Multinedia Systems

Lecture 6:Data Compression

Data and is information?

- Can we measure information?
- Consider the two following sentences:

1. There is a traffic jam on I5
2. There is a traffic jam on I5 near Exit 234

- Sentence 2 seems to have more information than that of sentence 1. From the semantic viewpoint, sentence 2 provides more useful information.

What is information?

- It is hard to measure the "semantic" information!
- Consider the following two sentences

1. There is a traffic jam on I5 near Exit 160
2. There is a traffic jam on I5 near Exit 234

- It's not clear whether sentence 1 or 2 would have more information!

What is information?

- Let's attempt at a different definition of information.
  - How about counting the number of letters in the two sentences:

1. **There is a traffic jam on I5 (22 letters)**
2. **There is a traffic jam on I5 near Exit 234 (33 letters)**

What is information?

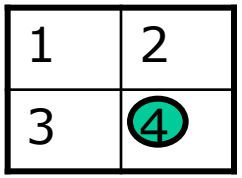
- It's interesting to know that log is the only function that satisfies $f(s^l) = lf(s)$

- **First attempt to quantify information by Hartley (1928).**
  - **Every symbol of the message has a choice of s possibilities.**
  - **A message of length l, therefore can have s^l distinguishable possibilities.**
  - **Information measure is then the logarithm of s^l**

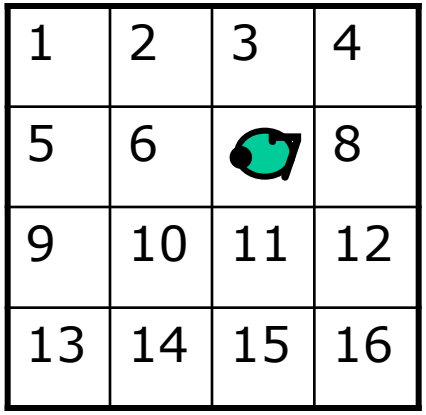
$$I = \log(s^l) = l\log(s)$$

- **Intuitively, this definition makes sense:**
- **one symbol (letter) has the information of $\log(s)$ then a sentence of length l**
- **should have l times more information, i.e. $l\log s$**

How about we measure information as the number of Yes/No questions one has to ask to get the correct answer to a simple game below



- How many questions?
  - 2



- How many questions?
  - 4

- Randomness due to uncerntainty of where the circle is!

Shannon's Information Theory

- Claude Shannon: A Mathematical Theory of Communication
  - Bell System Technical Journal, 1948

- Shannon's measure of information is the number of bits to represent the amount of uncertainty (randomness) in a data source, and is defined as entropy

$$H = -\sum_{i=1}^{n} p_i \log(p_i)$$

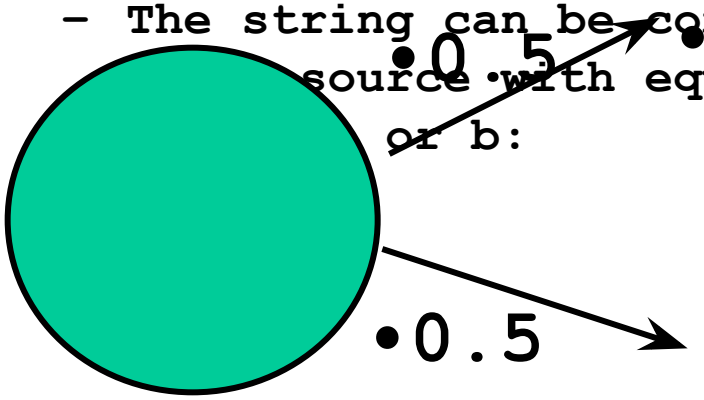
- Where there are n symbols 1, 2, … n, each with p_i

Shannon's Entropy

- Consider the following string consisting of symbols a and b:

  abaabaababbbaabbabab… ….

  - On average, there are equal number of a and b.
  - The string can be considered as an output of a source with equal probability of outputting a or b:



- We want to characterize the average information generated by the source!

Intuition on Shannon's Entropy

- **Why** $H = -\sum_{i=1}^{n} p_i \log(p_i)$

- Suppose you have a long random string of two binary symbols 0 and 1, and the probability of symbols 1 and 0 are p_1 and p_0
- Ex: 00100100101101001100001000100110001 ….
- If any string is long enough say N, it is likely to contain Np_0 0's and Np_1 1's. The probability of this string pattern occurs is equal to

$$p = p_0^{Np_0} p_1^{Np_1}$$

- Hence, # of possible patterns is

$$1/p = p_0^{-Np_0} p_1^{-Np_1}$$

- # bits to represent all possible patterns is

$$\log(p_0^{-Np_0} p_1^{-Np_1}) = -\sum_{i=0}^{1} Np_i \log p_i$$

- The average # of bits to represent the symbol is therefore

$$-\sum_{i=0}^{1} p_i \log p_i$$

More Intuition on Entropy

- Assume a binary memoryless source, e.g., a flip of a coin. How much information do we receive when we are told that the outcome is *heads*?
  - If it's a fair coin, i.e., $P(heads) = P(tails) = 0.5$, we say that the *amount of information is 1 bit*.
  - If we already know that it will be (or was) heads, i.e., $P(heads) = 1$, the *amount of information is zero!*
  - If the coin is not fair, e.g., $P(heads) = 0.9$, the *amount of information is more than zero but less than one bit!*
  - Intuitively, the amount of information received is the same if $P(heads) = 0.9$ or $P(heads) = 0.1$.

Self Information

- So, let's look at it the way Shannon did.
- Assume a memoryless source with
  - alphabet $\mathbf{A} = (a_1, \ldots, a_n)$
  - symbol probabilities $(p_1, \ldots, p_n)$.
- How much information do we get when finding out that the next symbol is a_i?
- According to Shannon the self information of a_i is

$$I(a_i) = \log \frac{1}{p_i}$$

Why?

- Assume two independent events A and B, with probabilities $P(A) = p_A$ and $P(B) = p_B$.
- For both the events to happen, the probability is $p_A \not\subset p_B$. However, the amount of information should be added, not multiplied.

$$I(p_A \cdot p_B) = I(p_A) + I(p_B)$$

- Logarithms satisfy this!

$$I(A) = \log(p_A)?$$

- No, we want the information to increase with decre

$$I(A) = -\log(p_A) = \log \frac{1}{p_A}$$

negative logarithm.

Self Information

- Example

$$p_i = 1 \Rightarrow I(1) = \log \frac{1}{1} = 0$$

- Example

$$p_i = 0.5 \Rightarrow I(0.5) = \log_2 \frac{1}{0.5} = 1 \text{ [bit]}$$

- *Which logarithm?* **Pick the one you like! If you pick the natural log, you'll measure in *nats*, if you pick the**

Self Information

- On *average over all the symbols*, we get:

$$H = \sum_1^N p_i I\left(a_i \atop p_i\right)$$

- $H(X)$ is called the first order *entropy* of the source.
- This can be regarded as the degree of *uncertainty* about the following symbol.

Entropy

- Example: Binary Memoryless Source

•0 1 1 0 1 0 0 0

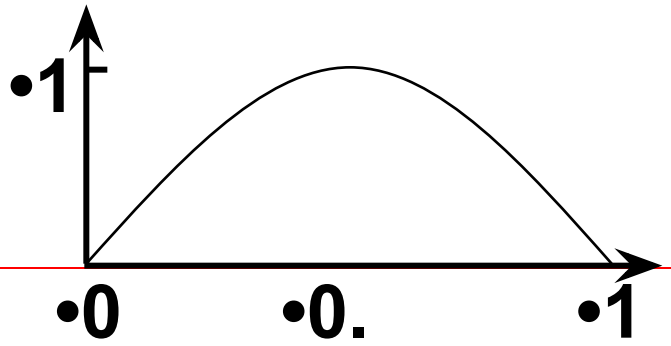
- Let

$$p = P(X_k = 1)$$
$$q = P(X_k = 0) = 1 - p$$

- Then

$$H = p\log\frac{1}{p} + (1-p)\log\frac{1}{1-p}$$

•Often denoted $h(p)$



- The uncertainty (information) is largest when $p = q = \frac{1}{2}$

Example

- Three symbols a, b, c with corresponding probabilities:
- P = {0.5, 0.25, 0.25}
- What is H(P)?
- Three weather conditions in Corvallis: Rain, sunny, cloudy with corresponding probabilities:
- Q = {0.48, 0.32, 0.20}
- What is H(Q)?

Entropy: Three properties

1. It can be shown that $0 \cdot H \cdot \log N$.

2. **Maximum entropy** ($H = \log N$) is reached when all symbols are **equiprobable**, i.e.,
   $p_i = 1/N$.

3. The difference $\log N - H$ is called the *redundancy* of the source.

Encoding and Compression of Data

- Fax Machines
- ASCII
- Variations on ASCII
  - min number of bits needed
  - cost of savings
  - patterns
  - modifications

- **Definition**: Compression means storing data in a format that requires less space than usual.

- Data compression is particularly useful in communications because it enables devices to transmit the same amount of data in fewer bits.

- **The bandwidth of a digital communication link can be effectively increased by compressing data at the sending end and decompressing data at the receiving end.**

- **There are a variety of data compression techniques, but only a few have been standardized.**

Types of Data Compression

- There are two main types of data compression : Lossy and Lossless.

- In Lossy data compression the message can never be recovered exactly as it was before it was compressed.

- In a Lossless data compression file the original message can be exactly decoded.

- Lossless compression is ideal for text.

- Huffman coding is type of lossless data compression.

Compression Algorithms

- Huffman Coding
- Run Length Encoding
- Arithmetic Codes
- Lempel ziv 77 and Variants

Huffman Coding

- Huffman coding is a popular compression technique that assigns variable length codes (VLC) to symbols, so that the most frequently occurring symbols have the shortest codes.

- On decompression the symbols are reassigned their original fixed length codes.

- **The idea is to use short bit strings to represent the most frequently used characters**

- **and to use longer bit strings to represent less frequently used characters.**

- That is, the most common characters, usually space, e, and t are assigned the shortest codes.

- In this way the total number of bits required to transmit the data can be considerably less than the number required if the fixed length ASCII representation is used.

- A Huffman code is a binary tree with branches assigned the value 0 or 1.

Huffman Coding

- **Proposed by Dr. David A. Huffman in 1952**
  - ***"A Method for the Construction of Minimum Redundancy Codes"***
- **Applicable to many forms of data transmission**
  - **Our example: text files**

The Basic Algorithm

- Huffman coding is a form of statistical coding
- Not all characters occur with the same frequency!
- Yet all characters are allocated the same amount of space
  - 1 char = 1 byte, be it e or x

The Basic Algorithm

- Any savings in tailoring codes to frequency of character?
- Code word lengths are no longer fixed like ASCII.
- Code word lengths vary and will be shorter for the more frequently used characters.

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

- Consider the following short text:

- *Eerie eyes seen near lake.*

- Count up the occurrences of all characters in the text

Building a Tree

Scan the original text

Eerie eyes seen near lake.

- What characters are present?

E e r i space

y s n a r l k .

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	1
e	8	s	2	.	1
r	2	n	2		
i	1	a	2		
space	4	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

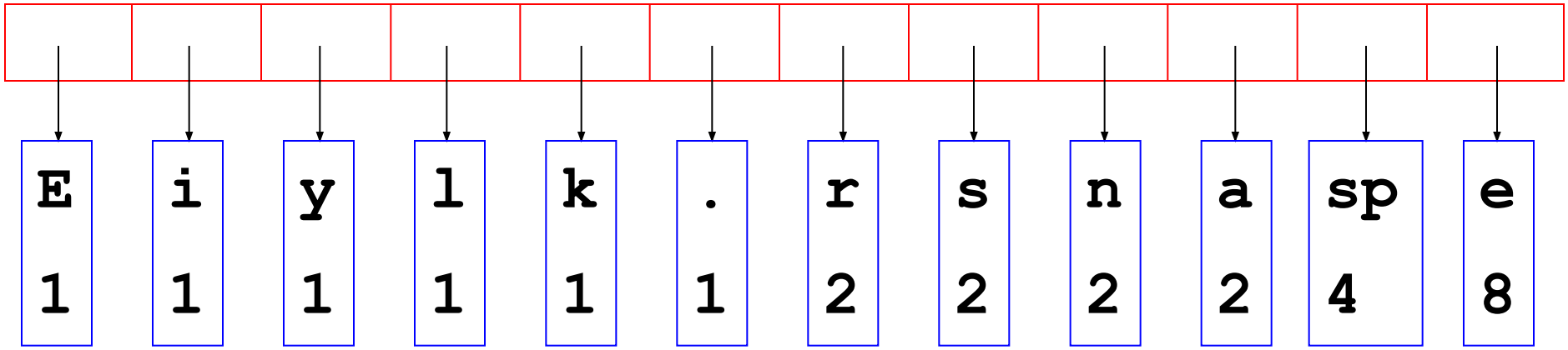
- Uses binary tree nodes

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

priorityQueue myQueue;
```

Building a Tree

- The queue after inserting all nodes

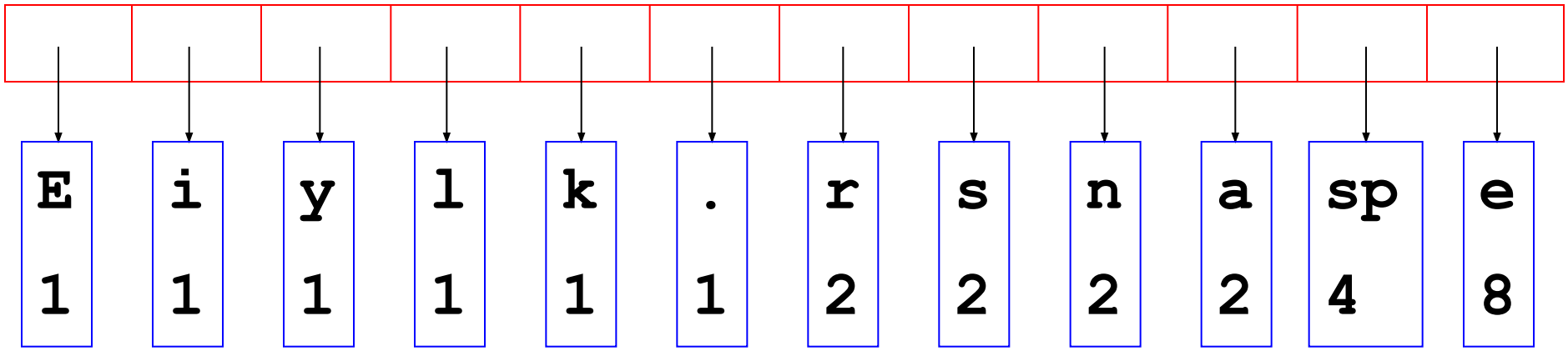


- Null Pointers are not shown

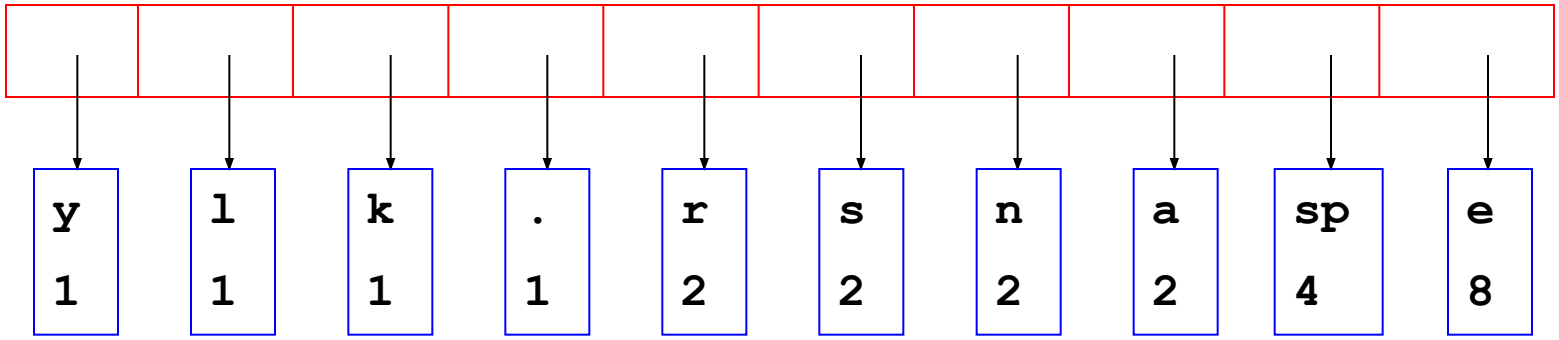
Building a Tree

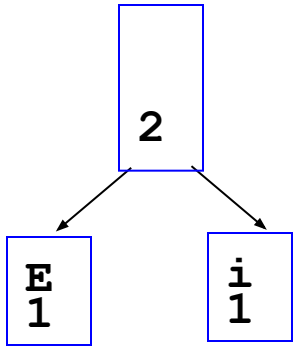
- While priority queue contains two or more nodes
  - Create new node
  - Dequeue node and make it left subtree
  - Dequeue next node and make it right subtree
  - Frequency of new node equals sum of frequency of left and right children
  - Enqueue new node back into queue

Building a Tree

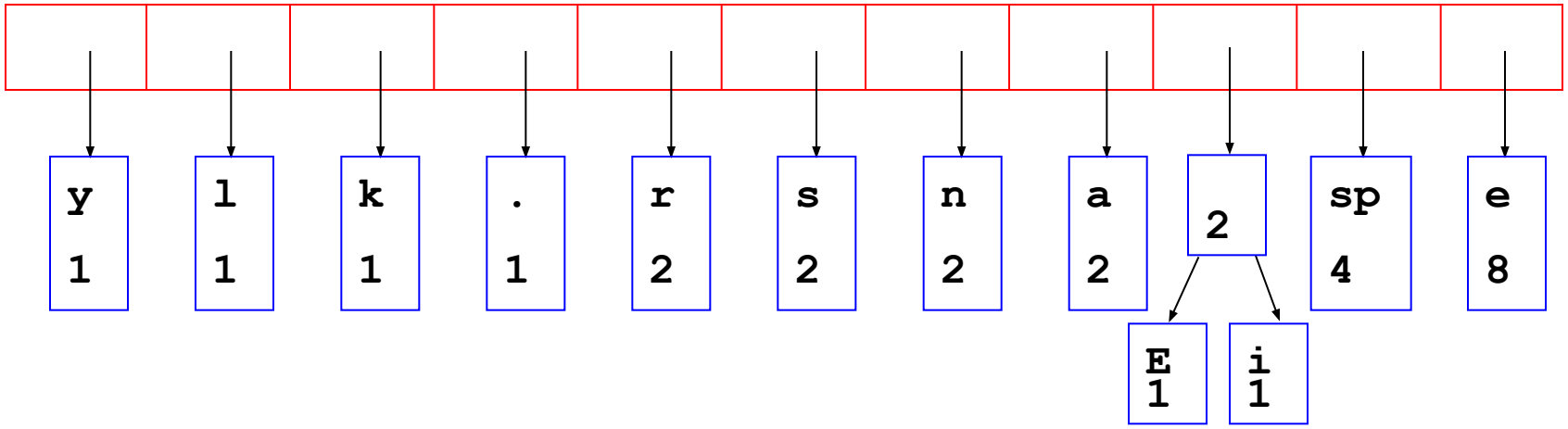


Building a Tree

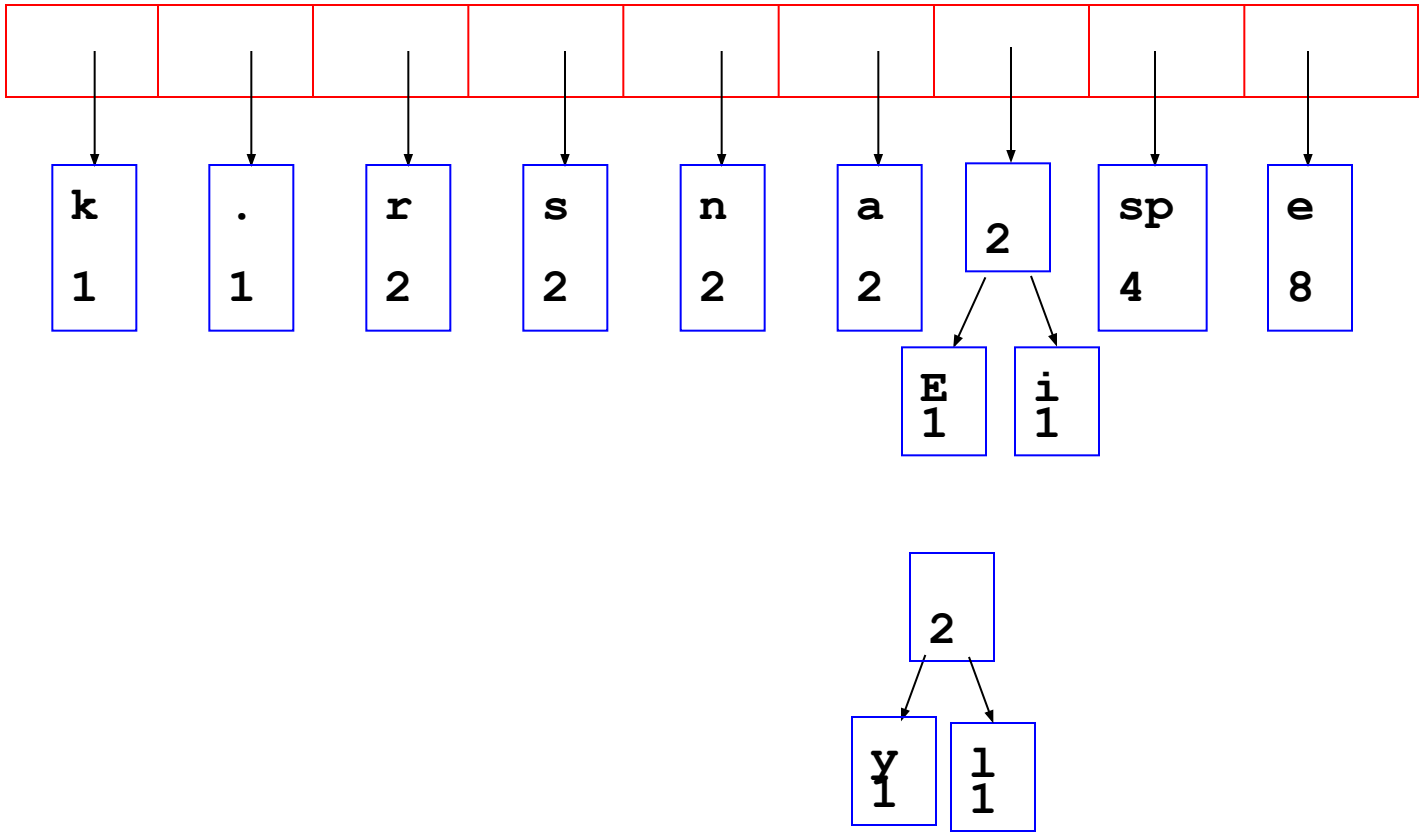




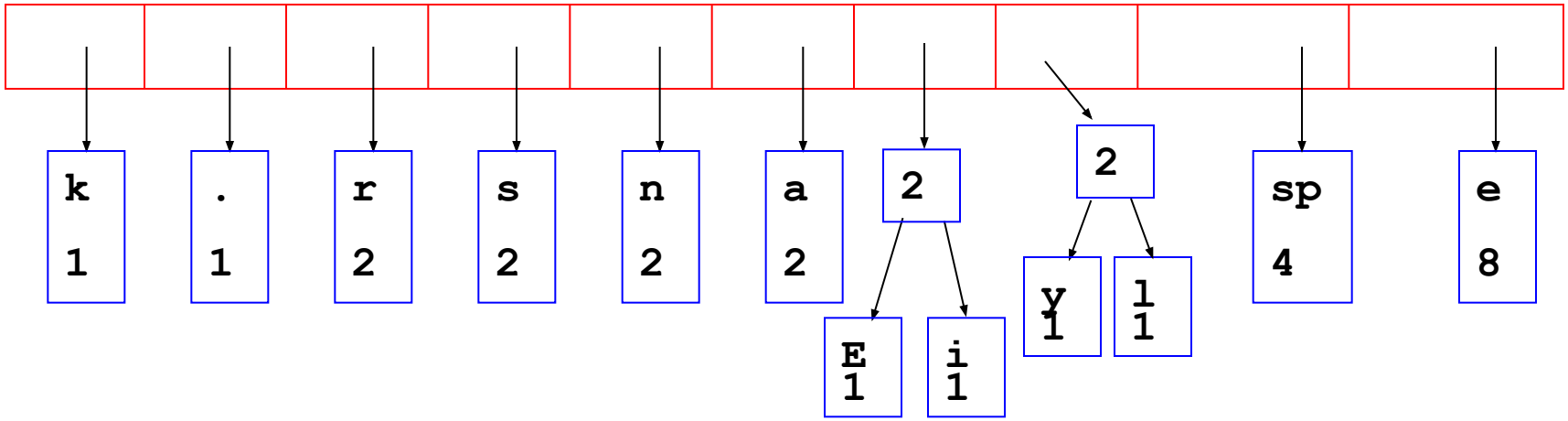
Building a Tree



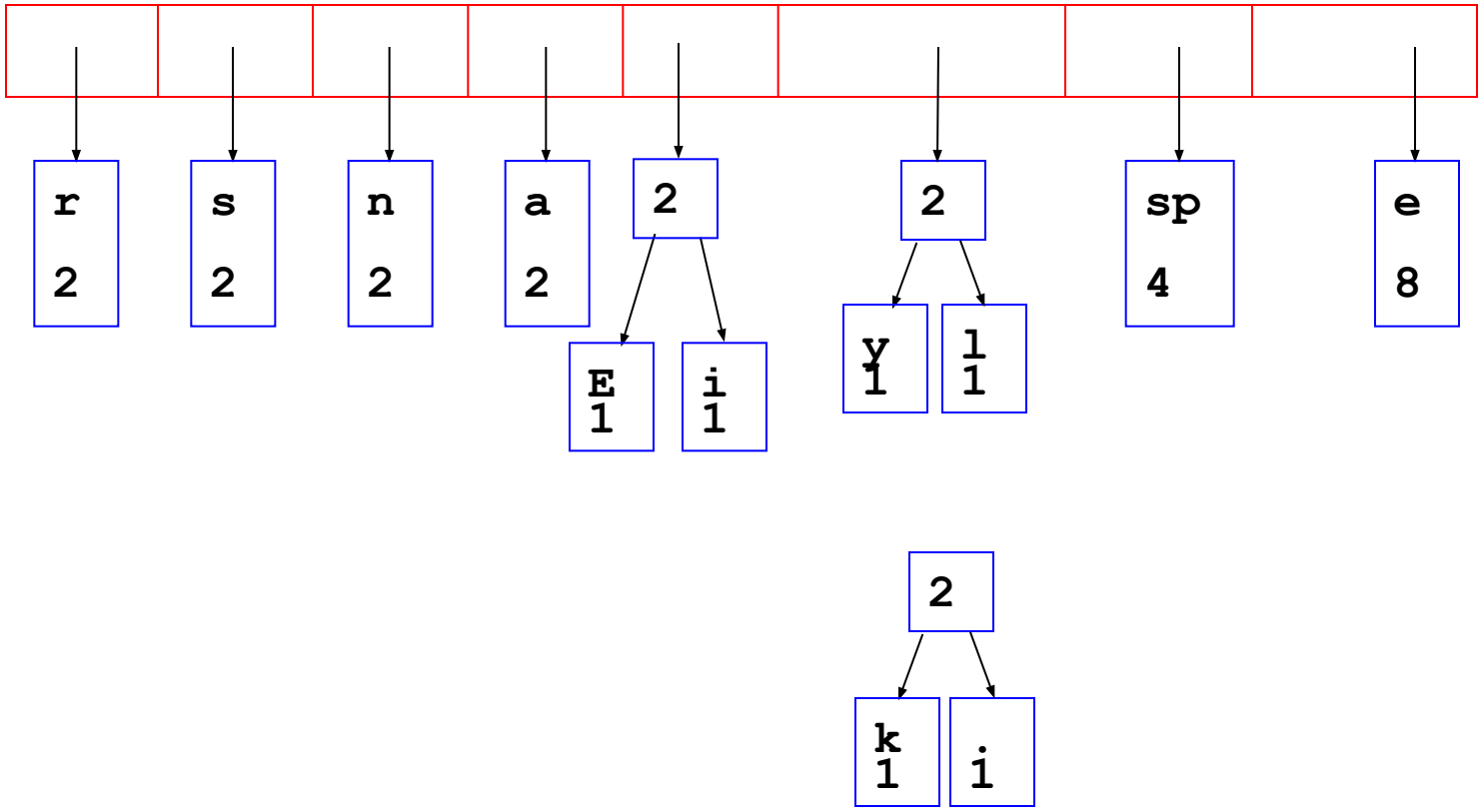
Building a Tree



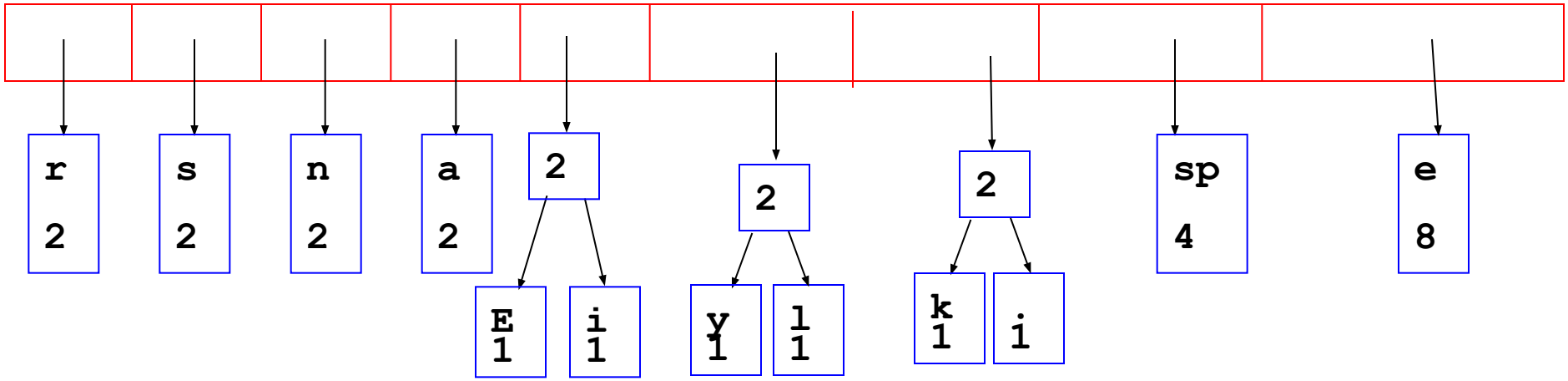
Building a Tree



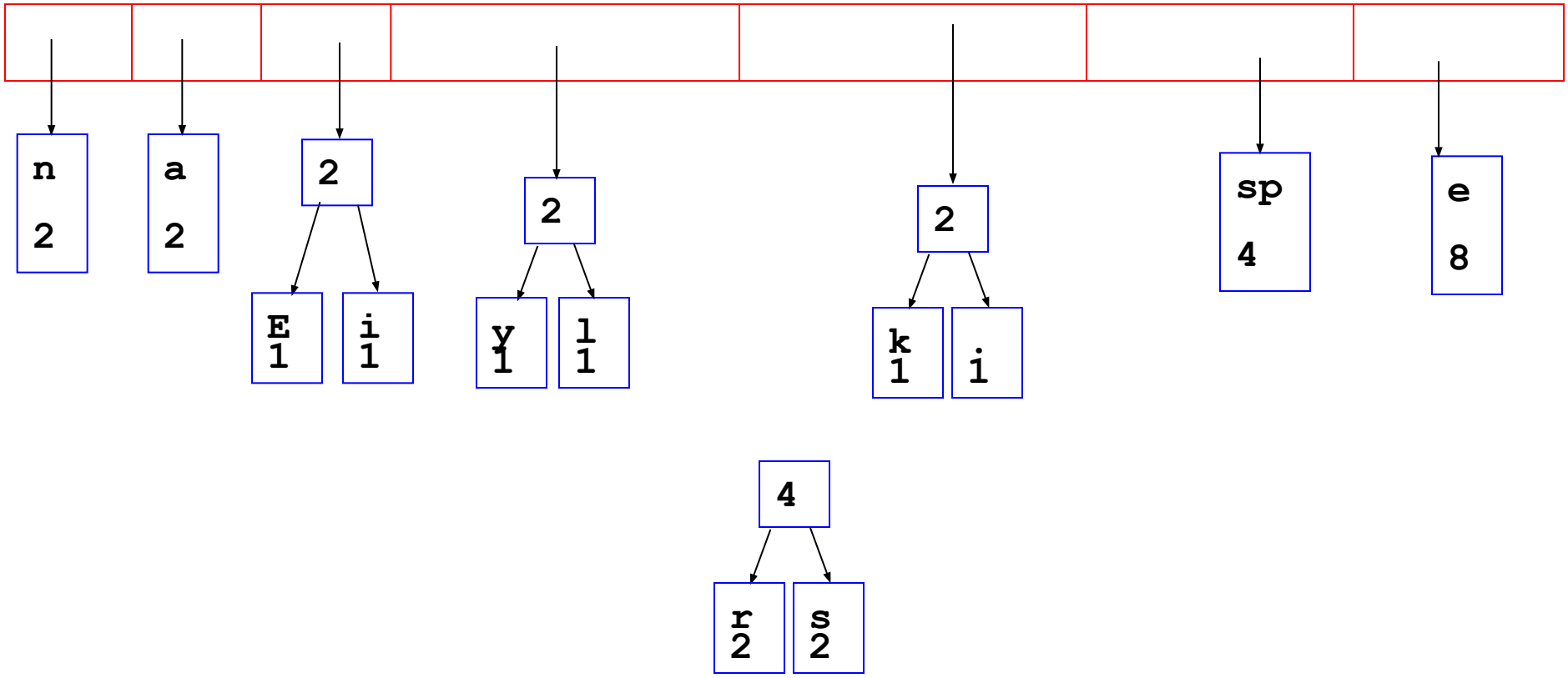
Building a Tree



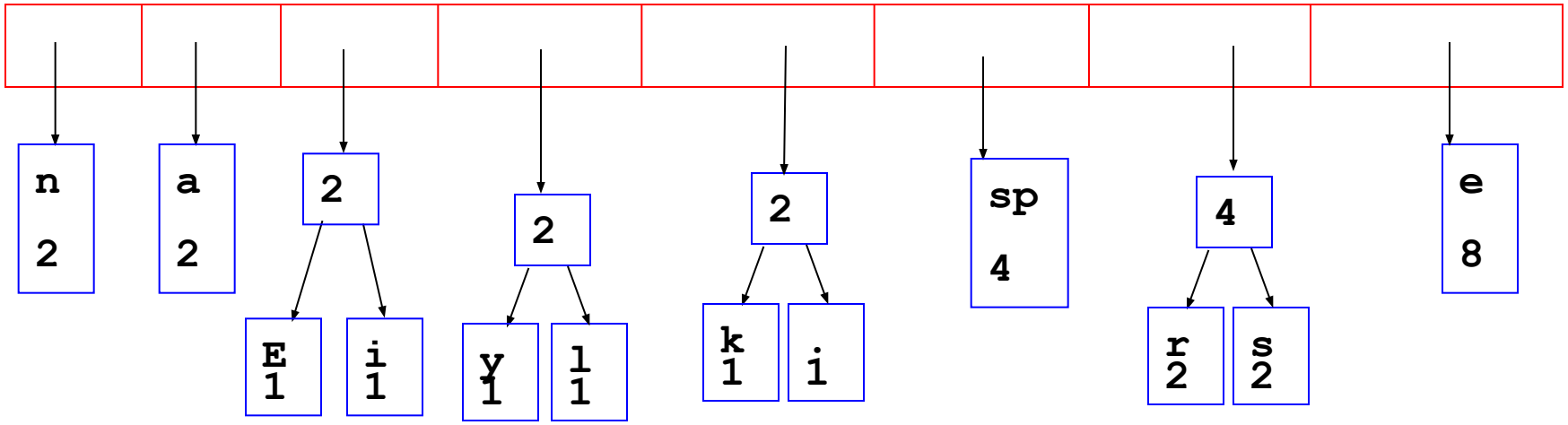
Building a Tree



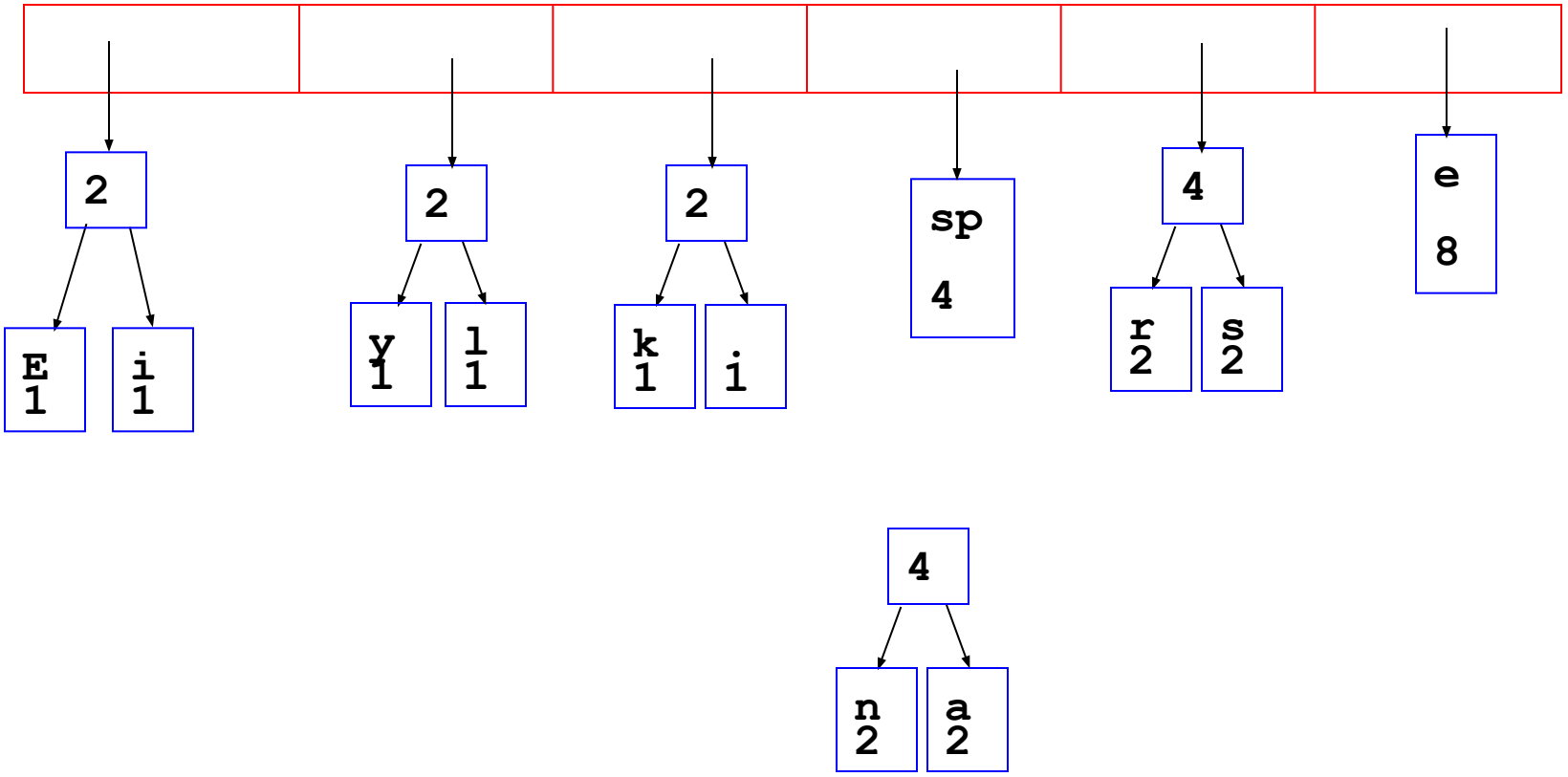
Building a Tree



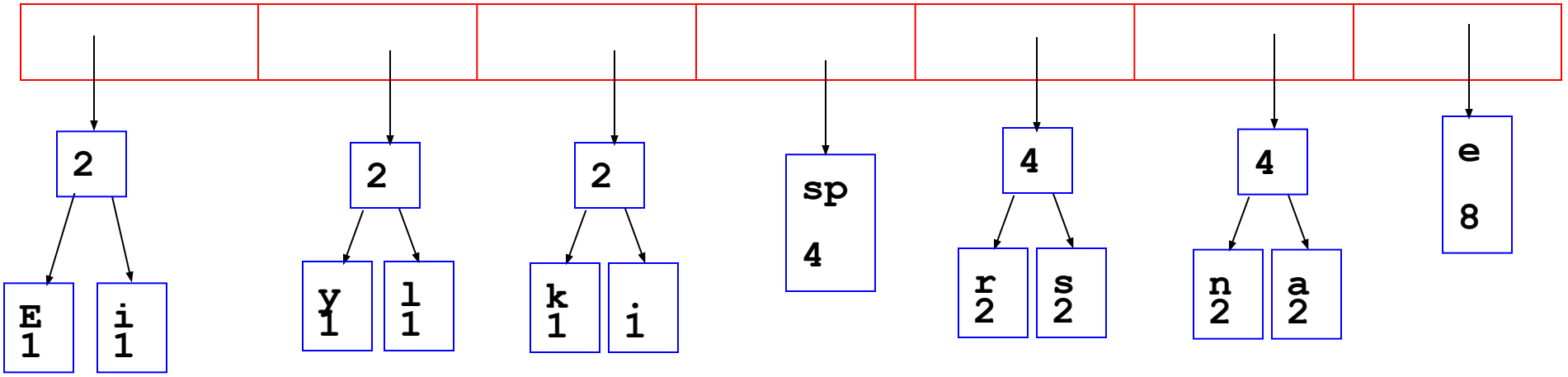
Building a Tree



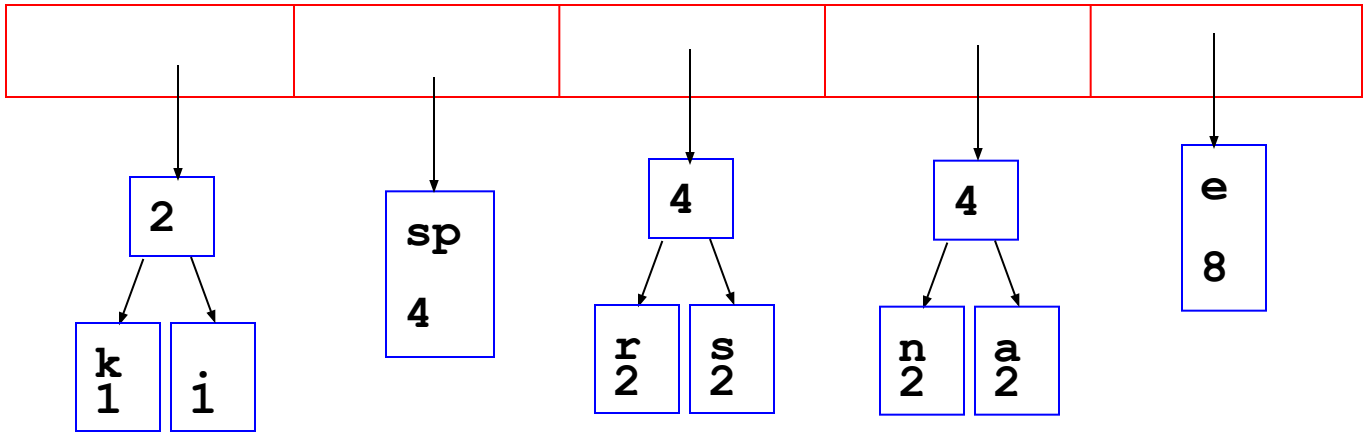
Building a Tree

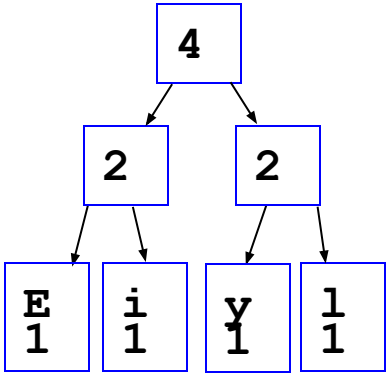


Building a Tree

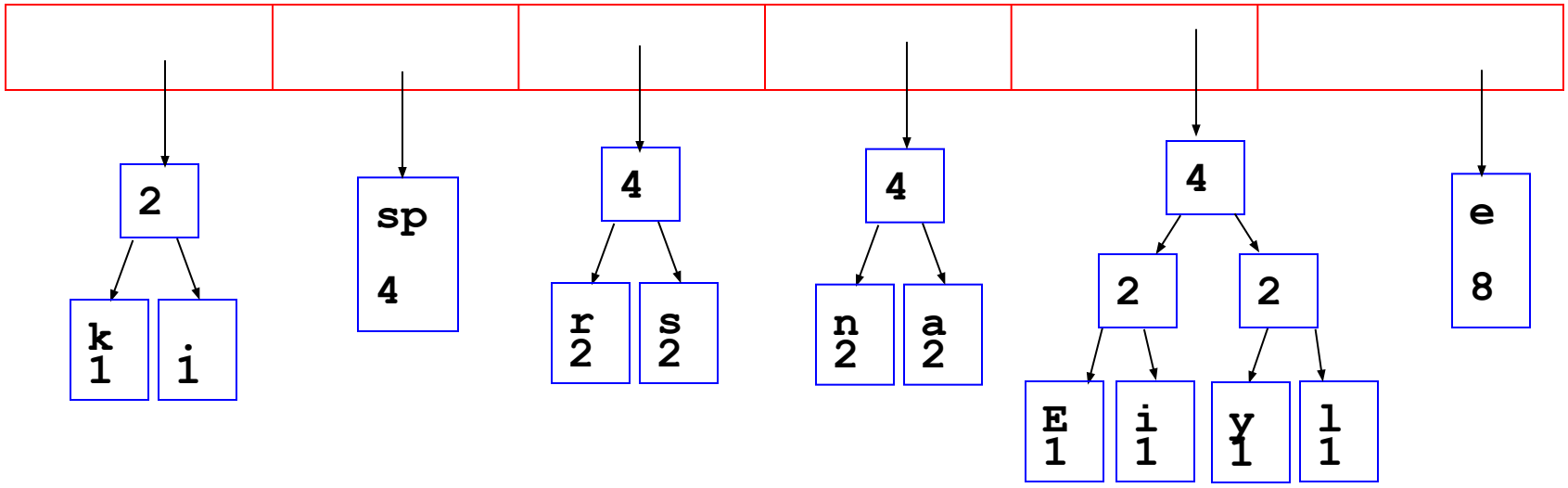


Building a Tree

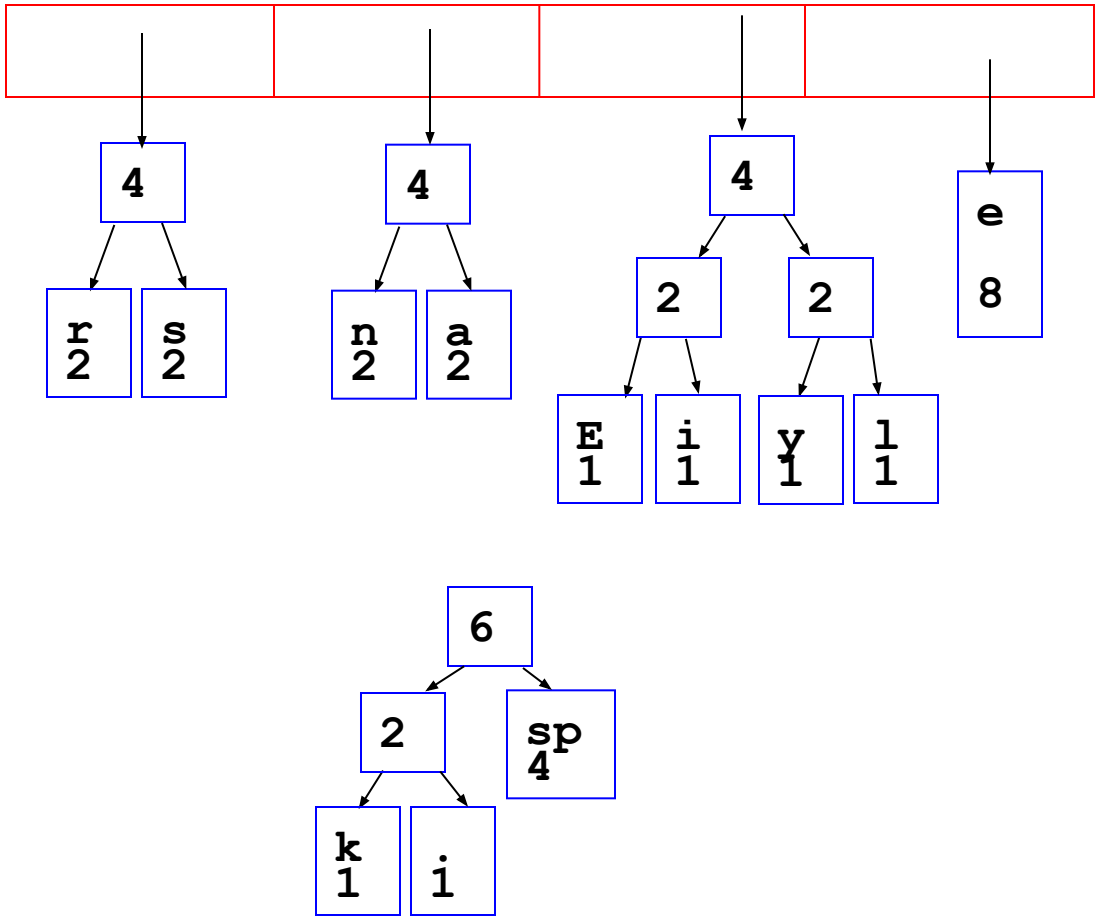




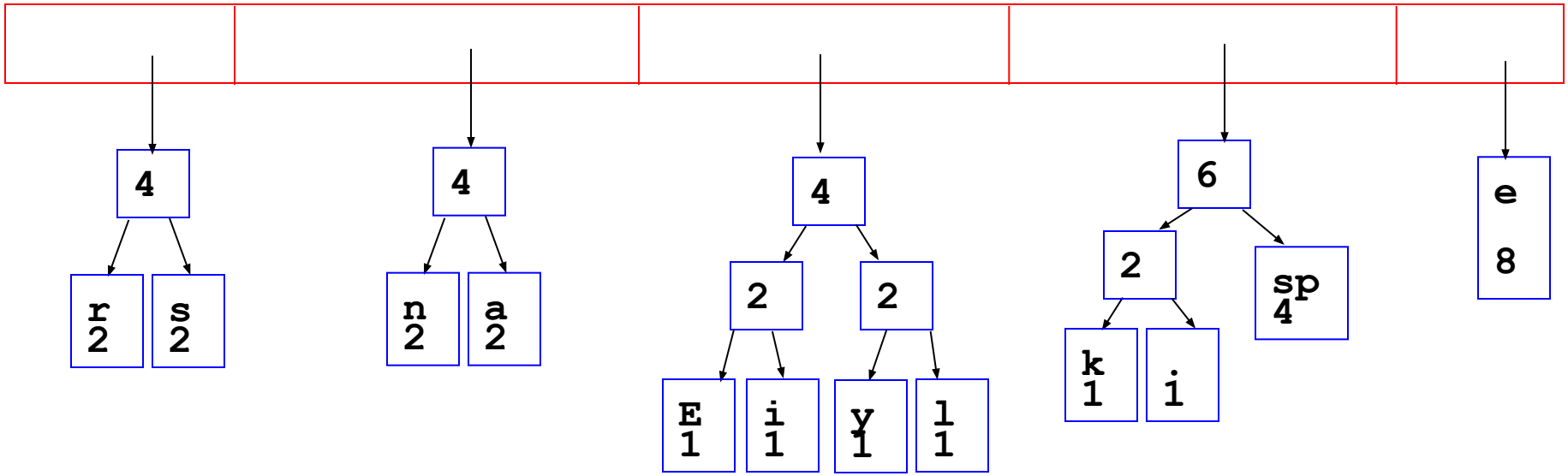
Building a Tree



Building a Tree

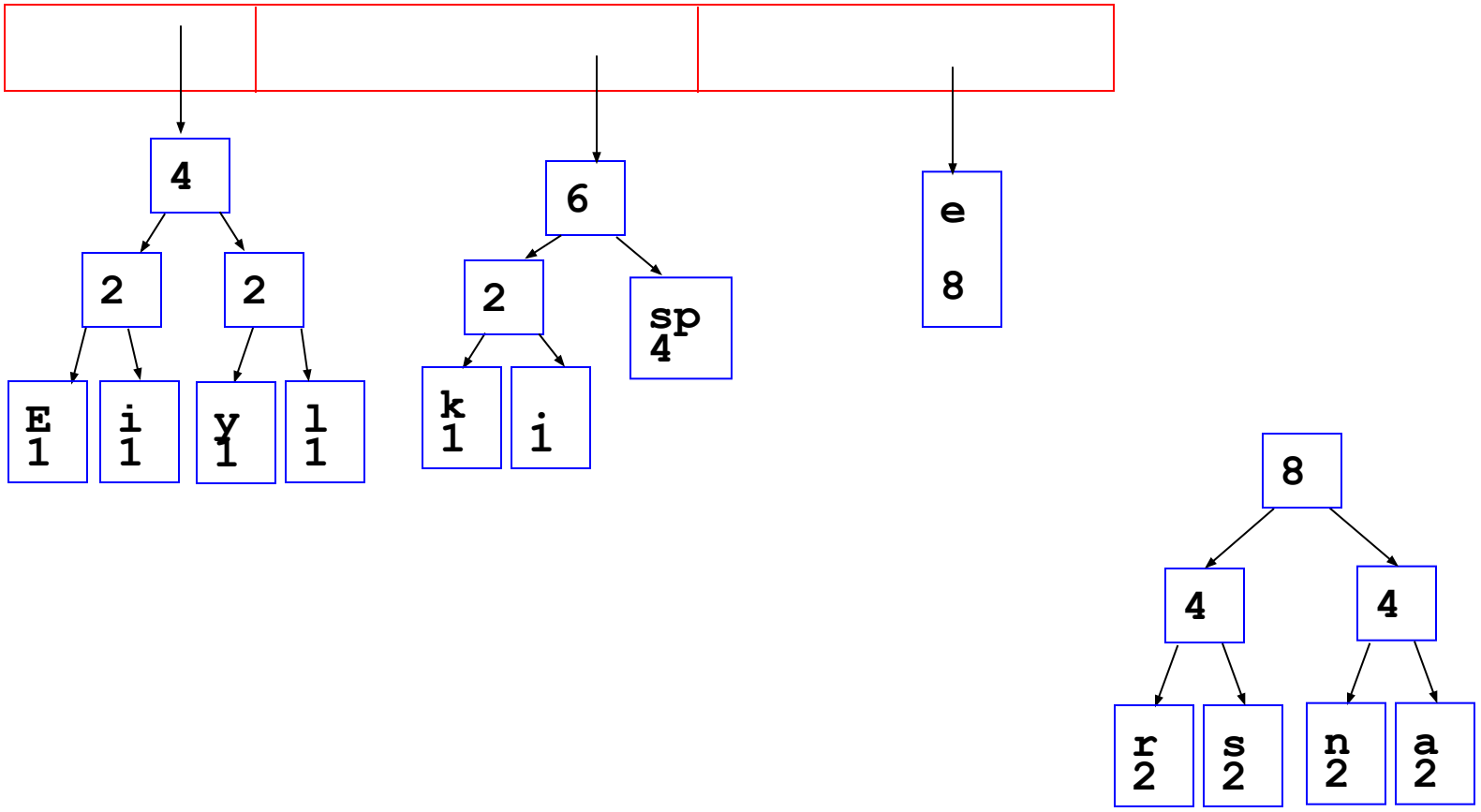


Building a Tree

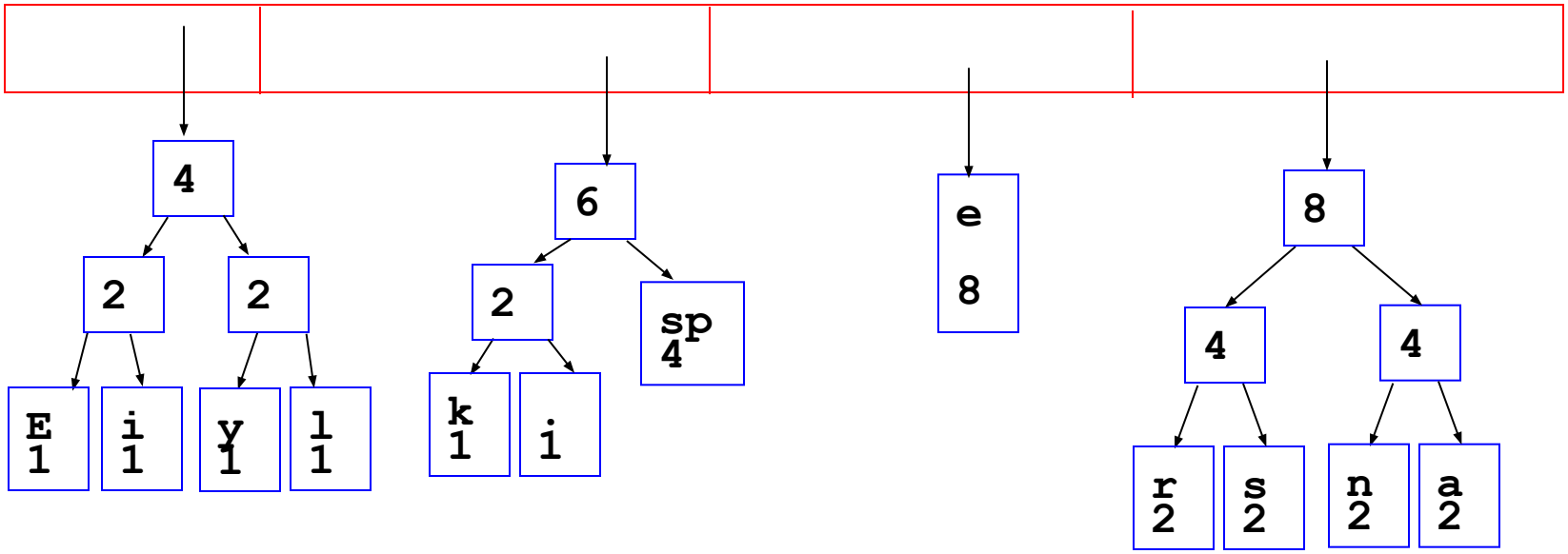


What is happening to the characters with a low number of occurrences?

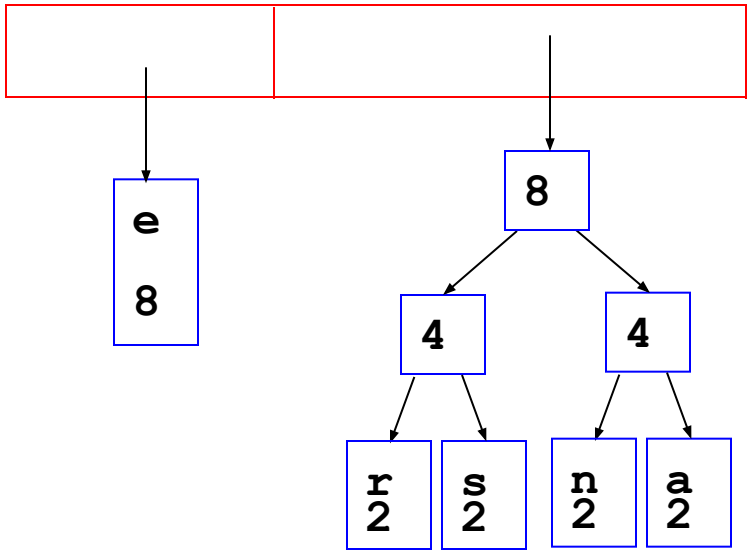
Building a Tree

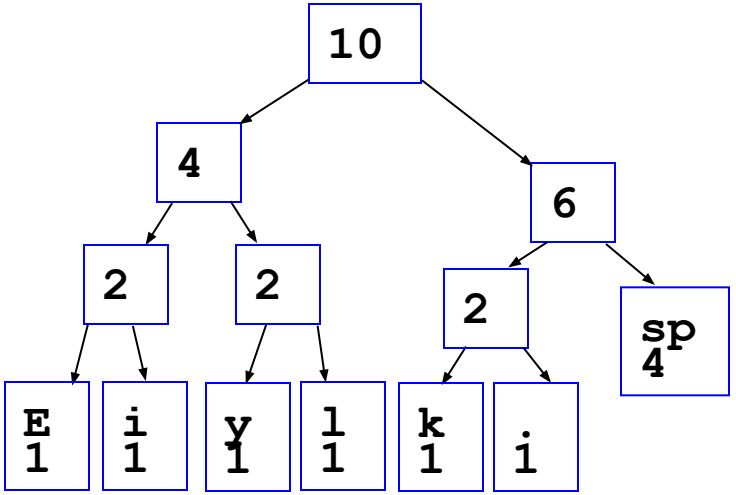


Building a Tree

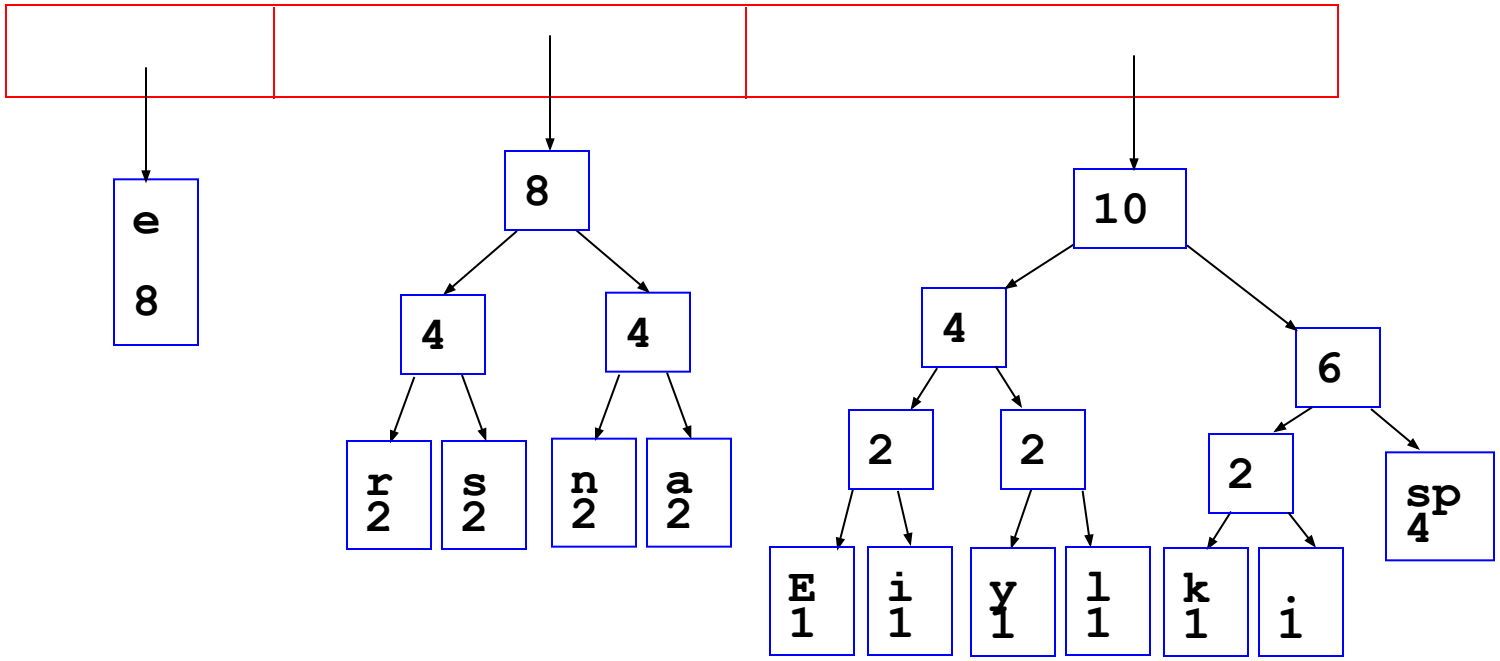


Building a Tree

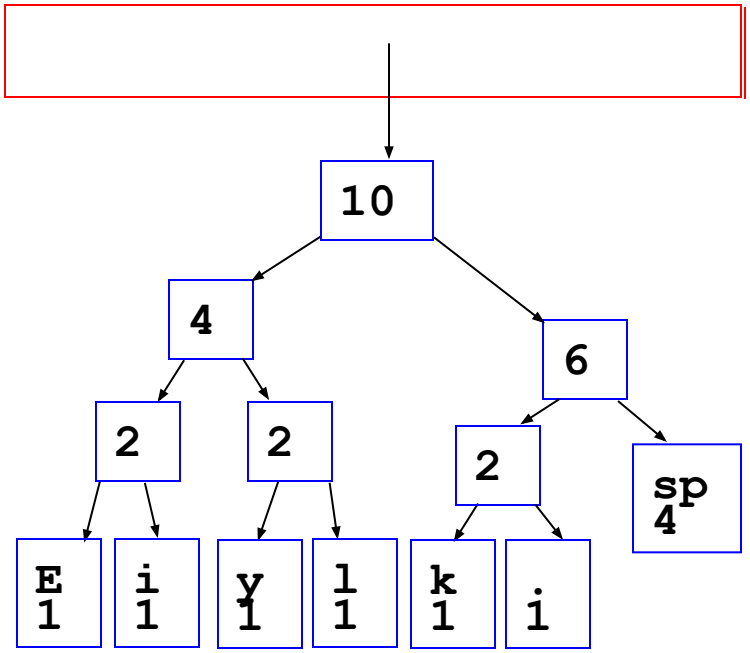


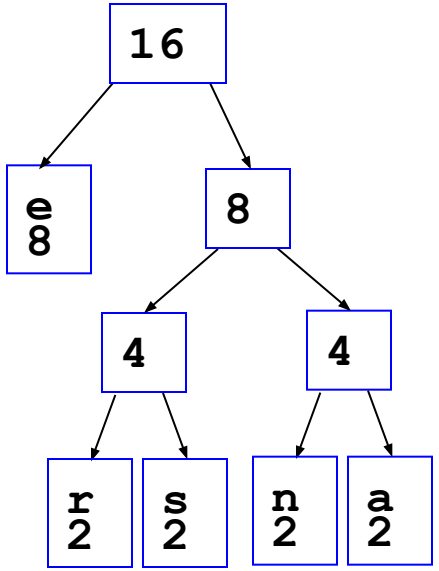


Building a Tree

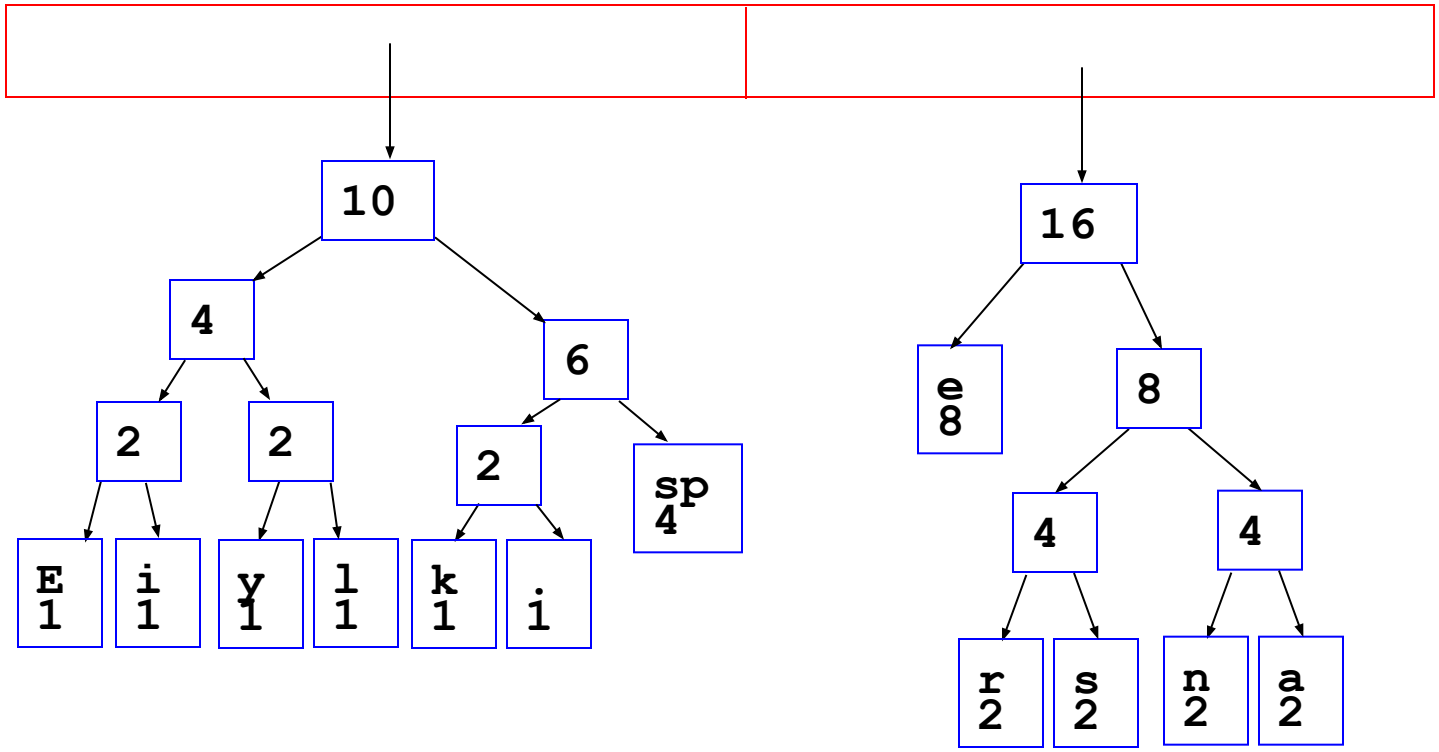


Building a Tree

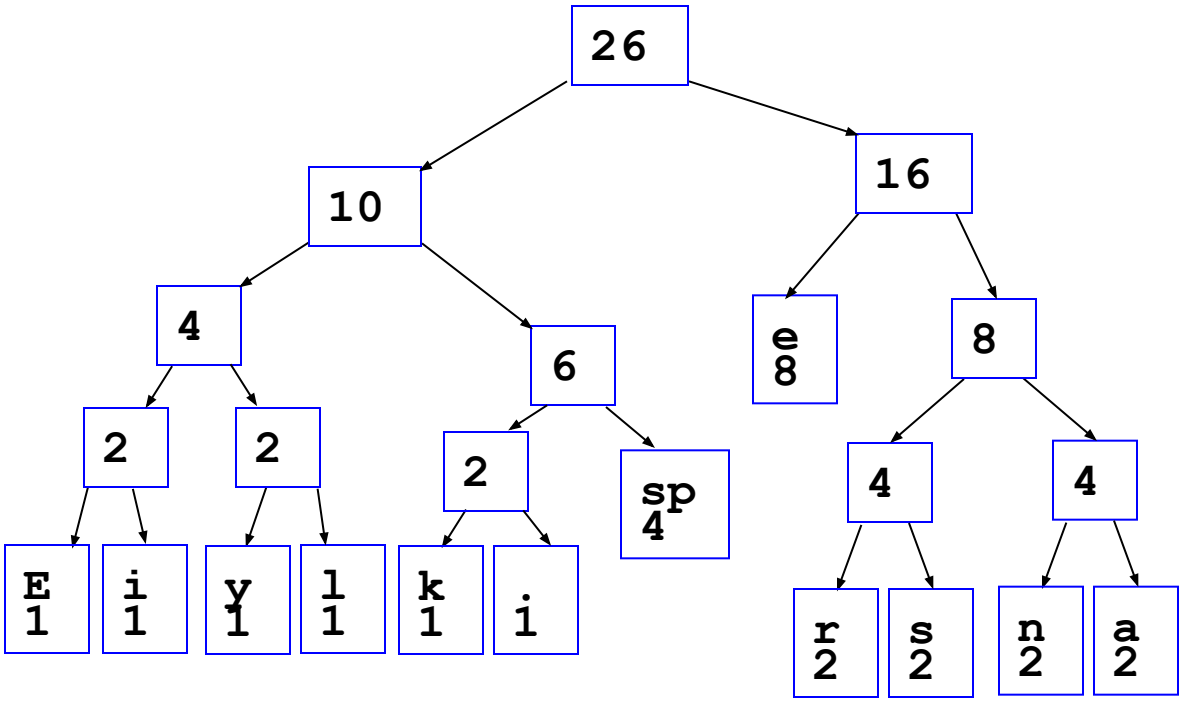




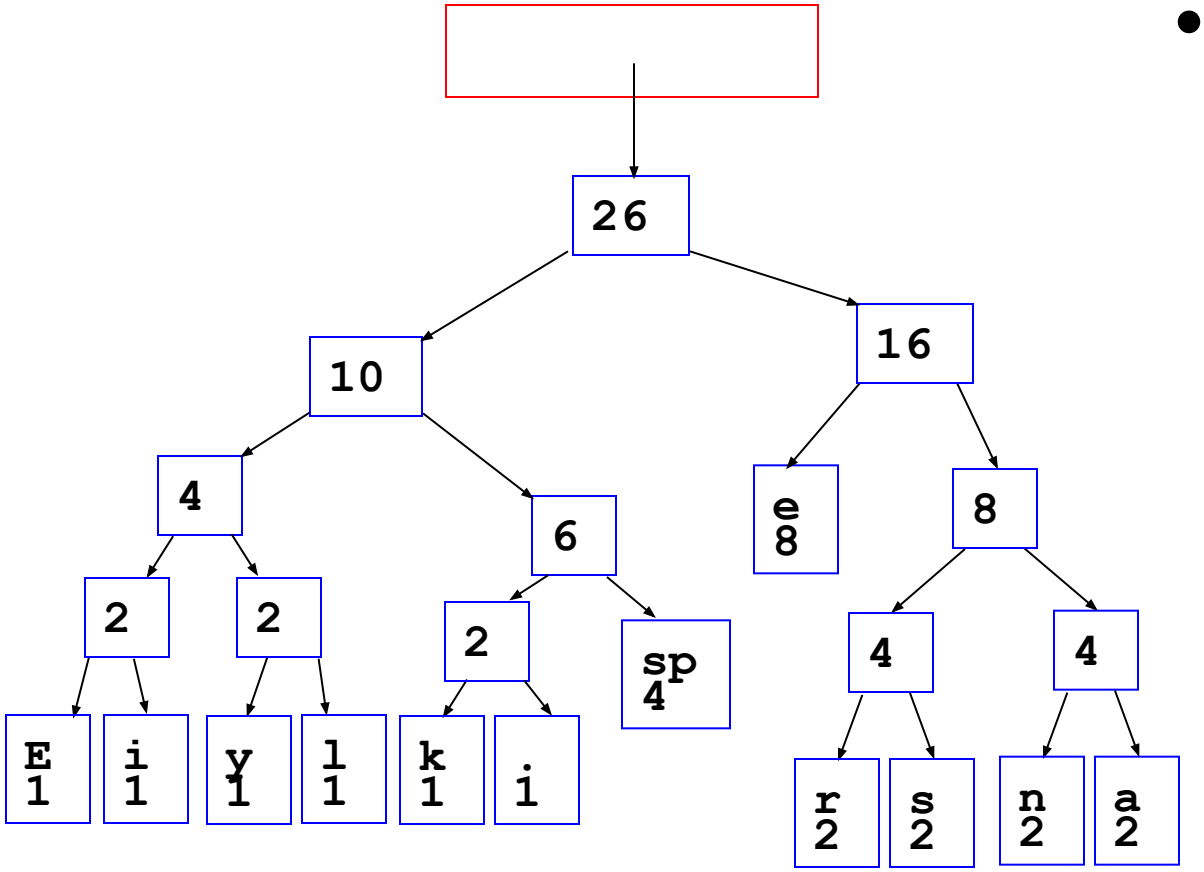
Building a Tree



Building a Tree



Building a Tree



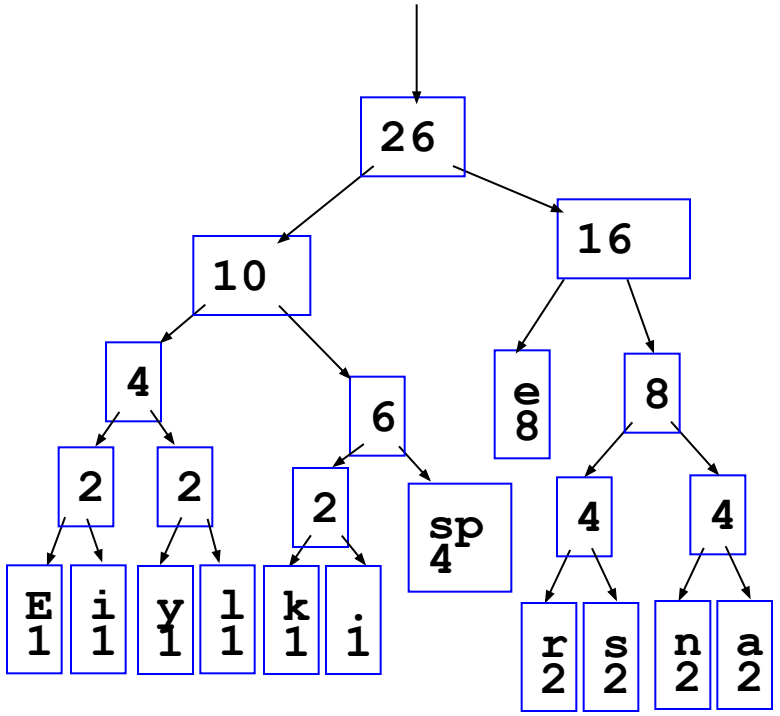
- After enqueueing this node there is only one node left in priority queue.

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.

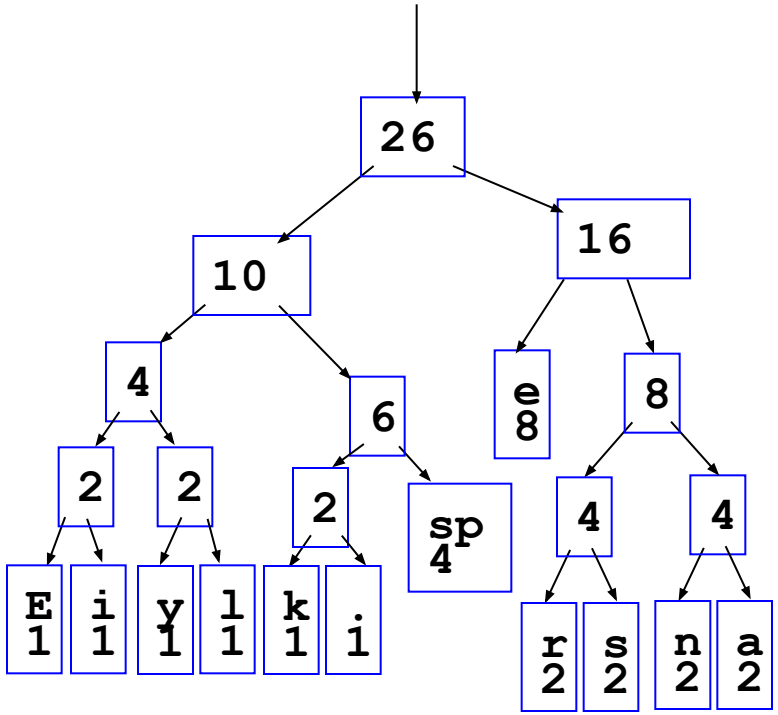


Eerie eyes seen near lake. □ 26 characters

Encoding the File

Traverse Tree for Codes

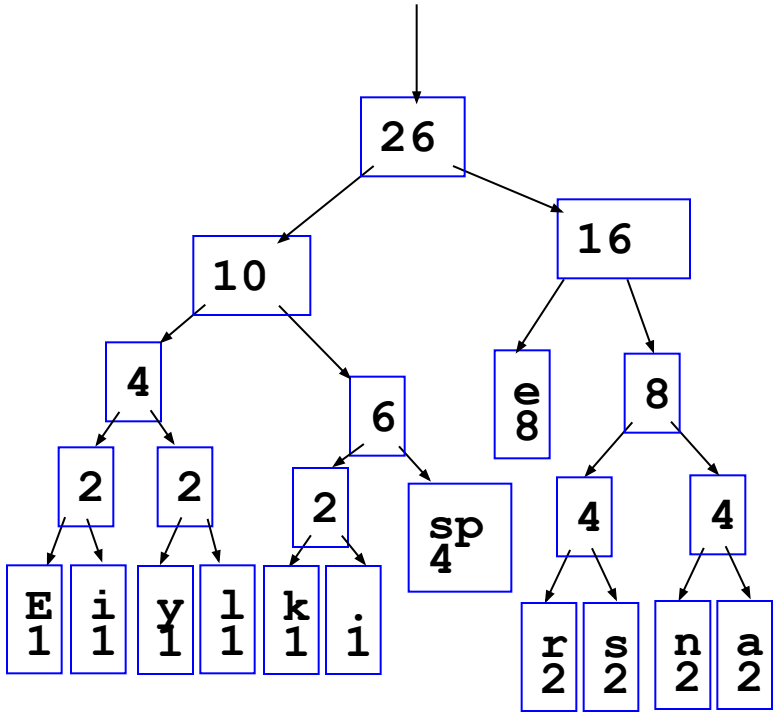
- Perform a traversal of the tree to obtain new code words
- Going left is a 0 going right is a 1
- code word is only completed when a leaf node is reached



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.

```
00001011000000110011
10001010110110100 11
11101011111110001100
11111101001001 01
```

- 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 any better?
- 73 bits to encode the text
- ASCII would take 8 * 26 = 208 bits

```
0000101100000110011
1000101011011010011
1110101111110001100
1111110100100101
```

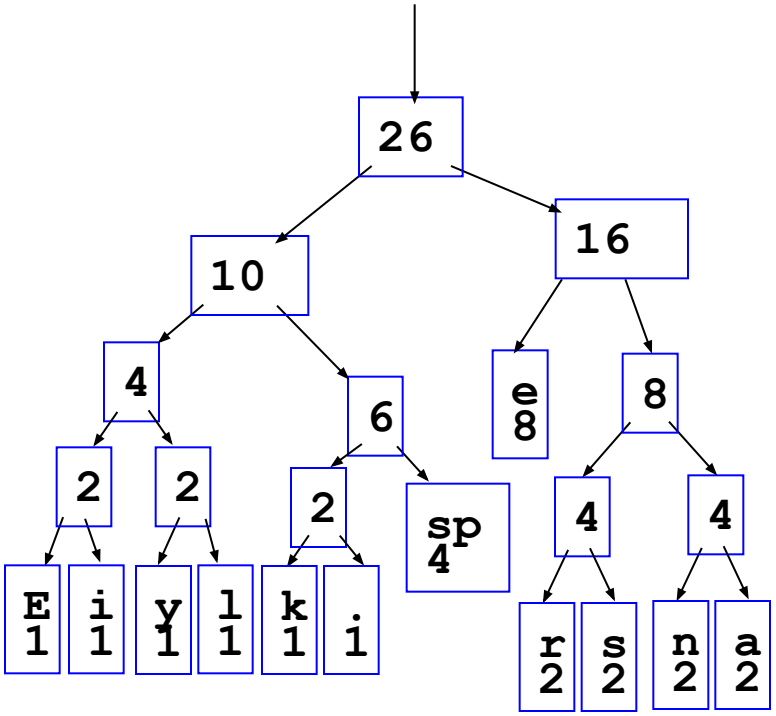
hIf modified code used 4 bits per character are needed. Total bits 4 * 26 = 104. Savings not as great.

Decoding the File

- How does receiver know what the codes are?
- Tree constructed for each text file.
  - Considers frequency for each file
  - Big hit on compression, especially for smaller files
- Tree predetermined
  - based on statistical analysis of text files or file types
- Data transmission is bit based versus byte based

Decoding the File

- Once receiver has tree it scans incoming bit stream
- 0 ⇒ go left
- 1 ⇒ go right




```
10100011011110111
01111110000110101
```

Summary

- Huffman coding is a technique used to compress files for transmission
- Uses statistical coding
  - more frequently used symbols have shorter code words
- Works well for text and fax transmissions
- An application that uses several data structures