Rope

Assignment 2

Definition

- A rope is an augmented binary tree that is used to represent very long strings
- Each leaf node in the rope stores a small part of the string
- Each node p is augmented with the length of the string represented by the rope rooted at p

Rope Node *

```
class Node
{
        private string s;           //  only stored in a leaf node
        private int length:         //  augmented data
        private Node left, right;
        ...
}
class Rope
{
        private Node root;
        ...
}
```

* Properties can be defined instead

Methods

Constructor: Rope(string S)

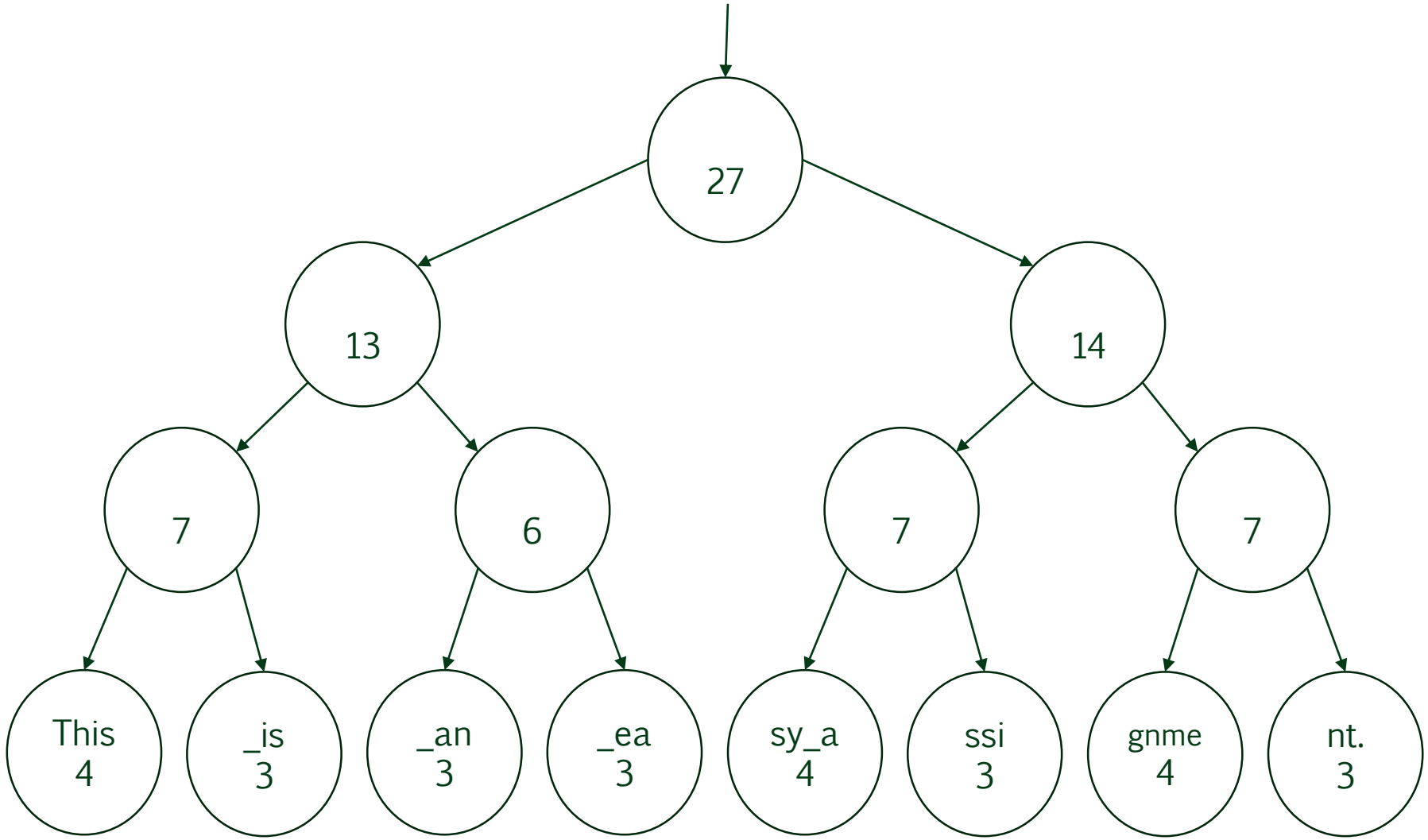
› Let's start with the string

"This_is_an_easy_assignment."

› Create a balanced rope (binary tree)

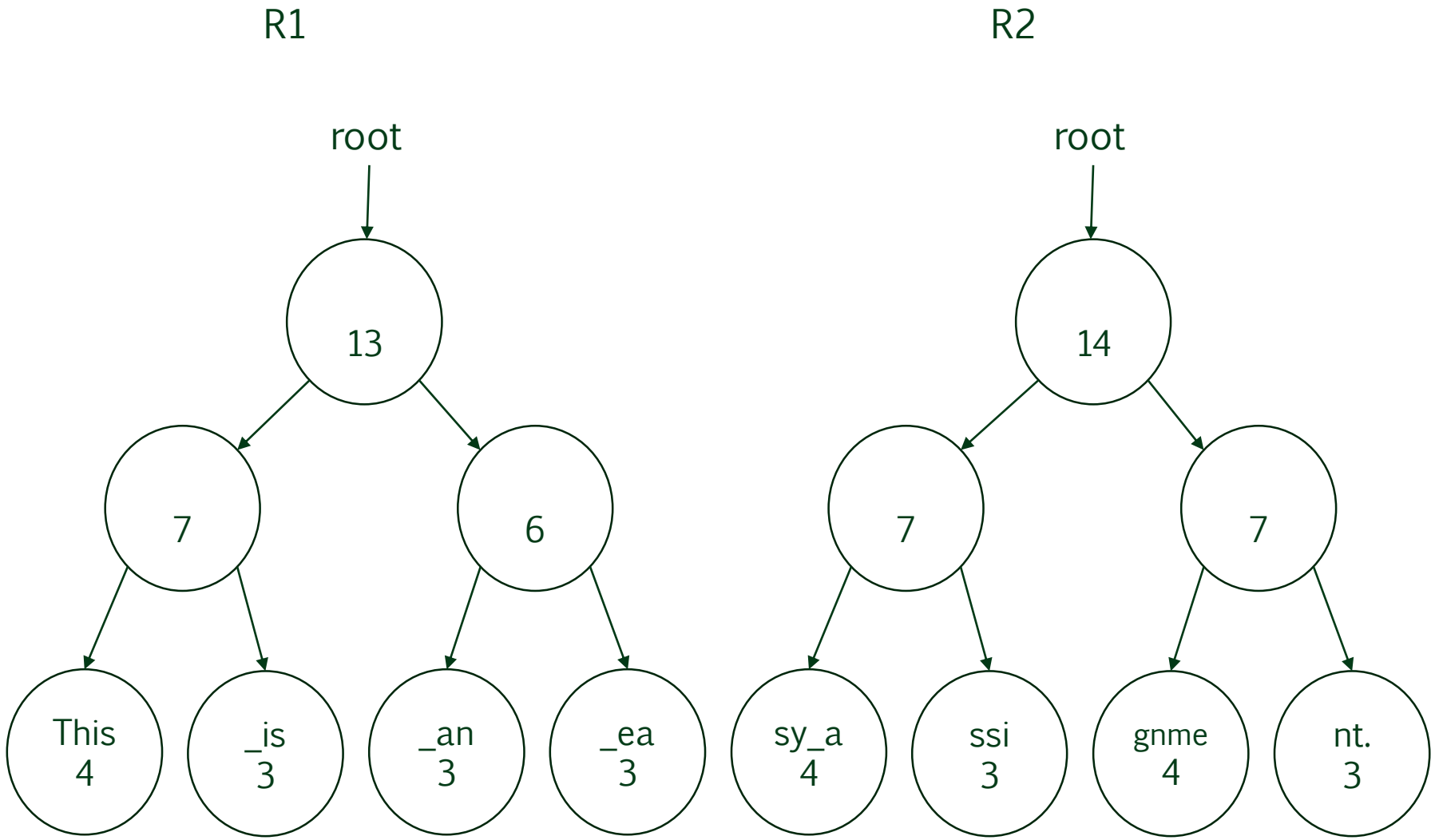
› Assume that the maximum length of a leaf substring is 4 (on the assignment it is 10)

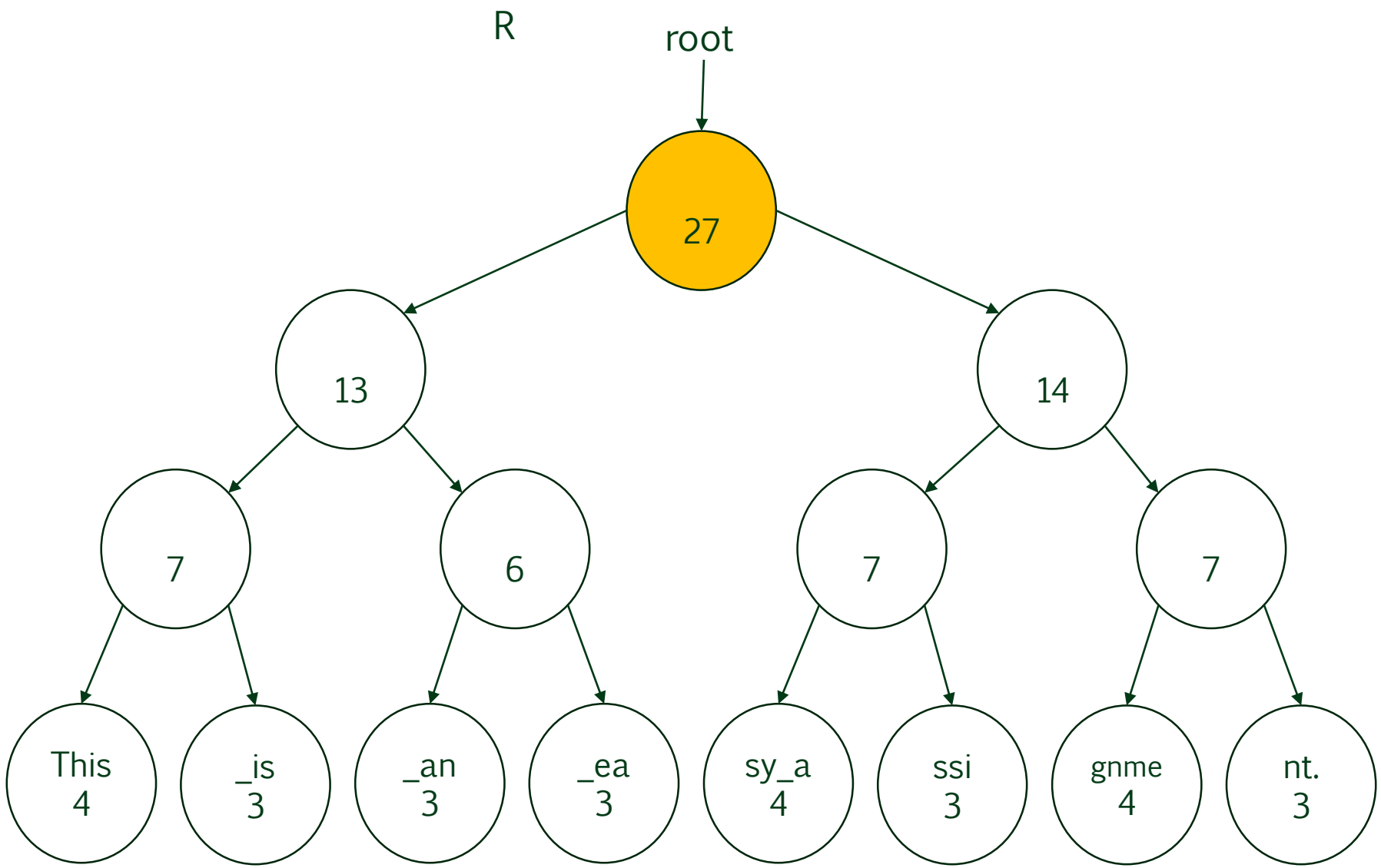
- Strategy
  - Create a root node
  - Recursively build the rope from the top down using the two halves of the given string to create the left and right ropes of the root


27
13
14
7
6
7
7
This
4
_is
3
_an
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3

Rope Concatenate(Rope R1, Rope R2)

- Strategy
  - Create a new rope R and set the left and subtrees of its root to the roots of R1 and R2 respectively

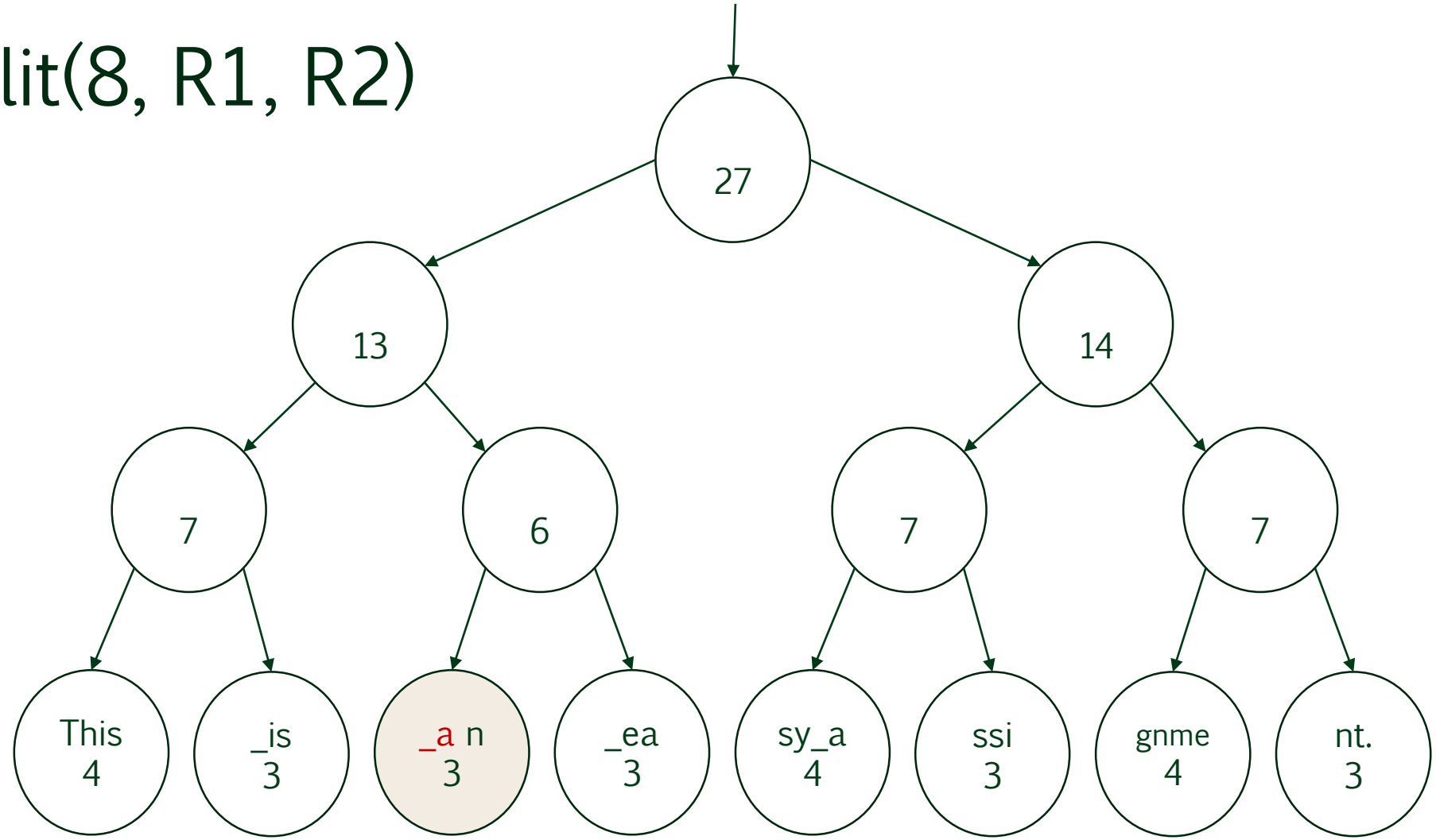

R1
root
13
7
6
This
4
_is
3
_an
3
_ea
3
R2
root
14
7
7
sy_a
4
ssi
3
gnme
4
nt.
3


R
root
27
13
14
7
6
7
7
This
4
_is
3
_an
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3

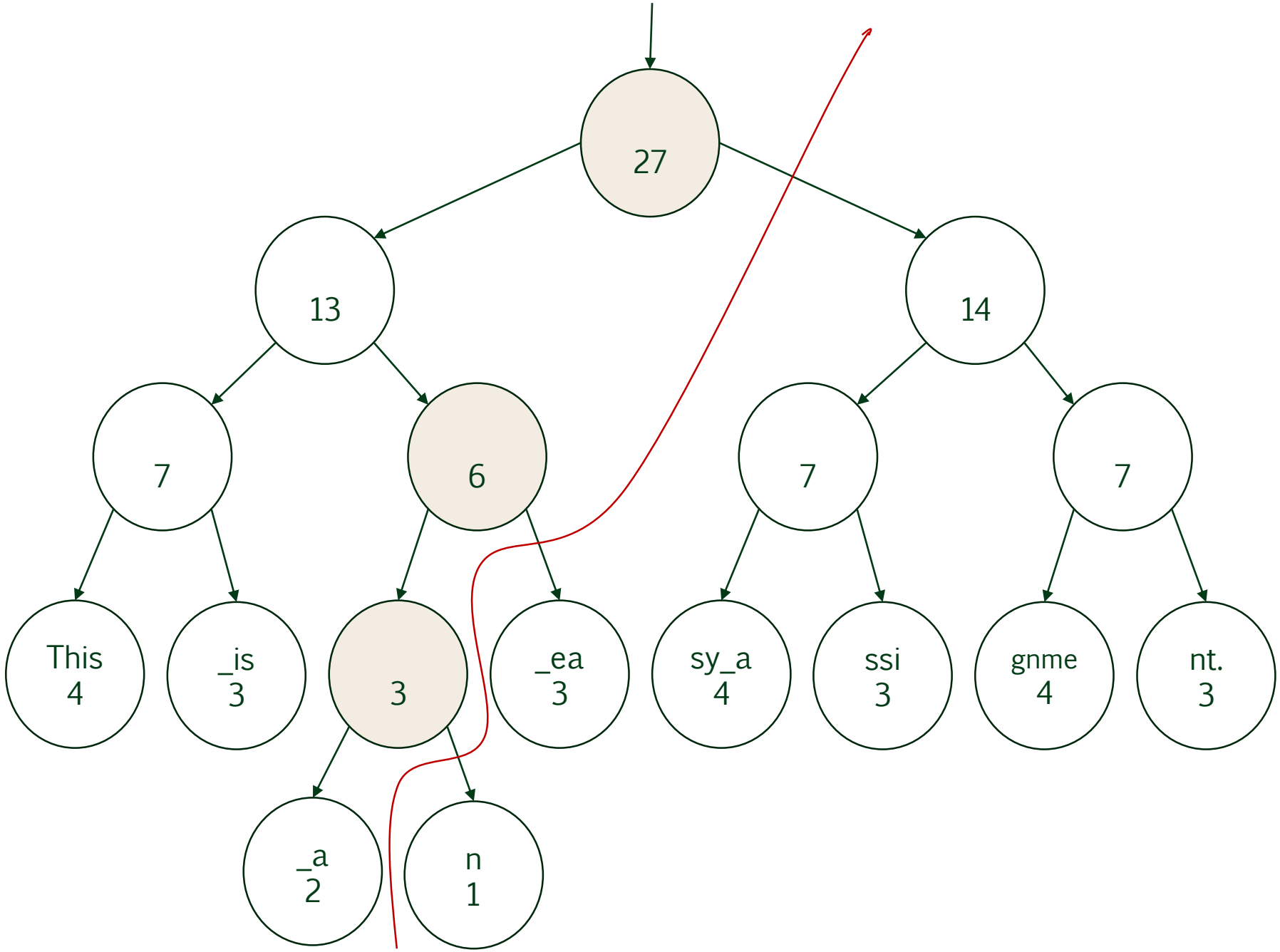
void Split(int i, Rope R1, Rope R2)

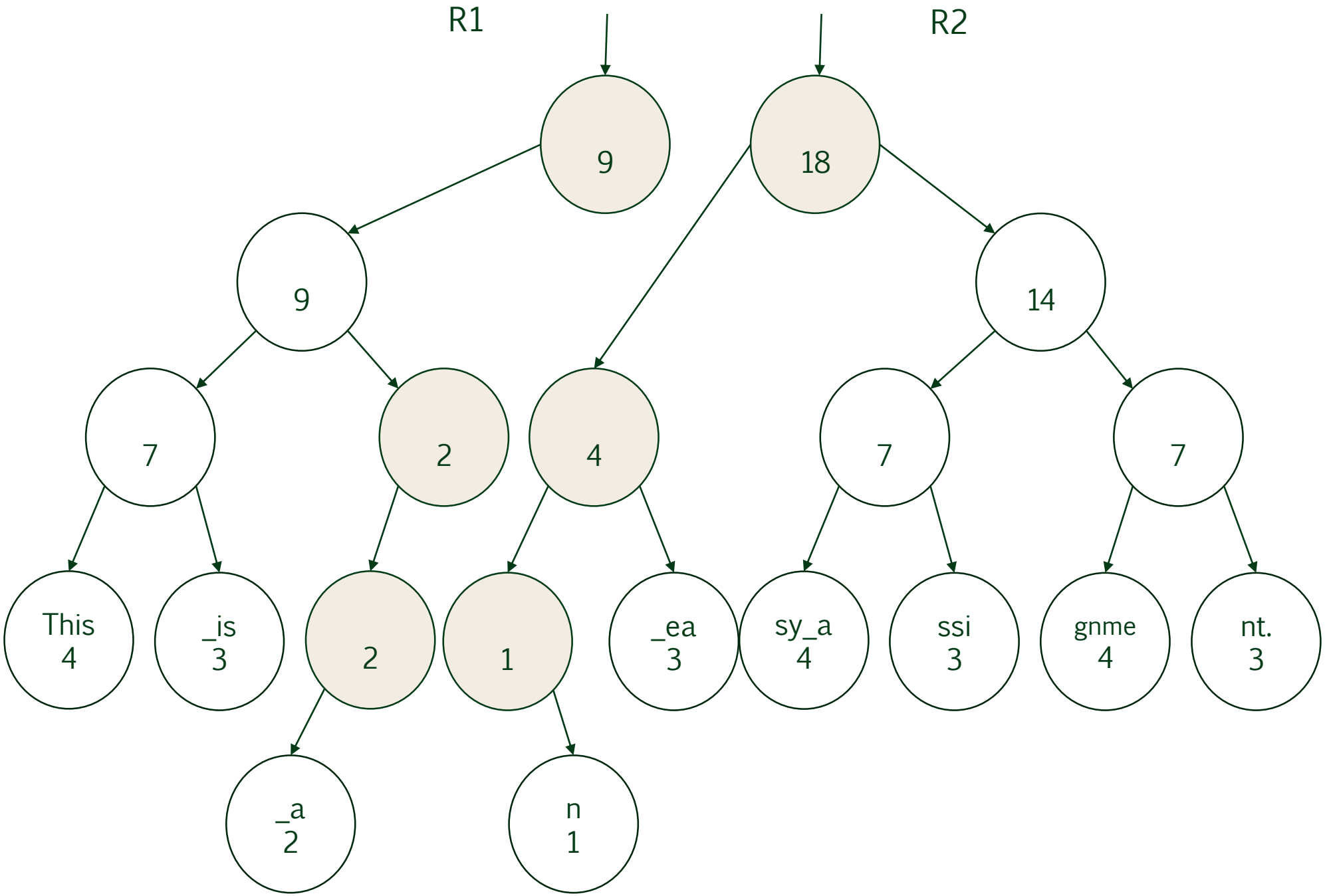
- Strategy
  - Find the node p with index i of the string
  - If the index splits the substring at p into two then insert the two parts of the string between (new) left and right subtrees of p
  - Break the rope into two parts, duplicating nodes whose right children cross the split

Split(8, R1, R2)

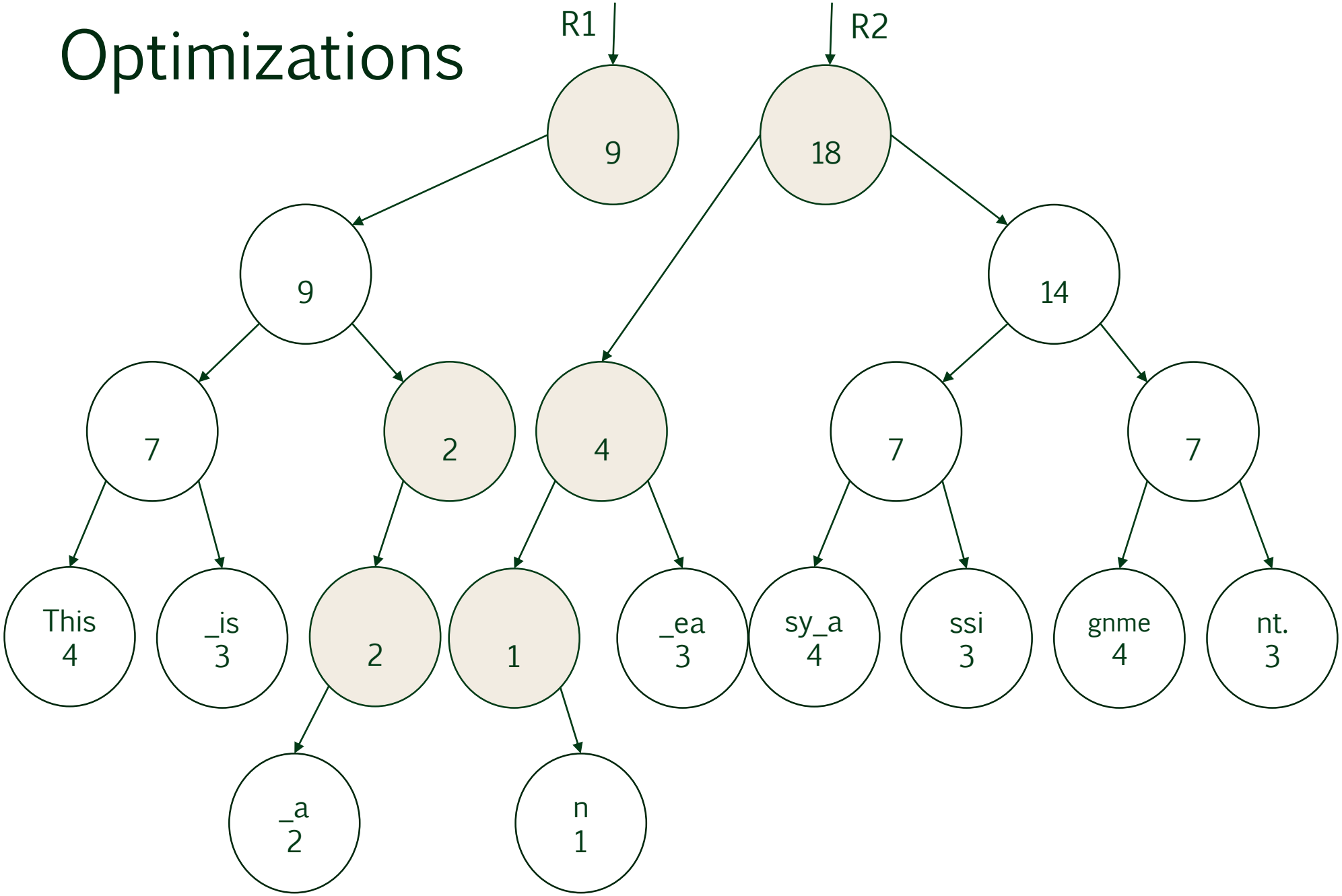


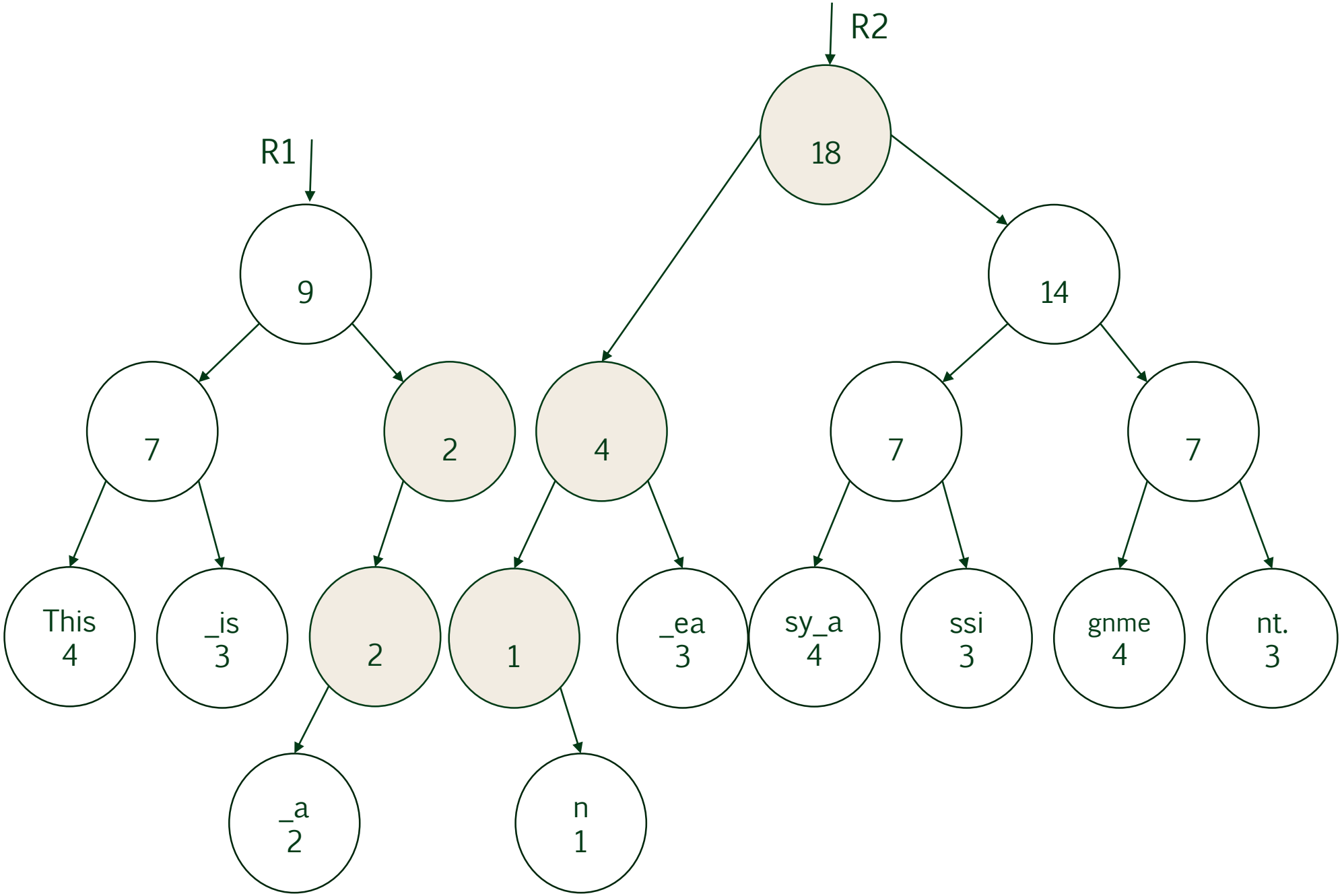
Assuming indices start at 0

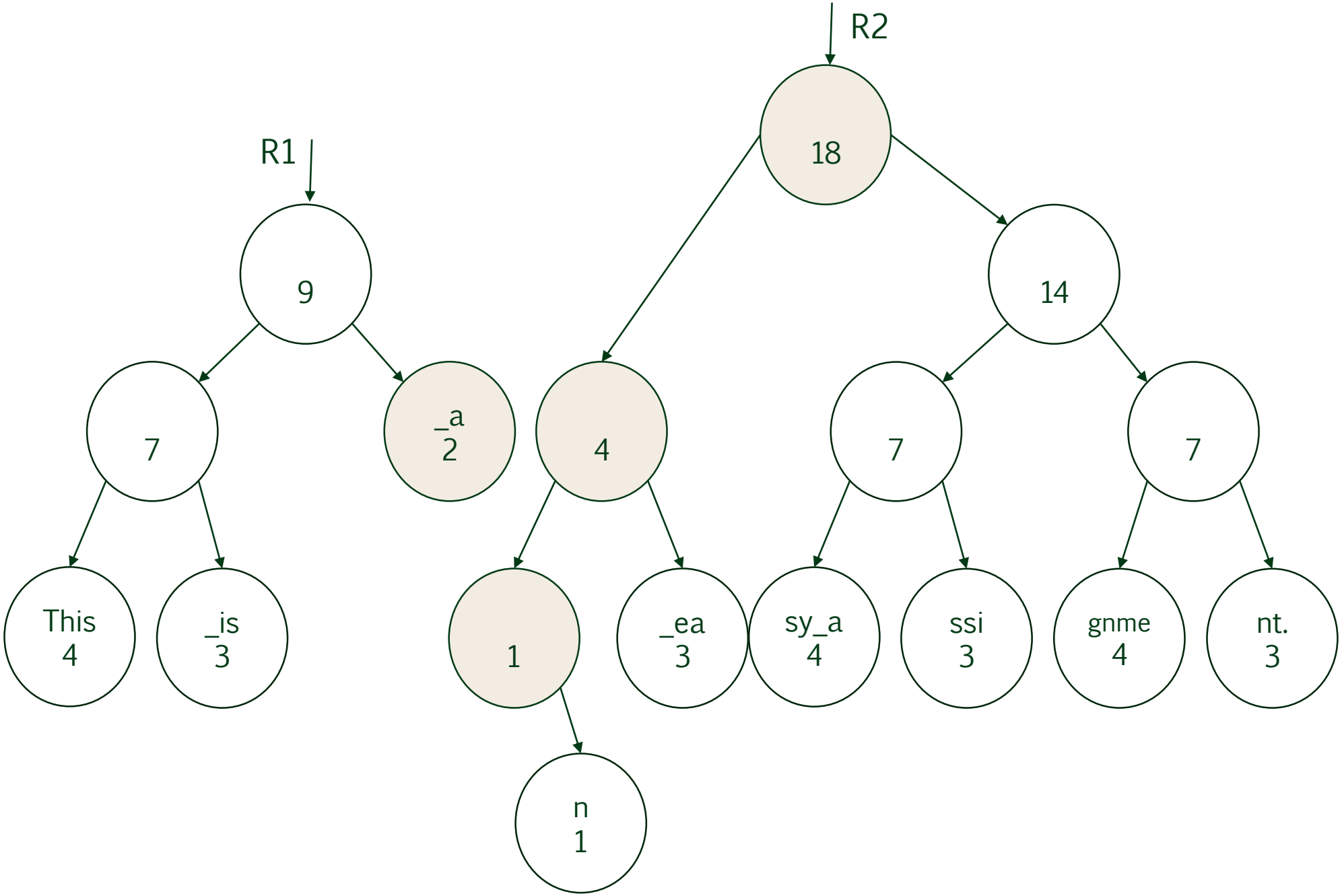

27
13
14
7
6
7
7
This
4
_is
3
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3
_a
2
n
1

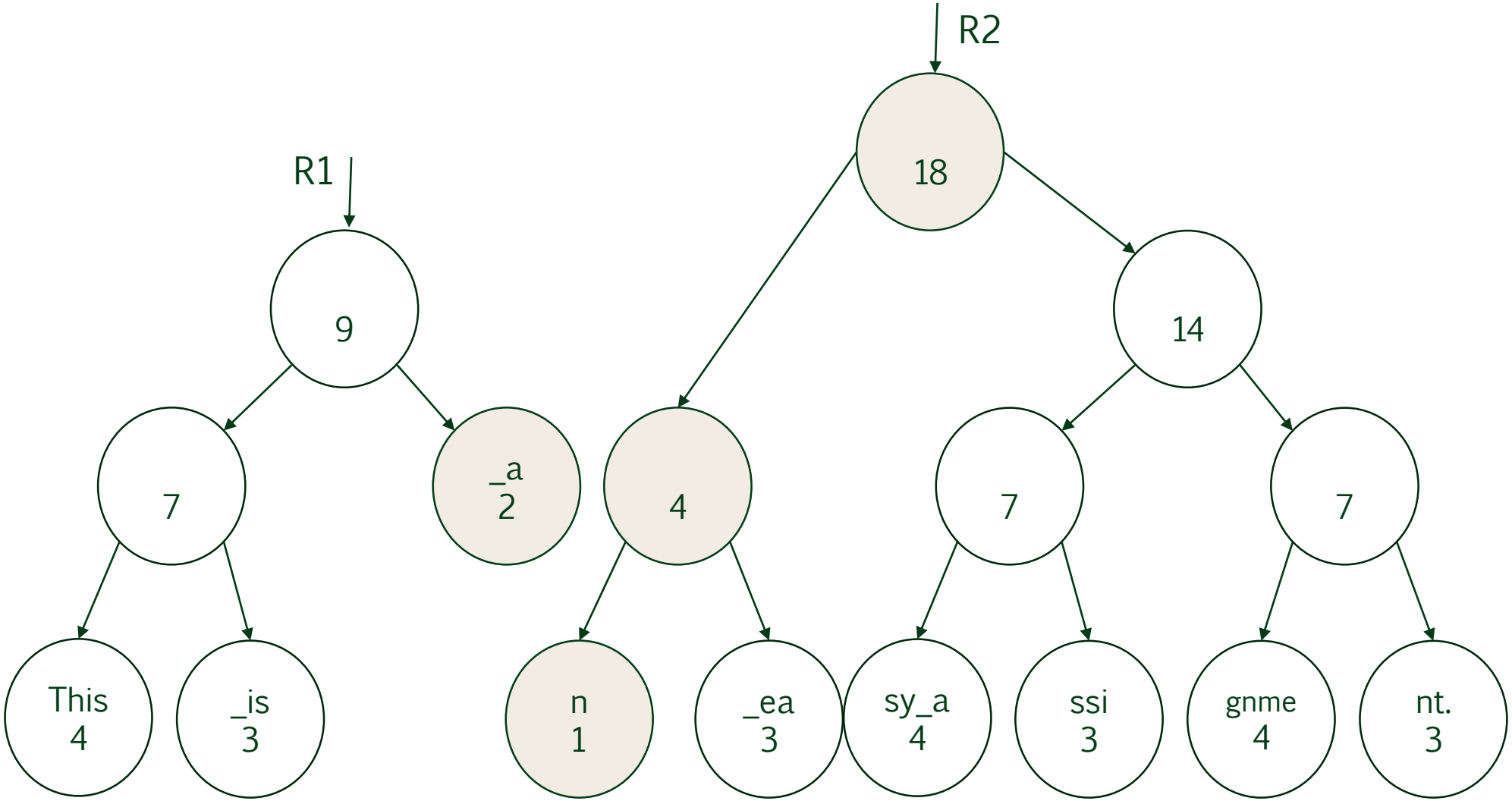

R1
R2
9
18
9
14
7
2
4
7
7
This
4
_is
3
2
1
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3
_a
2
n
1

Optimizations




R2
18
R1
9
14
7
2
4
7
7
This
4
_is
3
2
1
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3
_a
2
n
1


R2
18
R1
9
14
7
_a
2
4
7
7
This
4
_is
3
1
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3
n
1


R2
18
R1
9
14
7
_a
2
4
7
7
This
4
_is
3
n
1
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3

void Insert(string S, int i)

- Strategy (one split and two concatenations)
  - Create a rope R1 using S
  - Split current rope at index i into ropes R2 and R3
  - Concatenate R1, R2 and R3

void Delete(int i, int j)

- Strategy (two splits and one concatenation)
  - Split current rope at indices i - 1 and j to give ropes R1, R2 and R3
  - Concatenate R1 and R3

string Substring(int i, int j)

- Strategy
  - Carry out a “pruned” traversal of the rope using the indices i, j and the augmented data for length to curb unnecessary explorations of the rope

char CharAt(int i)

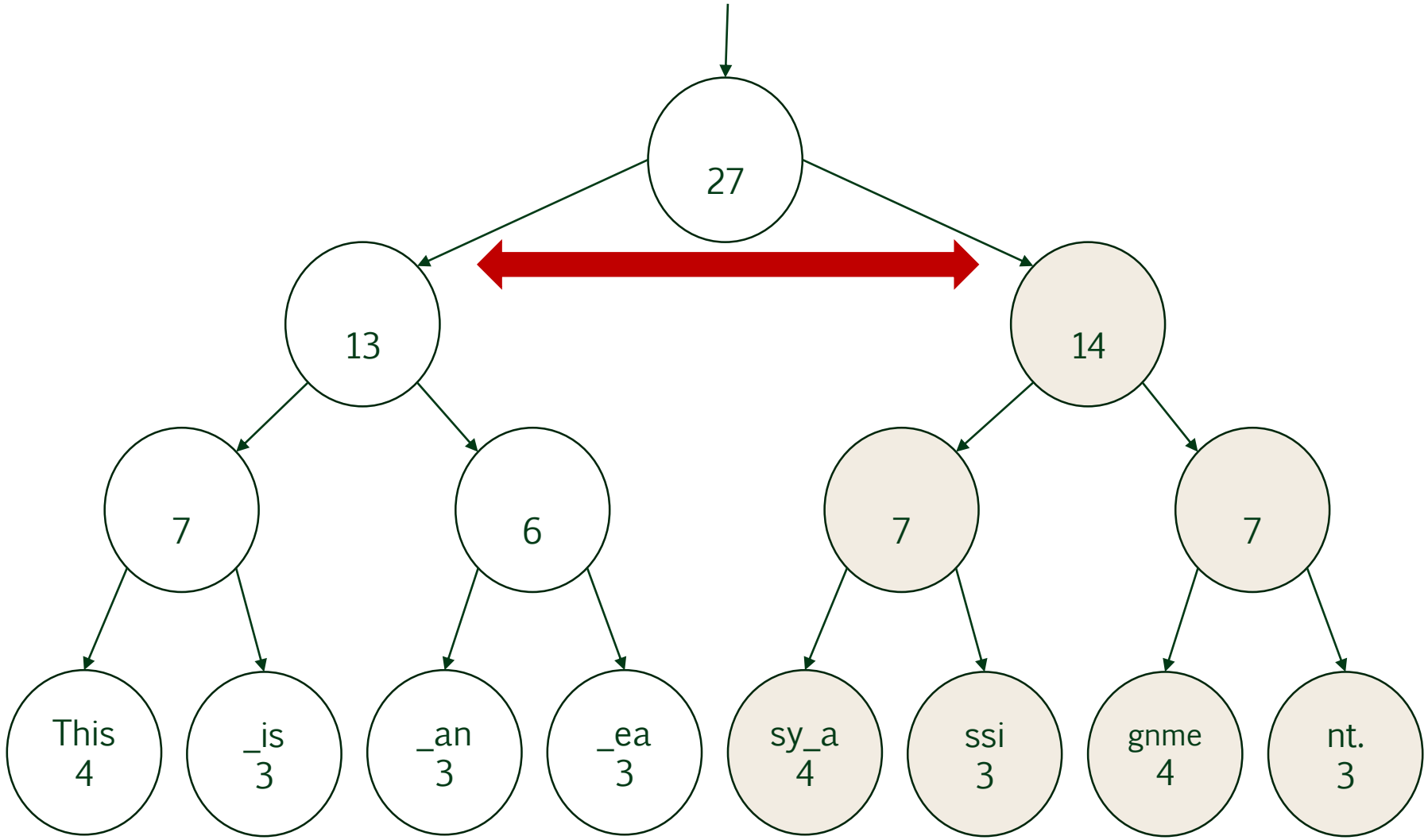
- Strategy
  - Using the augmented data for length, proceed either left or right at each node of the rope to find the character of the given index (cf Rank method of the Augmented Treap)

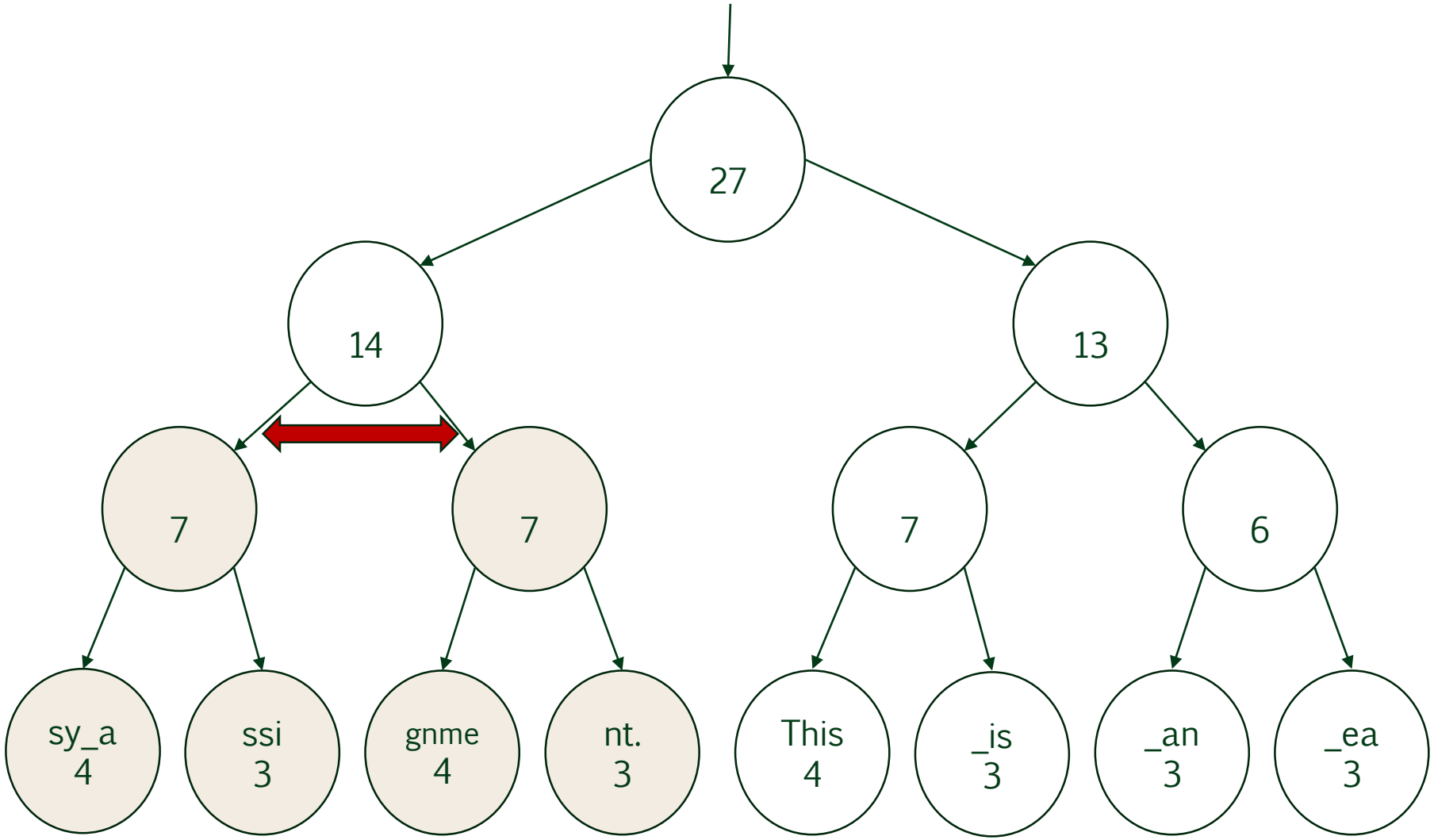
int IndexOf(char c)

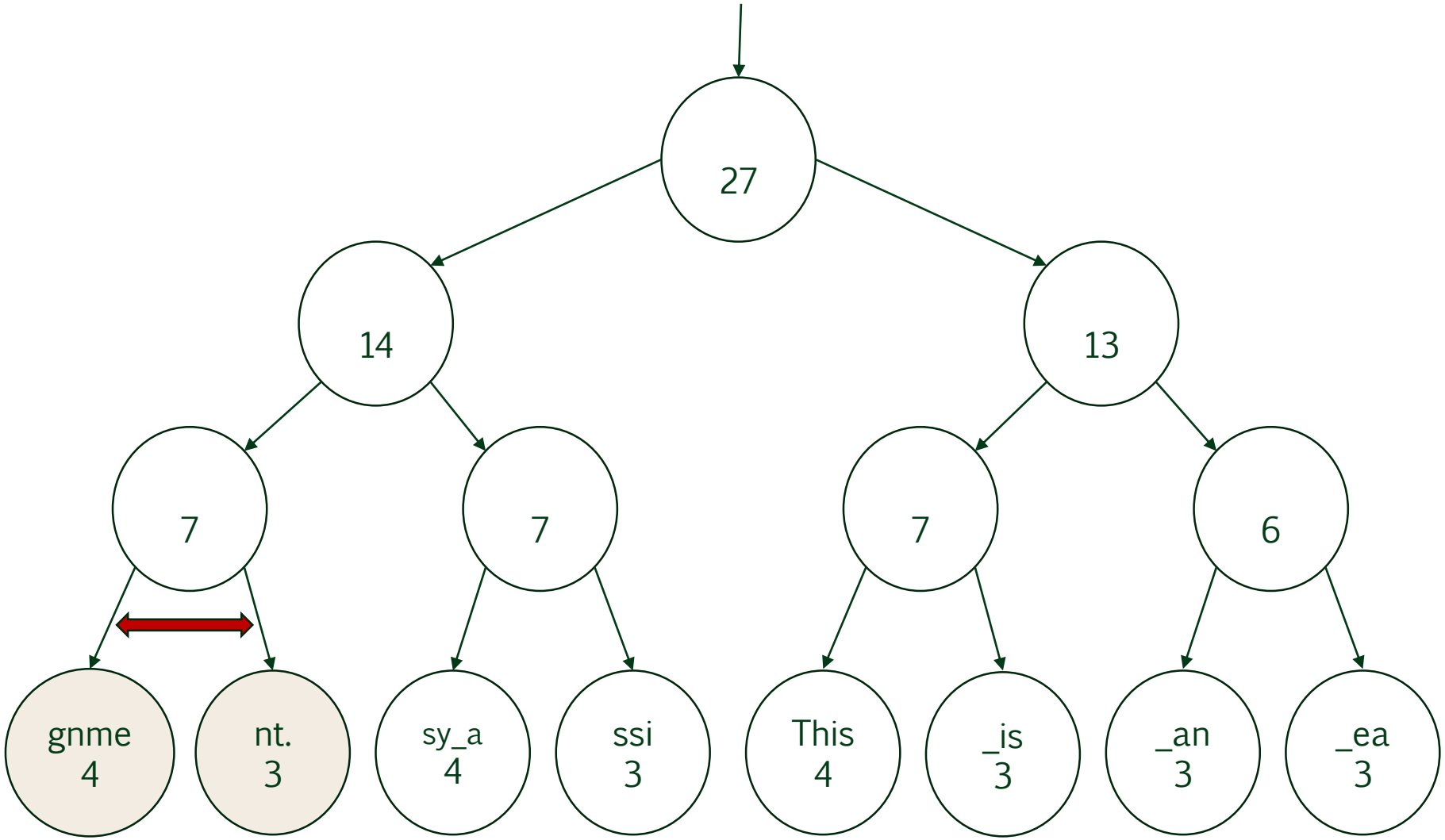
- Strategy
  - Carry out a traversal of the rope and a running count for the index until the first occurrence of character c is found

void Reverse( )

- Strategy
  - Modify the rope so that it represents the string in reverse
  - Carry out a preorder traversal of the rope and swap left and right children at each node
  - Reverse the substrings at each leaf node


27
13
14
7
6
7
7
This
4
_is
3
_an
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3


27
14
13
7
7
7
6
sy_a
4
ssi
3
gnme
4
nt.
3
This
4
_is
3
_an
3
_ea
3



and so on ...

int Length( )

- Strategy
  - Easy … where is the length of the full string stored?

string ToString( )

- Strategy
  - Carry out a traversal of the rope and concatenate (using the string method) the substrings at each leaf node
  - Return the concatenated string

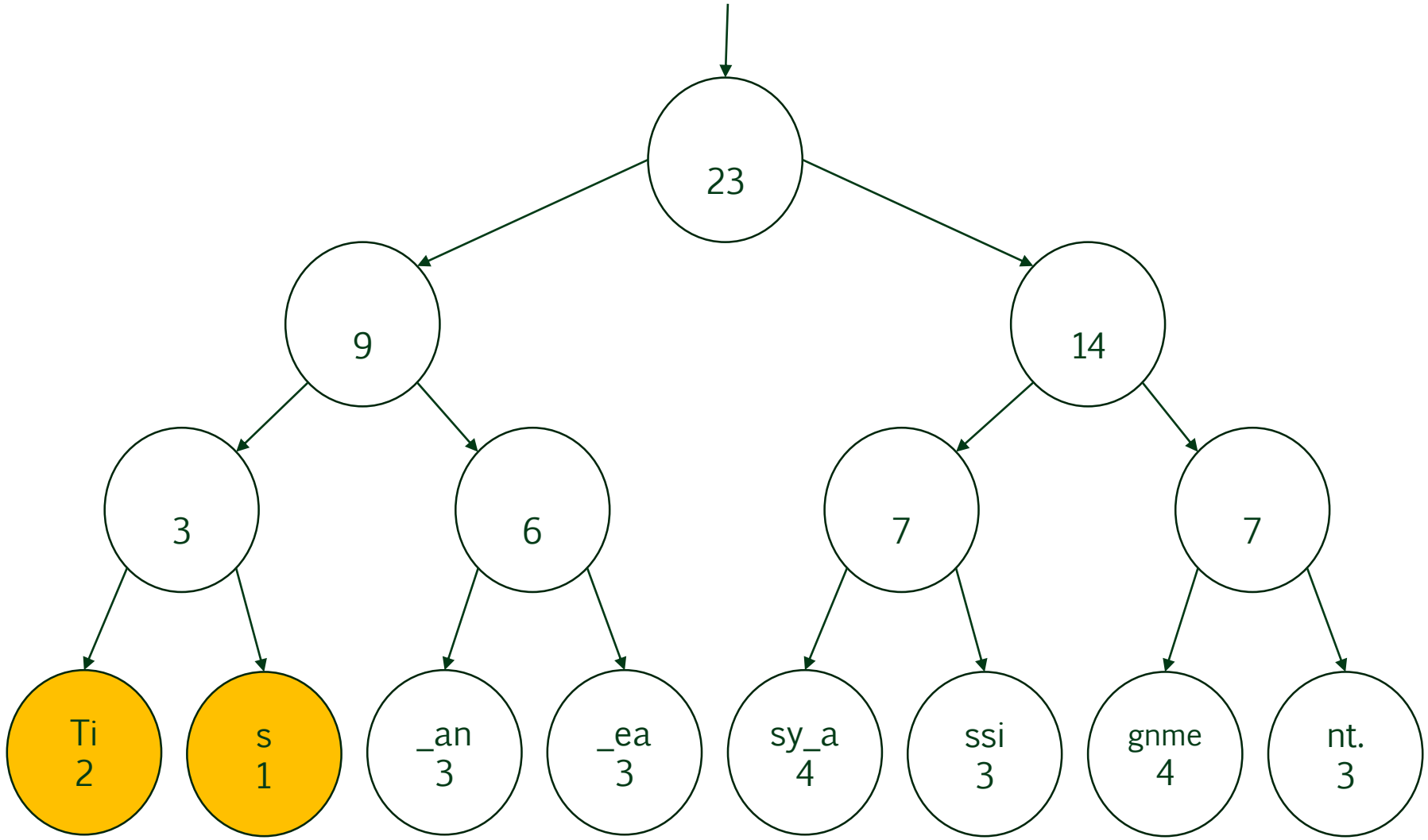
void PrintRope( )

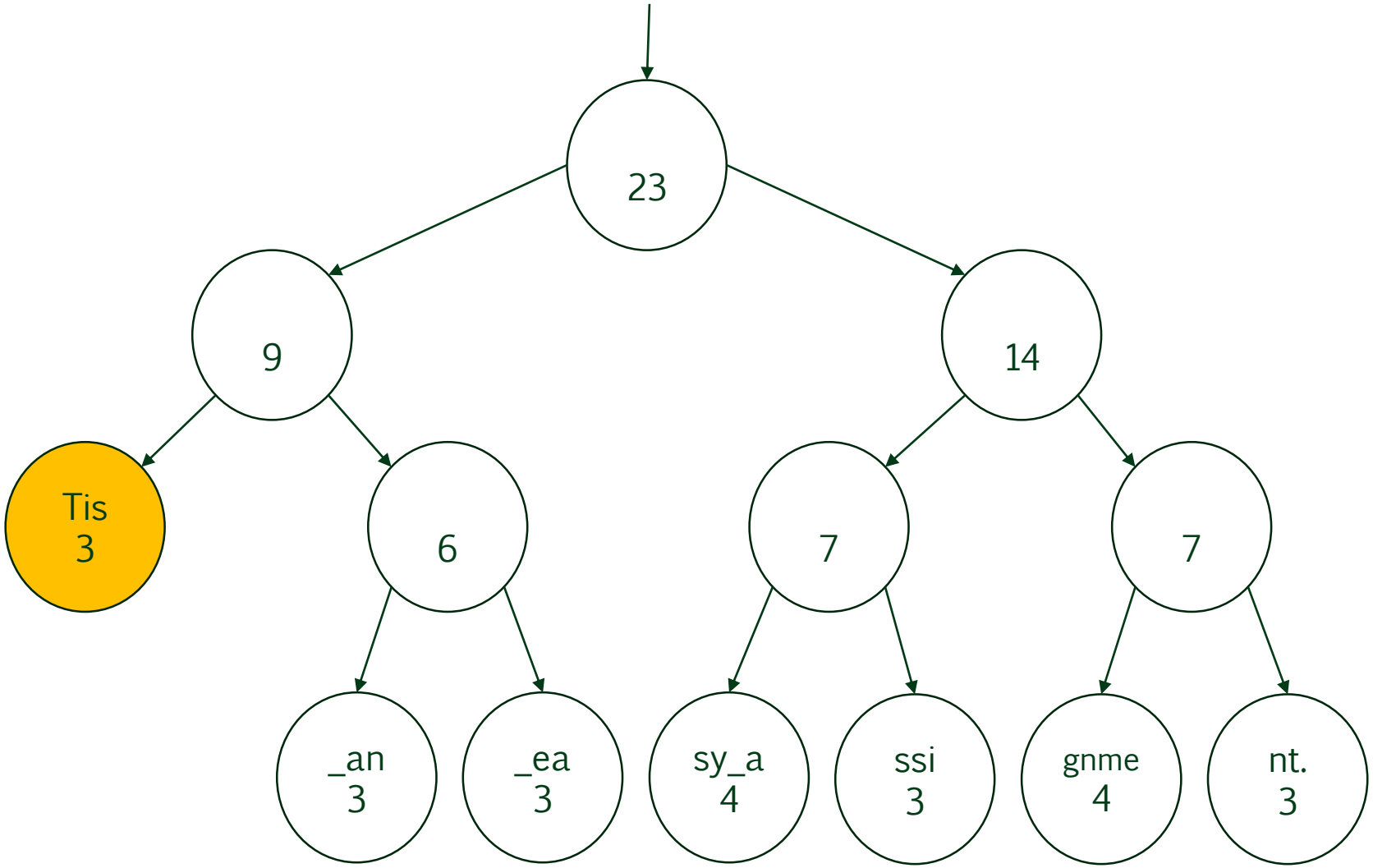
- Strategy
  - Print out the nodes of the rope inorder (cf Print of the Treap)

More Optimizations

Combining siblings with short strings

- Strategy
  - Concatenate two (leaf) sibling strings to create a longer substring and place it at the parent node


23
9
14
3
6
7
7
Ti
2
s
1
_an
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3


23
9
14
Tis
3
6
7
7
_an
3
_ea
3
sy_a
4
ssi
3
gnme
4
nt.
3

Rebalancing

- (Simple) Strategy
  - Concatenate the string represented by the rope and rebuild the rope (as done in the constructor)