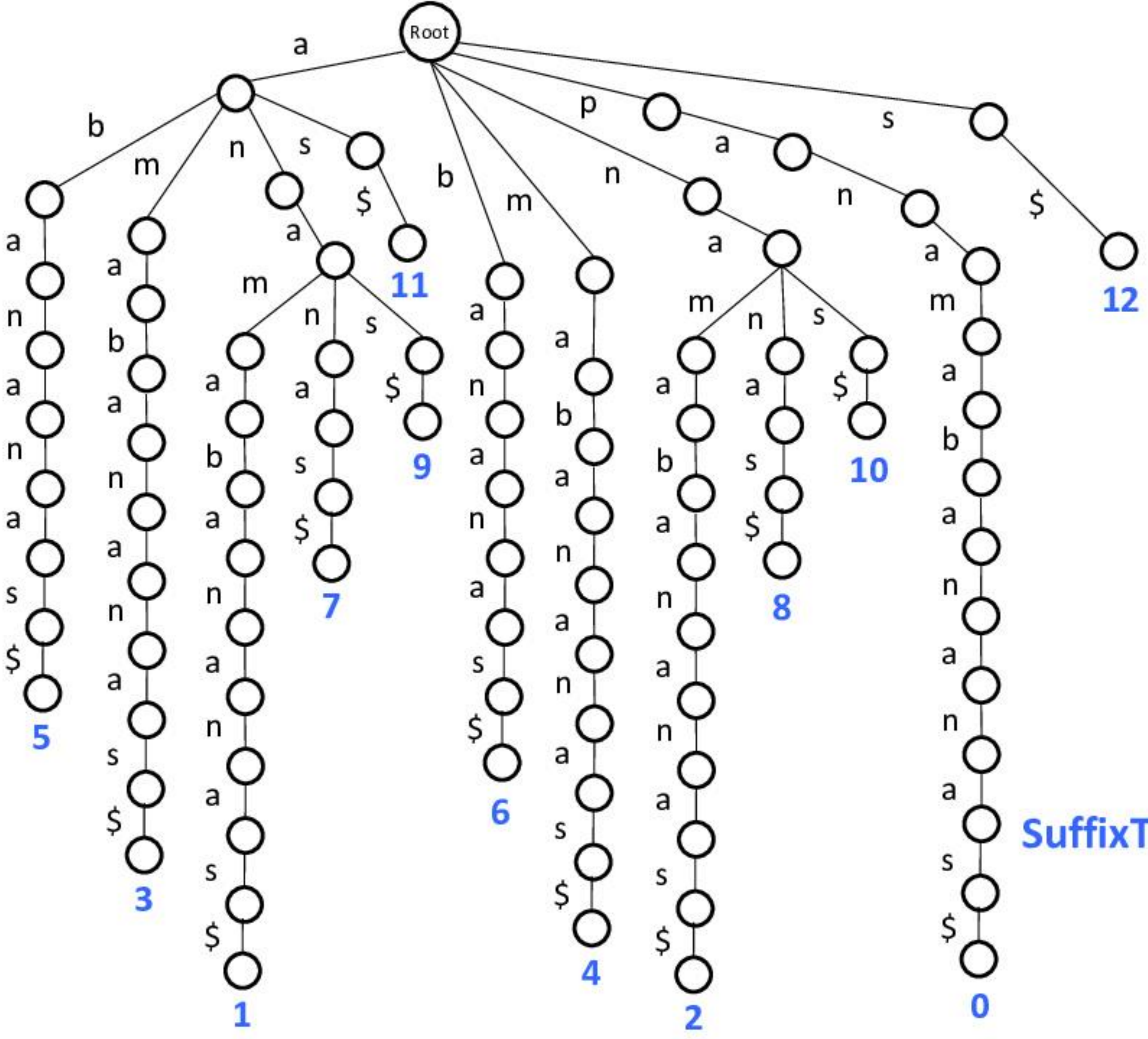
b a n a n a s $

Where Are the Matches???



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

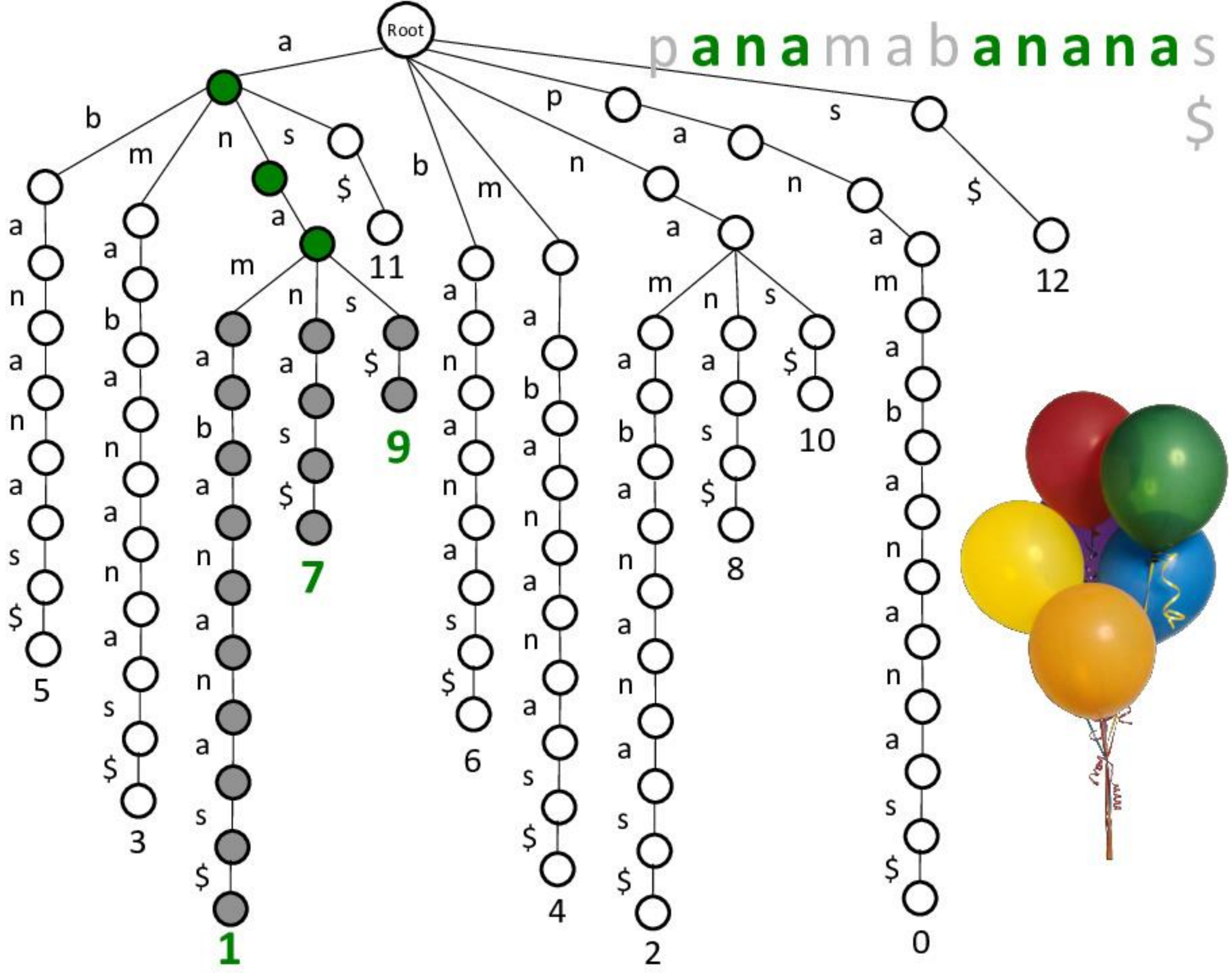
Idea: to find the positions of matches, add some information to leaves



SuffixTrie(*Text*)

Identifying position of match in text

- Once we find a match, walk down to the leaves to get the position(s) of the matches.


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

Memory Footprint of Suffix Trie

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

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

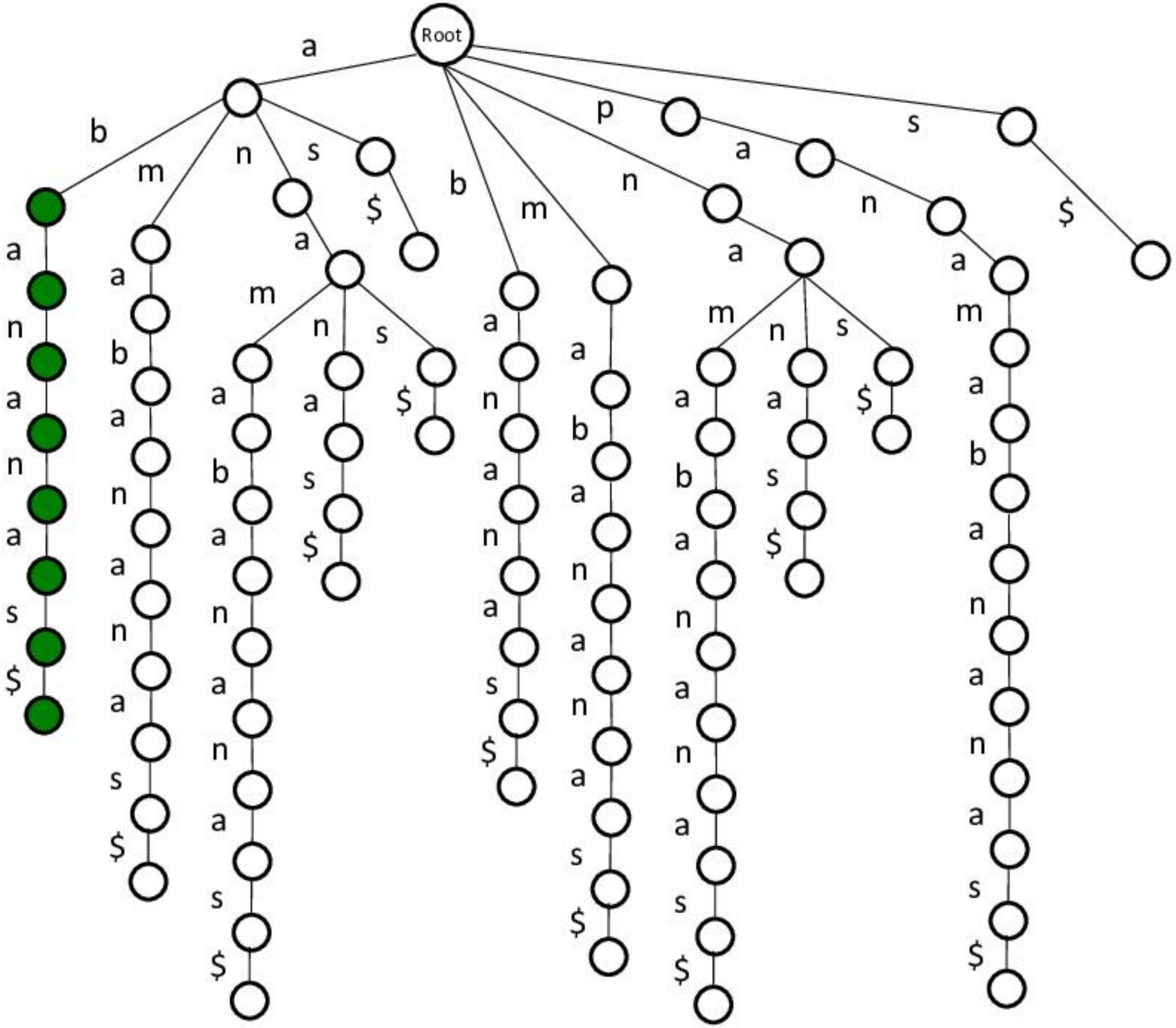
For human genome:

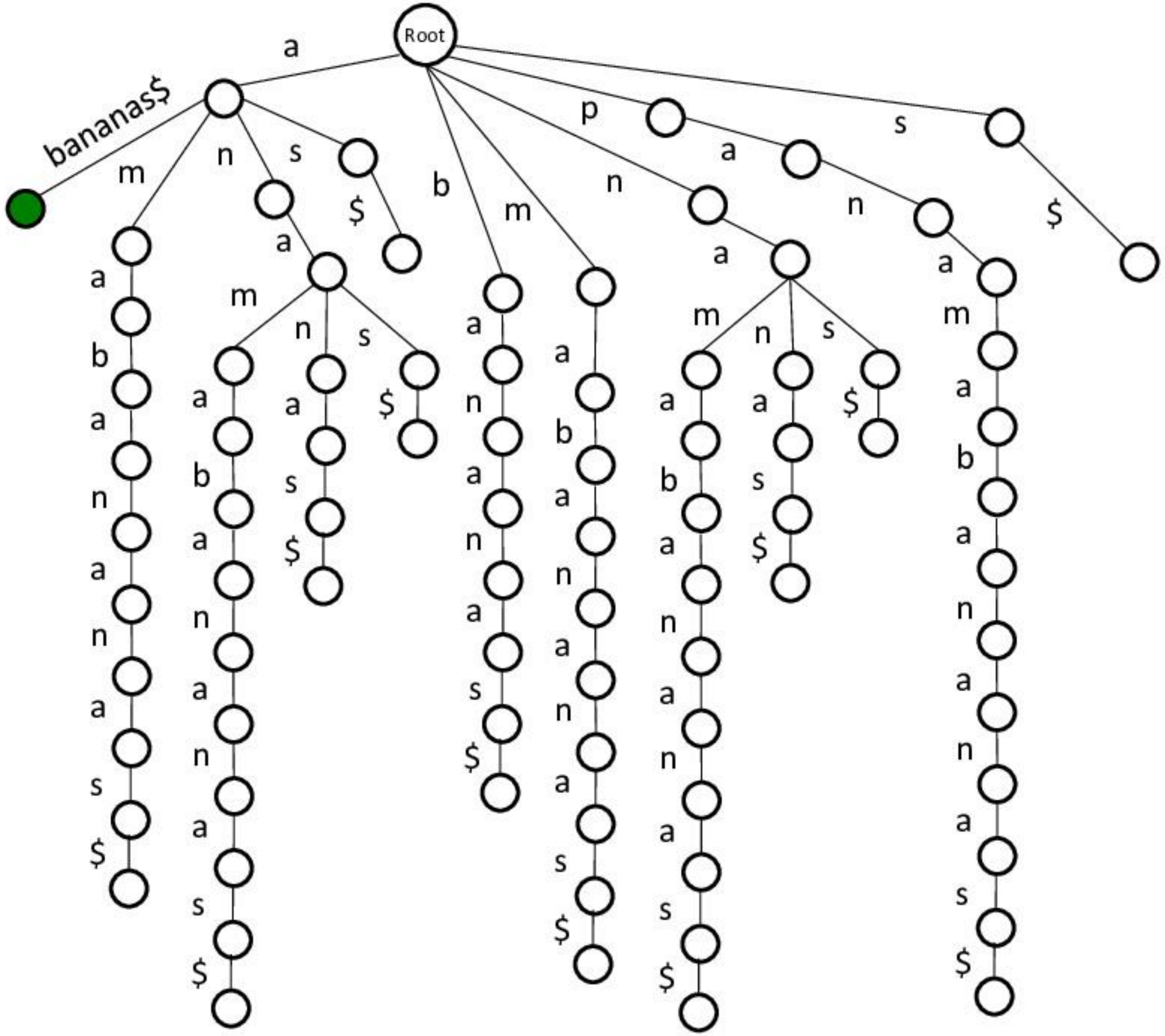
- $|Text| \approx 3*10^9$

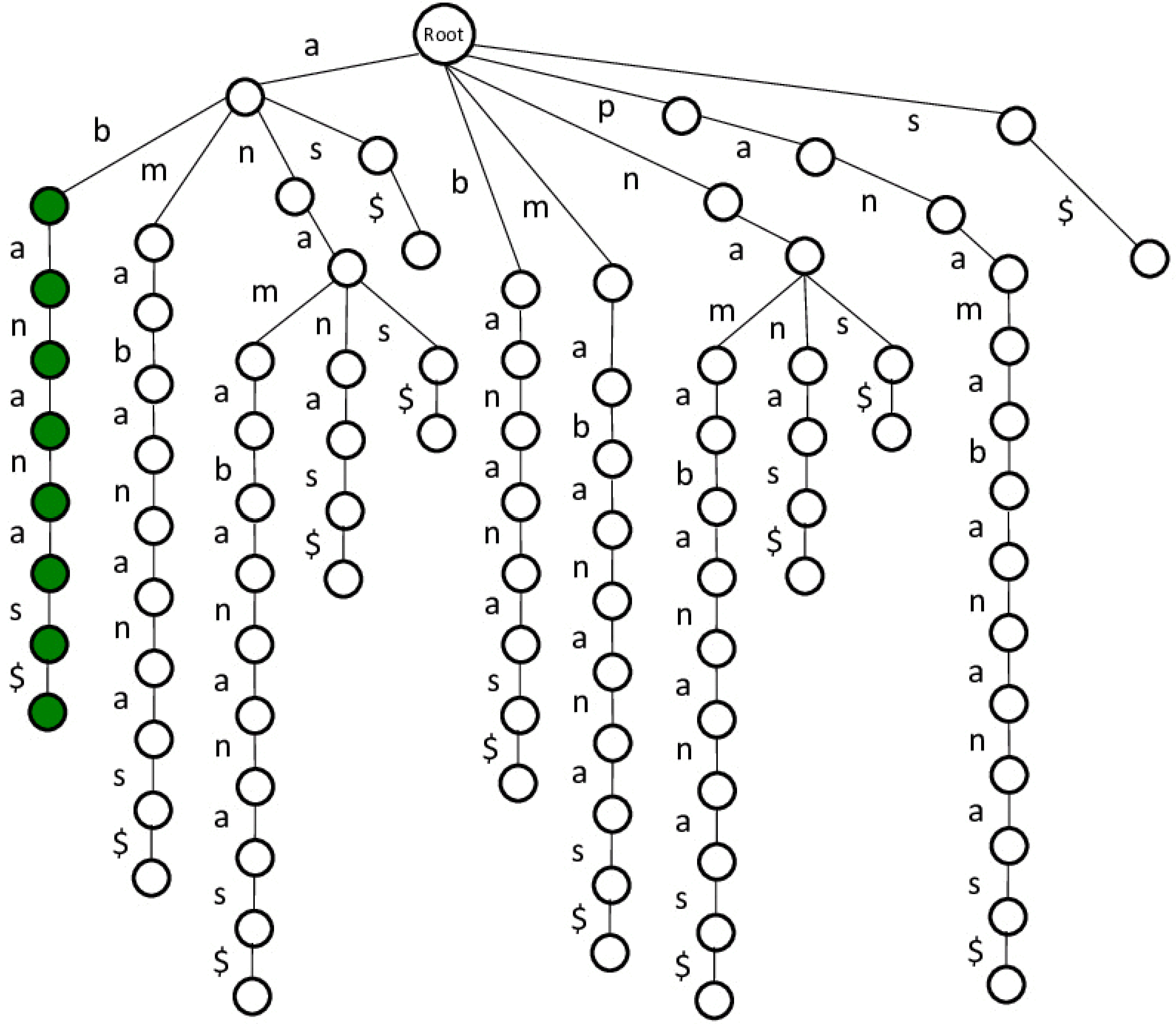
|*Text*| symbols

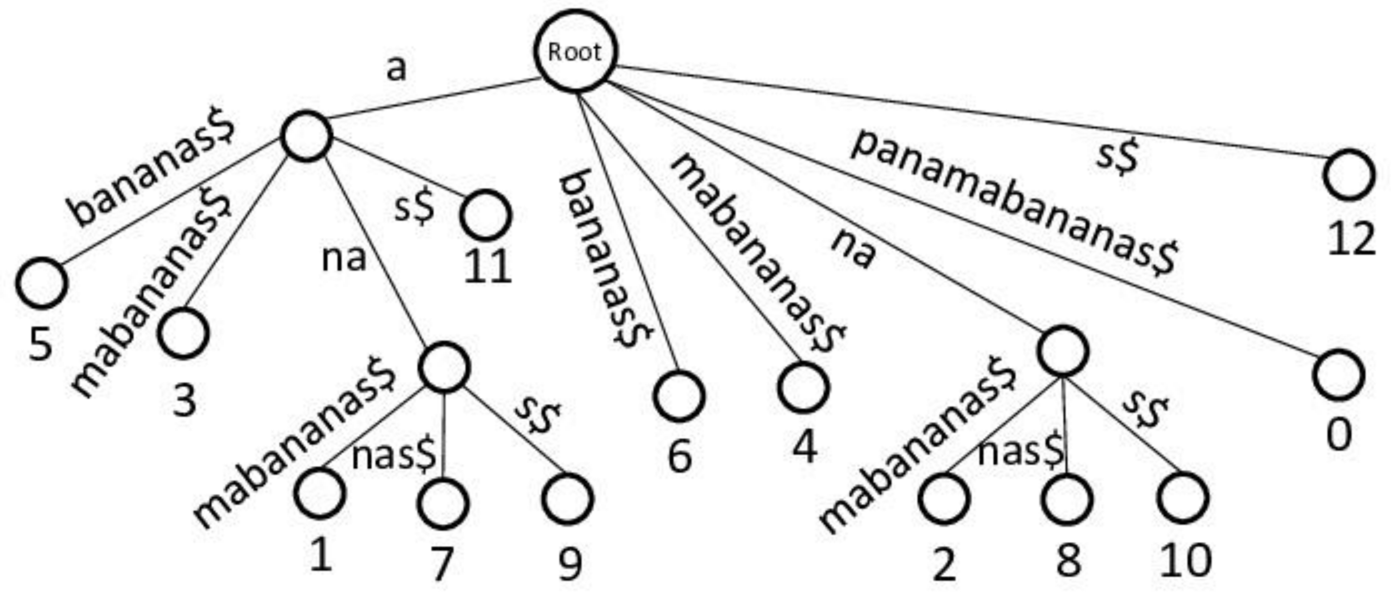
```
p a n a m a b a n a n a s $
  a n a m a b a n a n a s $
    n a m a b a n a n a s $
      a m a b a n a n a s $
        m a b a n a n a s $
          a b a n a n a s $
            b a n a n a s $
              a n a n a s $
                n a n a s $
                  a n a s $
                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

|*Text*| suffixes


Root

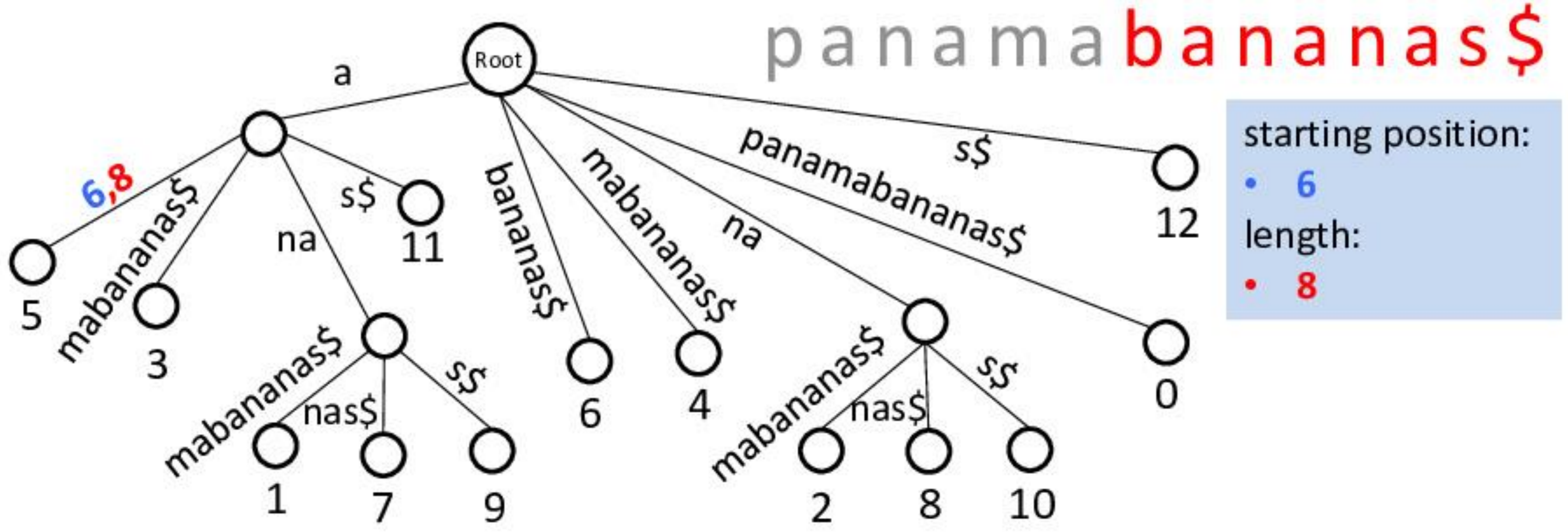

Root
a
bananas$
m
n
s
$
b
m
n
p
a
n
s
$
a
m
n
s
a
b
$


Root
a
b
m
n
s
p
$



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*|)
- storing edge labels

Overview of Suffix Tree

- Fast **Exact** Multiple Pattern Matching
  - Time: O(|text| + |patterns|)
  - Memory: O(|text|)
    - Actual implementation still too demanding in memory requirements. ~20 * |text|
- Construction: O(|text|)
- Need better method that:
  - Can handle mutations (approximate matching)
  - Has smaller memory footprint