Under Pressure

Compressing and Encoding Data

Charlie Chiccarine

PaTTAN Computer Science Praxis Prep

What is Compression?

Compression is taking data and making it smaller by removing redundancies

What are common applications of compression?

Making .zip folders

Encoding

- Encoding is the act of taking information and turning it into a code
- Unique pieces of data have unique codes
- Some common examples
  - Morse code
  - Braille
  - ASCII
  - Unicode

Run Length Encoding

What is it?

A way to encode data

Why would we use it?

When data is very repetitive, it compresses the file

How do we do it?

See the next slide

Run Length Encoding

2 2 2 2 2 2 2 2 5 6 6 6 3 3 3 2 2

Let's encode the above line of numbers using RLE

1. Count the amount of times the first element appears
2. Append the count and the element to your code
   - Since 2 appears 8 times, we would shorten the line to 8 2
3. Repeat steps 1 and 2 until you've finished

Completed Code:

8 2 1 5 3 6 3 3 2 2

Run Length Encoding - Good Case

- We've shortened the previous code from 17 elements to 10 elements and still kept all the data in tact
- RLE works better on codes that are very repetitive
- The best case possible, would be a code of only one element repeating
- In the best case, we would take a line of x elements and reduce it to only 2

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

▼

23 2

Run Length Encoding - Bad Case

- In the previous examples, we shortened the codes
- In many cases, RLE will actually increase the length of the codes
- This happens when there are many unique values
- In the worst case, we would take a line of x elements and increase it to $2x$ elements

2 5 2 6 2 7 2 8 2 9

▼

1 2 1 5 1 2 1 6 1 2 1 7 1 2 1 8 1 2 1 9

Run Length Encoding - Practice

Encode the following sequence using RLE:

a a a b c d d d d d d b c

What is the resulting encoding?
Was this effective in compressing the sequence?
Why or why not?

Run Length Encoding - Practice

What is the resulting encoding?

3 a 1 b 1 c 6 d 1 b 1 c

Was this effective in compressing the sequence?

Yes, the original was 13 elements long. Ours is 12 elements long

Why or why not?

Since there was a fair amount of repetiton, the sequence was able to be compressed

Run Length Encoding - Practice

Decode the following sequence using RLE:

5 a 2 b 1 c 1 z 3 y 1 a

What is the resulting decoding?
Was this effective in compressing the sequence?
Why or why not?

Run Length Encoding - Practice

What is the resulting decoding?

a a a a a b b c z y y y a

Was this effective in compressing the sequence?

Yes, the original was 13 elements long. The encoding is 12 elements long

Why or why not?

Since there was a fair amount of repetiton, the sequence was able to be compressed

Huffman Encoding

What is it?

A way to encode data with frequencies

Why would we use it?

More frequent data is given shorter codes, while less frequent data is given longer codes

How do we do it?

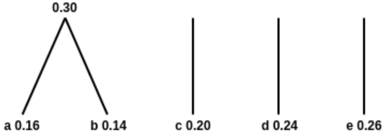
See the next slide

Huffman Encoding

- We are going to encode the following datapoints a, b, c, d, e
- These have the following frequencies, respectively, 0.16, 0.14, 0.20, 0.24, 0.26
- I attached stems to each value because we're gonna build a tree
- A tree is a way to organize data

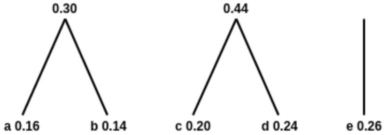
Huffman Encoding

- First we connect the two smallest frequencies (0.16 & 0.14)
- We now have a subtree and three leaves
  - A leaf is the smallest part of a tree
  - Each letter is a leaf
- Now we have 4 frequencies
  - 0.30, 0.20, 0.24, 0.26



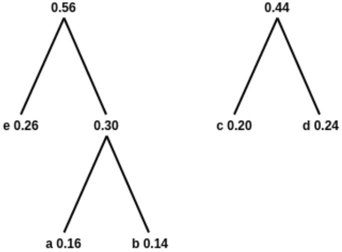
Huffman Encoding

- Now we connect the two smallest frequencies again (0.20 & 0.24)
- We have two subtrees and a leaf
- We have 3 frequencies
  - 0.30, 0.44, 0.26



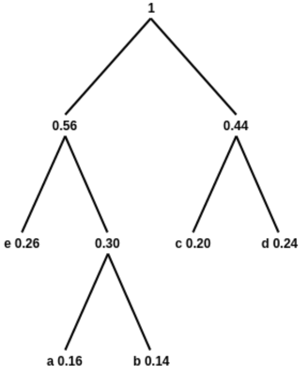
Huffman Encoding

- Once again, we combine the two smallest frequencies (0.30 & 0.26)
- We are down to two frequencies
  - 0.44 & 0.56



Huffman Encoding

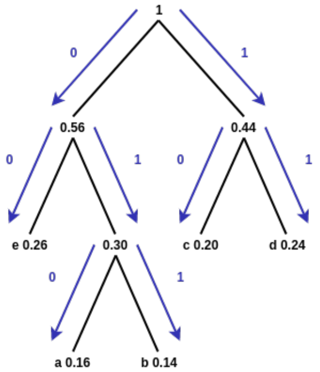
- Finally, we combine the last two subtrees
- Now we have a full tree
- What do we do with this tree?
- What does any of this have to do with encoding?



Huffman Encoding

- Now we find our values to get their encoding
- If we fall left down the tree, we add a 0 to the code
- If we fall right, we add a 1
- We get the following encoding for each data point

Data	Frequency	Code
a	0.16	010
b	0.14	011
c	0.20	10
d	0.24	11
e	0.26	00



Huffman Encoding - Conclusions

- The smaller the frequency, the longer the code
- No code is the prefix for another code
  - The prefix for a and b is 01
  - 01 is not a code for another letter

Data	Frequency	Code
a	0.16	010
b	0.14	011
c	0.20	10
d	0.24	11
e	0.26	00

Lossy Compression

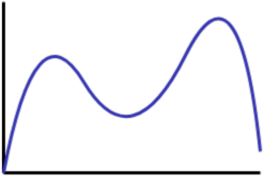
- Run Length Encoding and Huffman Tree Encoding are examples of lossless compression
- Lossy compression is when you compress data, but lose information
- There is a video about lossy compression in the Data Representation folder

Voice Transmission

- Phone companies use this when you talk on the telephone
- When you have a sound, it's analog, when we make the signal digital, we lose data
- Phone companies use lossy compression in a clever way so that while information is lost, quality is still preserved

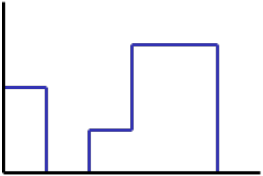
Digital vs Analog

Analog



- Can represent a wide range of values
- Much smoother than digital
- This is what sound normally is

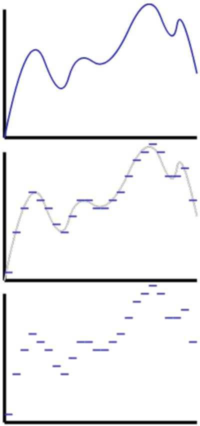
Digital



- Discrete in nature
- Typically just on and off
- For our purposes, whole numbers
- What we turn sound into

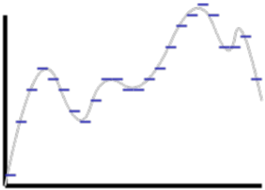
Voice Transmission

- The wavy sound is taken and digitized
- This means chopping it up and only taking certain levels
- This is where we lose data
- Data is sliced 8000 times per second
  - 1 every 125 microseconds

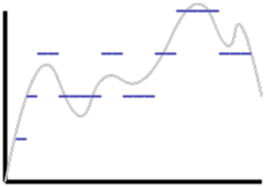


Voice Transmission

- Each horizontal level is a different value
- More bits = more levels
- 12 bits would give us 4096 different levels for sound
- The sound is more accurate, so there's less static
- More Bits, More Accurate, More Information, Less Static
- Less Bits, Less Accurate, Less Information, More Static



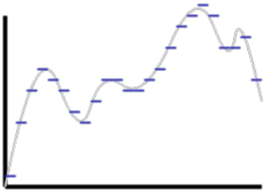
More bits



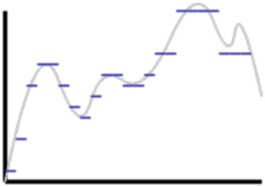
Less bits

Voice Transmission

- Since there's static when there are less bits, phone companies found a way to circumvent that
- Static is only annoying when sound is quiet
- When sound is loud, the static can be ignored
- So phone companies reserve more values for quieter levels and less bits for louder levels
- It looks more like what you see on the right
- Lossy compression, but clever



All levels same size



Mixed level size

Recap

What you need for the exam:

- What compression and encoding are
- The difference between lossy and lossless compression

What you don't need to know, but I told you about anyway:

- How to compress voice transmissions and all the information relating to that