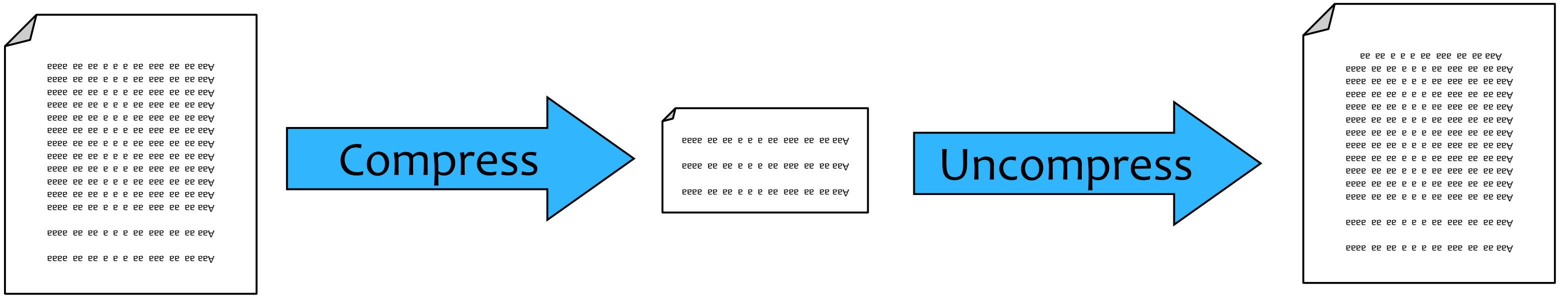
Data Compression

Mohammad Asad Abbasi

Lecture 16

Compression

- **Definition:** process of encoding with fewer bits
- **Reason:** to save valuable resources such as communication bandwidth or hard disk space



Compression Types

1. **Lossy**
   - **Loses** some information during compression which means the exact original can **not** be **recovered** (jpeg)
   - Normally provides better compression
   - Used when loss is acceptable - image, sound, and video files

Compression Types

2. **Lossless**
   - Exact original can be recovered
   - Used when loss is not acceptable - data

Basic Term: **Compression Ratio** - ratio of the number of bits in original data to the number of bits in compressed data

For example: **3:1** is when the original file was **3000** bytes and the compression file is now only **1000** bytes.

Huffman Codes

- Invented by Huffman as a class assignment in 1950
- Used in many, if not most, compression algorithms
- gzip, bzip, jpeg (as option), fax compression,…

- **Properties:**
  - Generates optimal prefix codes
  - Cheap to generate codes
  - Cheap to encode and decode
  - $l_a = H$ if probabilities are powers of 2

The (Real) Basic Algorithm

1. Scan text to be compressed and tally occurrence of all characters.
2. Sort or prioritize characters based on number of occurrences in text.
3. Build Huffman code tree based on prioritized list.
4. Perform a traversal of tree to determine all code words.
5. Scan text again and create new file using the Huffman codes.

Building a Tree

- Scan the original text

  Eerie eyes seen near lake.

  - What is the frequency of each character in the text?

Char	Freq.	Char	Freq.	Char	Freq.	
E	1	Y	1	k		
e	8	s	2	.	1	
r	2	n	2	space	4	
i	1	a	2			
		l	1			

Building a Tree

- **Prioritize characters**
  - Create binary tree nodes with character and frequency of each character
  - Place nodes in a priority queue
    - The lower the occurrence, the higher the priority in the queue

Building a Tree

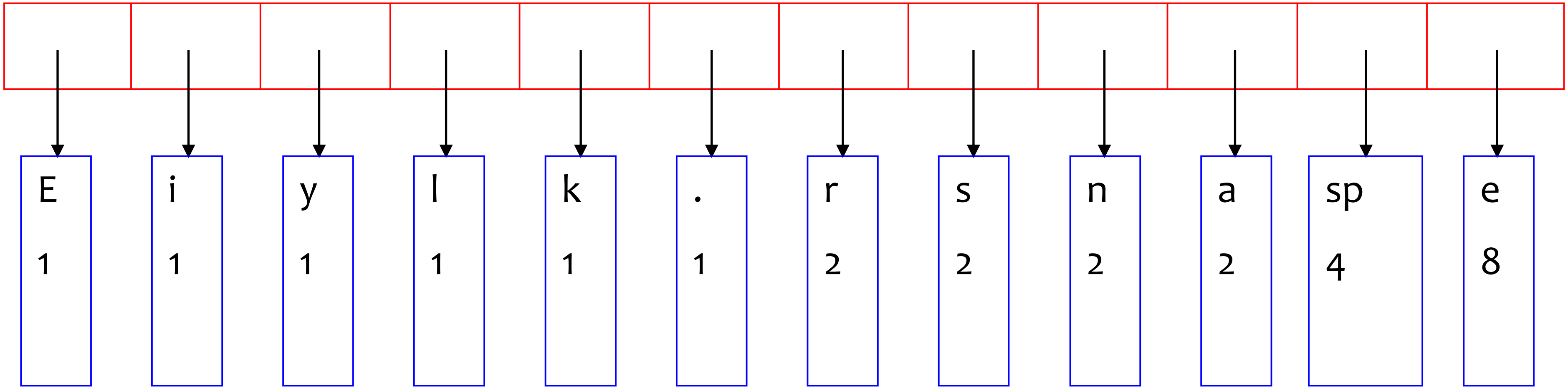
- **Prioritize characters**

Uses binary tree nodes

```
public class HuffNode
{
    public char myChar;
    public int myFrequency;
    public HuffNode myLeft, myRight;
}
priorityQueue myQueue;
```

Building a Tree

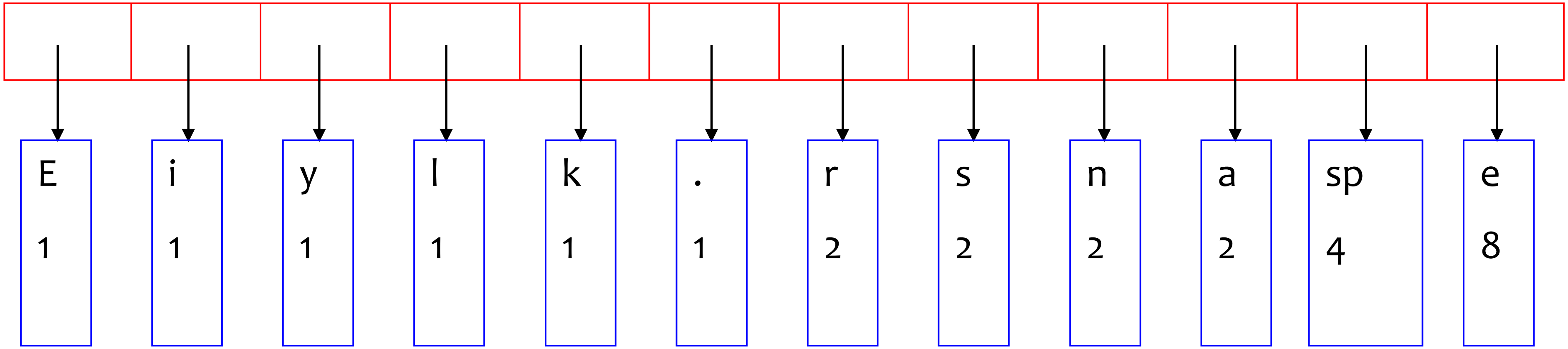
- The queue after inserting all nodes
- Null Pointers are not shown



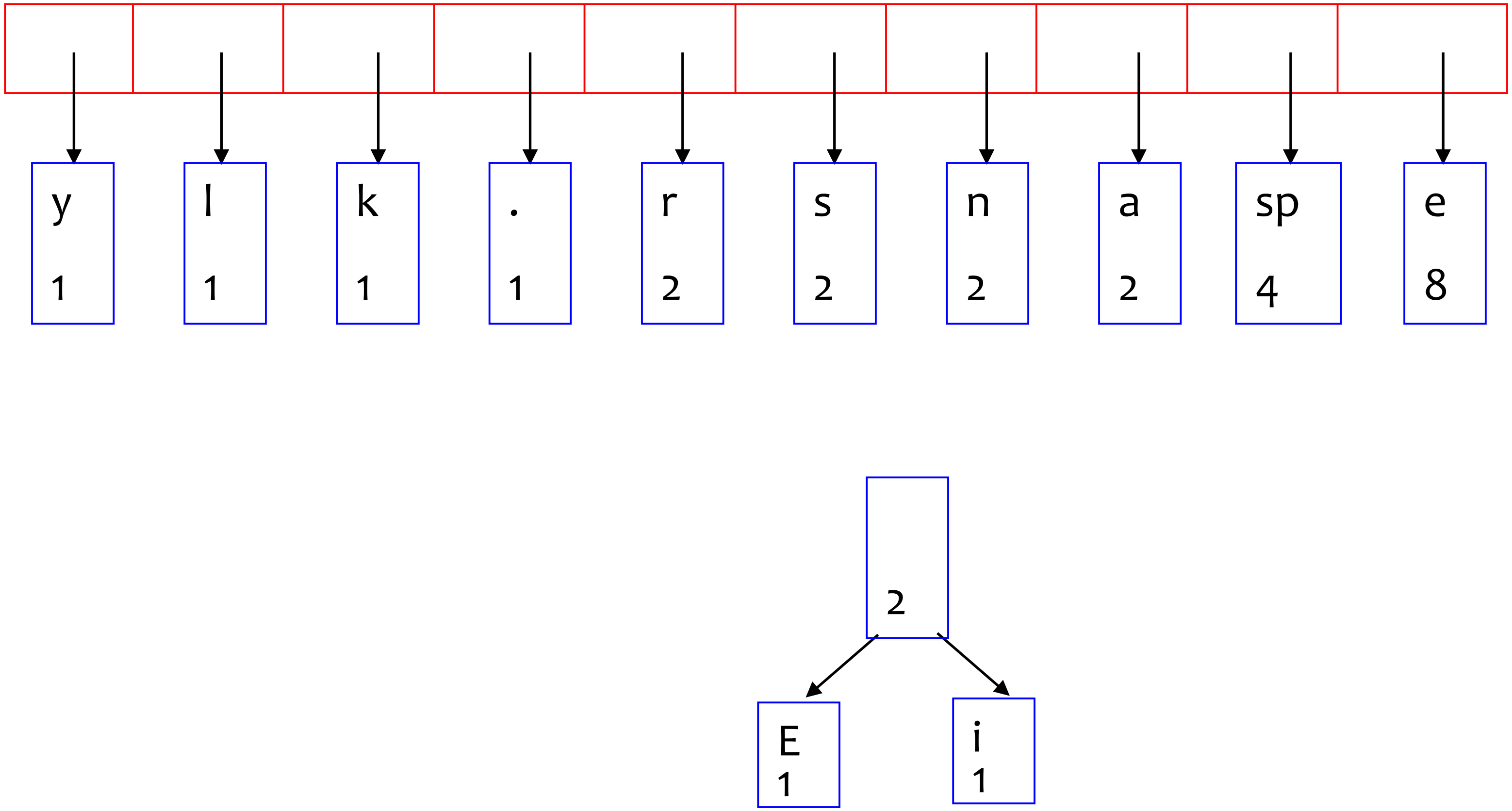
Building a Tree

- While priority queue contains two or more nodes
  - Create new node
  - Dequeue node and make it left subtree
  - Dequeue next node and make it right subtree
  - Frequency of new node equals sum of frequency of left and right children
  - Enqueue new node back into queue

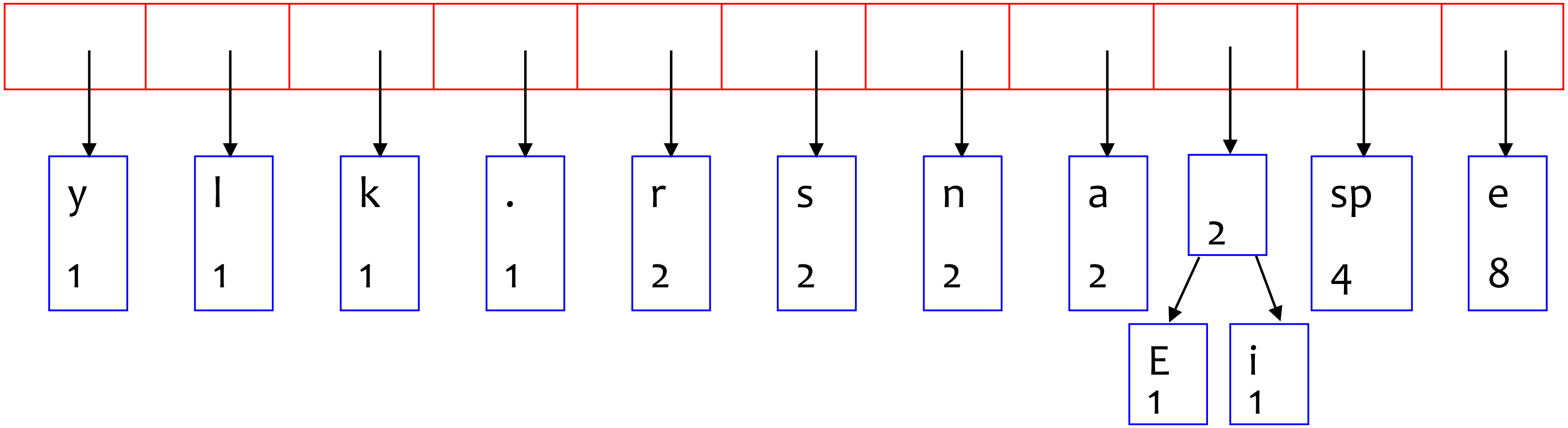
Building a Tree



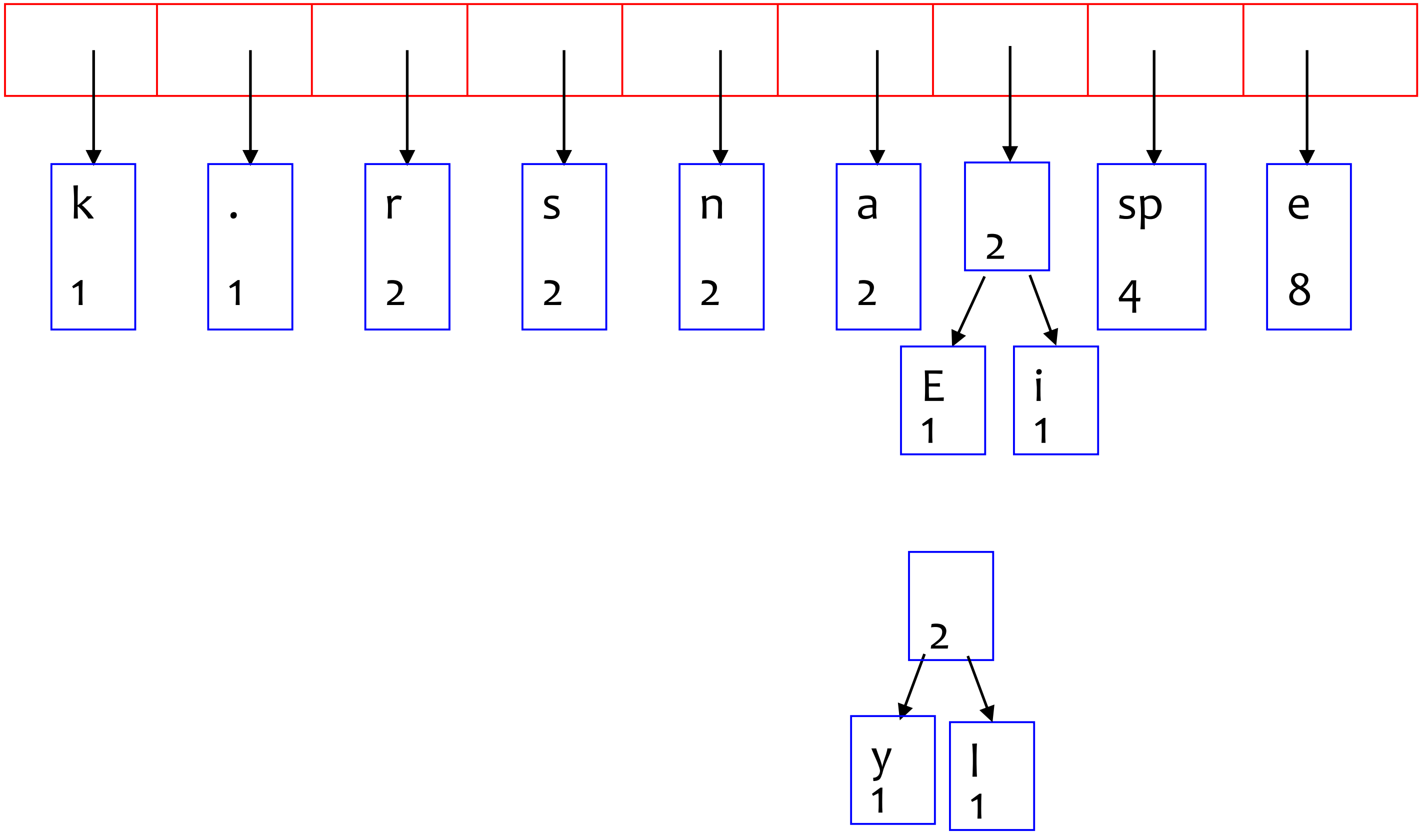
Building a Tree



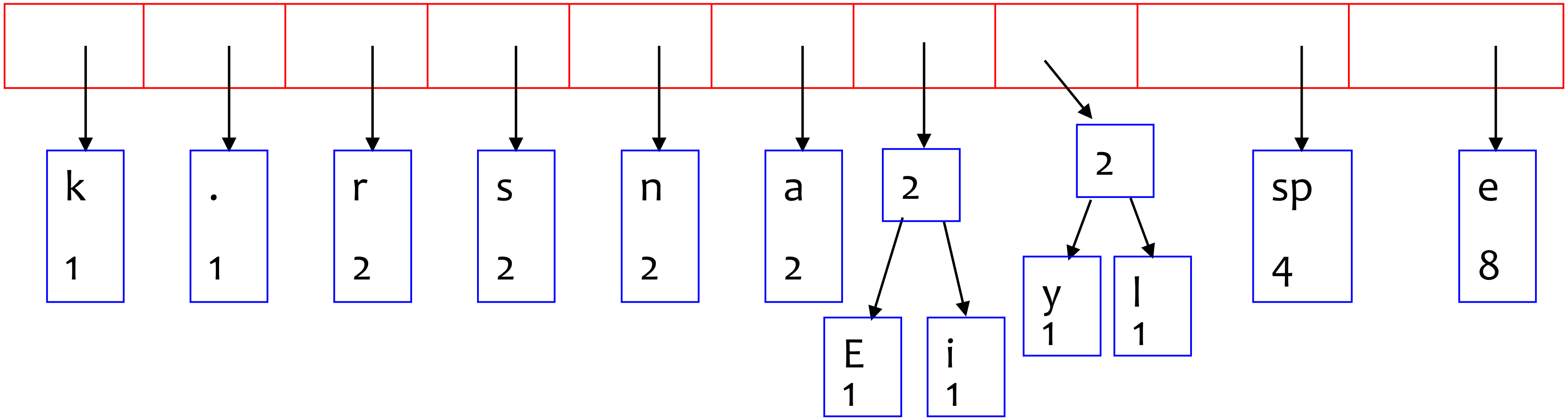
Building a Tree



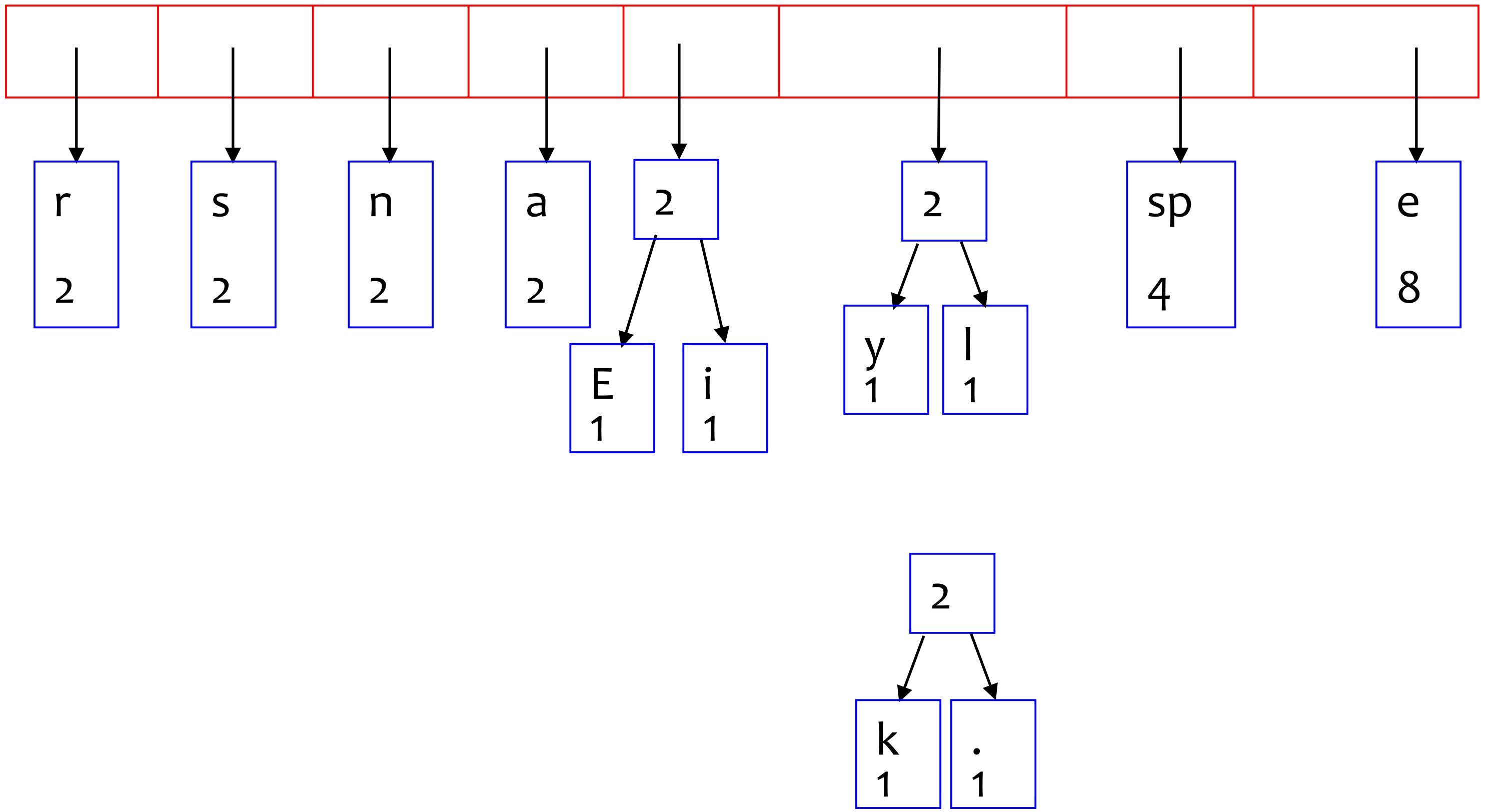
Building a Tree



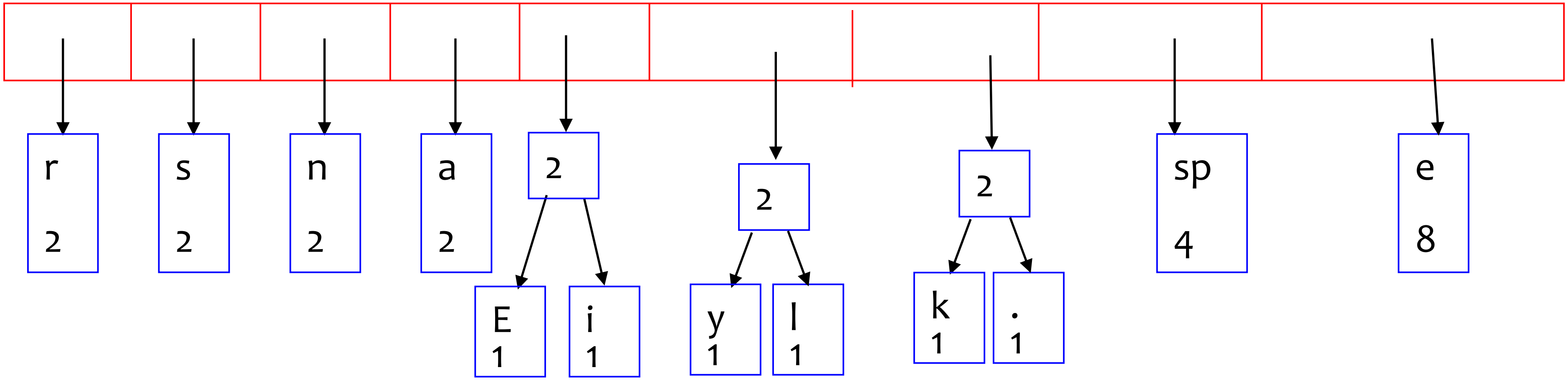
Building a Tree



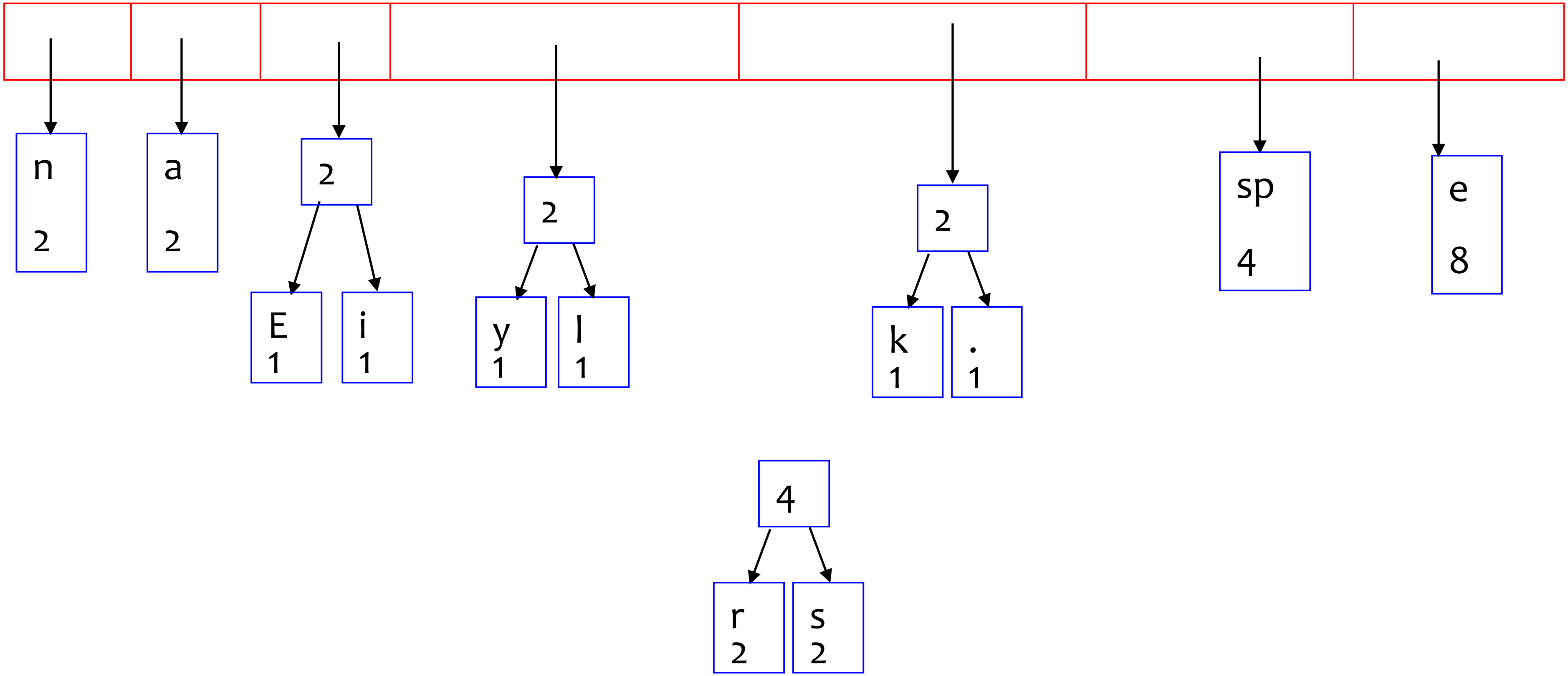
Building a Tree



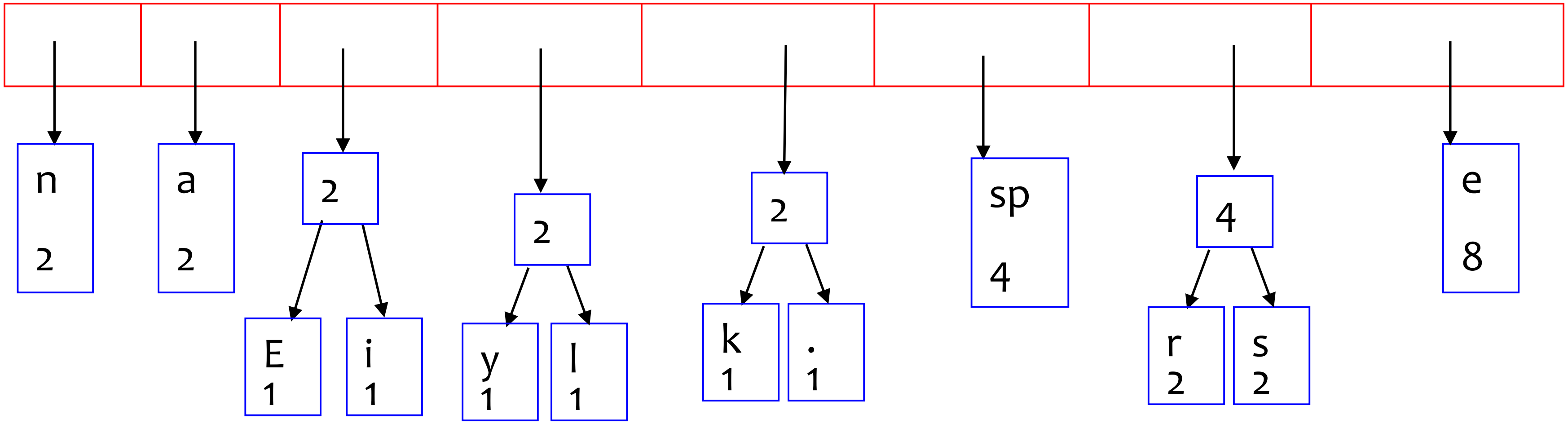
Building a Tree



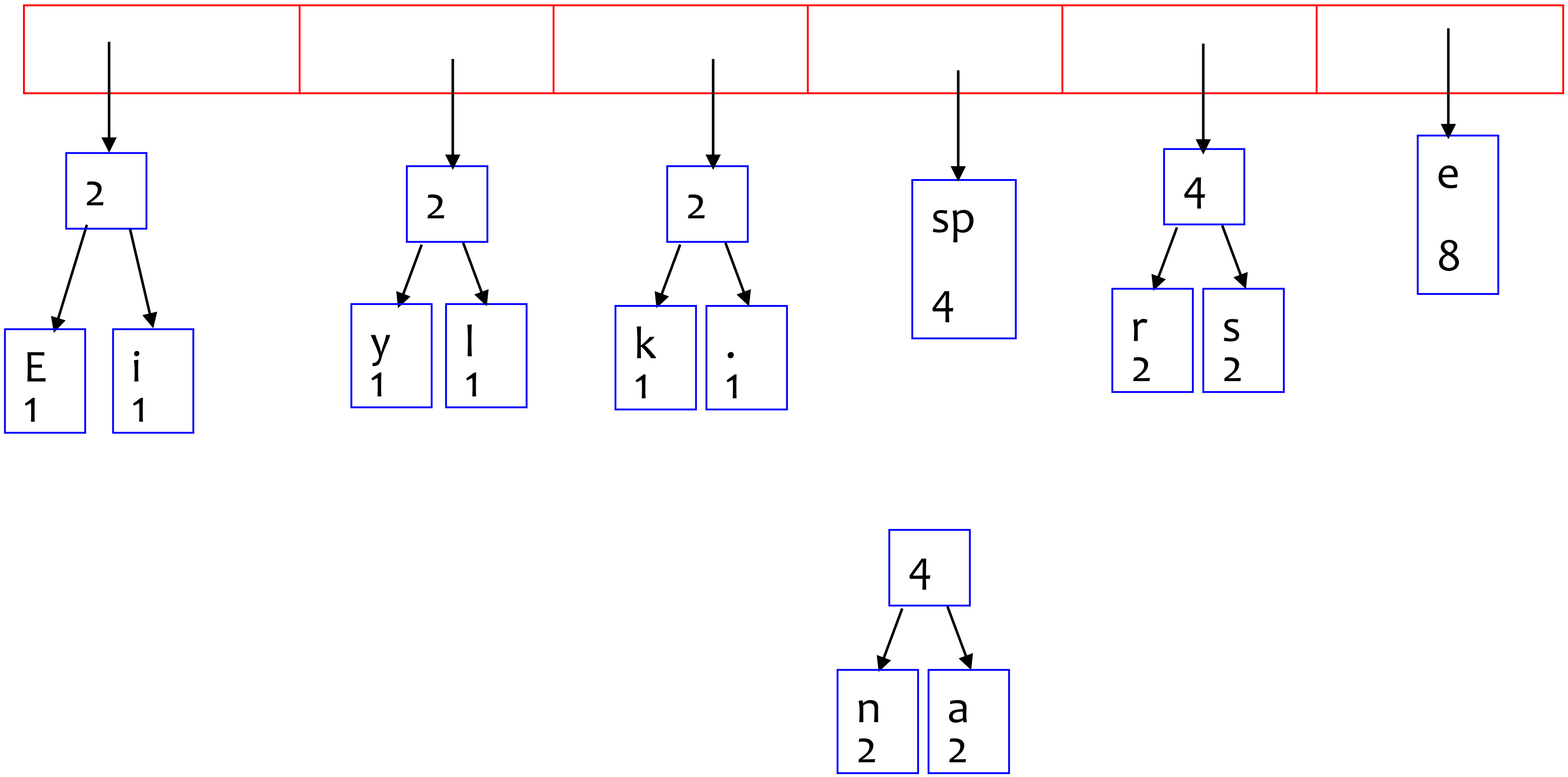
Building a Tree



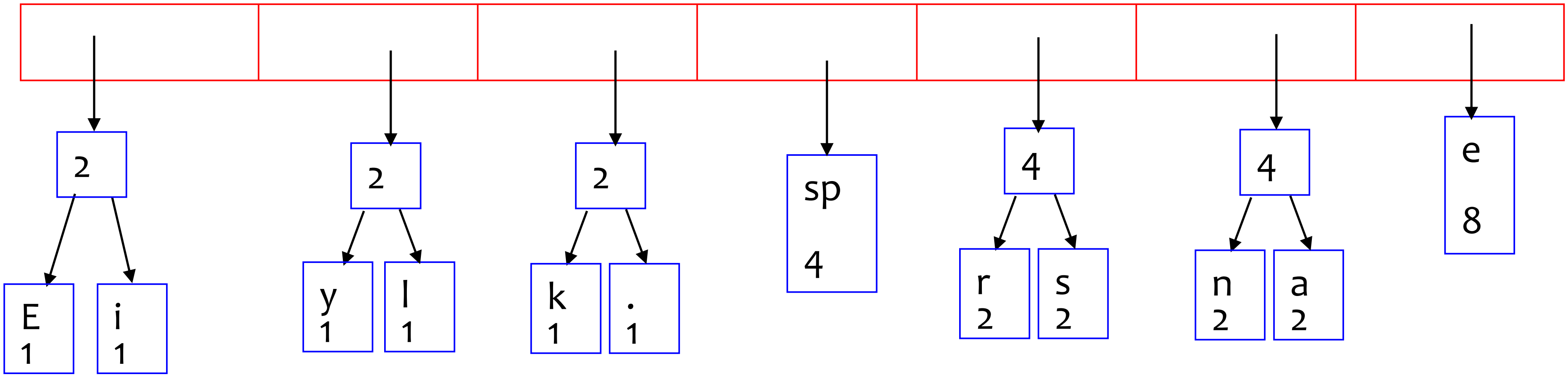
Building a Tree



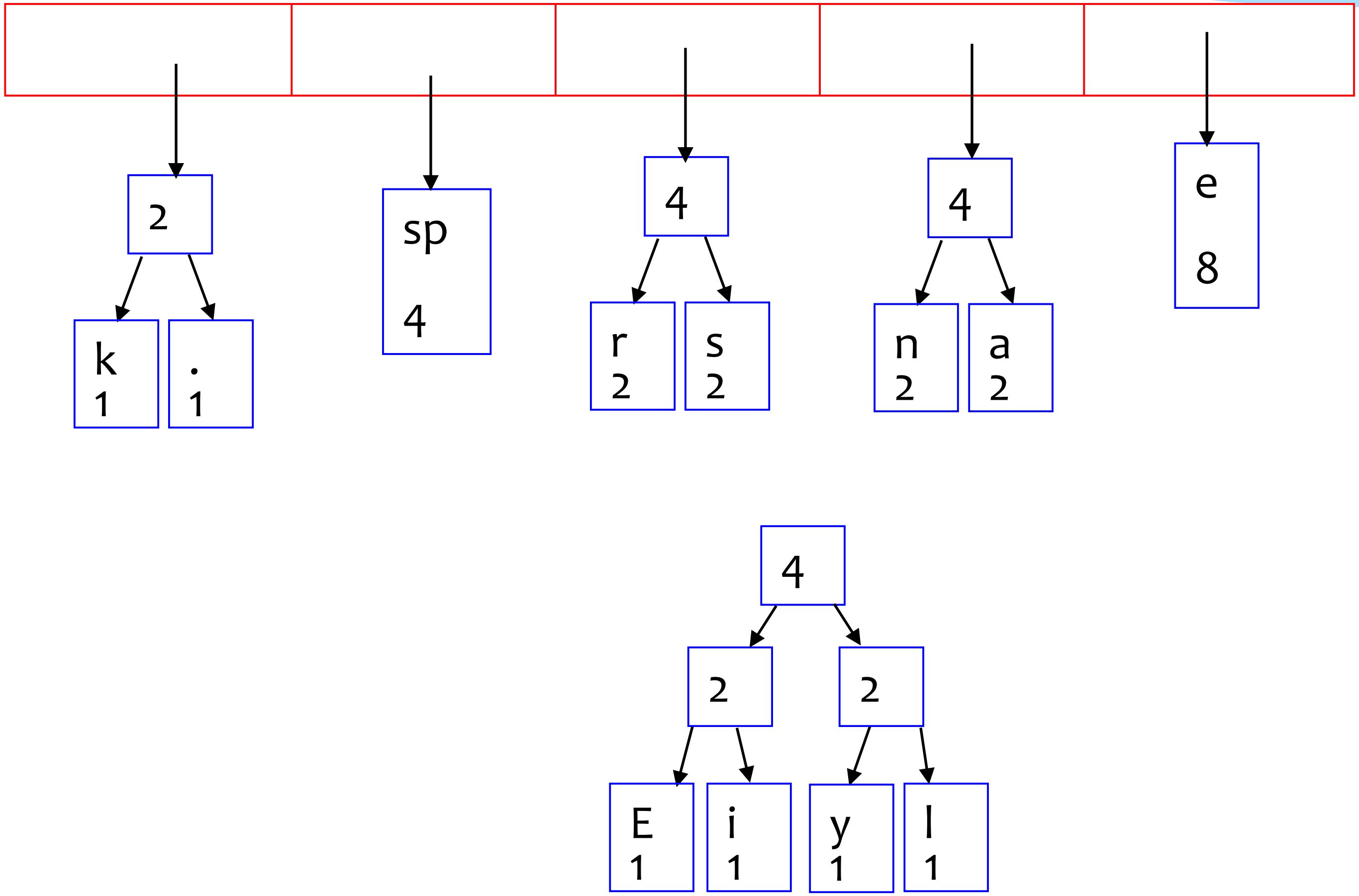
Building a Tree



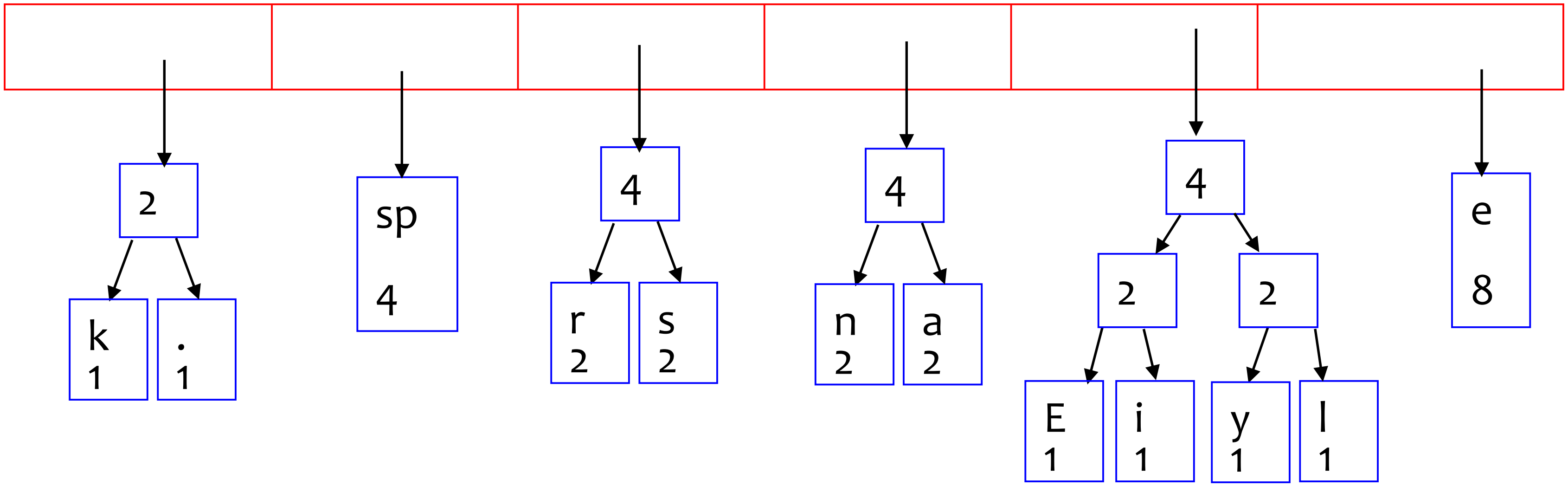
Building a Tree



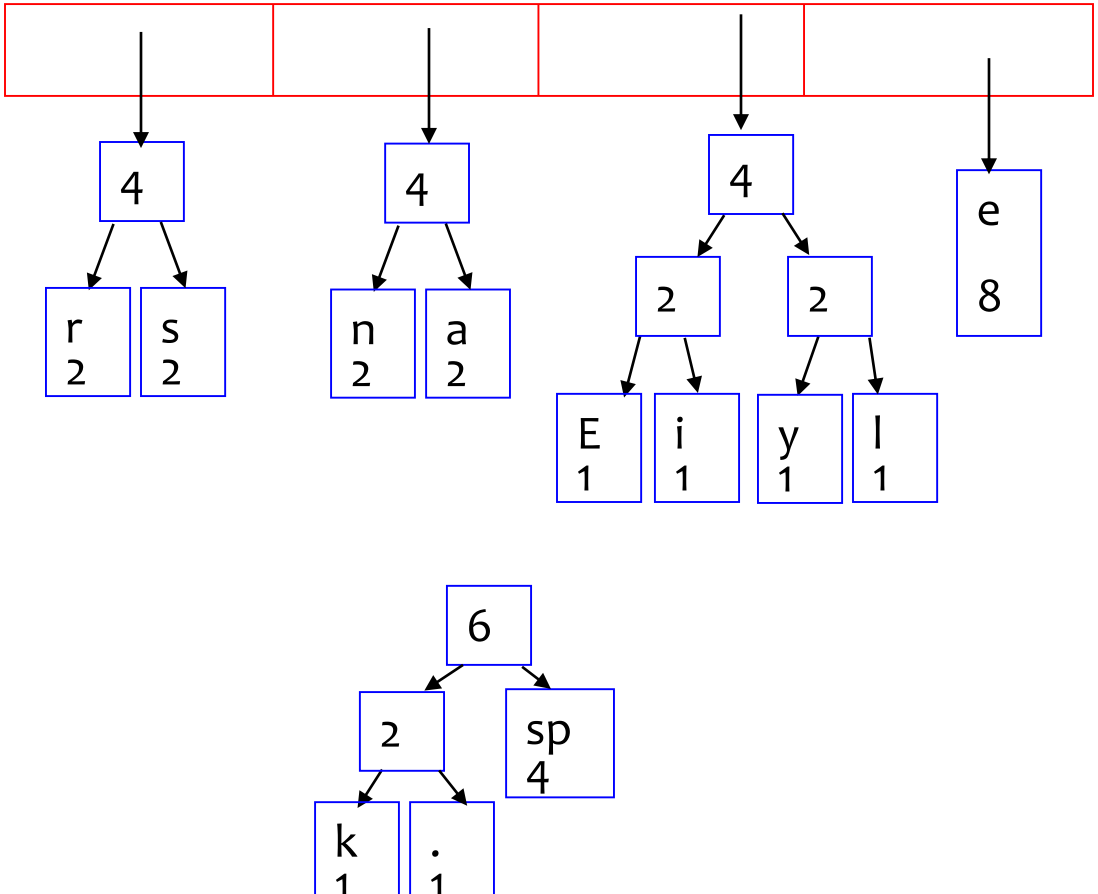
Building a Tree



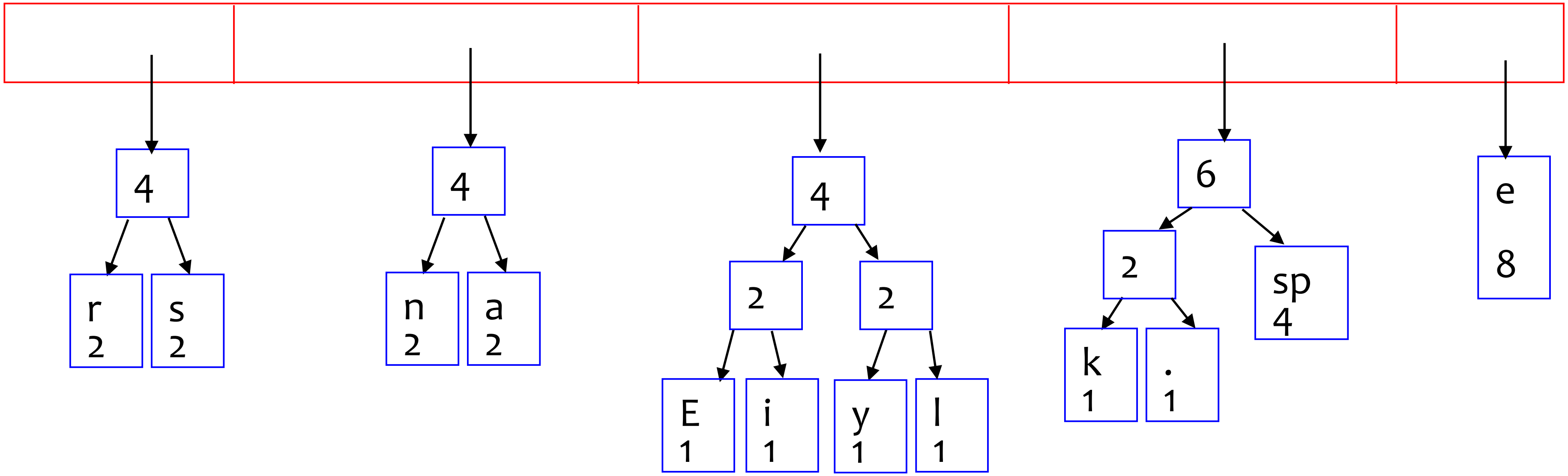
Building a Tree



Building a Tree

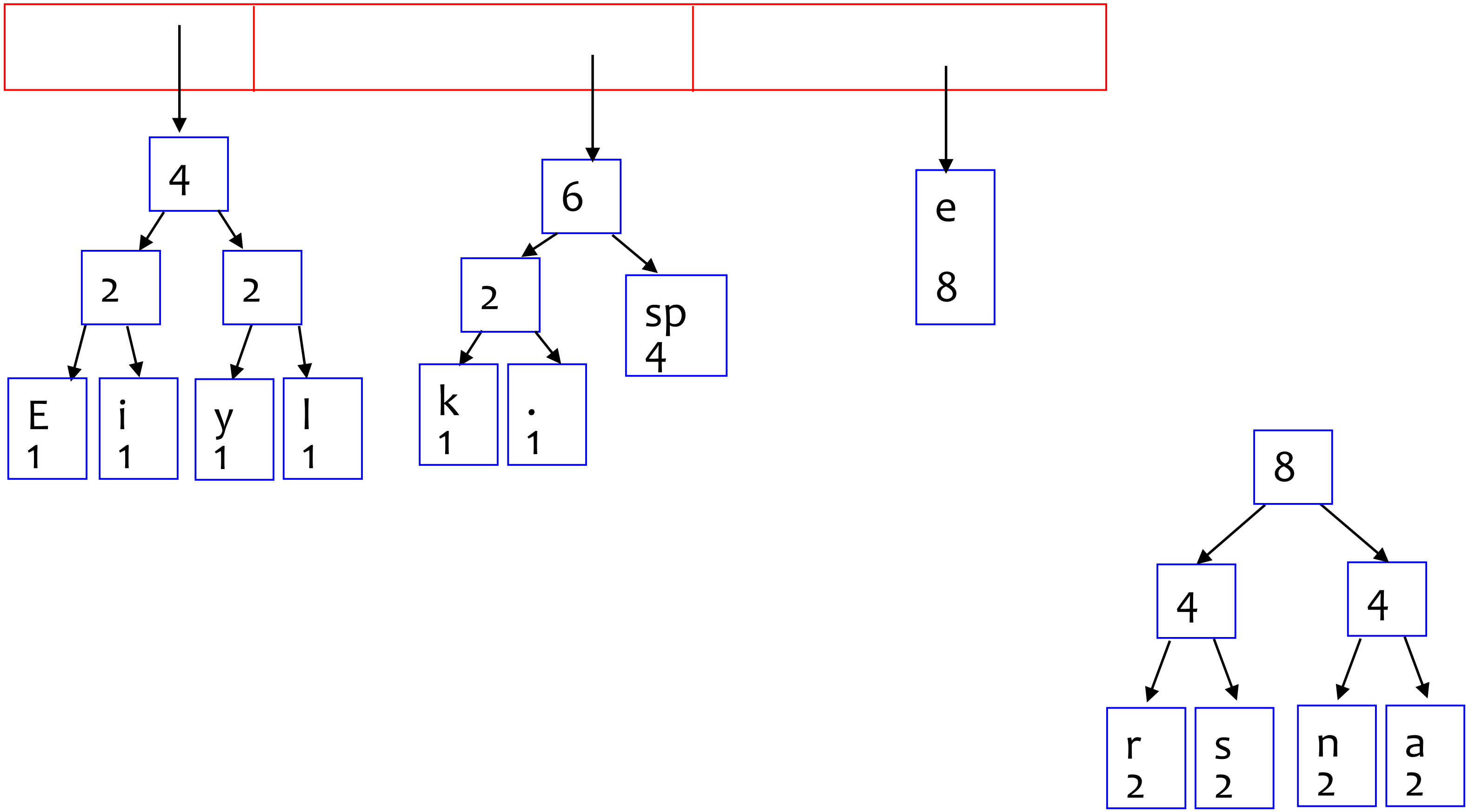


Building a Tree

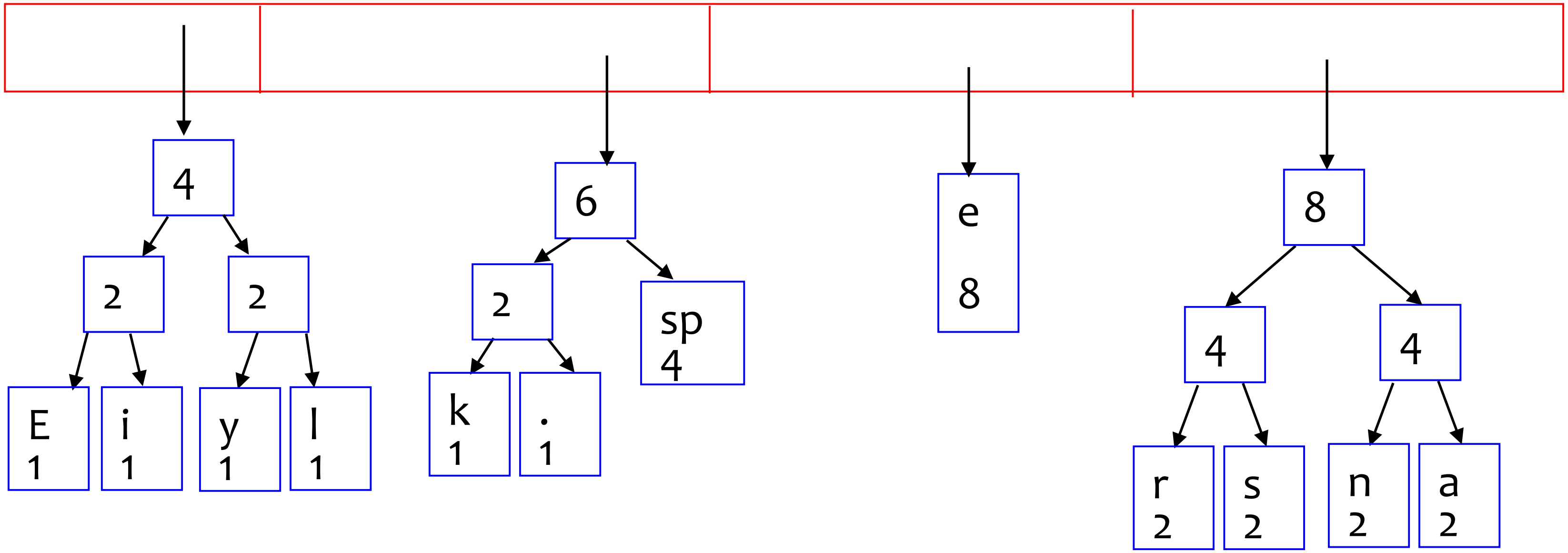


What is happening to the characters with a low number of occurrences?

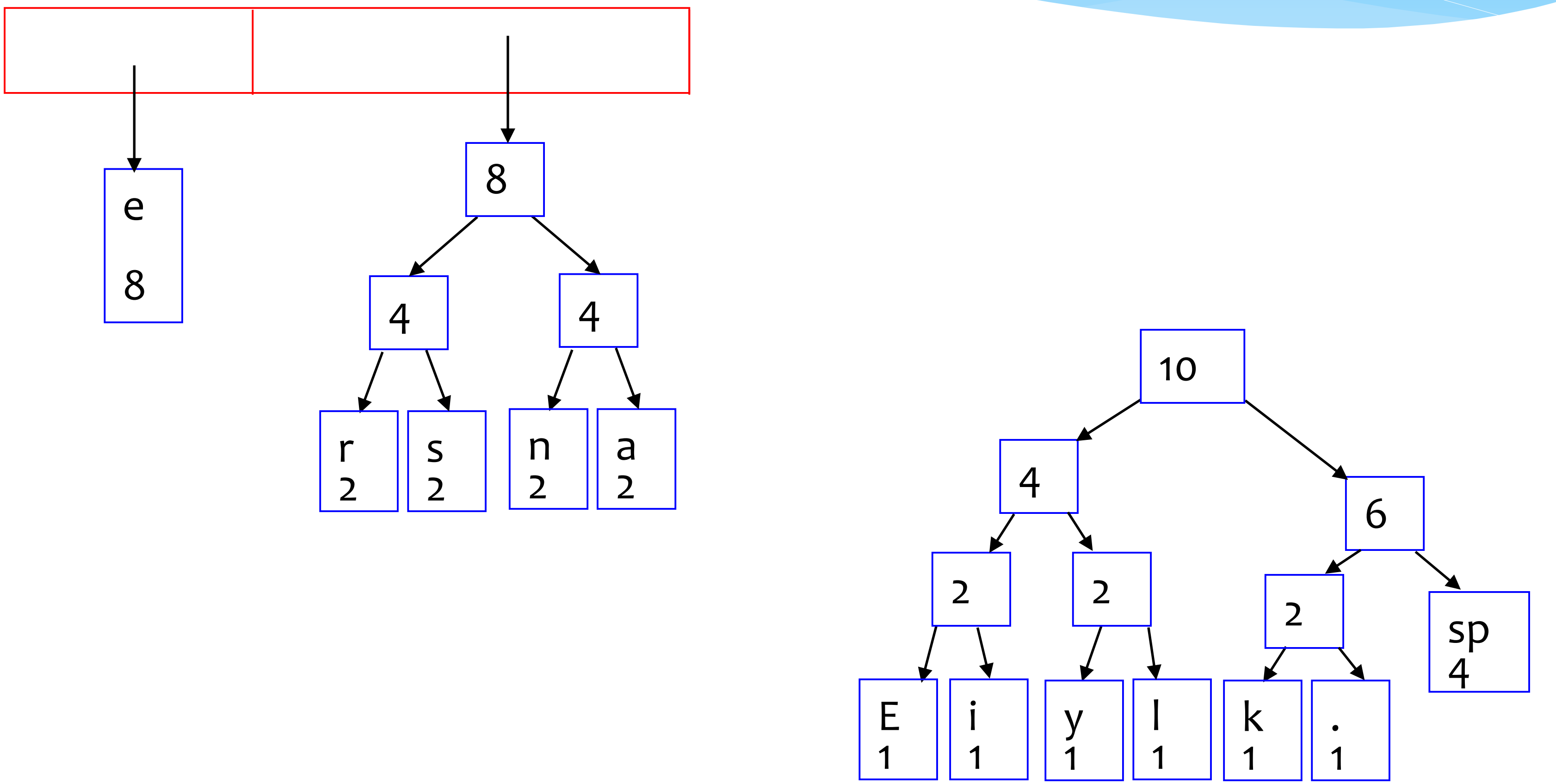
Building a Tree



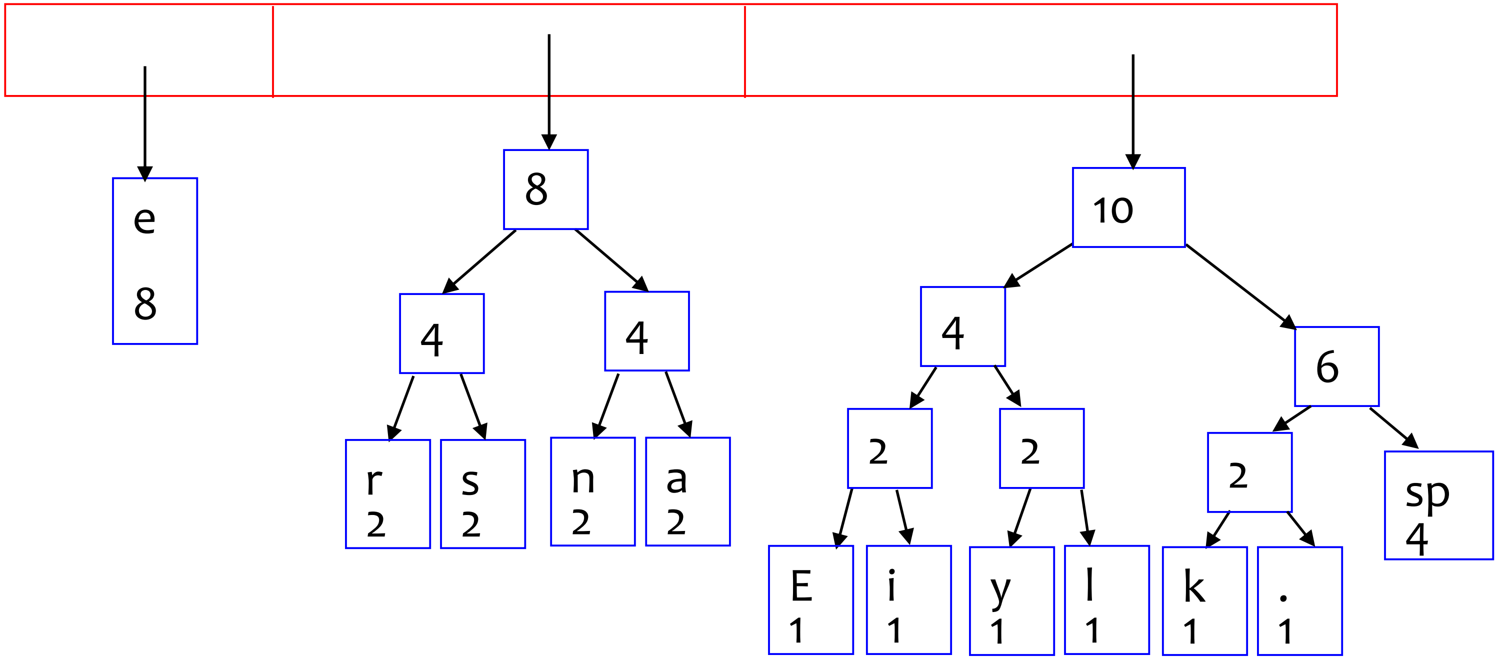
Building a Tree



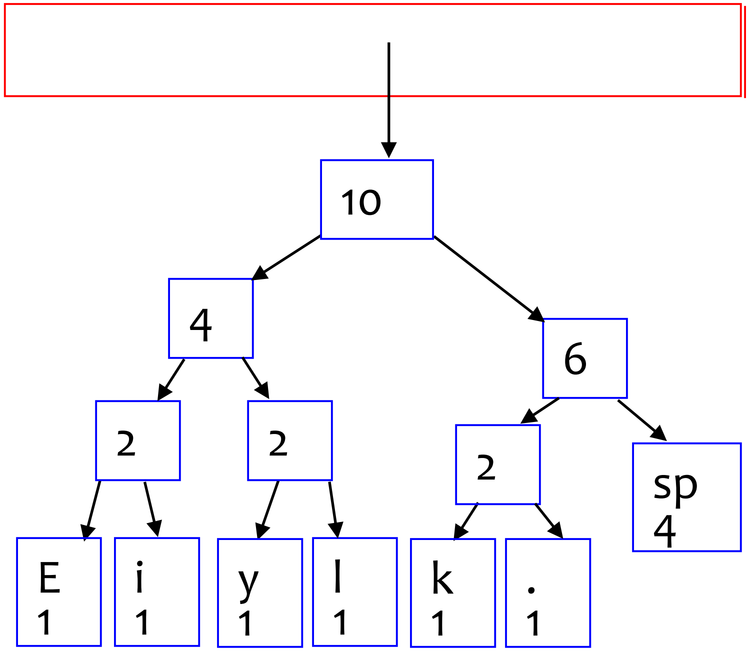
Building a Tree

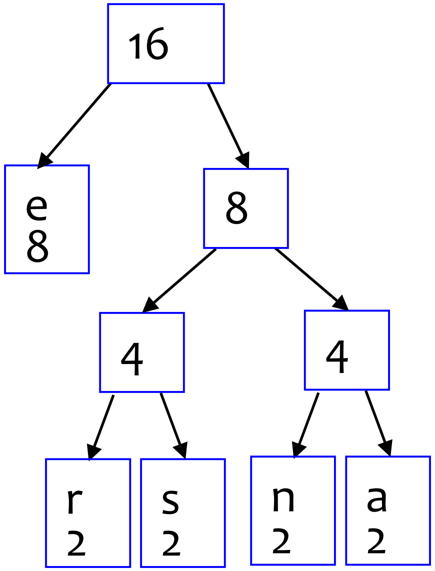


Building a Tree

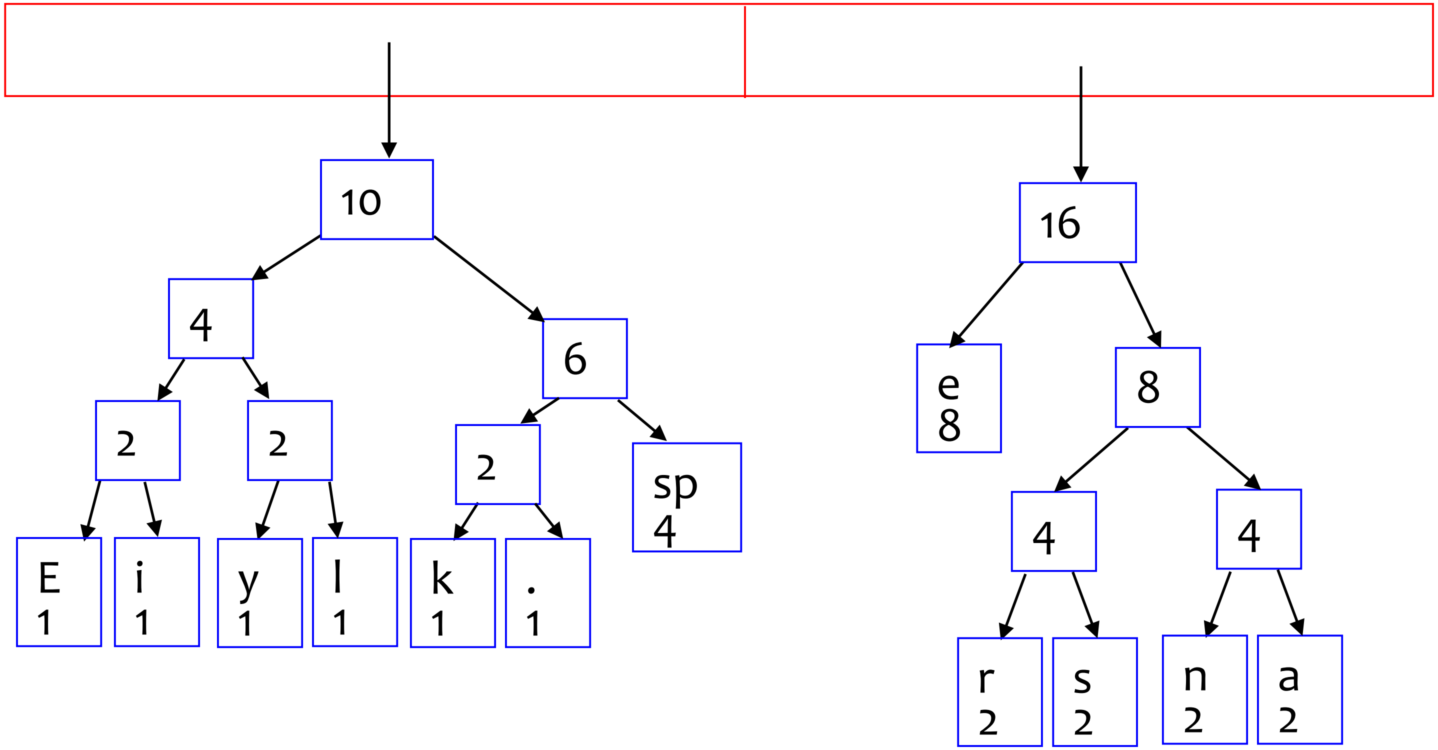


Building a Tree

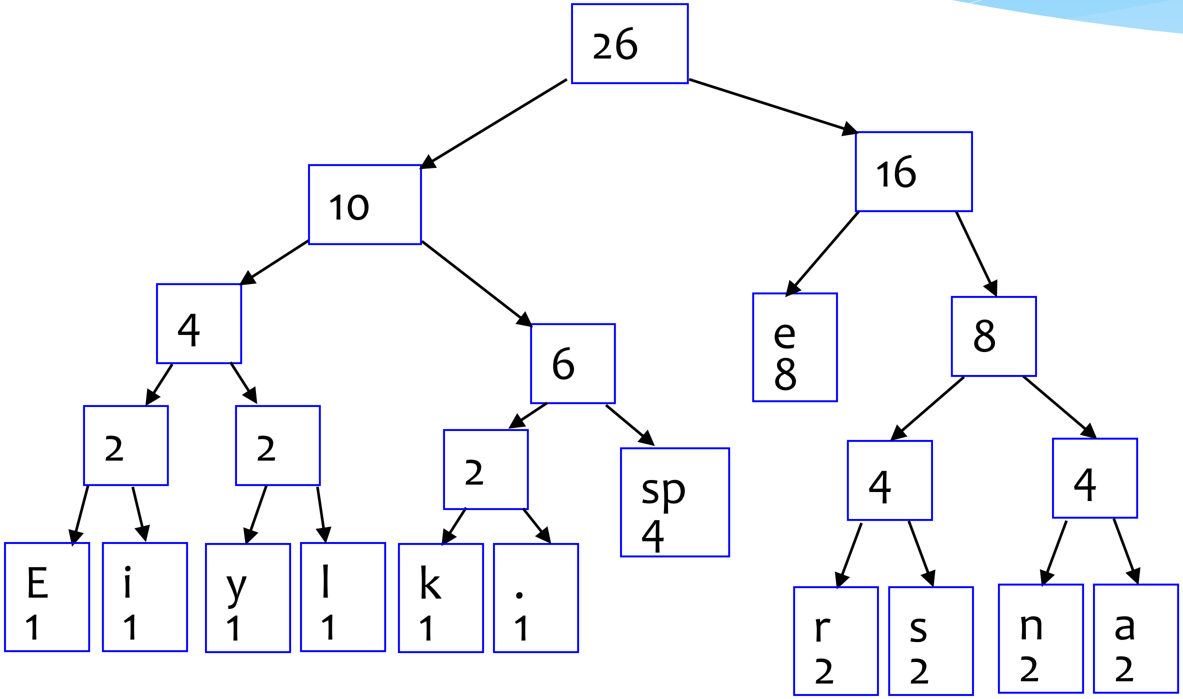




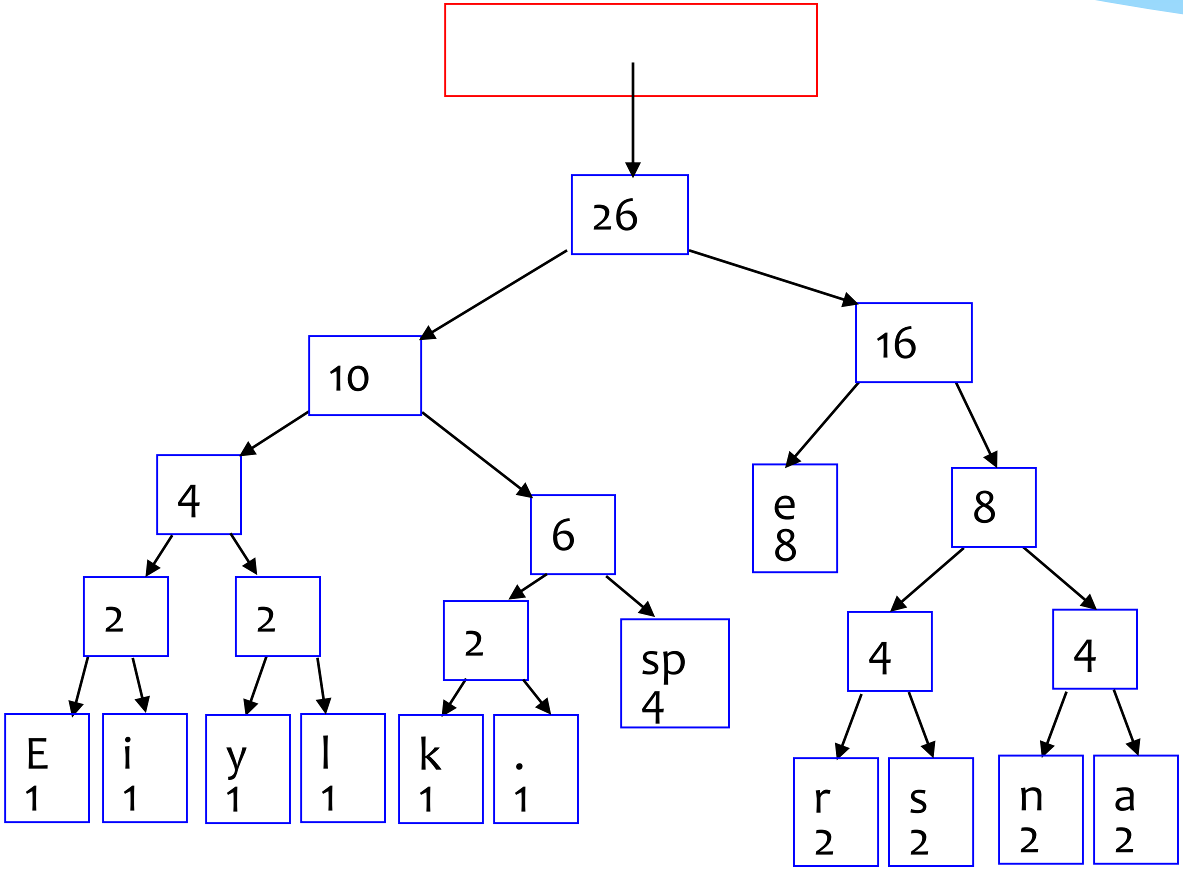
Building a Tree



Building a Tree



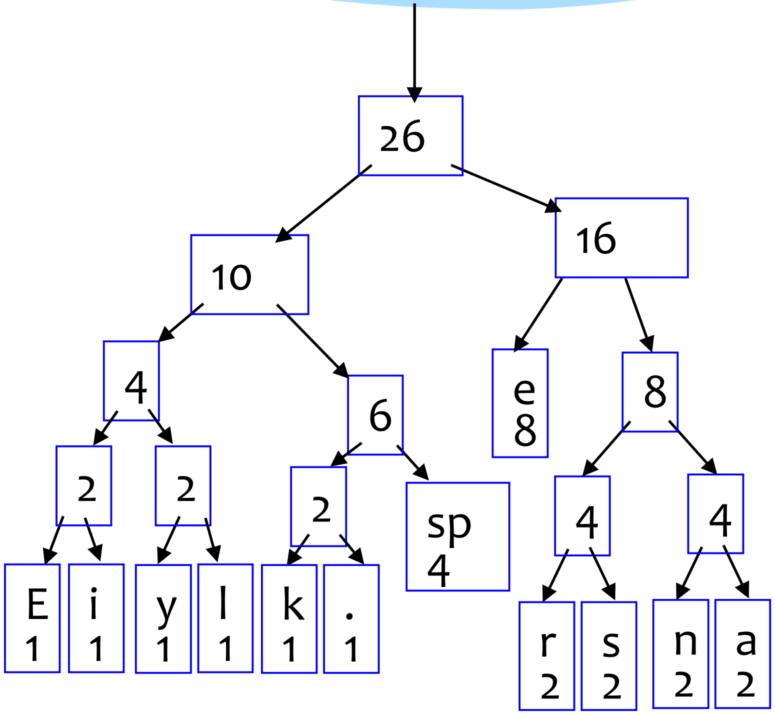
Building a Tree



After enqueueing this node there is only one node left in priority queue.

Building a Tree

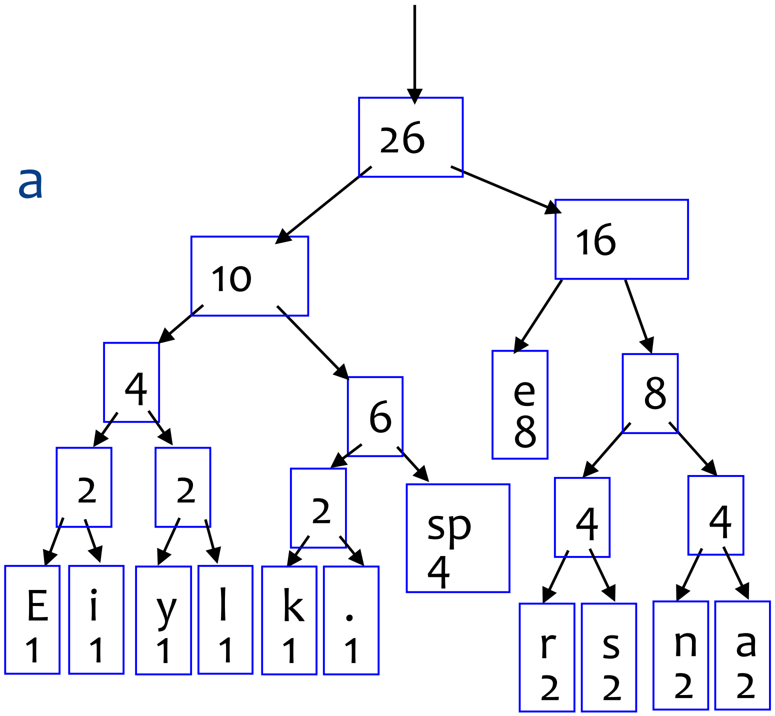
- Dequeue the single node left in the queue.
- This tree contains the new code words for each character.
- Frequency of root node should equal number of characters in text.



Eerie eyes seen near lake. 🗐 26 characters

Encoding the File
Traverse Tree for Codes

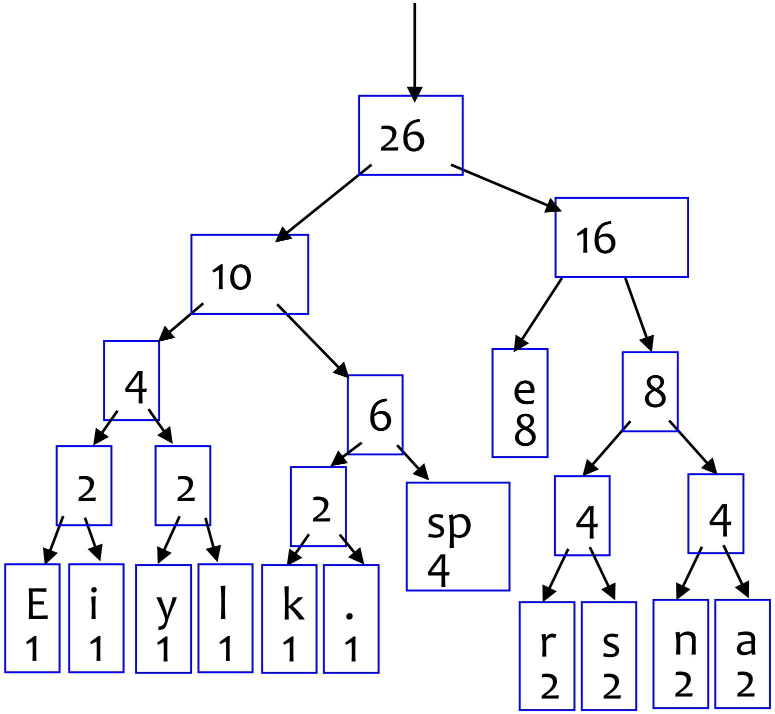
- Perform a traversal of the tree to obtain new code words
- Going left is a 0 going right is a 1
- Code word is only completed when a leaf node is reached



Encoding the File

Traverse Tree for Codes

Char	Code
E	0000
i	0001
y	0010
l	0011
k	0100
.	0101
space	011
e	10
r	1100
s	1101
n	1110
a	1111



Encoding the File

- Rescan text and encode file using new code words

Eerie eyes seen near lake.

```
000010110000001100111000101011011010011111
01011111100011001111110100100101
```

- Why is there no need for a separator character?

Char	Code
E	0000
i	0001
y	0010
l	0011
k	0100
.	0101
space	011
e	10
r	1100
s	1101
n	1110
a	1111

Encoding the File

Results

- Have we made things better?
- 73 bits to encode the text
- ASCII would take 8 * 26 = 208 bits

```
000010110000001100111000101011011010011111
01011111100011001111110100100101
```