CSCI 241

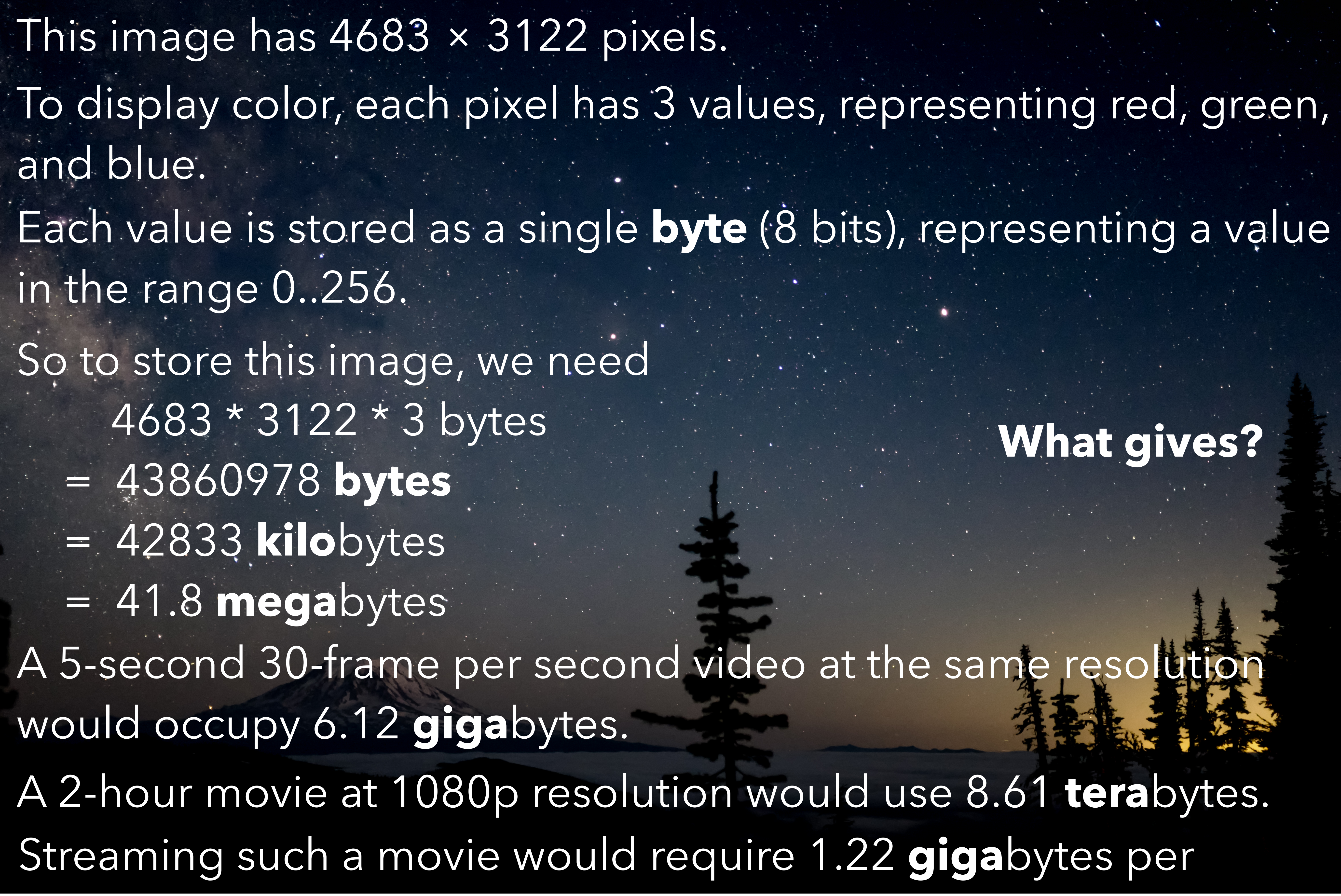
Scott Wehrwein

Coding Trees:
Motivation, Encoding, Decoding

Goals

Understand the motivation for entropy coding.

Be able to use a coding tree to decode and encode a message.



This image has 4683 × 3122 pixels.

To display color, each pixel has 3 values, representing red, green, and blue.

Each value is stored as a single **byte** (8 bits), representing a value in the range 0..256.

So to store this image, we need

4683 * 3122 * 3 bytes
= 43860978 **bytes**
= 42833 **kilo**bytes
= 41.8 **mega**bytes

What gives?

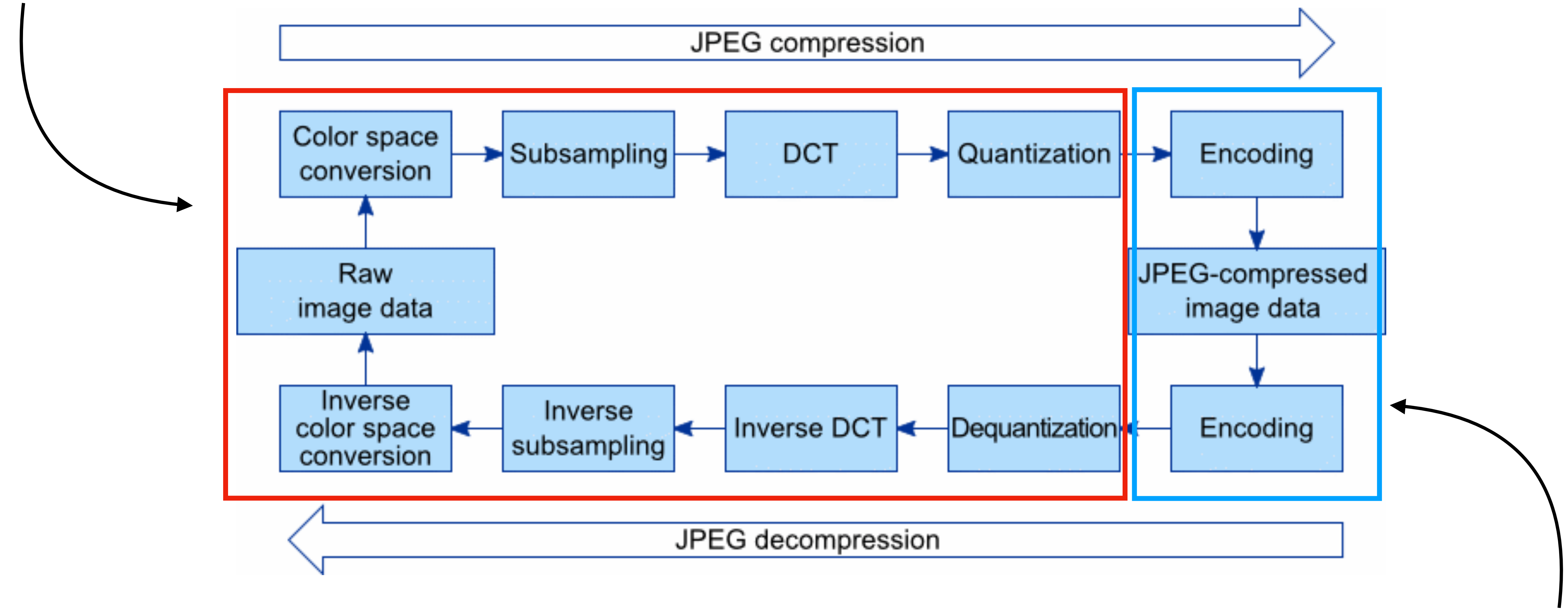
A 5-second 30-frame per second video at the same resolution would occupy 6.12 **giga**bytes.

A 2-hour movie at 1080p resolution would use 8.61 **tera**bytes.

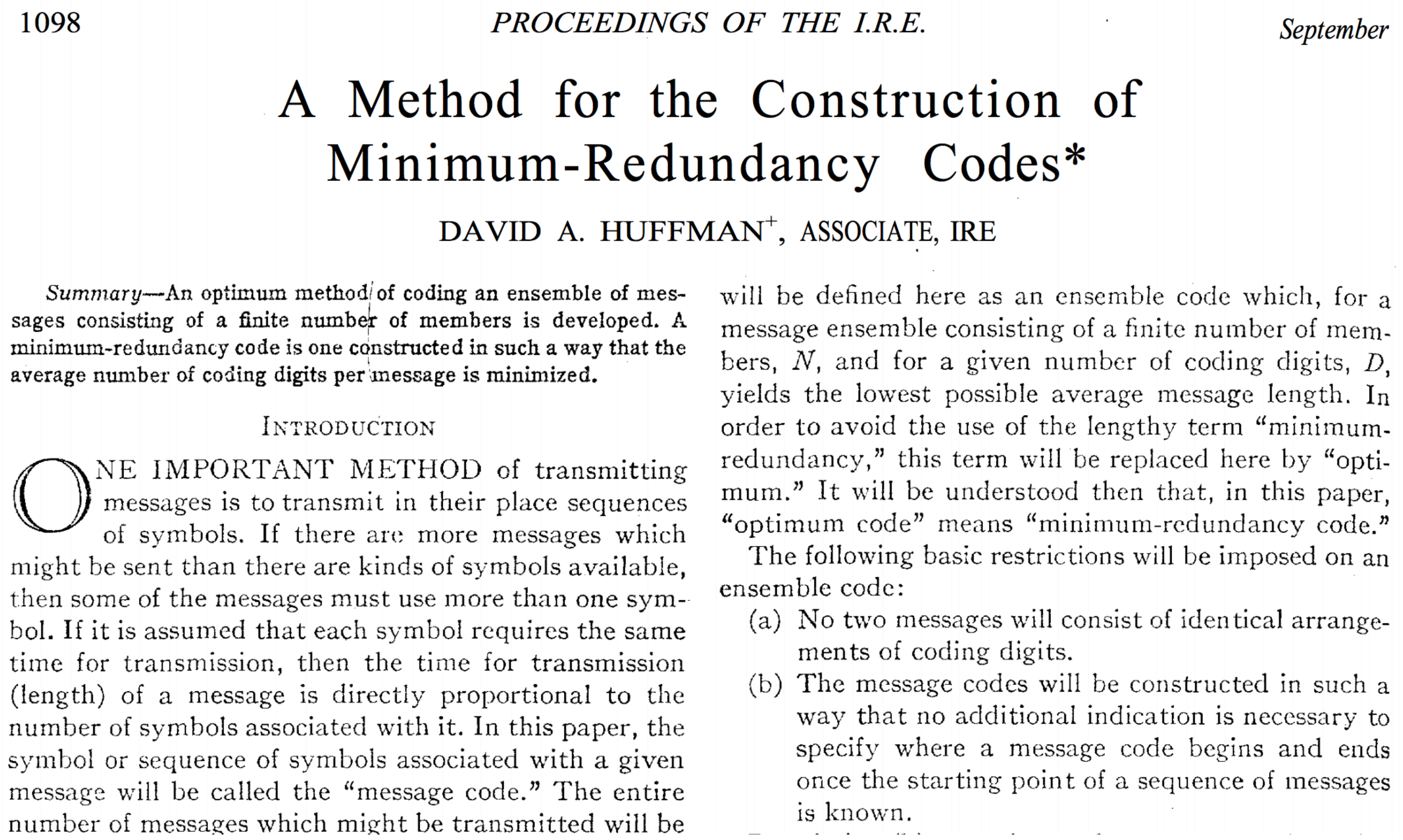
Streaming such a movie would require 1.22 **giga**bytes per second of bandwidth. The fastest home internet connection money can buy is 625 **mega**bytes per second.

Image Compression: JPEG

This part takes advantage of the fact that it's image data; it may be **lossy**.



This part uses Entropy Coding to **losslessly** represent n bits using fewer than n bits.



A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN[+], ASSOCIATE, IRE

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.

Introduction

ONE IMPORTANT METHOD of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

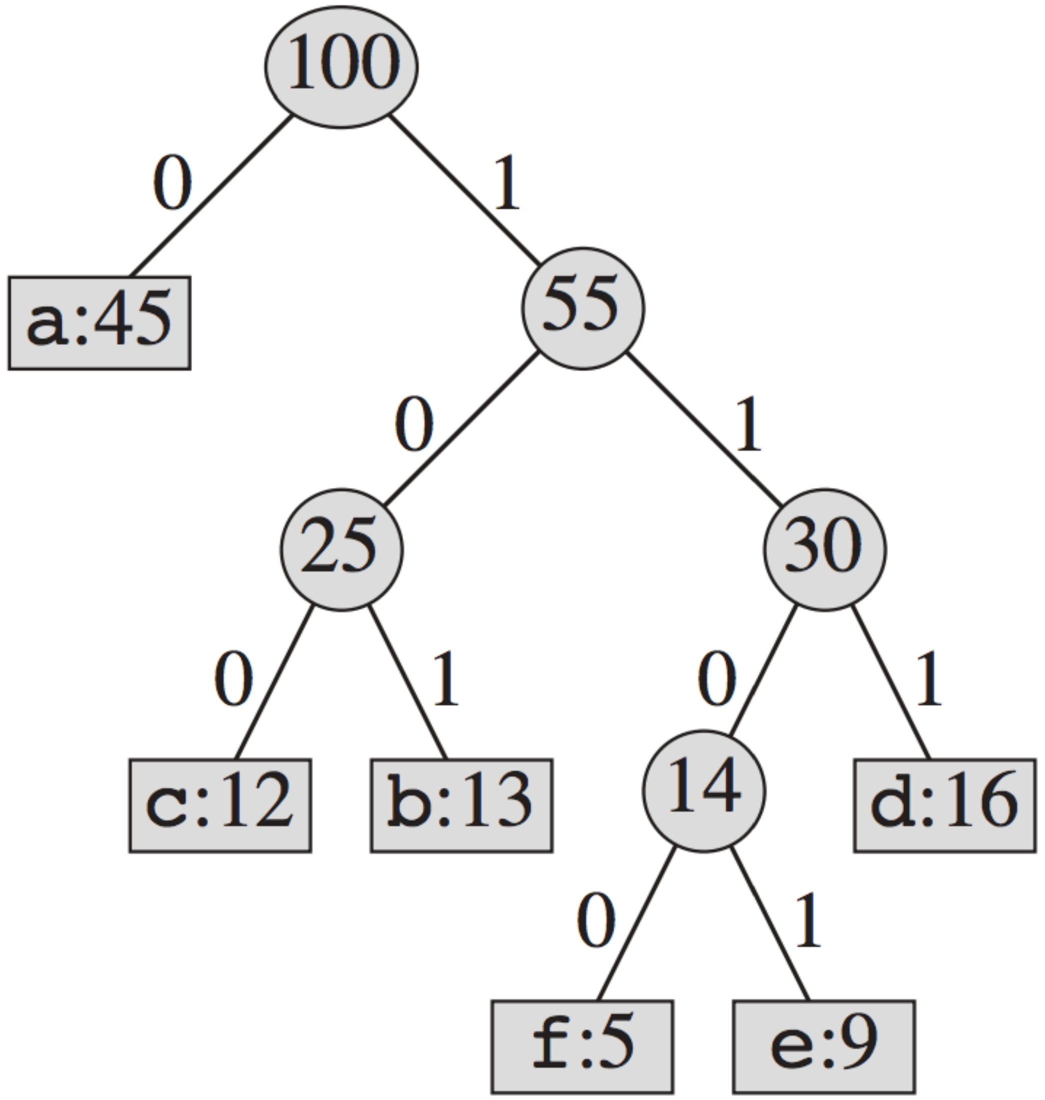
(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Publication date: September 1952

New tech at the time:

- Compilers!
- Programs stored in computer
- Tic-tac-toe program

This is a **coding tree**.

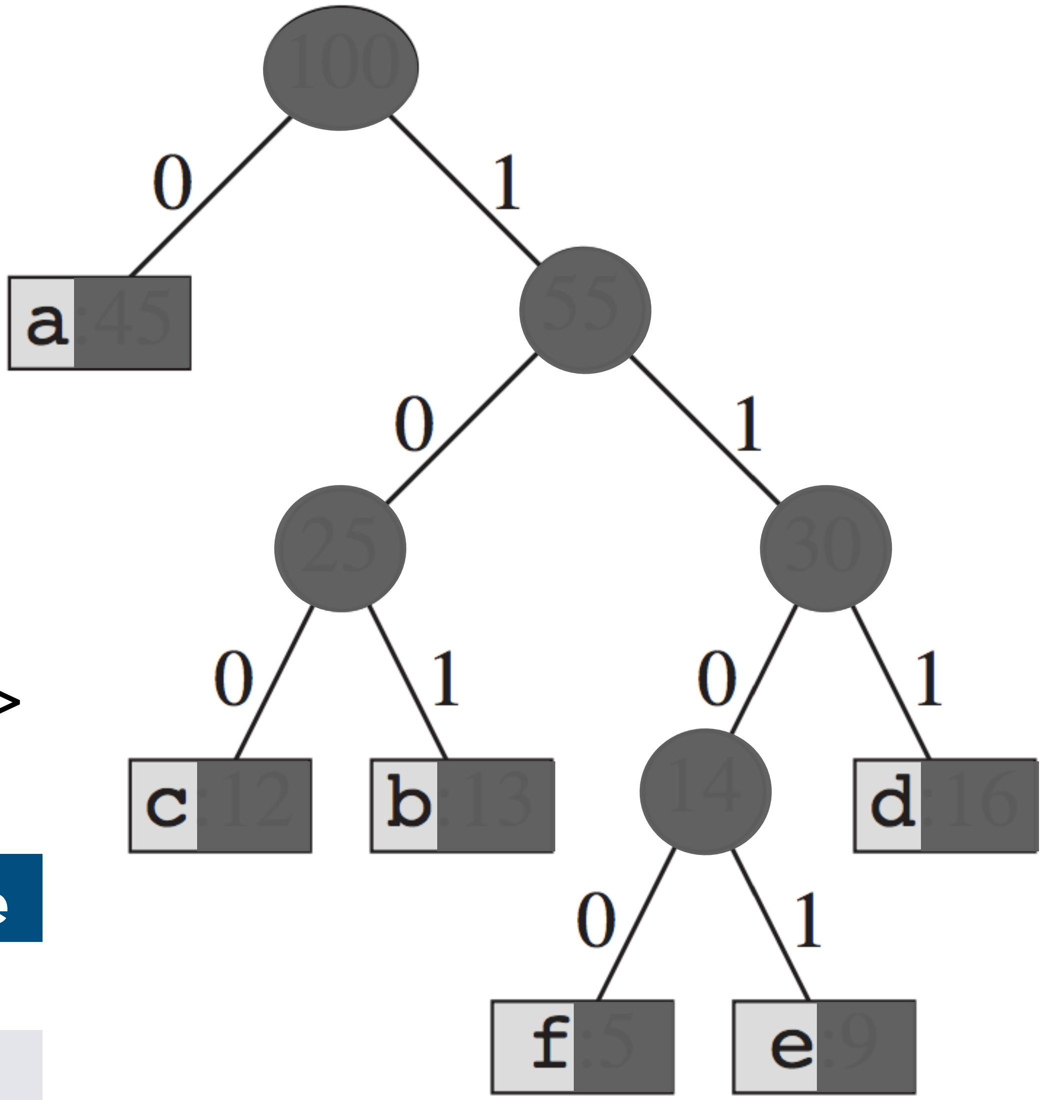


This is a **coding tree**.

It's a trie!

Things to notice:

- Digits are binary
- Nodes store characters
- This is a Map<Bitstring, Character>



Aside: we chose Characters here, but could use anything, such as whole words, image pixels, raw bytes, etc.

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

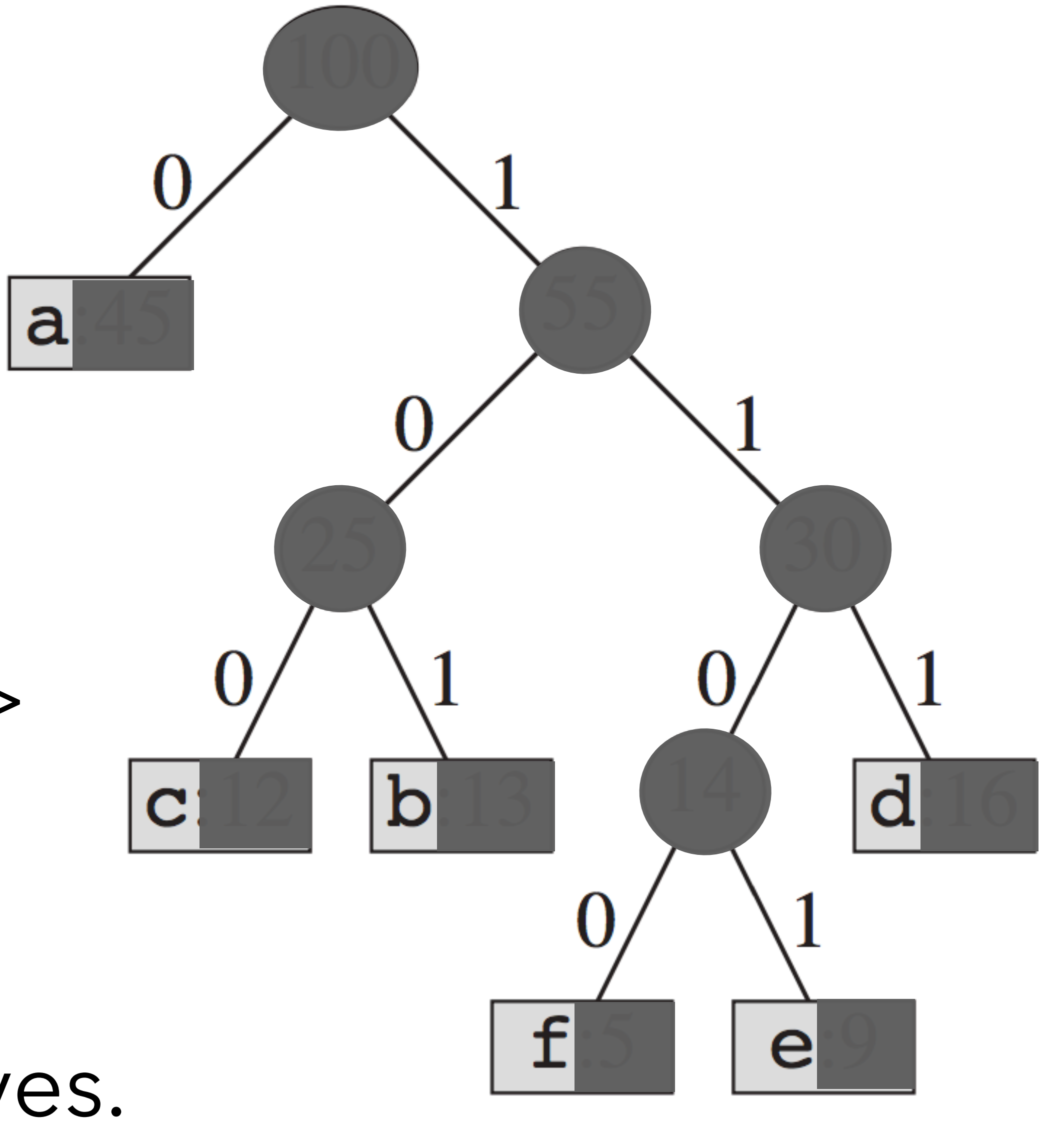
This is a **coding tree**.

It's a trie!

Things to notice:

- Digits are binary
- Nodes store characters
- This is a Map<Bitstring, Character>

- All nodes that `terminate` are leaves.

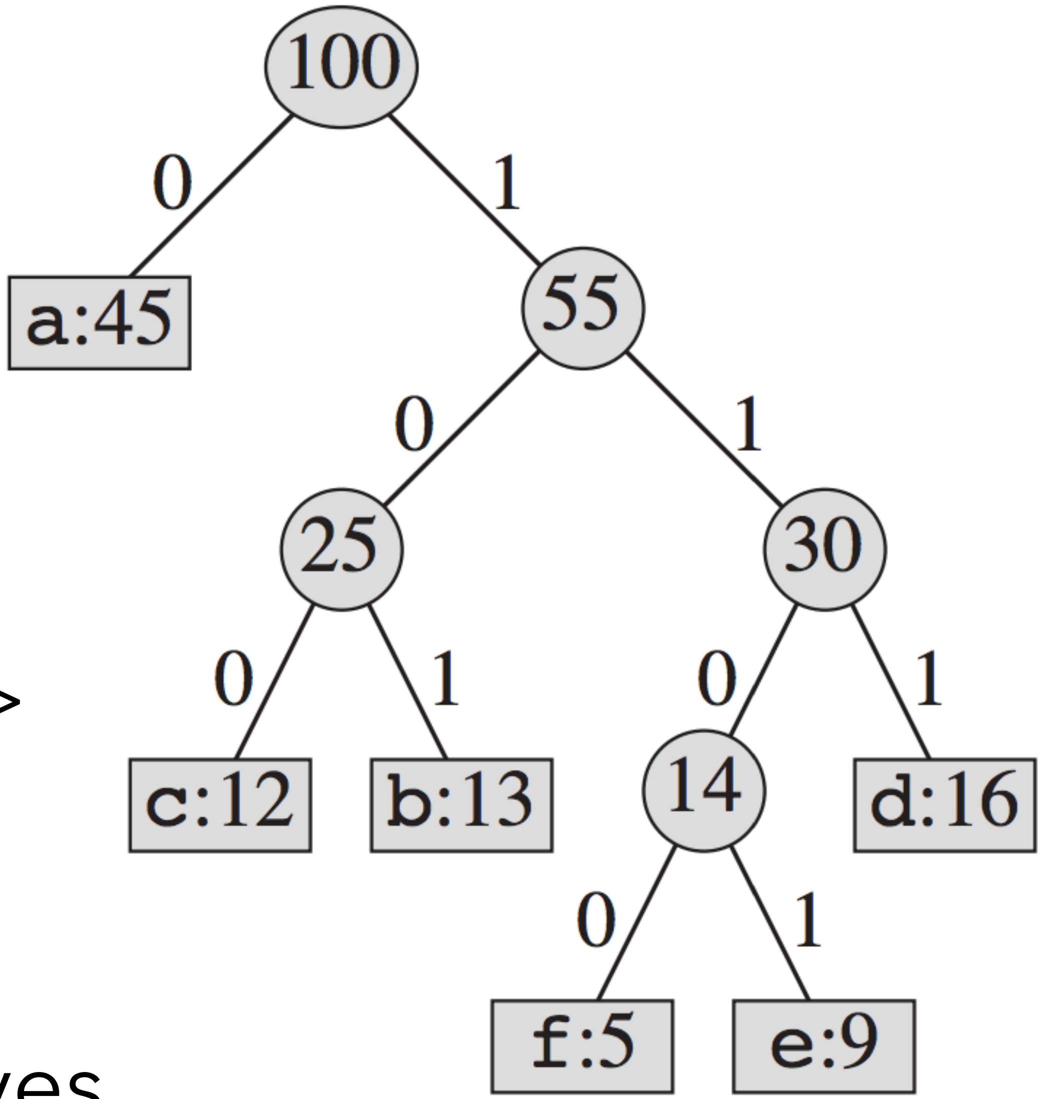


This is a **coding tree**.

It's a trie!

Things to notice:

- Digits are binary
- Nodes store characters
- This is a Map<Bitstring, Character>

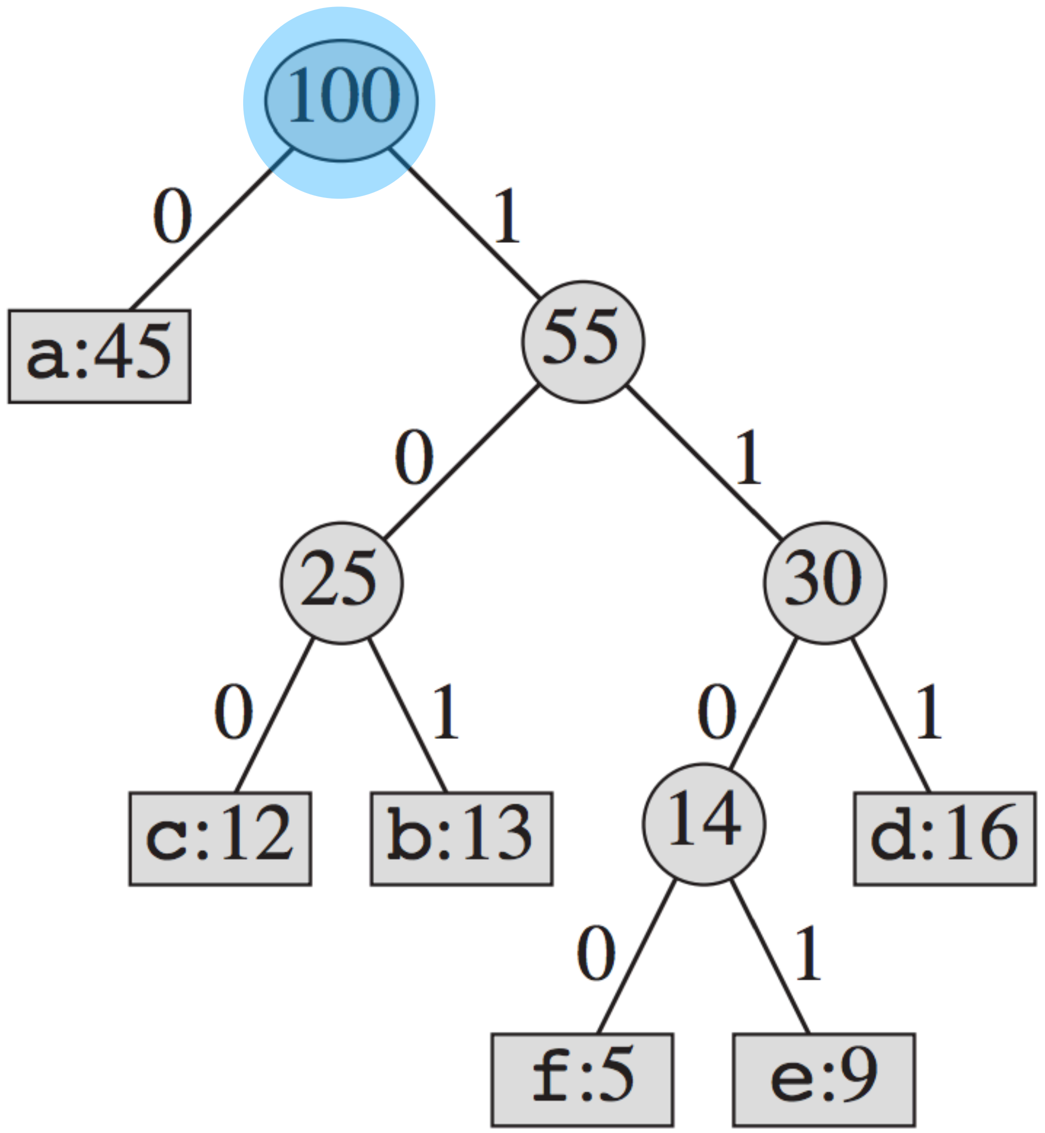


- All nodes that `terminate` are leaves.
- There's an extra number stored at each node.

Decoding

Use the trie as a codebook.

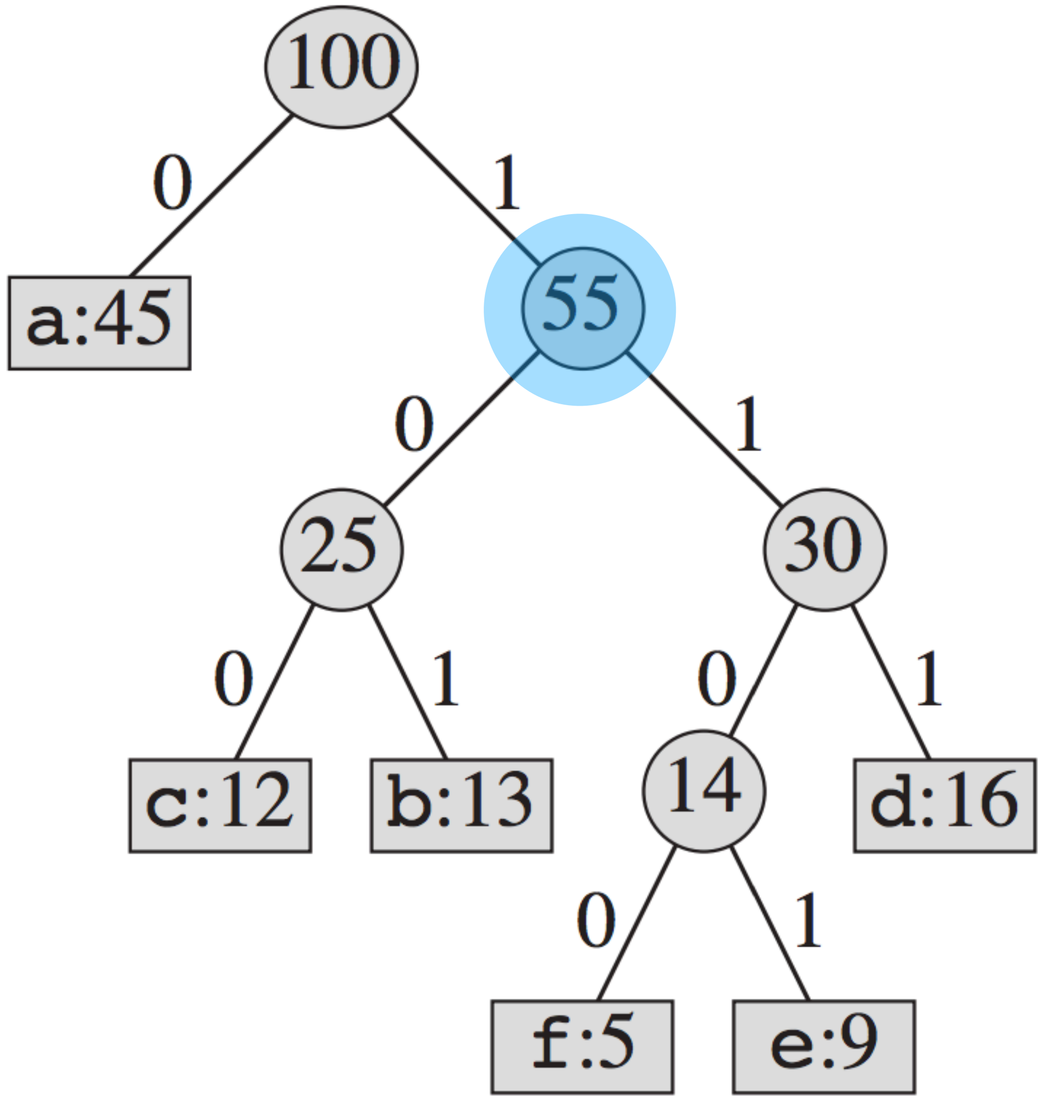
Example: decode 11000101



Decoding

Use the trie as a codebook.

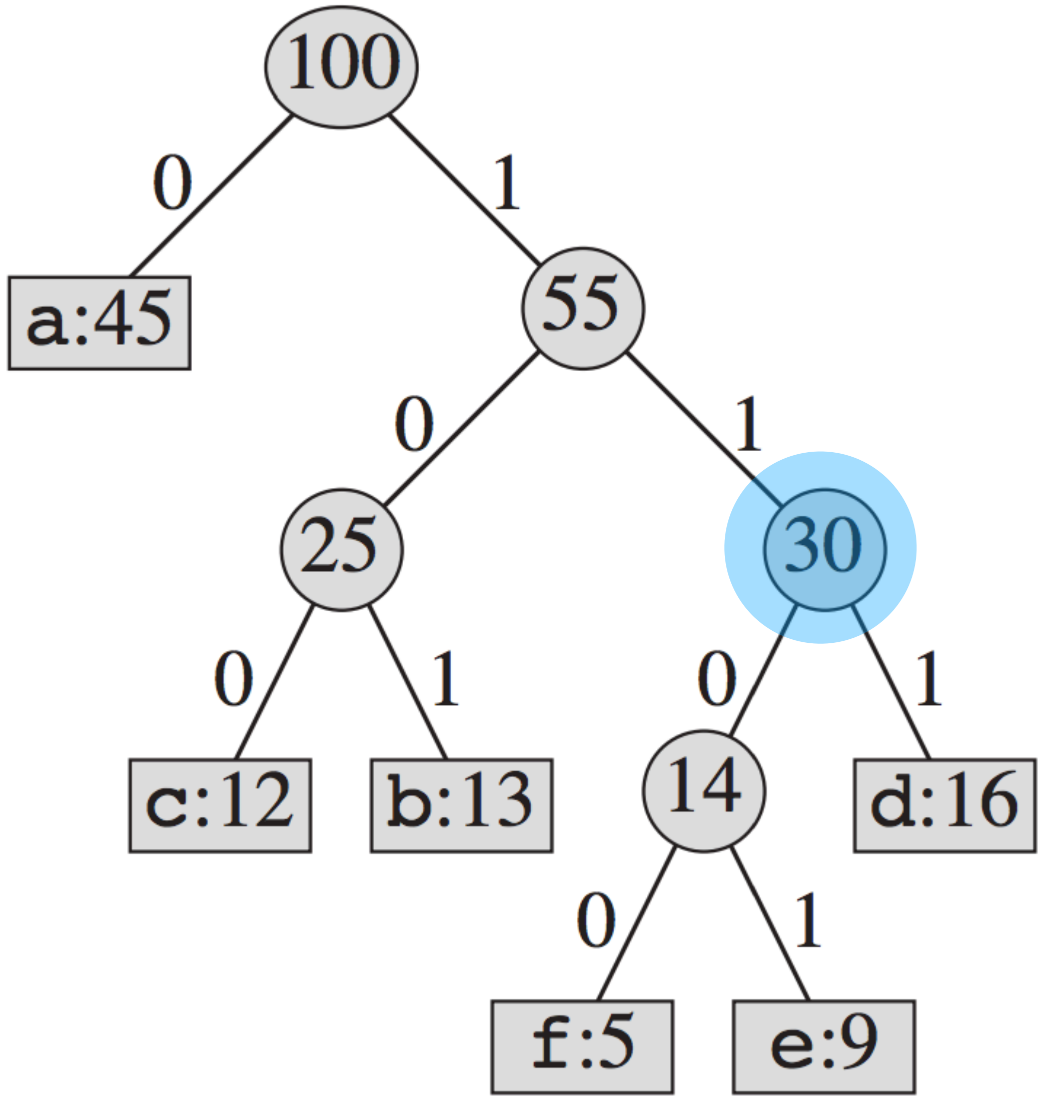
Example: decode 11000101



Decoding

Use the trie as a codebook.

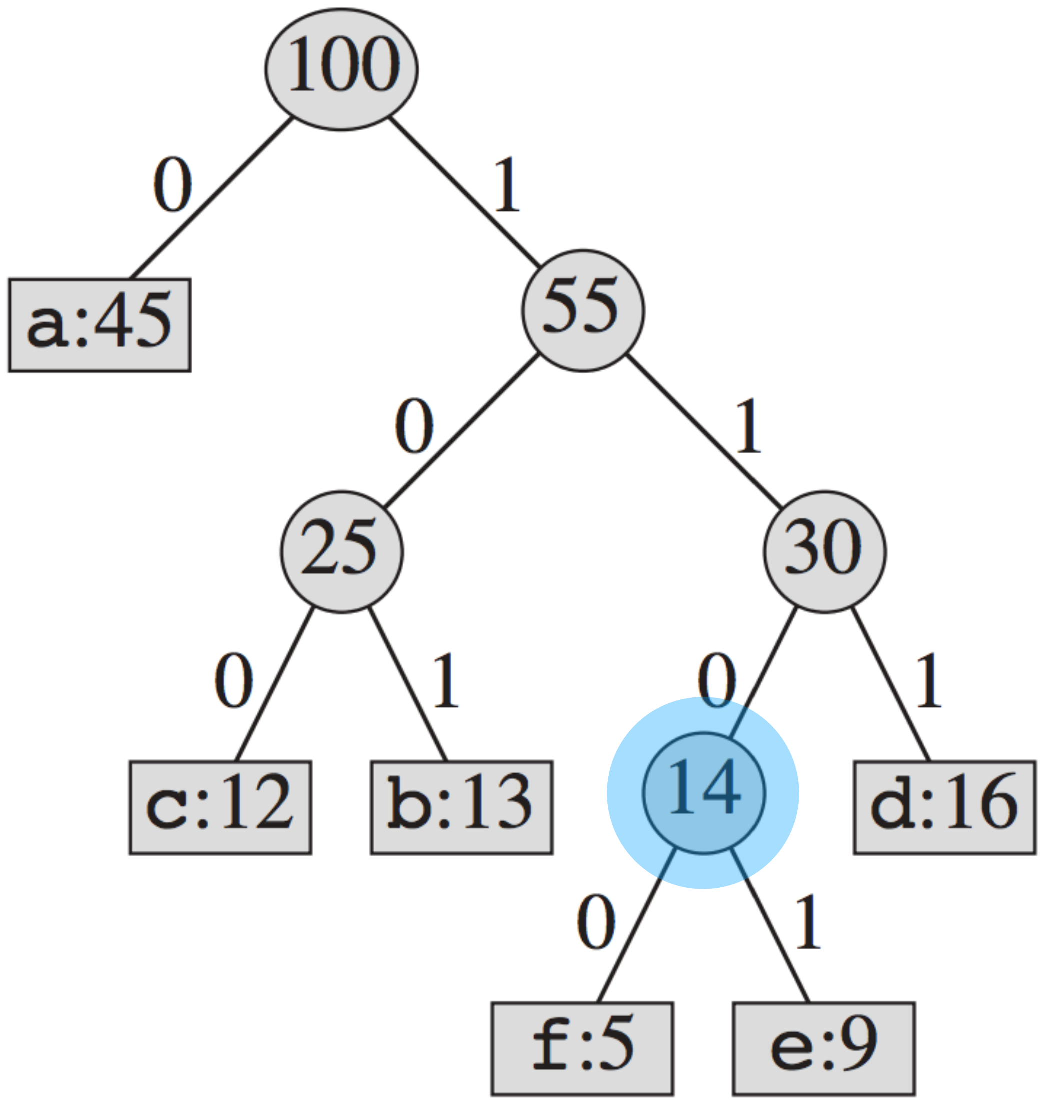
Example: decode 11000101



Decoding

Use the trie as a codebook.

Example: decode 11000101

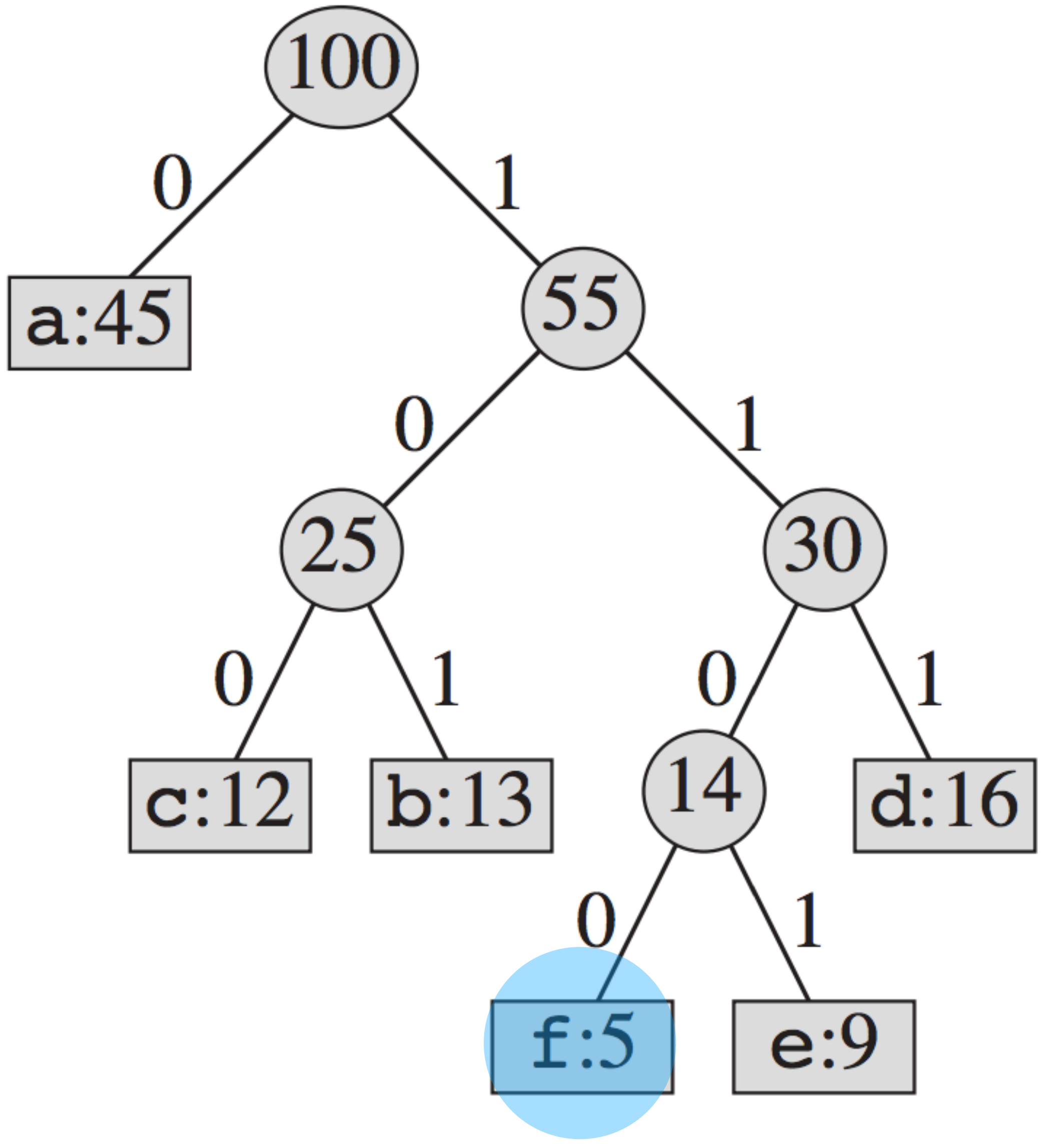


Decoding

Use the trie as a codebook.

Example: decode 11000101

f

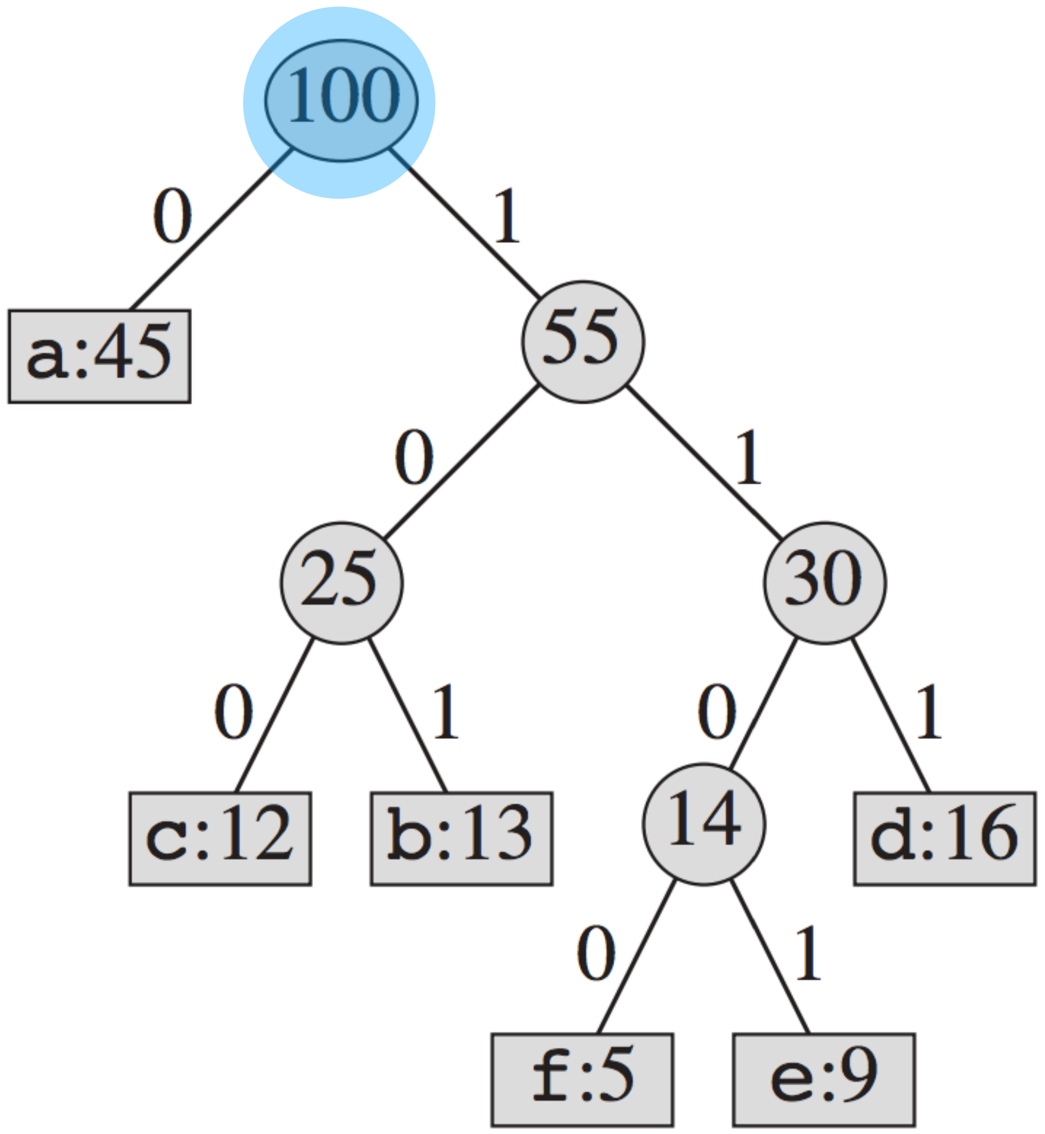


Decoding

Use the trie as a codebook.

Example: decode 11000101

`f`

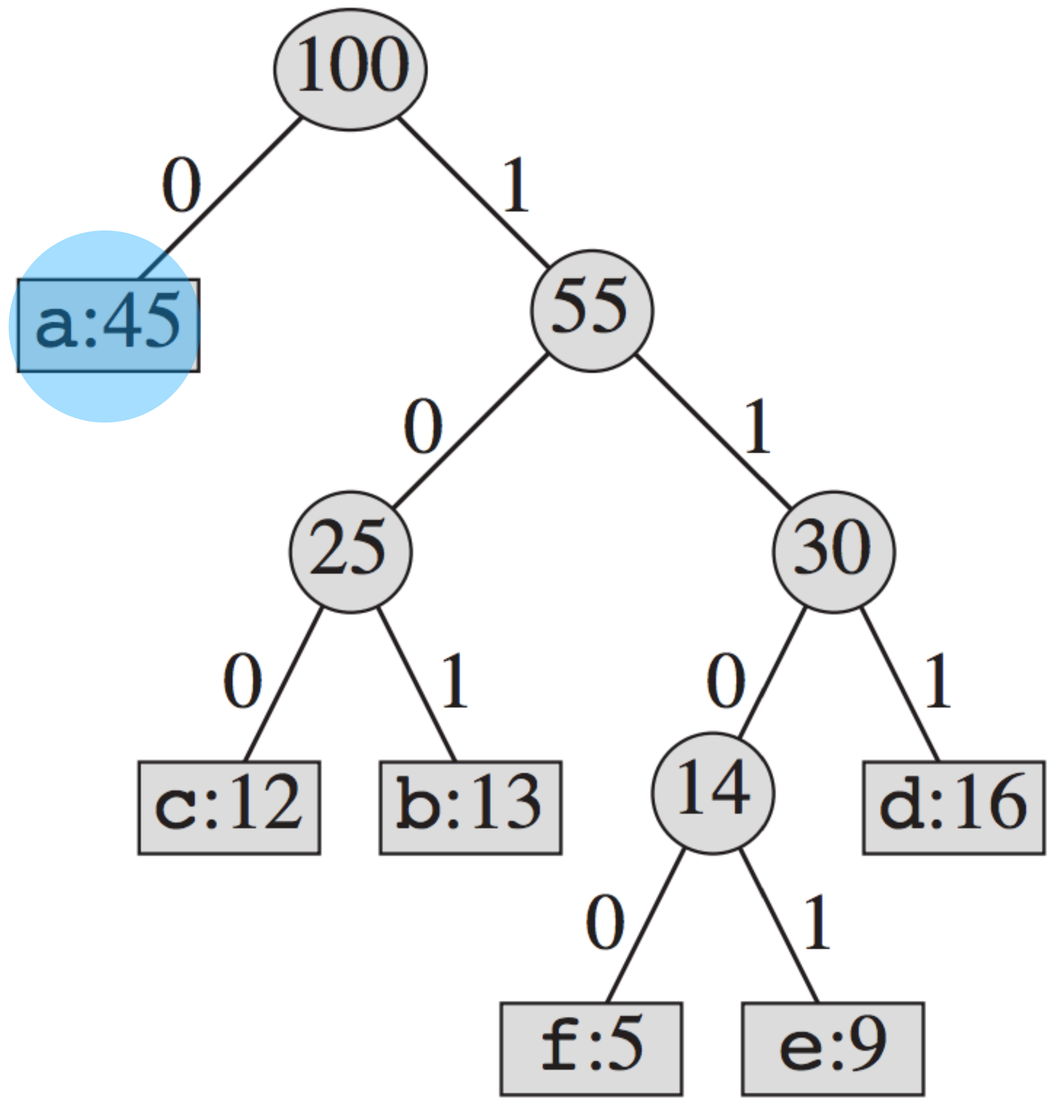


Decoding

Use the trie as a codebook.

Example: decode 11000101

`fa`

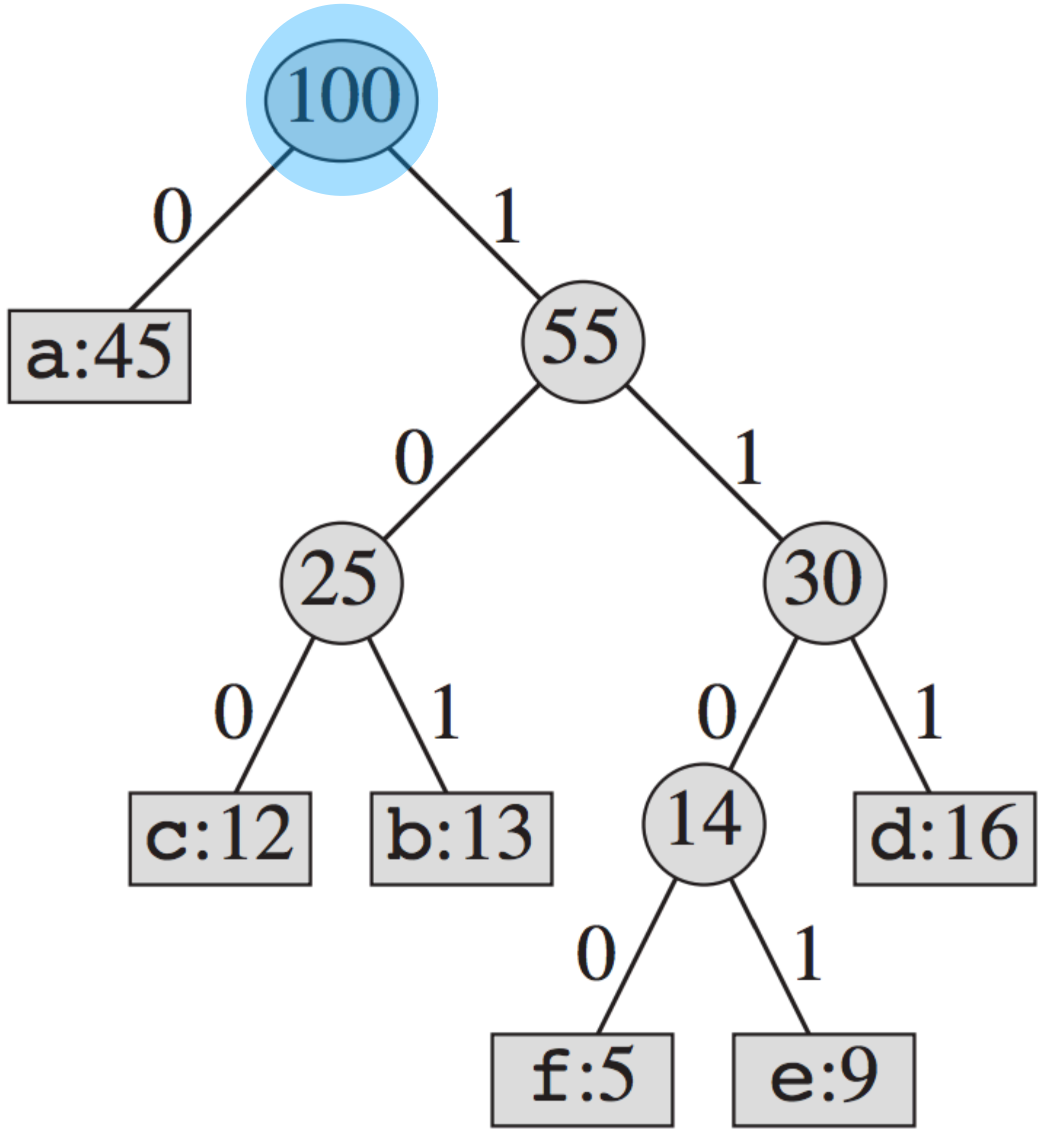


Decoding

Use the trie as a codebook.

Example: decode 11000101

fa

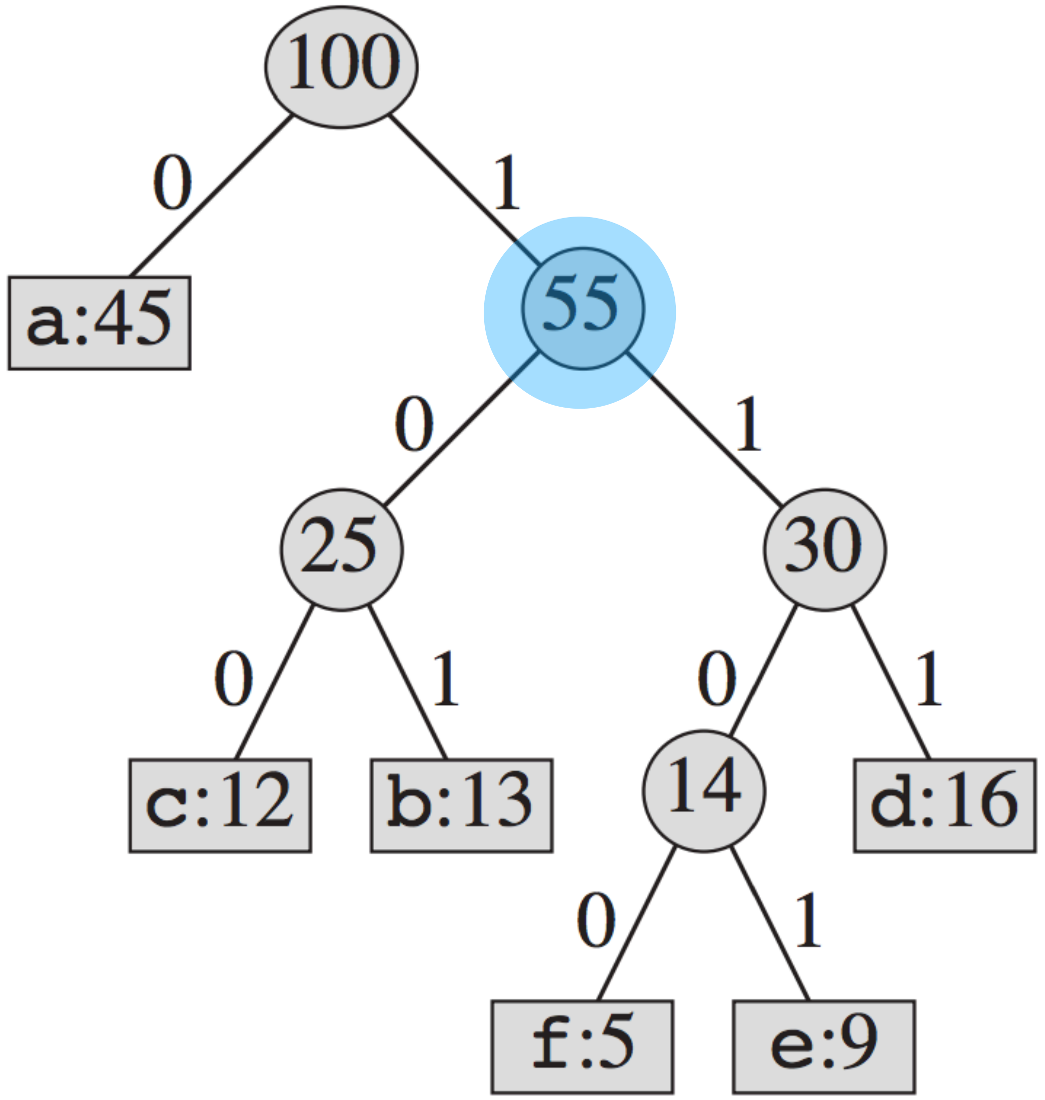


Decoding

Use the trie as a codebook.

Example: decode 11000101

fa

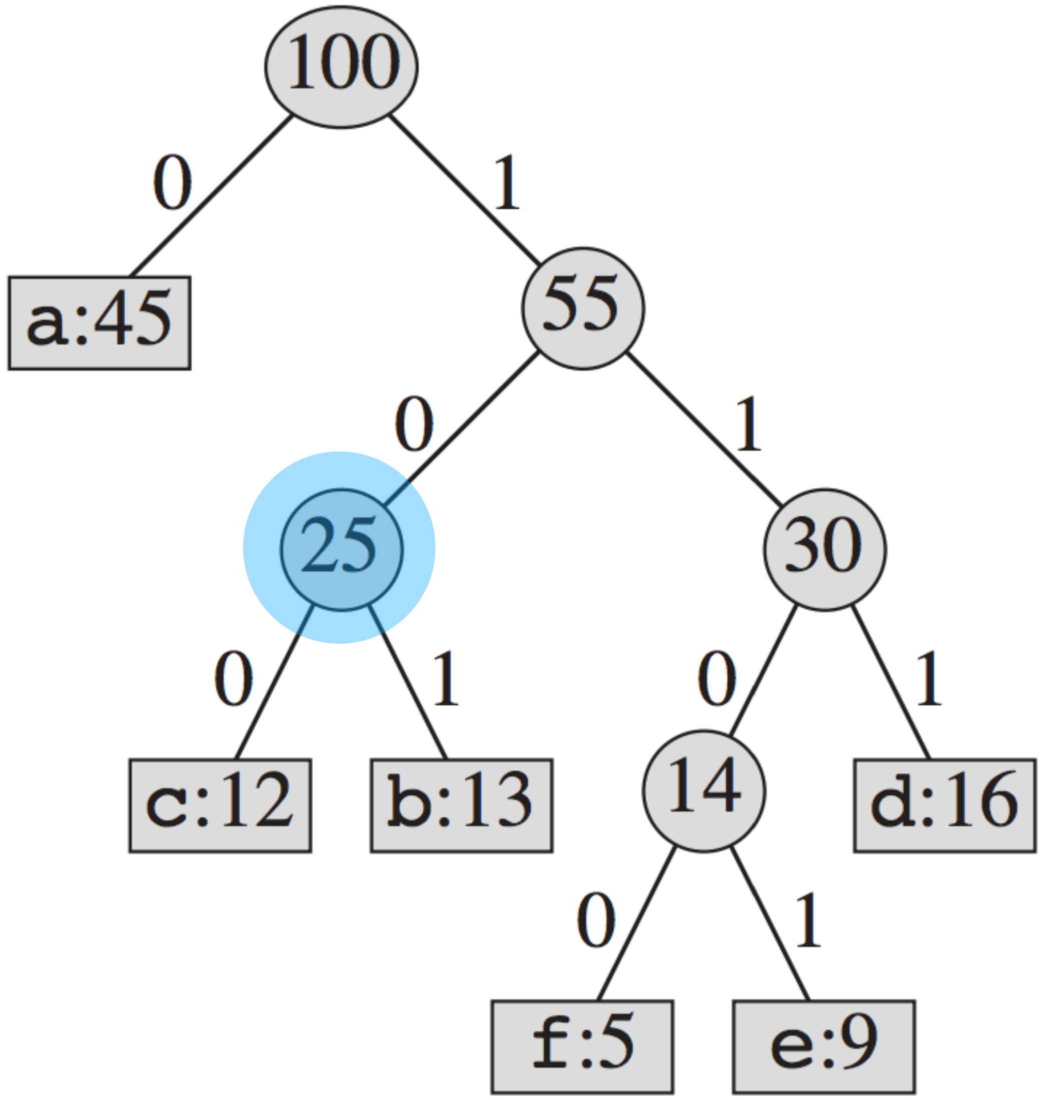


Decoding

Use the trie as a codebook.

Example: decode 11000101

`fa`

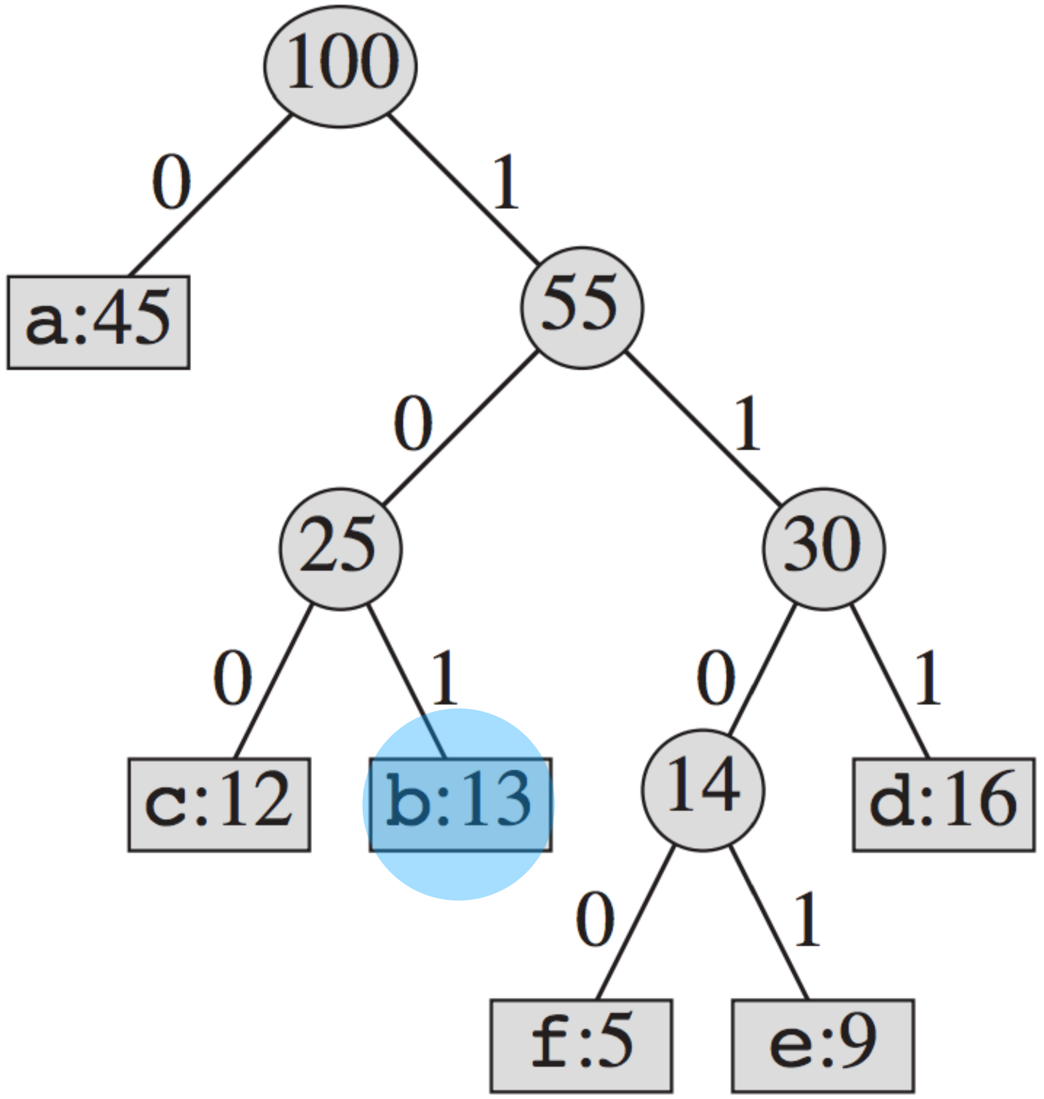


Decoding

Use the trie as a codebook.

Example: decode 11000101

`fab`



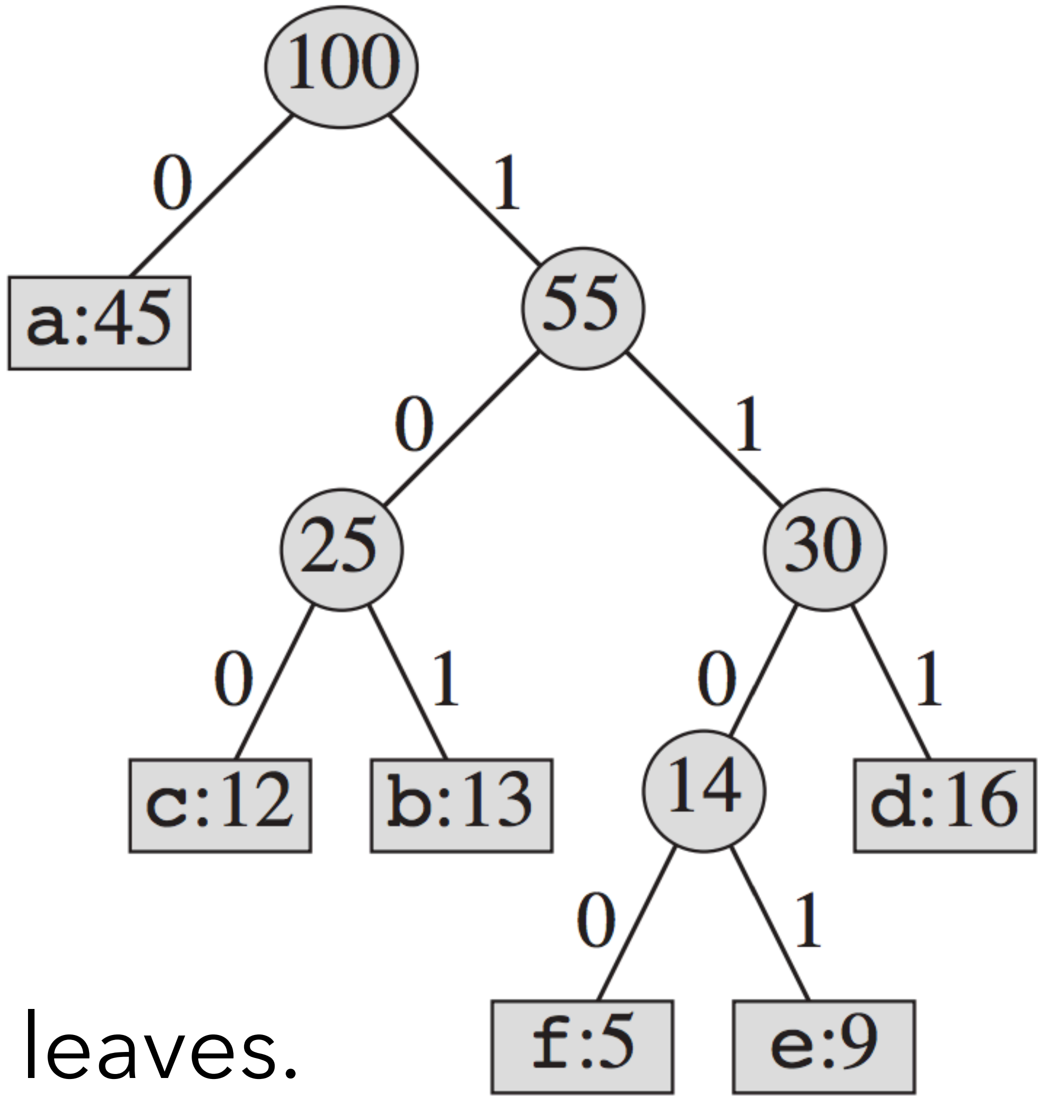
Decoding

Use the trie as a codebook.

Example: decode $\underline{1100}\,\underline{0}\,\underline{101}$

`fab`

Recall: all nodes that `terminate` are leaves.



These are called *prefix codes* - no code is a prefix of another.
This yields a **unique** decoding with *variable-length* codes.

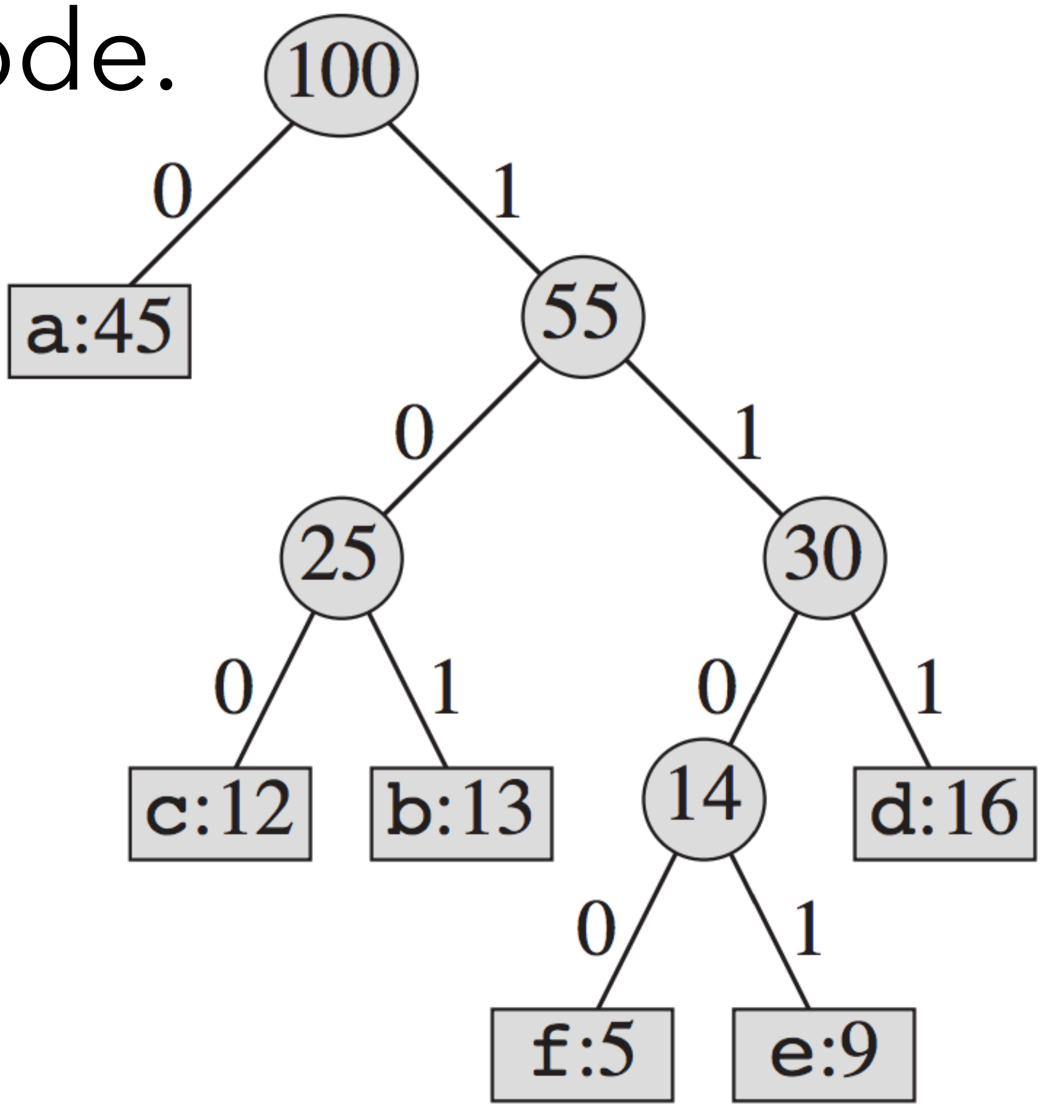
Encoding

Apply the inverse map to encode.

Example: encode `bad`

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

Character	Encoding
a	0
c	100
b	101
d	111
f	1100
e	1101



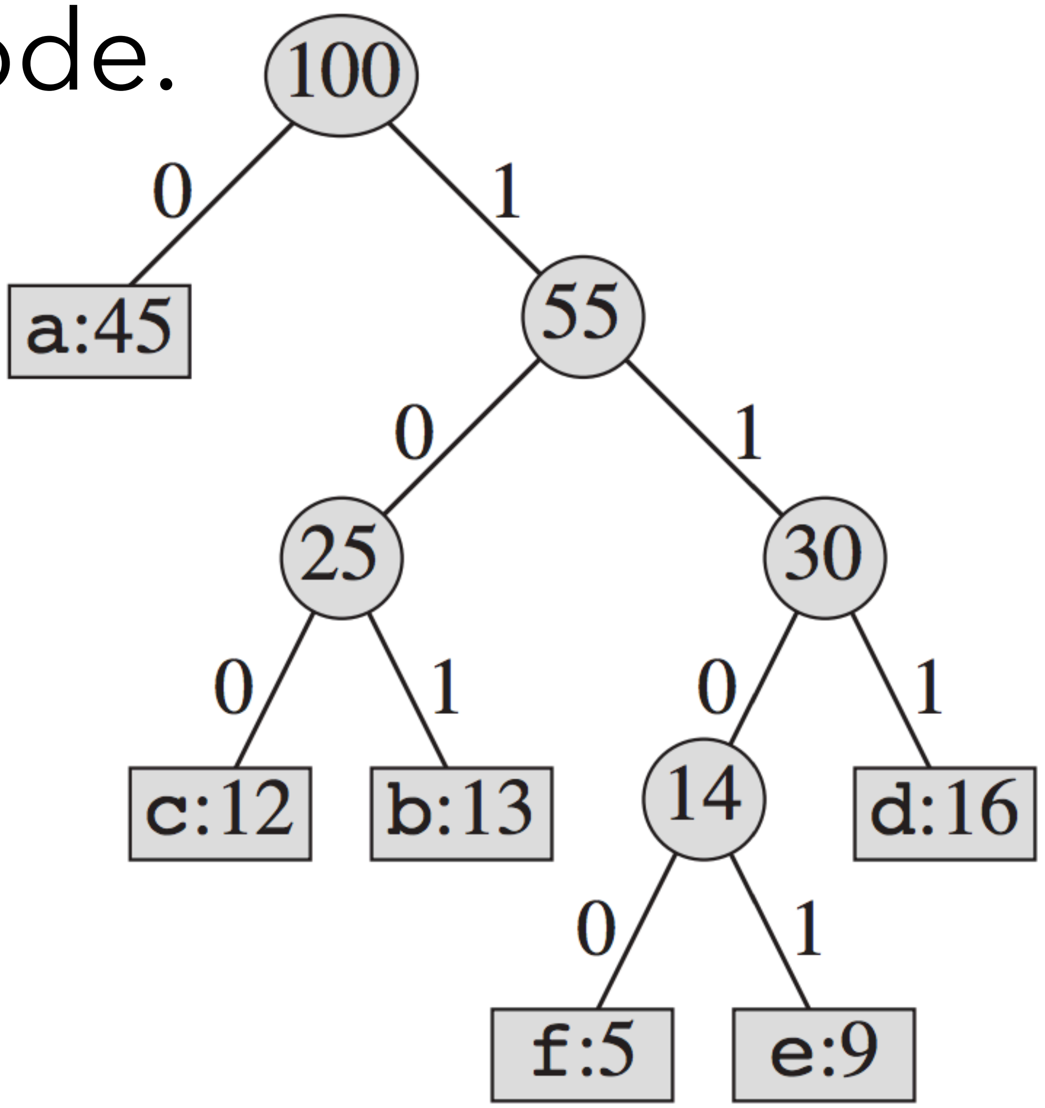
Encoding

Apply the inverse map to encode.

Example: encode `bad`

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

Character	Encoding
a	0
c	100
b	101
d	111
f	1100
e	1101



```
101
```

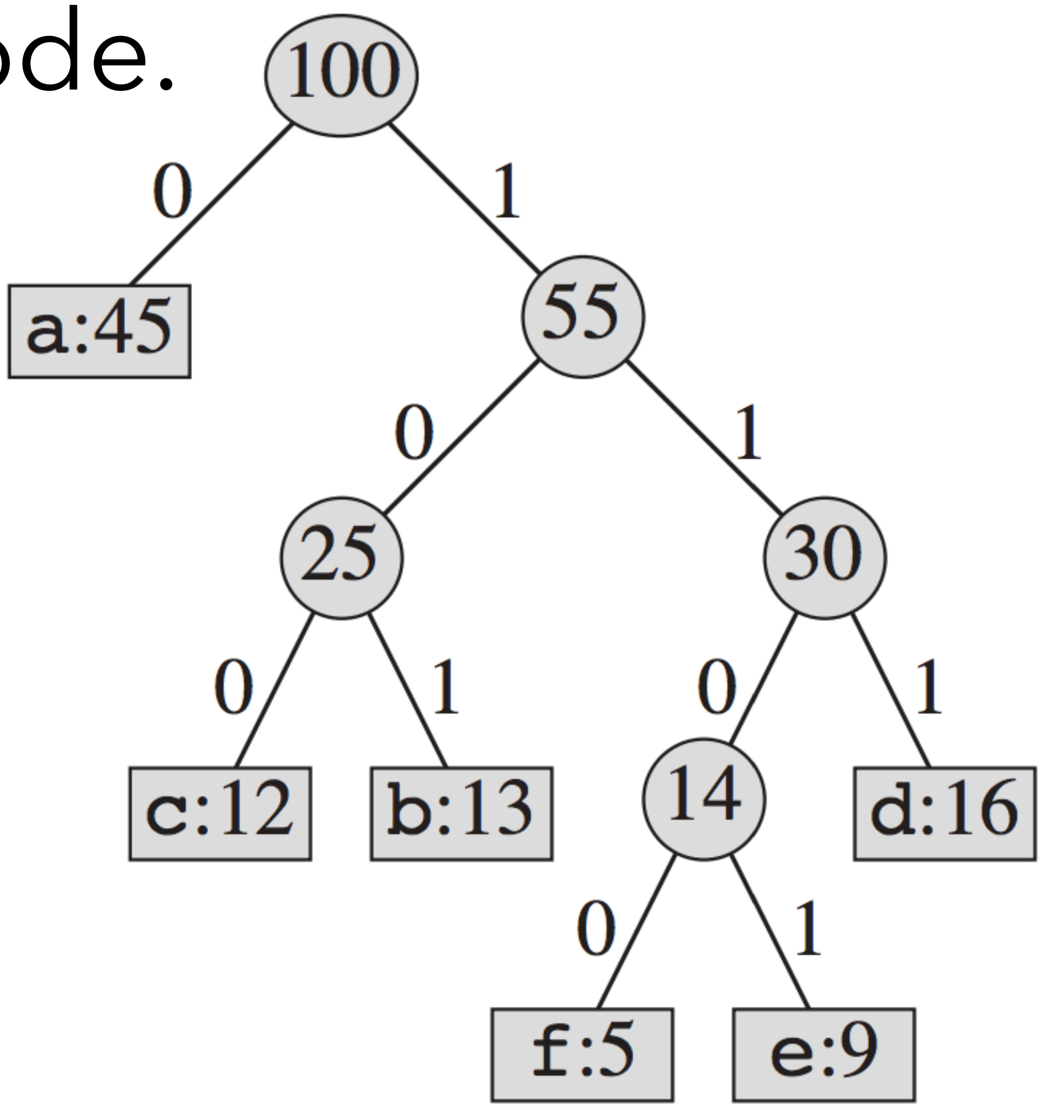
Encoding

Apply the inverse map to encode.

Example: encode `bad`

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

Character	Encoding
a	0
c	100
b	101
d	111
f	1100
e	1101



```
1010
```

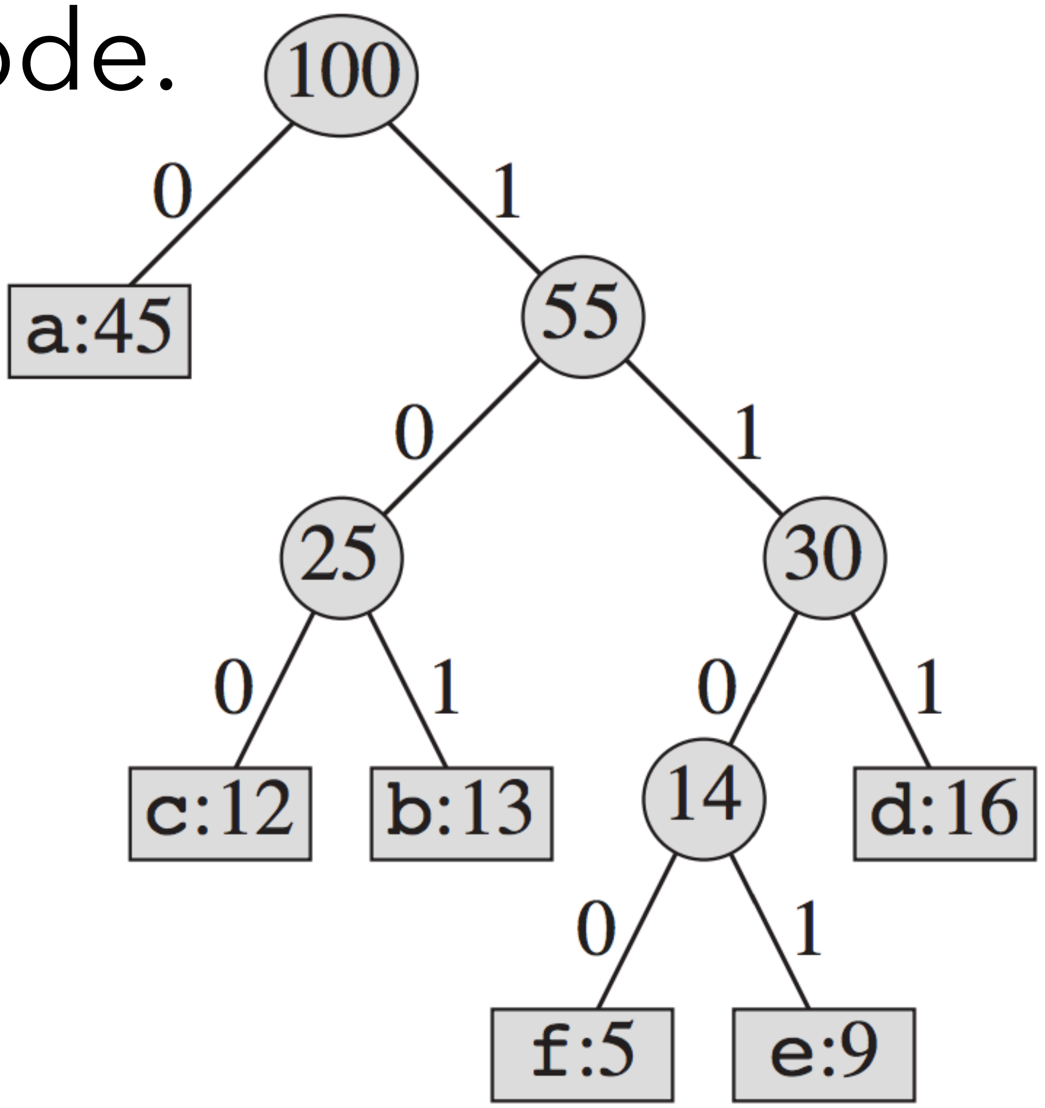
Encoding

Apply the inverse map to encode.

Example: encode `bad`

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

Character	Encoding
a	0
c	100
b	101
d	111
f	1100
e	1101



```
1010111
```

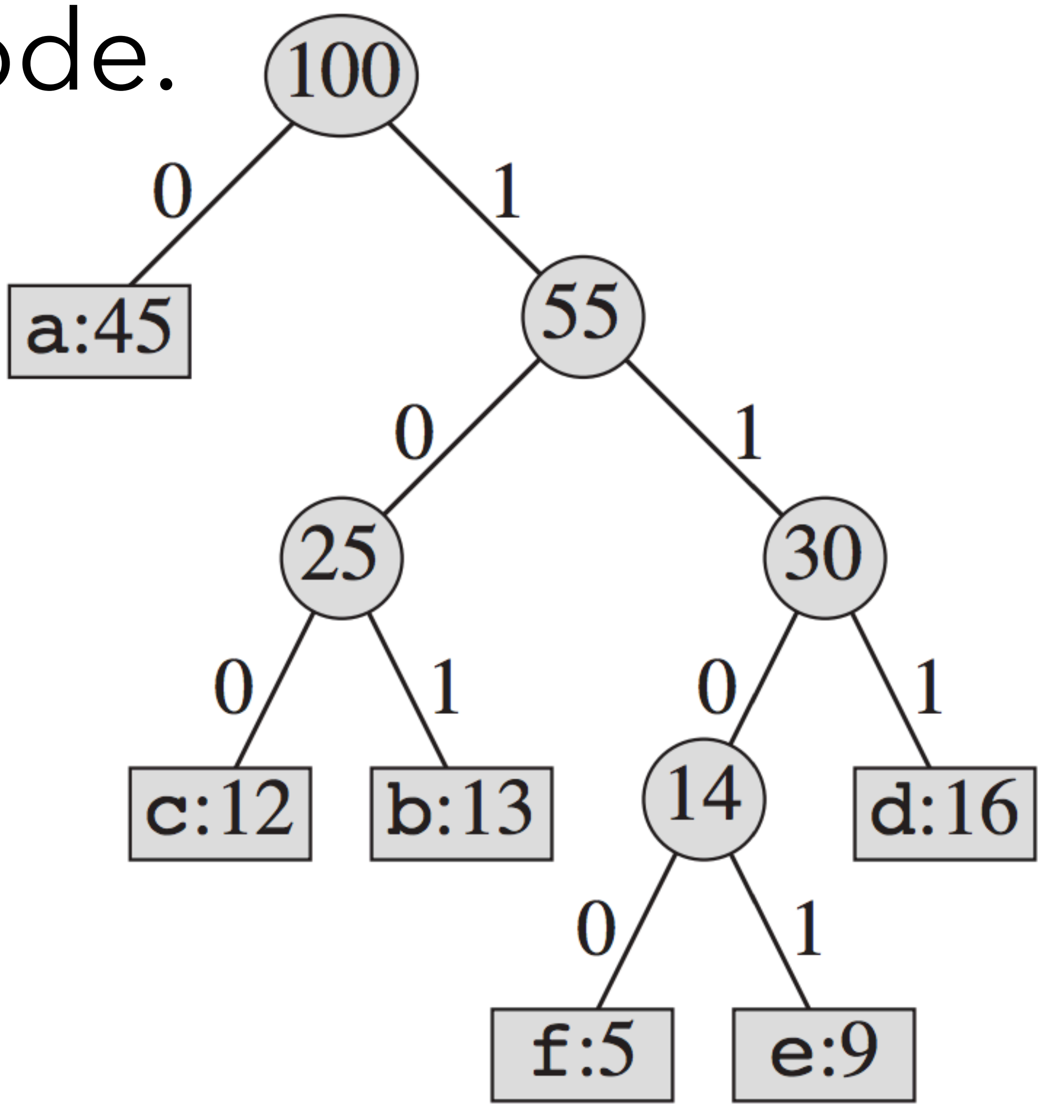
Encoding

Apply the inverse map to encode.

Example: encode **bad**

Key	Value
0	a
100	c
101	b
111	d
1100	f
1101	e

Character	Encoding
a	0
c	100
b	101
d	111
f	1100
e	1101



$\underline{101}\,\underline{0}\,\underline{111}$