

# Data Compression

A Brief Introduction

Scotty Smith

11/18/2010

# Goals

- What is Data Compression?
- Lossy vs. Lossless Compression
- Huffman Coding
- Further Discussion

# What is Data Compression

The process of encoding information using fewer bits (or other information-bearing units) than an unencoded representation would use, through use of specific encoding schemes.<sup>1</sup>

## Sample Schemes:

- JPEG
- MPEG
- MP3
- Zip

# What is Data Compression

Consider an image (In an uncompressed format):

- 640 pixels by 480 pixels.
- 16 bit color depth (5 bits each Red/Blue, 6 bits Green)

Size: 37.9 KB



$$640 \times 480 = 307,200 \times 2 = 614,400 \text{ bytes}$$

# Exercise

Find an image on the Internet, and paste it into Paint. Save it in a couple different file formats.

- Which formats lower the file size?
- Do any of the image formats achieve a file size close to the calculated one?
- Do all of the image formats represent the image in the same way?

Try compressing a text file with zip and compress. Do these compression program recover the original file exactly?

# Lossy vs. Lossless Compression

Lossless compression loses no information after encoding.

Lossy compression loses information, but is generally tuned so that the loss cannot be perceived

A lot of media compression techniques are lossy:

- JPEG
- MP3

Some are not:

- FLAC
- PNG

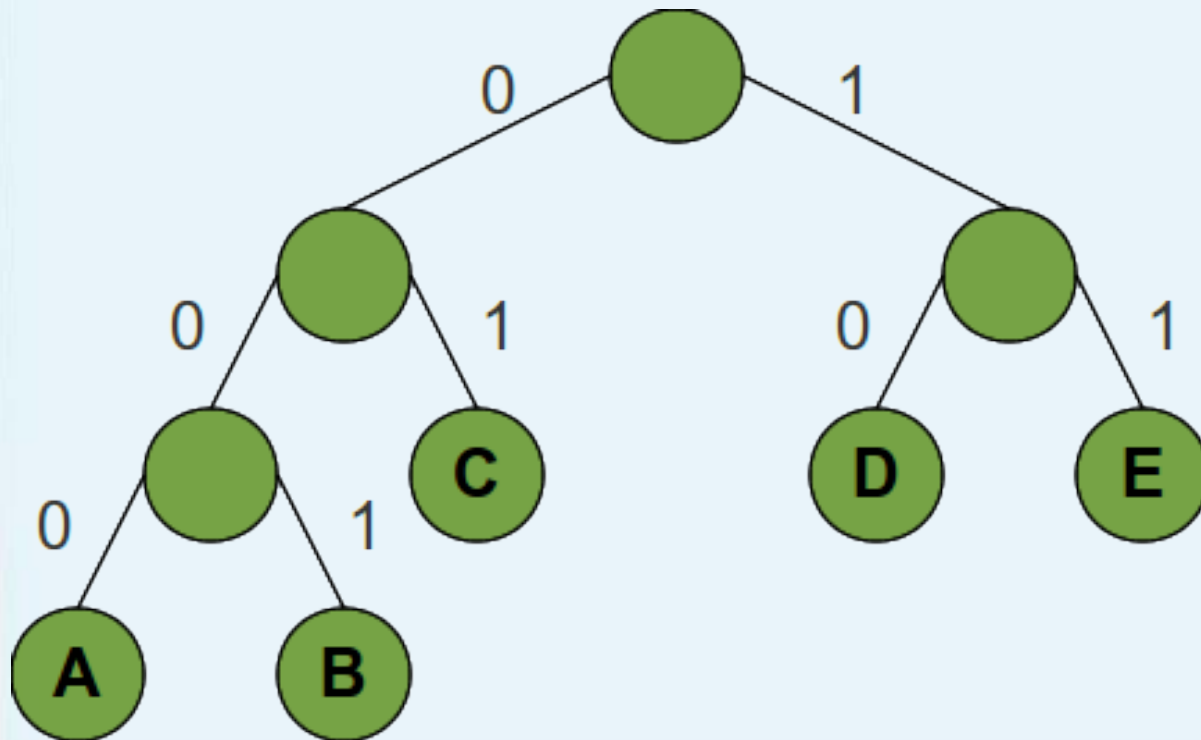
# Lossless Compression

An optimal coding technique satisfies two properties:

- 1) For any two letters  $a_i$  and  $a_j$ , if  $p[a_i] \leq p[a_j]$  then the code word length  $l_i \geq l_j$  for all  $i, j$ .
- 2) The two letters with the lowest probabilities have the same code length (Max in the tree).

# Huffman Coding from 10,000 Feet

- Create a Binary Tree (Will discuss later)
- Each path from root to leaf represents a bit code for the element at the leaf.

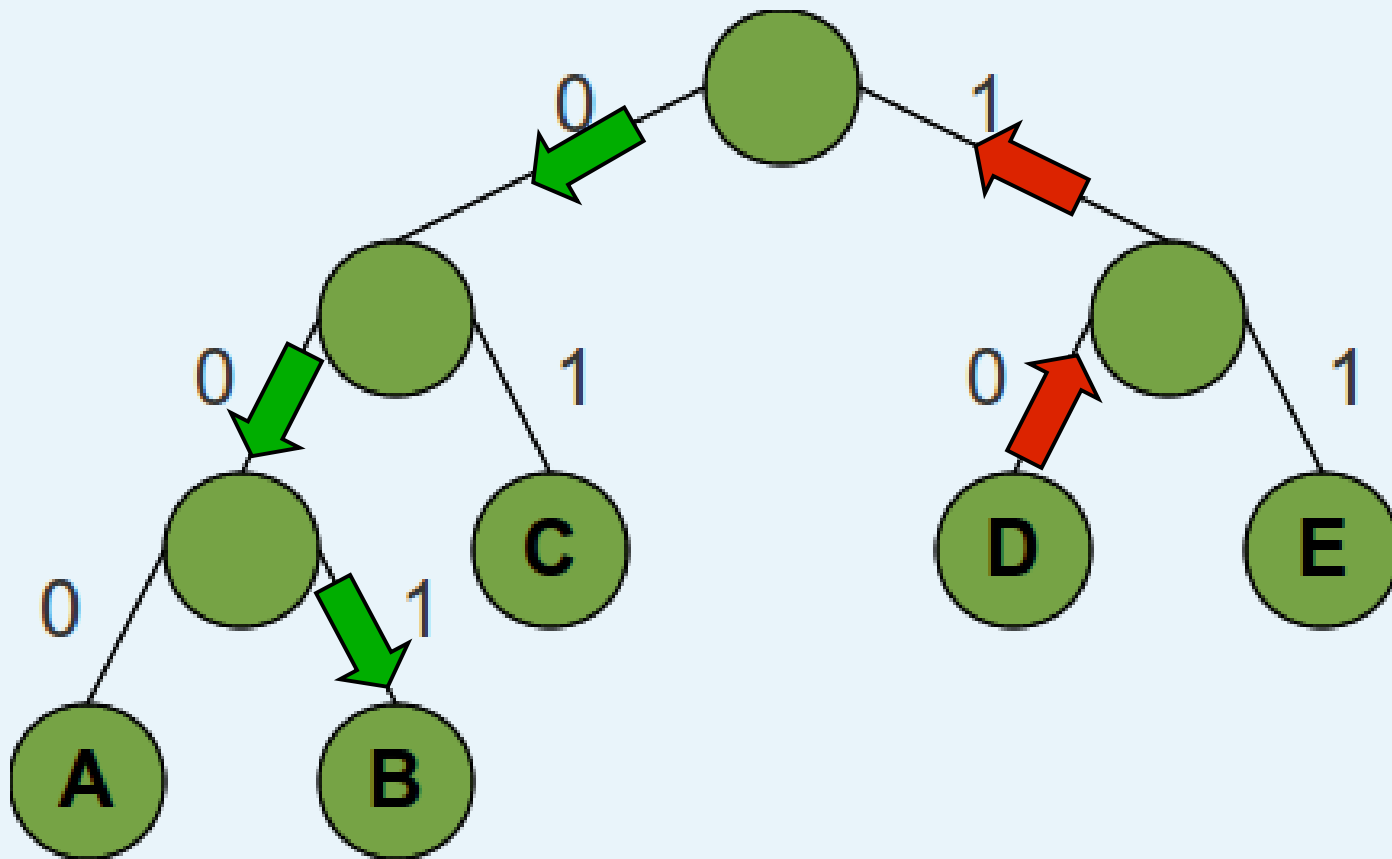


A	000
B	001
C	01
D	10
E	11

# Huffman Coding from 10,000 Feet

 Encoding: D = 10

 Decoding: 001 = B



# Exercise

With the provided handout, encode your favorite 5-15 character word/phrase.

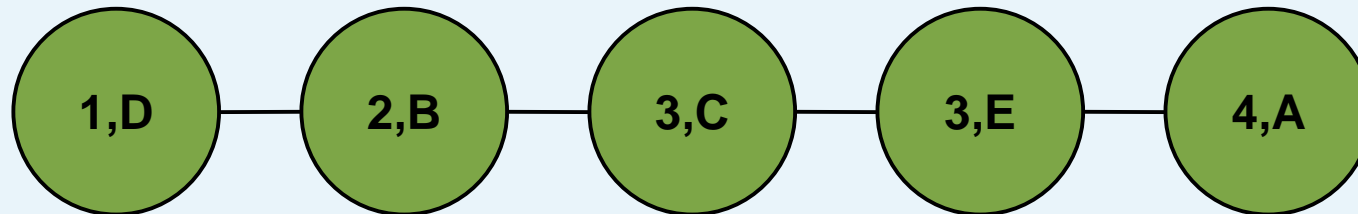
- How many bits did you have to use?
- How many bits would have been used assuming ASCII (7 bits per word)?
- How many bits would have been used using the minimal fixed length code words?  
(26 characters implies 5 bits)

Exchange bit patterns with someone near you, and decode their word.

# Building a Huffman Tree

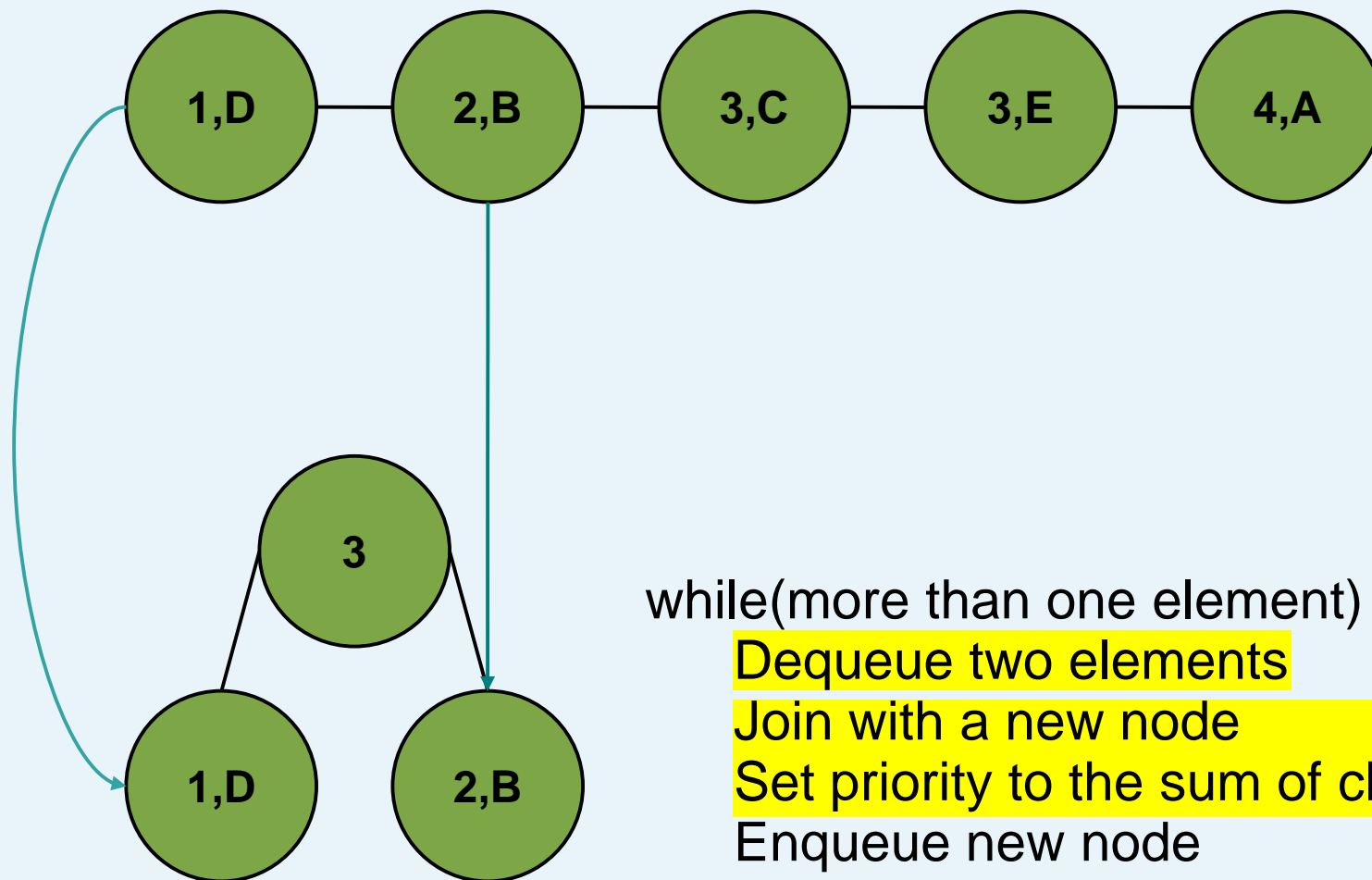
Build a Priority Queue

A	B	C	D	E
4	2	3	1	3



What data structures could we use to represent a Priority Queue?

# Building a Huffman Tree



while(more than one element)

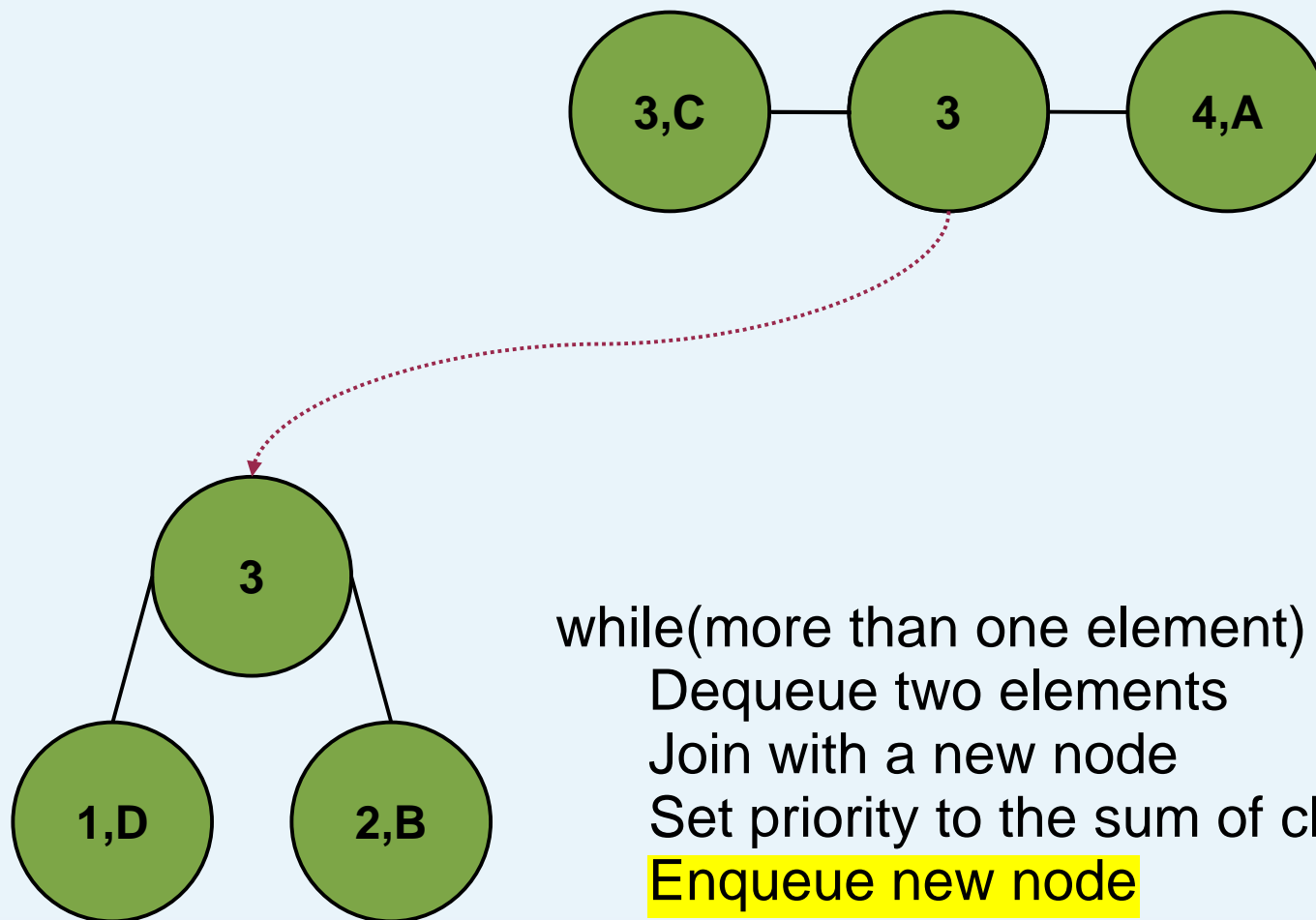
Dequeue two elements

Join with a new node

Set priority to the sum of children

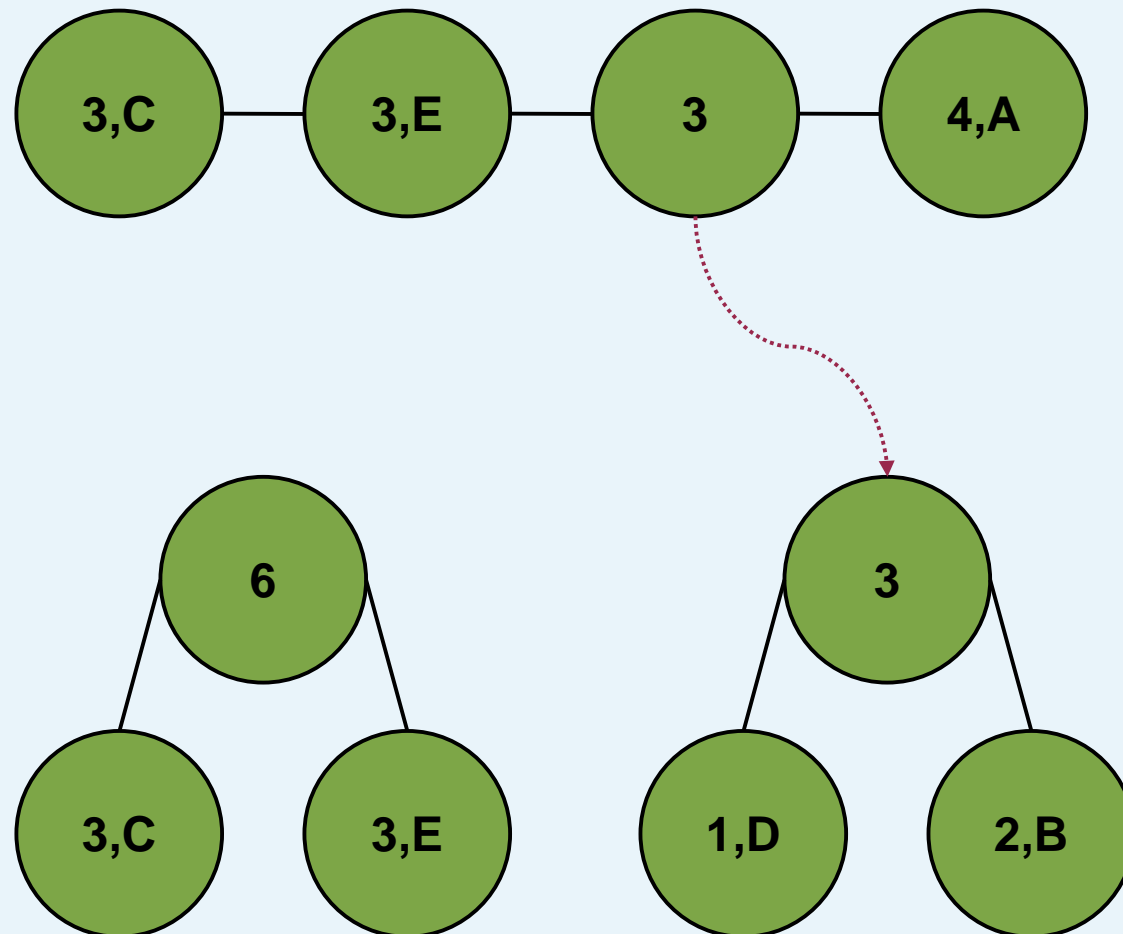
Enqueue new node

# Building a Huffman Tree

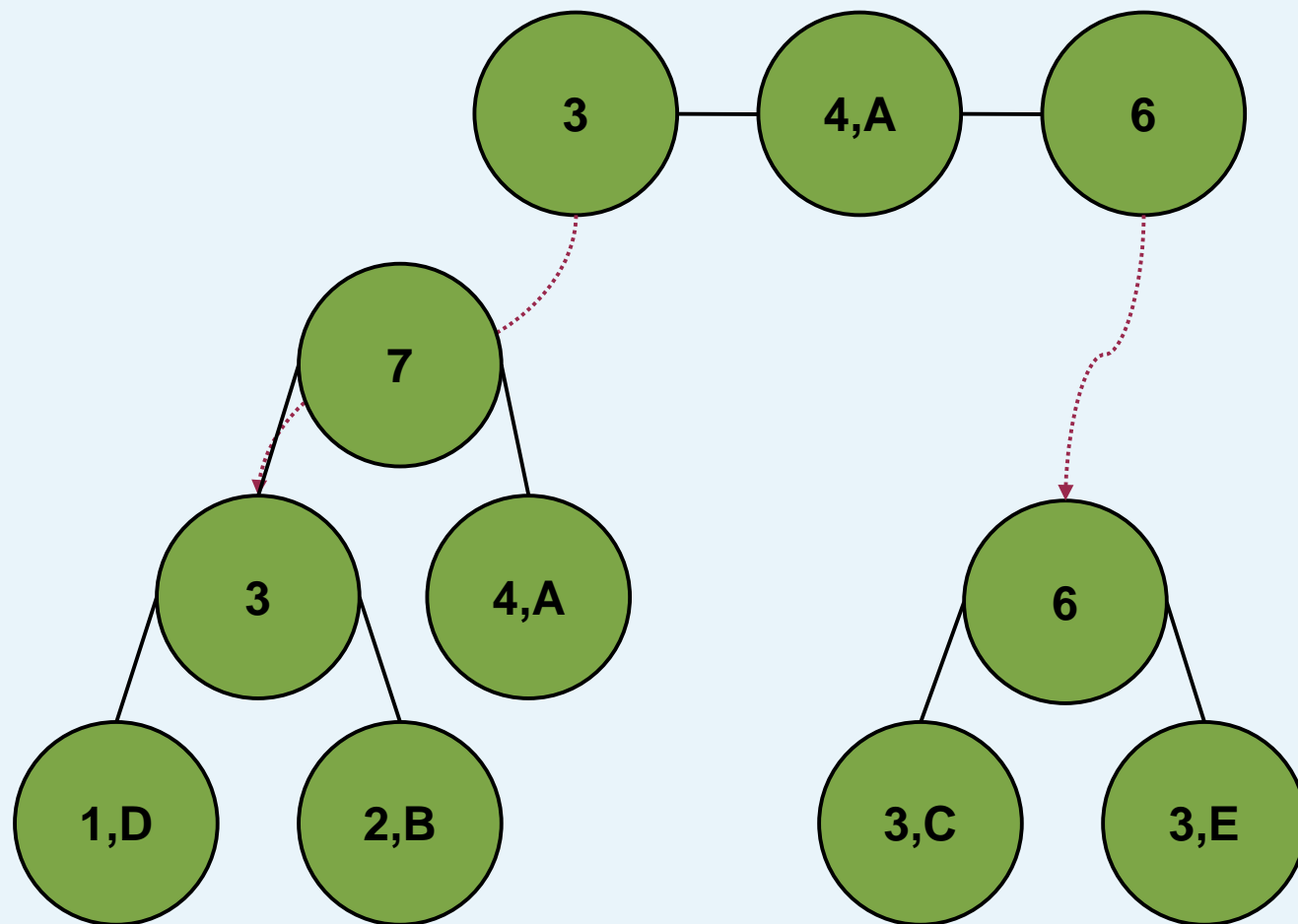


while(more than one element)  
Dequeue two elements  
Join with a new node  
Set priority to the sum of children  
**Enqueue new node**

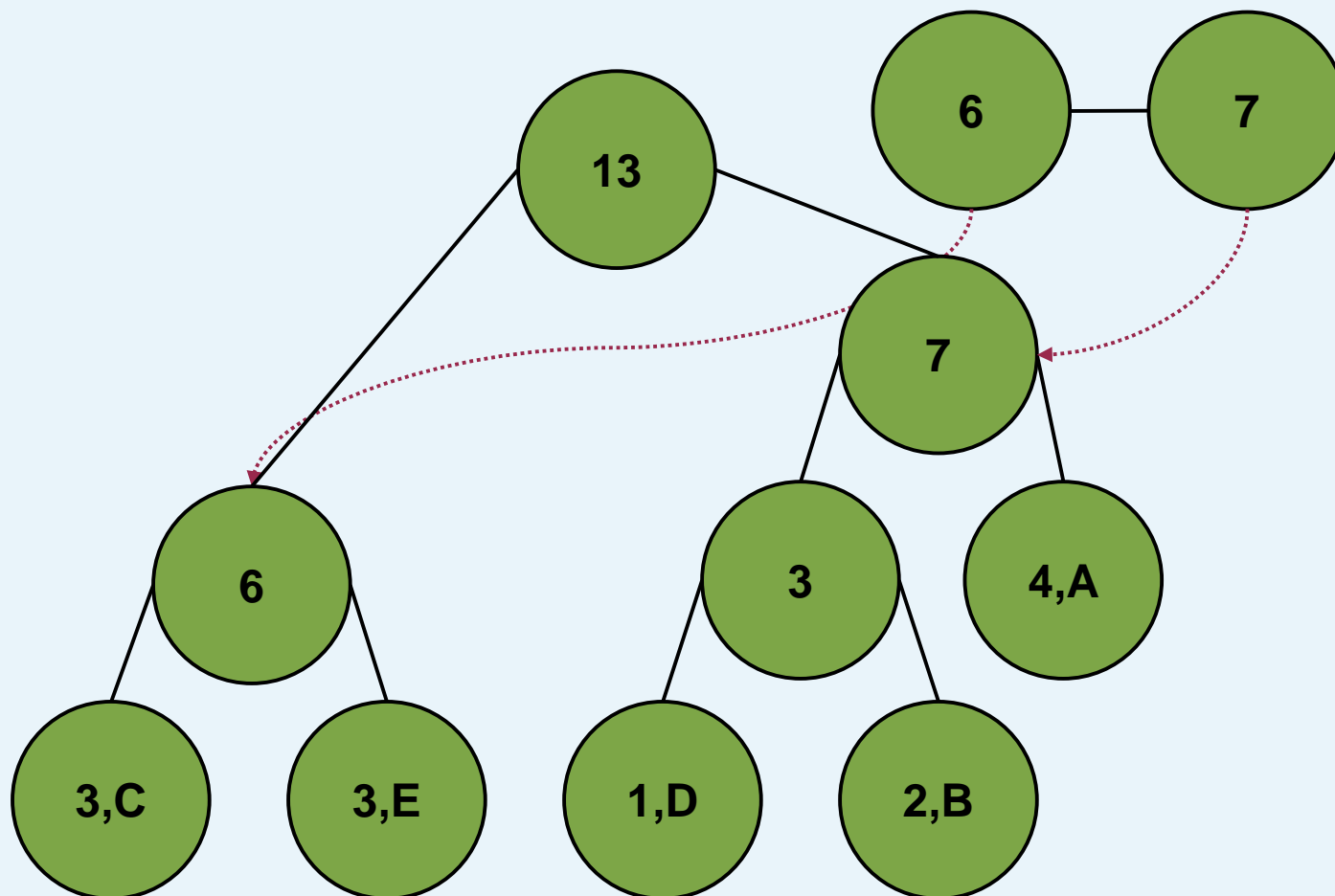
# Building a Huffman Tree



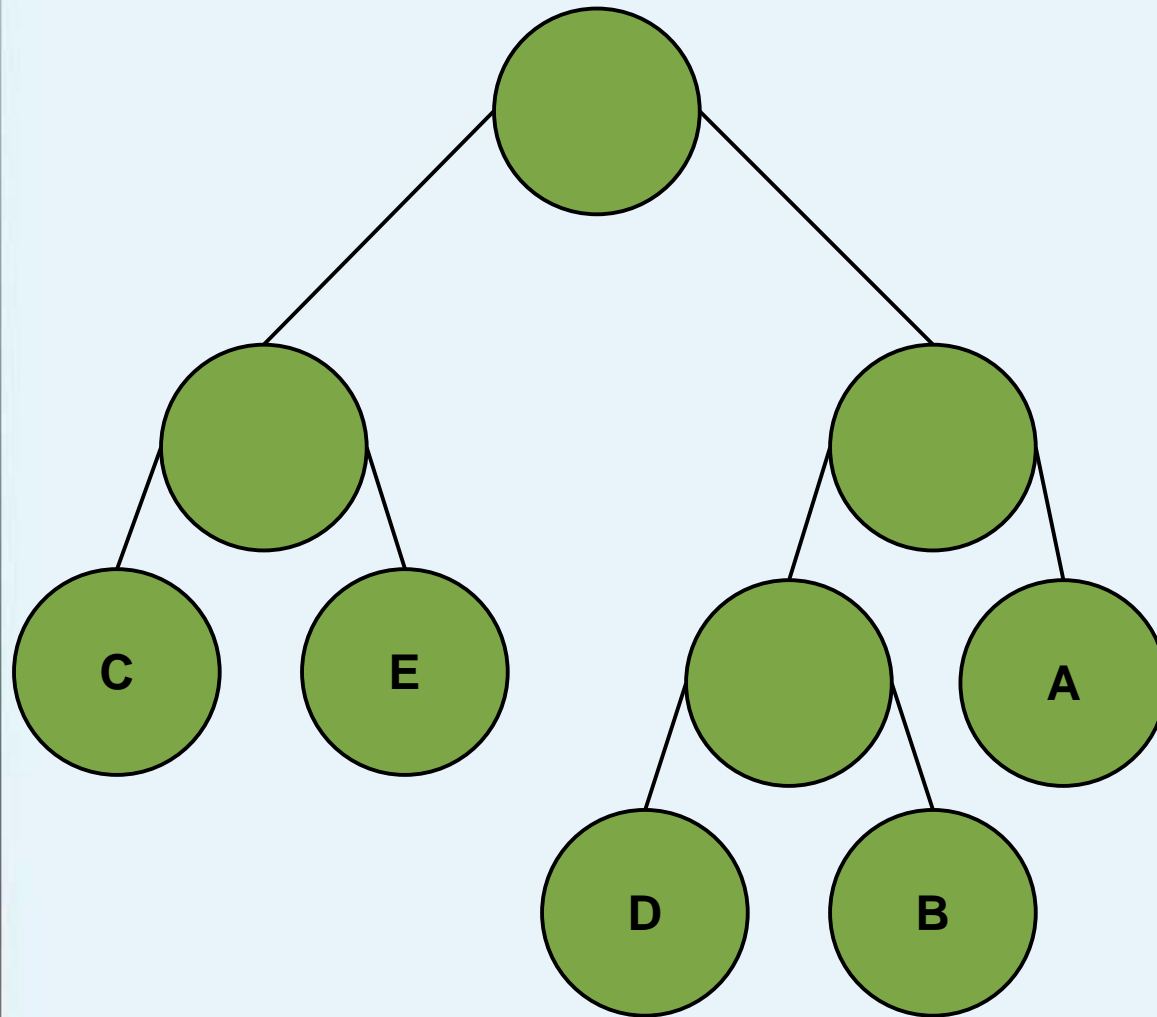
# Building a Huffman Tree



# Building a Huffman Tree



# Building a Huffman Tree



A	11
B	101
C	00
D	100
E	10

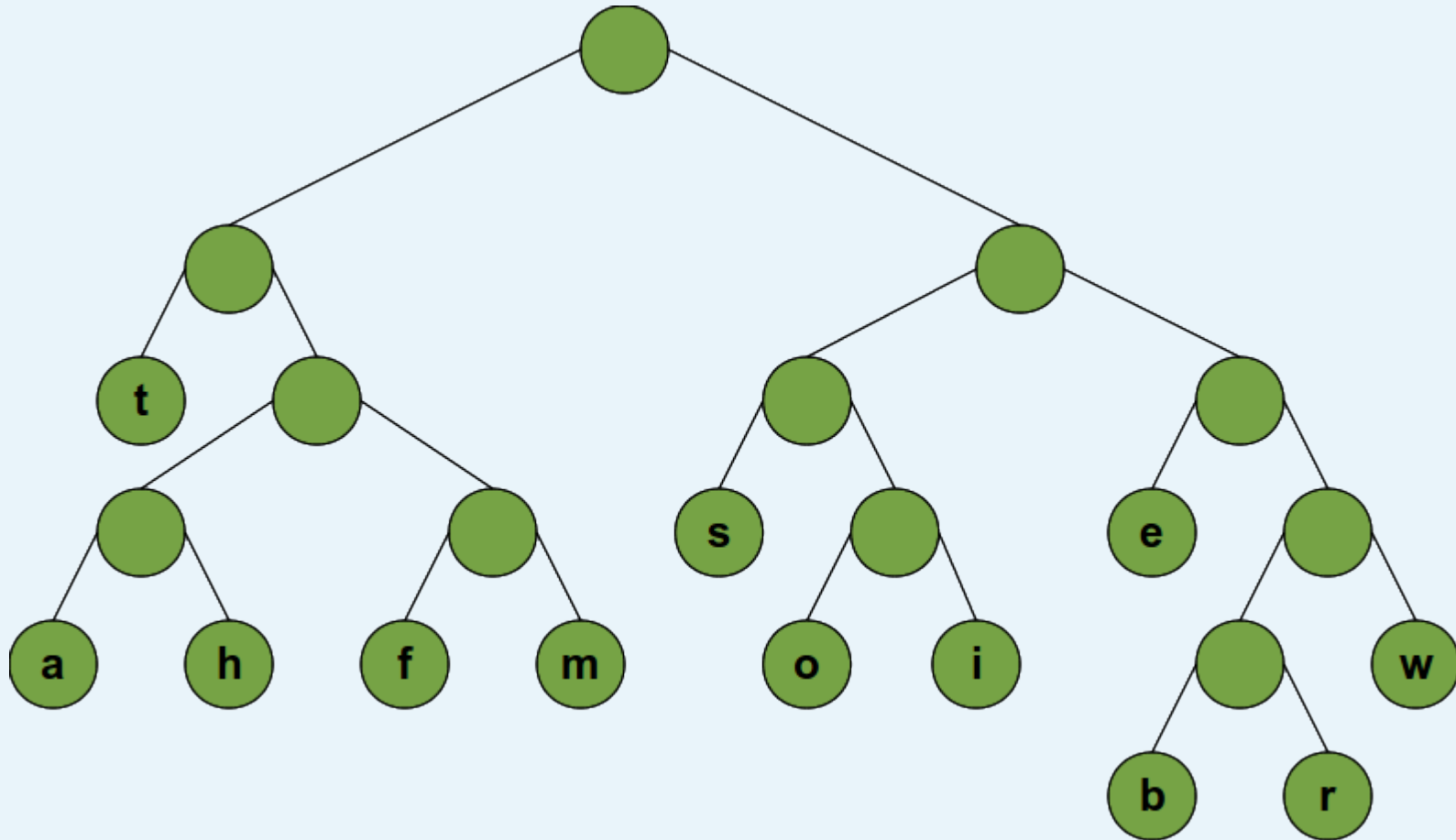
# Exercise

Build a Huffman tree for the following phrase:

“it was the best of times, it was the worst of times”

Ignore spaces, omit characters not found in the phrase

# Exercise Results



3.8 bits per character  
4 bits per character (fixed width)

128 bits encoded  
156 bits fixed  
28 bits saved

# Further Discussion

## Missing Pieces

- What is needed to decode a file encoded with Huffman?
- Tree building run-time complexity?
- Optimality?

Create a Priority Queue  
while(more than one element)  
  Dequeue two elements  
  Join elements with new node  
  Set priority to the sum of children  
  Enqueue new node

# Huffman Optimality

Recall the two features of an optimal code:

1) For any two letters  $a_i$  and  $a_j$ , if  $p[a_i] \leq p[a_j]$  then the code word length  $l_i \geq l_j$  for all  $i, j$ .



2) The two letters with the lowest probabilities have the same code length (Max in the tree)

Proof of (1) is by contradiction

# Further Discussion

What do mainstream compression programs use?

- Zip/GZip/7-zip – DEFLATE (Lempel-Ziv '77 and Huffman)
- PNG – DEFLATE
- Fax Machines – Modified Huffman (Combines Huffman with Run-Length Encoding)

# Lempel-Ziv

Consider a file containing the phrase:

“**it was the** best **of times**, **it was the** worst **of times**”

This phrase has a lot of redundancy

# Lempel-Ziv

Consider a file containing the phrase:

**“it was the best of times, (26, 11) worst (18, 8) ”**

This phrase has a lot of redundancy

Lempel-Ziv actually operates at character (Or bit) level, so an encoding of the above would look more like this:

$(0,0,i)(0,0,t)(0,0,\' \')\dots(12,2,i)\dots(26,10,w)\dots$

# Further Discussion

What about Lossy Compression?

Most lossy techniques use algorithms from lossless compression

Example:

Most MP3's are ultimately encoded with Huffman.

Data loss comes from removing data that the human ear cannot hear.

# Want to Learn More?

Data Compression (Grad Course: CSCI 6351)

Course Website: <http://www.seas.gwu.edu/~ayoussef/cs225/>

<http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf>

[The \\$5000 Compression Challenge](#)

[The Million Random Digit Challenge Revisited](#)