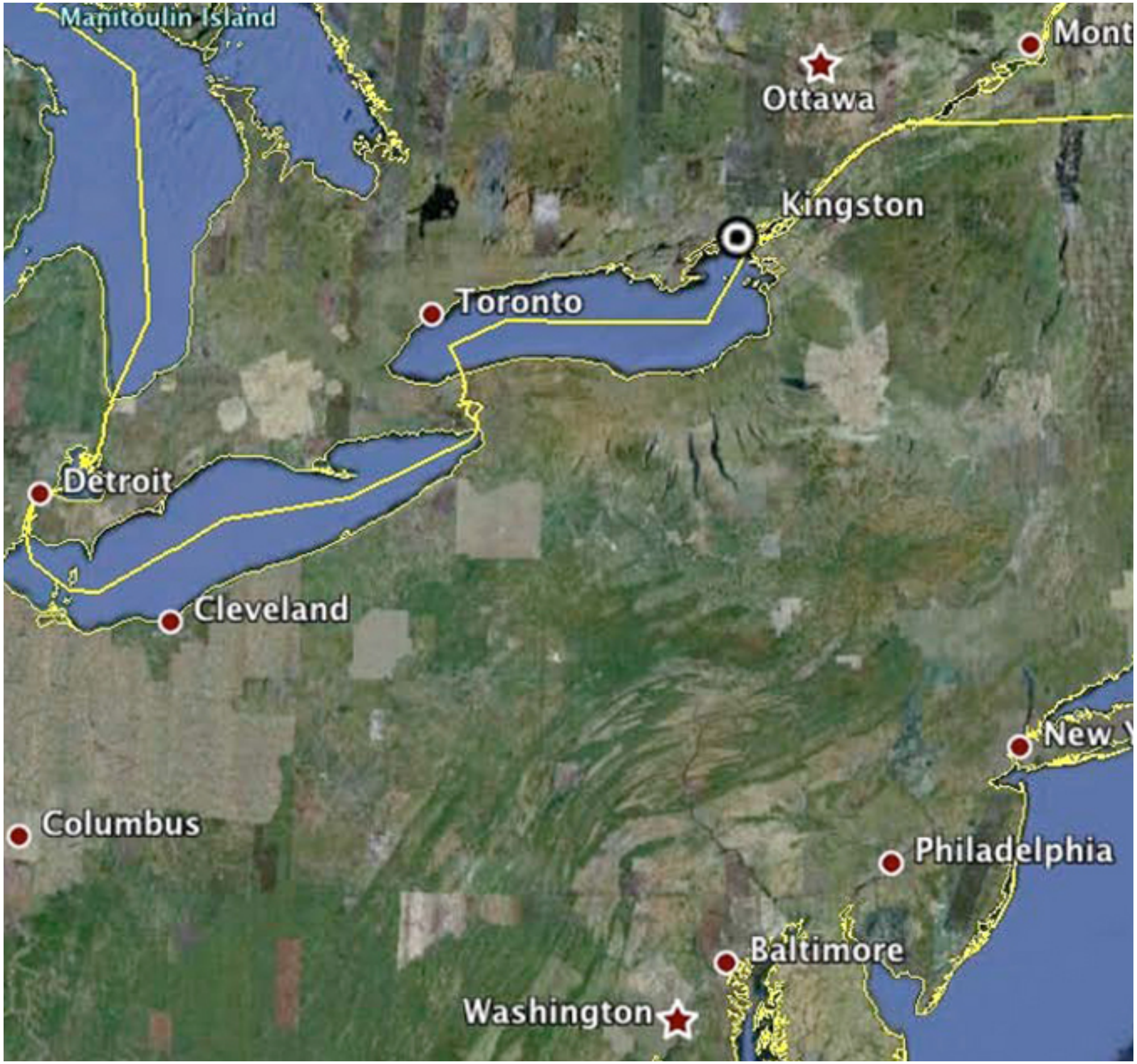
David Rappaport
School of Computing
Queen's University
CANADA


Manitoulin Island
Ottawa
Kingston
Toronto
Detroit
Cleveland
Columbus
Philadelphia
Baltimore
Washington

Data Compression

There are two broad categories of data compression:

- Lossless Compression *e.g.* gif, gzip
- Lossy Compression *e.g.* mp3, jpeg

Lossless and Lossy Compression

Lossless	**Lossy**
An exact copy of the original data is obtained after decompression	Original information content is lost.
Structured data can be compressed to 40-60 percent of original size	Any data can be compressed. Sometimes by 90% or more.

What is information?

The colour of the hat is red.

The colour of the book is red.

The colour of the table is red.

The colour of the hat is red.
The ---- " ------ book is " .
The ---- " ------ table is " .

The book, hat, and table are red.

Modelling Information

- The area of information theory explores the information content of data.
- We can predict the size of the content by modelling the data.
- Within a given model we can obtain lower bounds on the number of bits required to represent the data.

Modelling Information

Consider the following *message*

$x_1, x_2, \ldots, x_{12}$:

9 11 11 11 14 13 15 17 16 17 20 21

Using binary encoding we can store 0..21 using 5 bits per number.

Block Encoding

A decimal number say 1042 can be thought of as
$(1 \times 1000)+(0 \times 100)+(4 \times 10)+(2 \times 1)$
$= (1 \times 10^3)+(0 \times 10^2)+(4 \times 10^1)+(2 \times 10^0)$

With 4 decimal digits we can store values [0 .. 9999] or 10^4 different values.

Block Encoding

A binary number say 1001 can be thought of as
$(1 \times 2^3)+(0 \times 2^2)+(0 \times 2^1)+(1 \times 2^0) = ?$

With 4 binary bits we can store values $[0 .. 2^4 - 1]$ or $2^4 = 16$ different values.

With 5 binary bits we can store 32 different values.

Modelling Information

9 11 11 11 14 13 15 17 16 17 20 21

Or we can store:

```
0 2 2 2 5 4 6 8 7 8 11 12
```

A value n in our message actually represents the number n+9. Using binary encoding we can store 0..12 using 4 bits per number.

Modelling Information

9 11 11 11 14 13 15 17 16 17 20 21

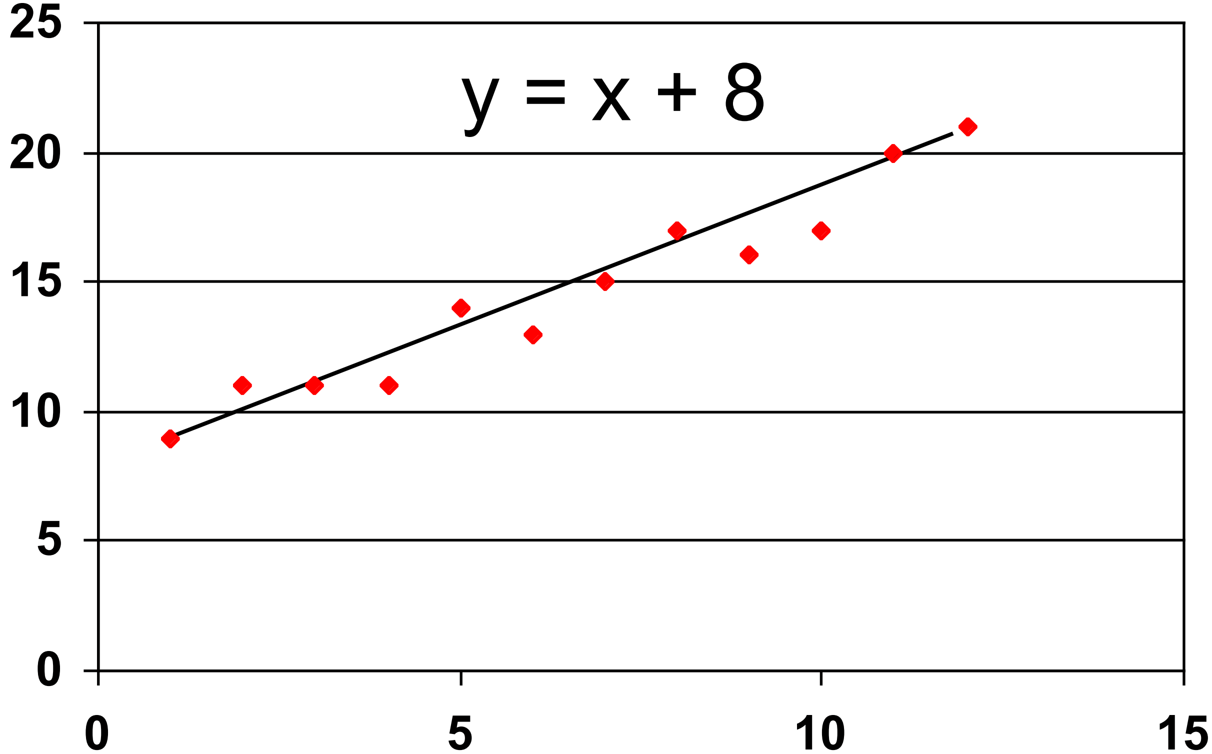
Or we can store:

9 2 0 0 3 -1 2 2 -1 1 3 1

x_1 is stored as is. To encode x_i we use $x_i - x_{i-1}$. We have 5 distinct values which can be encoded with 3 bits.

Modelling Information

9 11 11 11 14 13 15 17 16 17 20 21



Modelling Information

9 11 11 11 14 13 15 17 16 17 20 21

Let x_i denote the ith number in our message,

and let y_i be set to i+8.

We can encode $e_i=x_i-y_i$ giving:

0 1 0 -1 1 -1 0 1 -1 -1 1 1

There are 3 distinct values so 2 bits per number suffices.

Minimum Redundancy Coding

Let’s take a more organized approach with the next example: Suppose we have a message consisting of 5 distinct characters, that appear with the given frequencies.

A-15, B-7, C-6, D-6, E-5.

First Attempt

A-15, B-7, C-6, D-6, E-5.

A -- 000, B--001, C--010, D--011, E—100.

Three bits are needed to encode 5 distinct characters

Variable Length Code

A-15, B-7, C-6, D-6, E-5.

A -- 0, B--1, C--00, D--01, E--10.

Why won’t this code work?

Variable Length Code

A-15, B-7, C-6, D-6, E-5.

A -- 0, B--1, C--00, D--01, E--10.

Why won’t this code work?

Is 00 AA or C?

Second attempt

A-15, B-7, C-6, D-6, E-5.

A -- 01, B--000, C--001, D--111, E--100.

This code can be unambiguously decoded. Why?

Shannon-Fano Encoding

A-15, B-7, C-6, D-6, E-5.

Split sorted list of codes into roughly equal parts.

Assign first bit of left side as 0, and first bit of right side as 1.

Repeat on each side until every symbol is encoded.

Shannon-Fano Encoding

A-15, B-7, C-6, D-6, E-5.

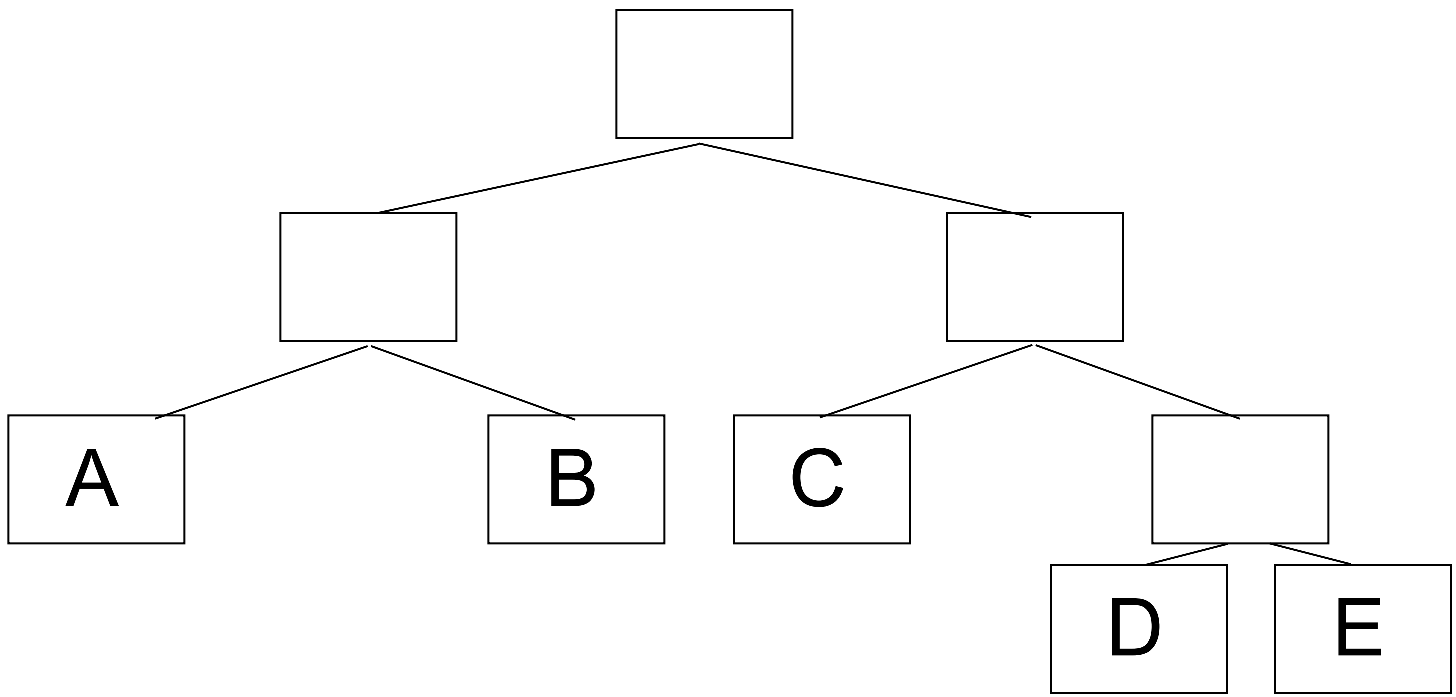
A B || C D E

A || B C || D E

D || E

Shannon-Fano encoding

This can be represented as a binary tree.



Prefix Code

A-00, B-01, C-10, D-110, E-111.

No symbol is encoded as the prefix of any other symbol. A prefix code can be decoded with no ambiguity.

Shannon-Fano Code

A-00, B-01, C-10, D-110, E-111.

Consider the following string:

0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1 0

If we are given, the frequencies, the tree, or the codes, we can decode the message.

Shannon-Fano Code

A-00, B-01, C-10, D-110, E-111.

For example try to decode the following string:

0001010010100101111110

Prefix Code

A-00, B-01, C-10, D-110, E-111.

Answer:

```
0001010010100101111110
 A  B  B  A  C  C  B  B   E   D
```

Huffman Algorithm

Encoding

Step 1. Determine the frequency of each symbol in the message. Each symbol can be thought of as a tree with a weight.

A-15	B-7	C-6	D-6	E-5

Huffman Algorithm

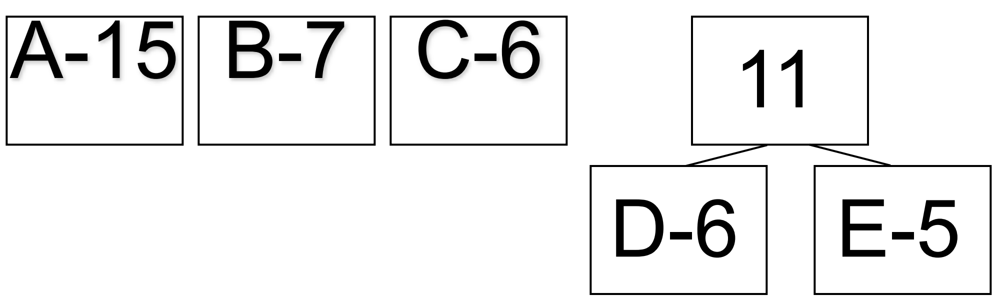
Encoding

Step 2. While there is more than one tree create a single tree from the two trees of least weight.

Huffman Algorithm

Encoding

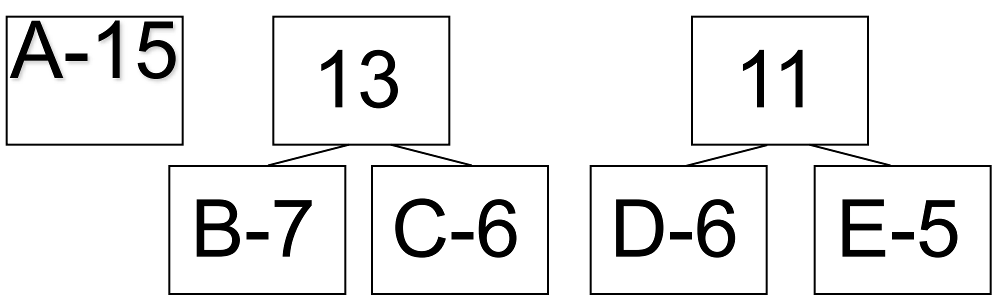
Step 2.



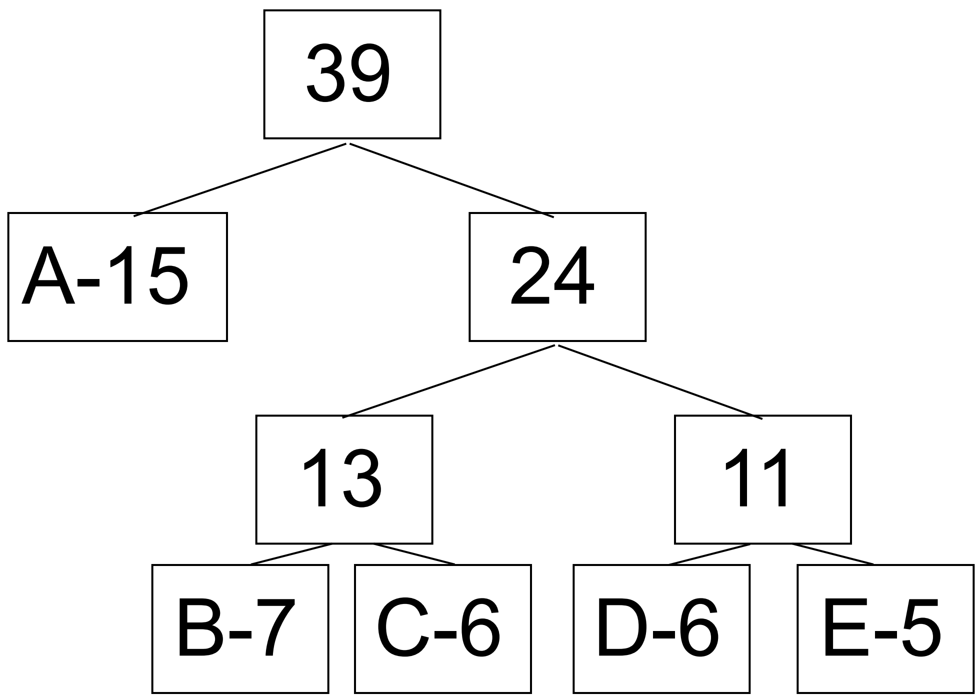
Huffman Algorithm

Encoding

Step 2.

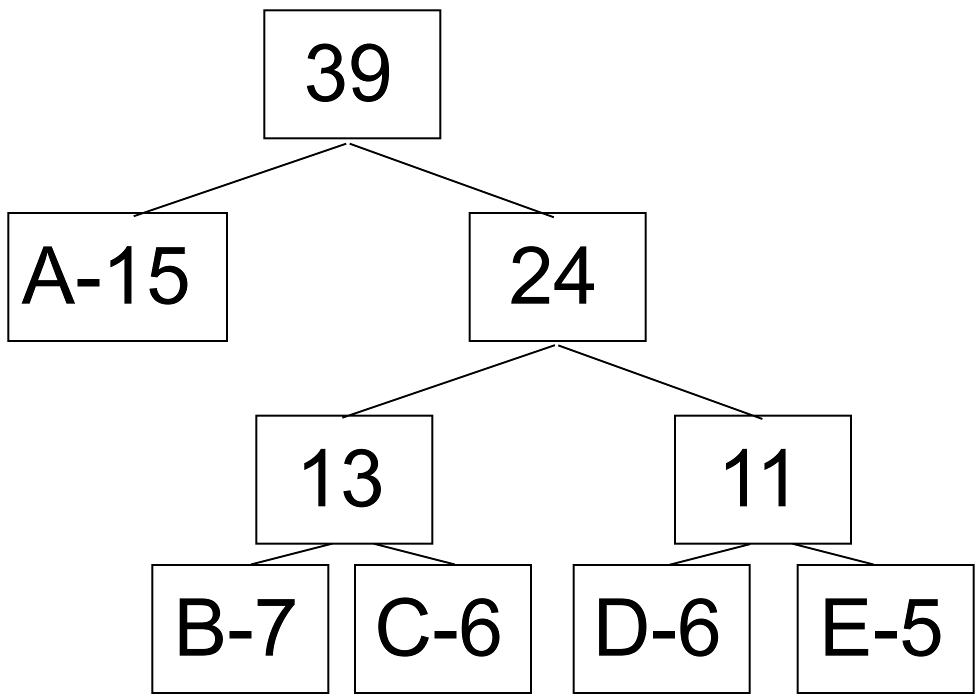


Huffman Algorithm



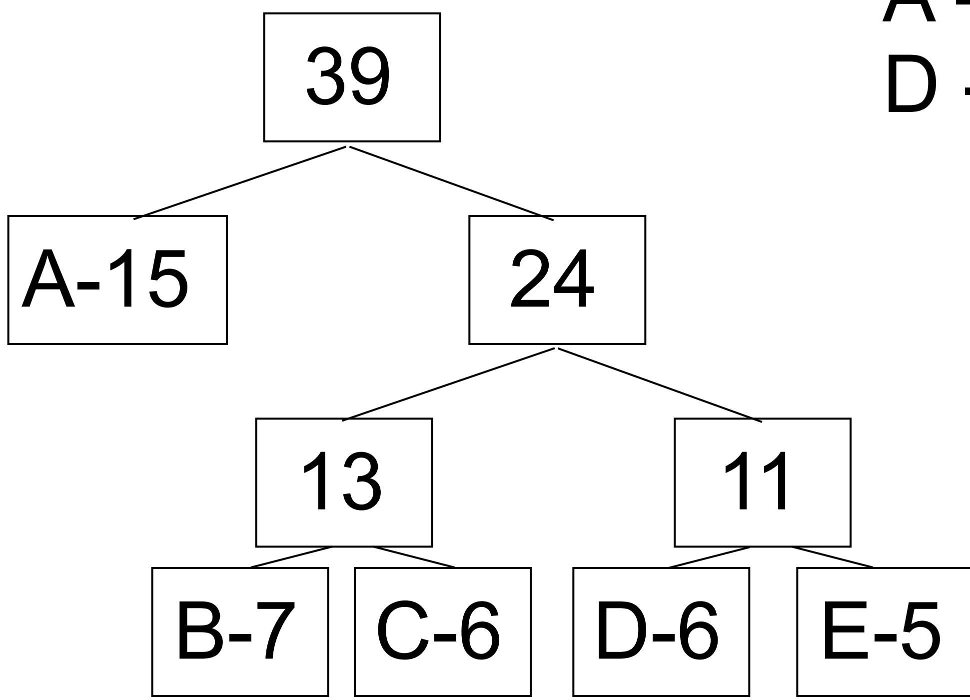
Final Huffman Tree

Huffman Algorithm



A Huffman code is obtained by following links from the root of the tree to each leaf, with a left link representing 0 and a right link 1.

Huffman Algorithm



A - 0, B - 100, C - 101, D - 110, E - 111.

Storing the Huffman Tree

To successfully decode the compressed file we need the Huffman tree. An alternate method is to simply store the counts and rebuild the tree in the decompression stage.

Huffman Algorithm

We can compare the Huffman code with the Shannon-Fano code used in our previous example.

Frequencies:

A-15, B-7, C-6, D-6, E-5.

Codes:

A-0, B-100, C-101, D-110, E-111

A-00, B-01, C-10, D-110, E-111.

Huffman Algorithm

Frequencies:

A-15, B-7, C-6, D-6, E-5.

Codes:

A - 0, B - 100, C - 101, D - 110, E - 111

15 + 3*(7 + 6 + 6 + 5) = 87

A-00, B-01, C-10, D-110, E-111.

2*(15 + 7 + 6) + 3(6 +5) = 89

Unresolved

- Can we do better that Huffman?
- How do we measure information content?

Entropy

Recall our example: A-15, B-7, C-6, D-6, E-4.

If we have a string of 38 symbols from {ABCDE} the probability that an A occurs is

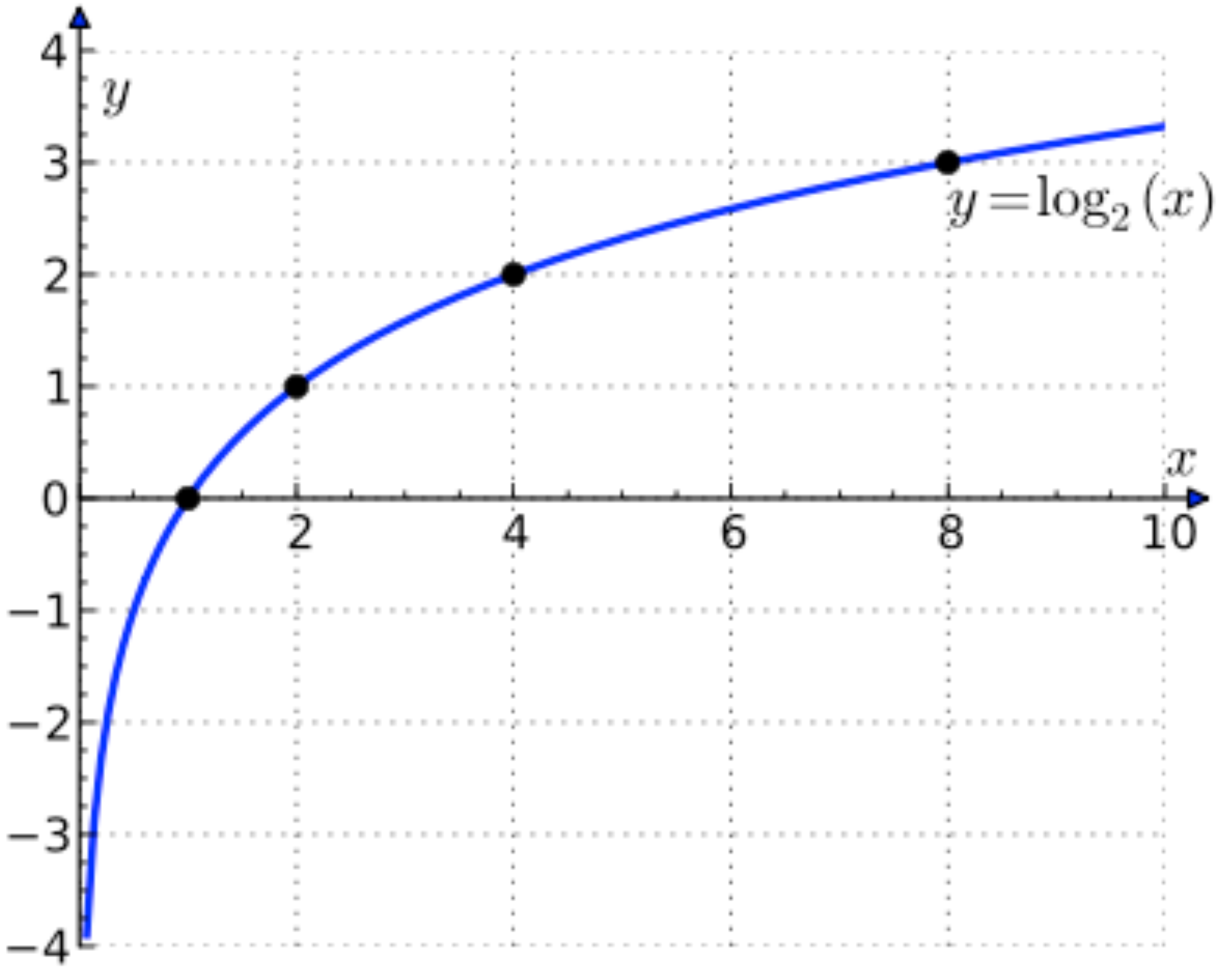
P(A) = 15/38. Similarly: P(B) = 7/38, P(C) = 6/38, P(D) = 6/38, P(E) = 4/38.

Self-information

Claude Shannon defined a quantity *self-information* associated with a symbol in a message. The self-information of a symbol X is given by the formula:

$$i(X) = \log_b \frac{1}{P(X)} = -\log_b P(X)$$

Log Function



Self-information

<u>Intuition:</u> The higher the probability of a character occurance, the lower the information content. At the extreme if P(X)=1 then there is nothing learned when receiving an X since that is the only possibility.

Entropy

Shannon also defined a quantity *entropy* associated with a message, representing the average self information per symbol in the message expressed in radix *b*.

$$H = \sum P(X_i) \log_b \frac{1}{P(X_i)}$$

$$= -\sum P(X_i) \log_b P(X_i)$$

Entropy

Entropy provides an absolute limit on the best possible lossless encoding or compression of any message, assuming that the message may be represented as a sequence of <u>independent</u> and identically distributed random variables.

Entropy

The entropy of the message in our example is computed as 2.16 binary bits (*i.e.* b = 2). The Huffman code obtained for this example uses an average of 2.23 bits per symbol. The Shannon-Fano code uses an average of 2.28 bits per symbol

Entropy

- It can be shown that a Shanon-Fano encoding of a message S produces a code with average bit length SF(S) that satisfies:

$$H(S) \leq SF(S) \leq H(S) + 1$$

Entropy

- Let ṗ denote the probability of the least likely symbol in message S
- It can be shown that a Huffman encoding of a message S produces a code with average bit length R(S) that satisfies:

$$H(S) \leq R(S) \leq H(S) + \dot{p} + 0.086 \leq H(S) + .586$$

Entropy

- Let ṗ denote the probability of the least likely symbol in message S
- Furthermore ṗ ≤ 1/m where m is the number of symbols in the alphabet:

  $H(S) \leq R(S) \leq H(S) + 1/m + 0.086$

Entropy

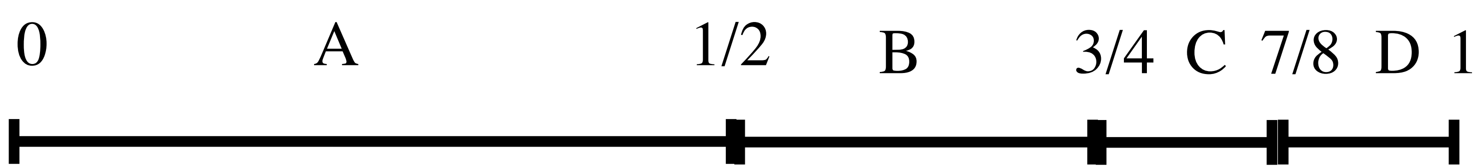
- For $m = 2^8$ (8 bits), $1/m = .0039$

$$H(S) \le R(S) \le H(S) + .0039 + 0.086$$
$$\le H(S) + .0899$$

Arithmetic Encoding

P(A) = 1/2, P(B) = 1/4 P(C) = P(D) = 1/8.

Assign symbols to intervals in 0 .. 1



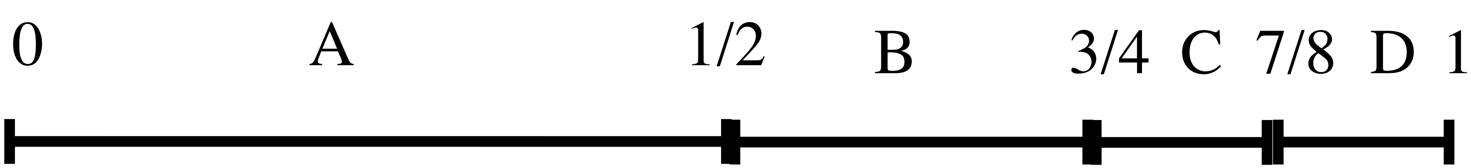
Arithmetic Encoding

P(A) = 1/2, P(B) = 1/4 P(C) = P(D) = 1/8.

Assign symbols to intervals in 0 .. 1

A_lo = 0, B_lo = 1/2, C_lo = 3/4, D_lo = 7/8

A_r = 1/2, B_r= 1/4, C_r= 1/8, D_r =1/8



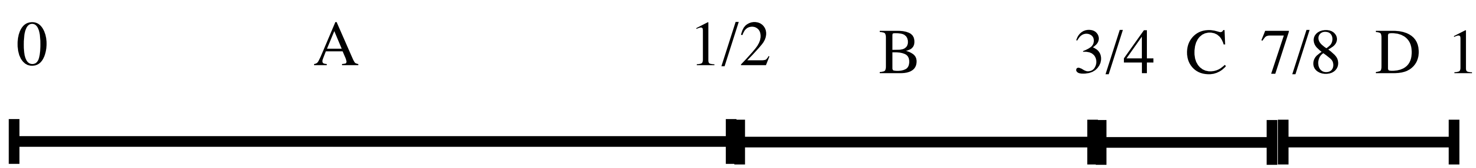
Arithmetic Encoding

```
lo = 0;
r = 1;
while symbols remain
    s = next symbol
    \\adjust range
    lo = lo  + r * s_lo
    r = r * s_r
```

After all symbols are read we are left with an interval defined between lo .. lo+r . We can represent the entire message by any value, call it v, in that interval.

Arithmetic Encoding

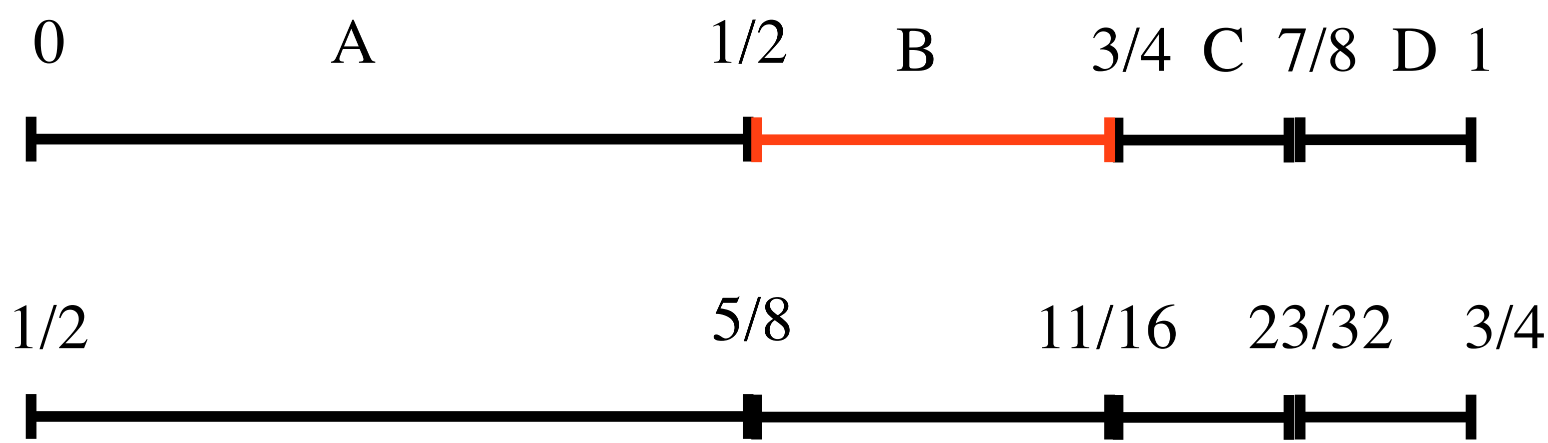
Encode message
BAABCADA



Arithmetic Encoding

Encode message

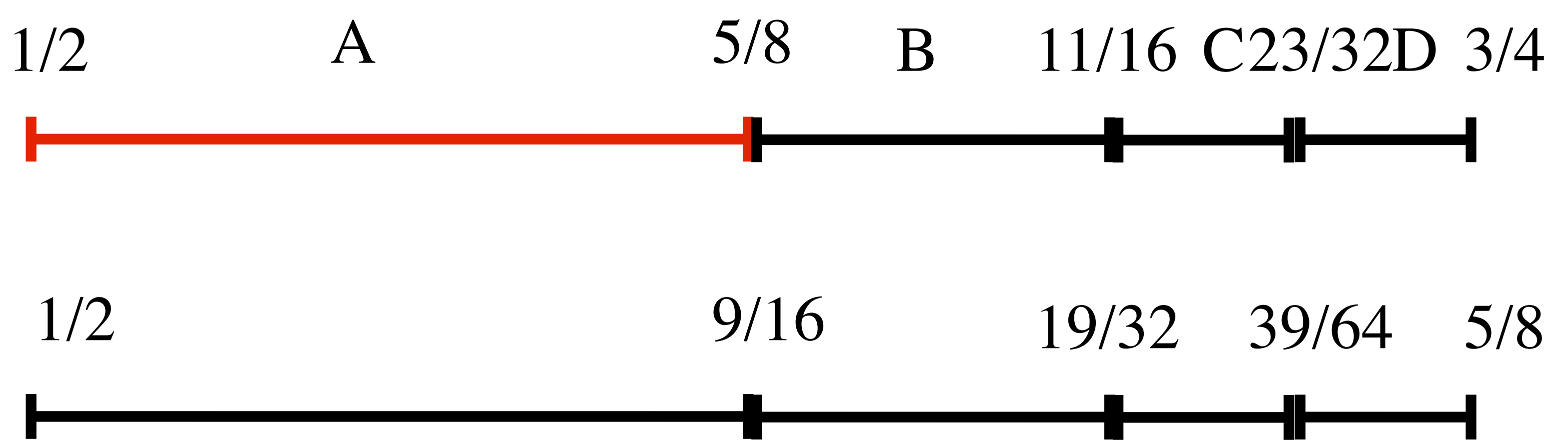
~~B~~AABCADA



Arithmetic Encoding

Encode message

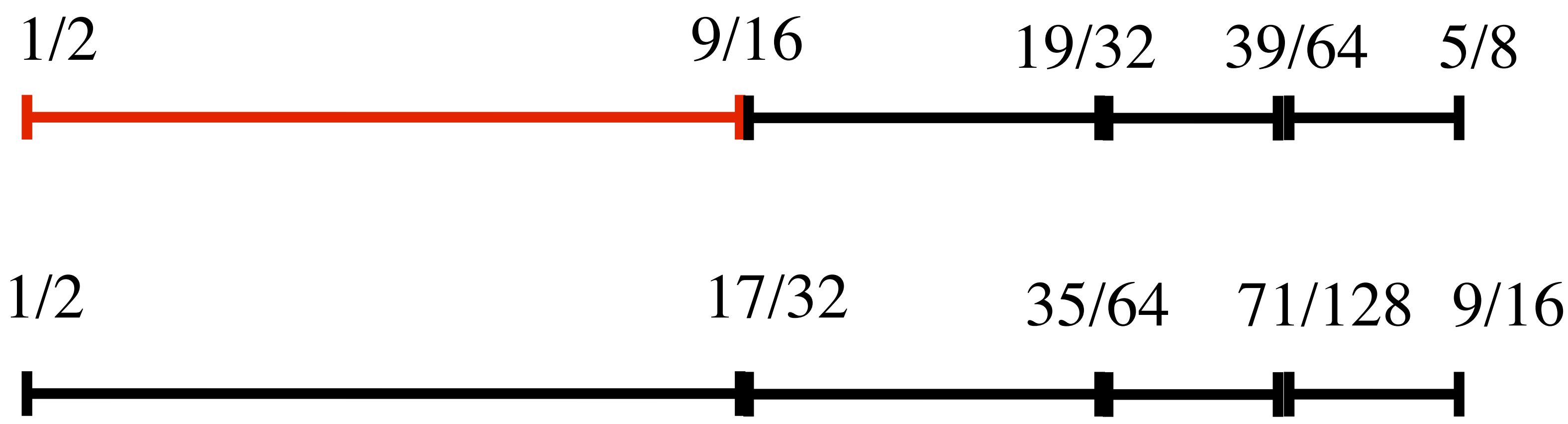
~~BA~~ABCADA



Arithmetic Encoding

Encode message

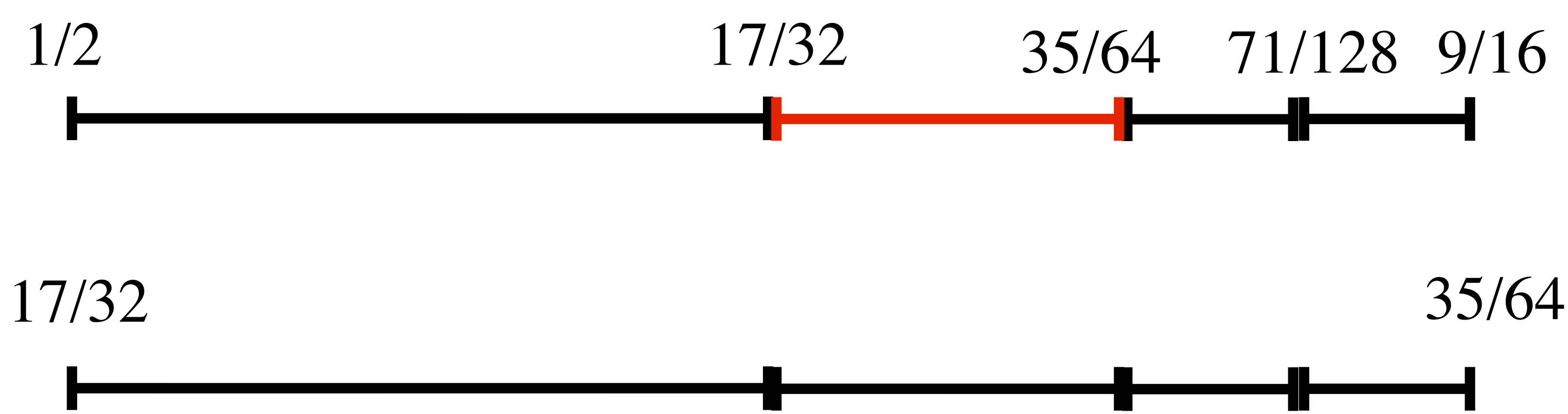
~~BAA~~BCADA



Arithmetic Encoding

Encode message

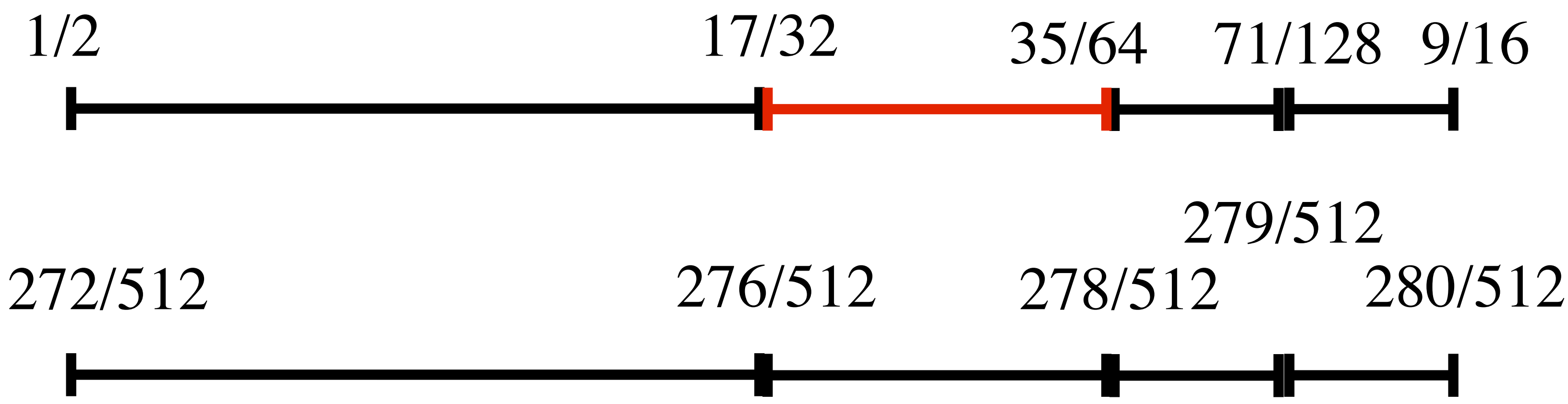
~~BAAB~~CADA



Arithmetic Encoding

Encode message

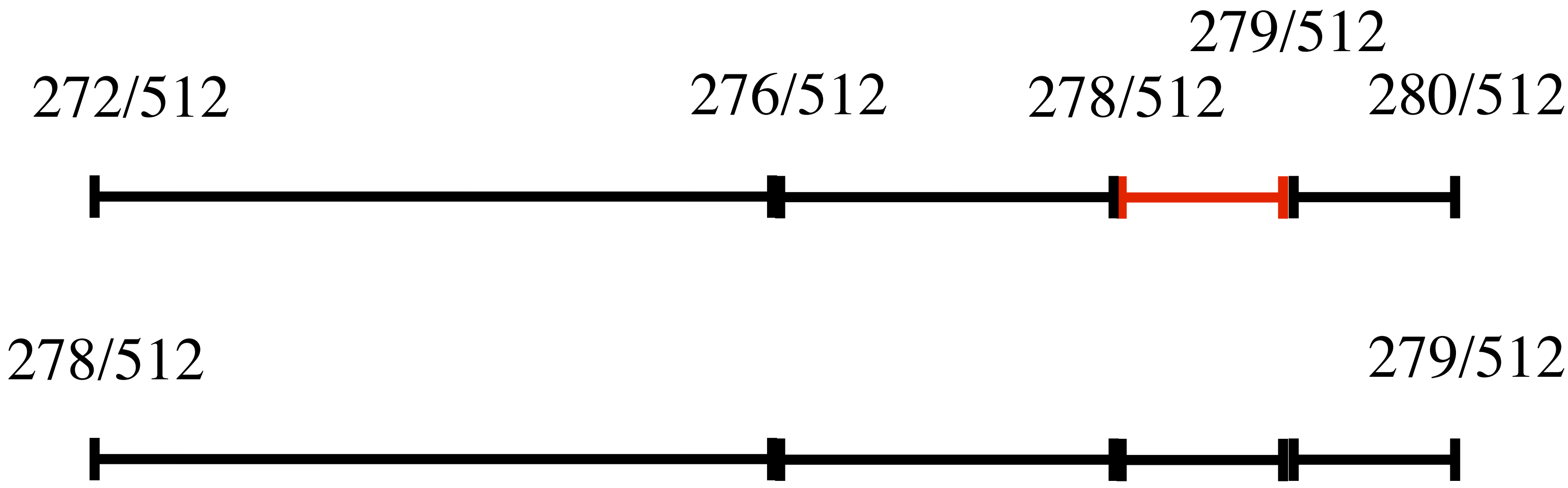
~~BAAB~~CADA



Arithmetic Encoding

Encode message

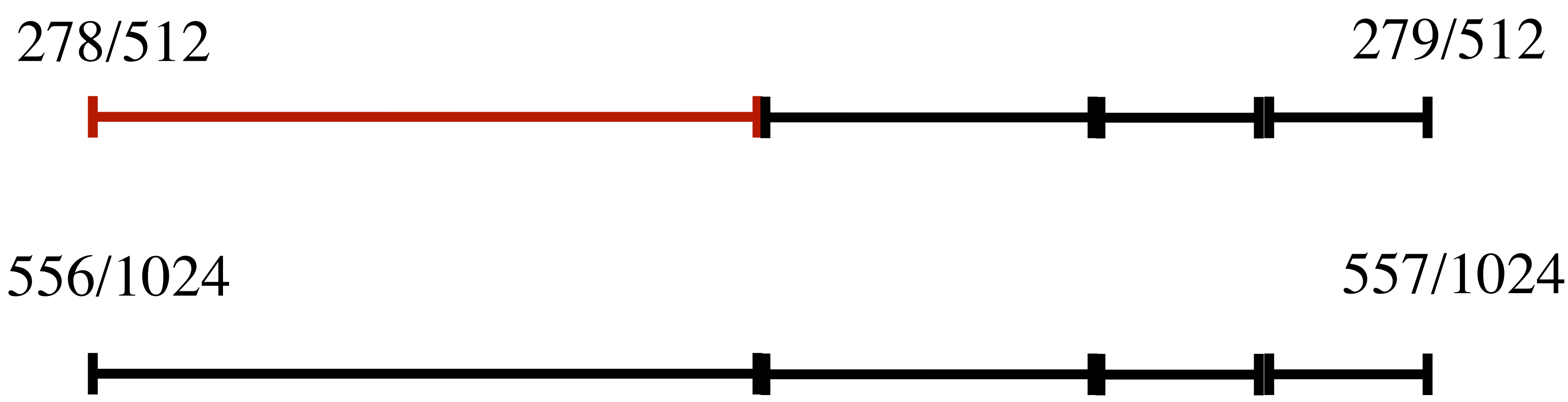
~~BAABC~~ADA



Arithmetic Encoding

Encode message

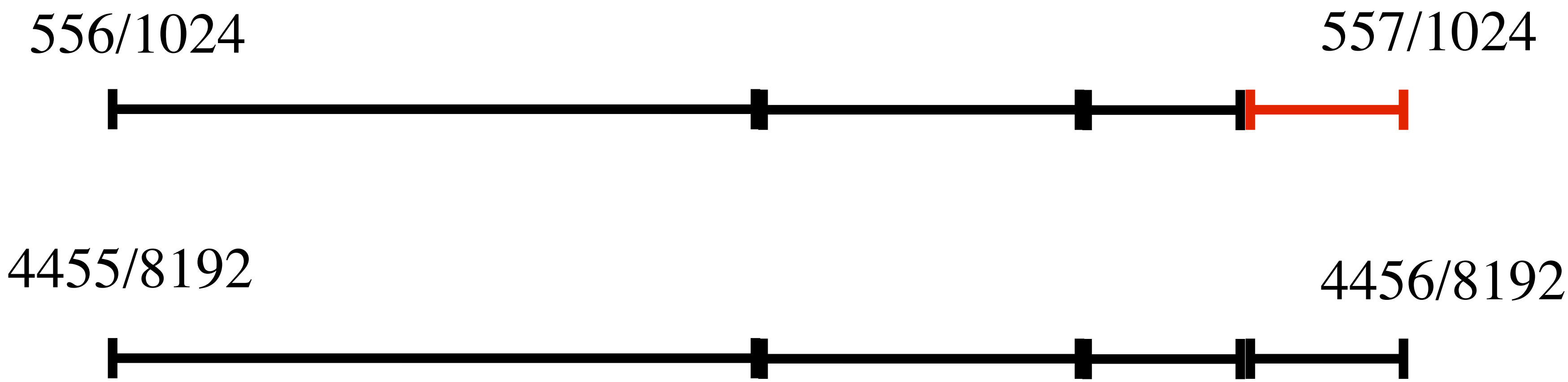
~~BAABCA~~DA



Arithmetic Encoding

Encode message

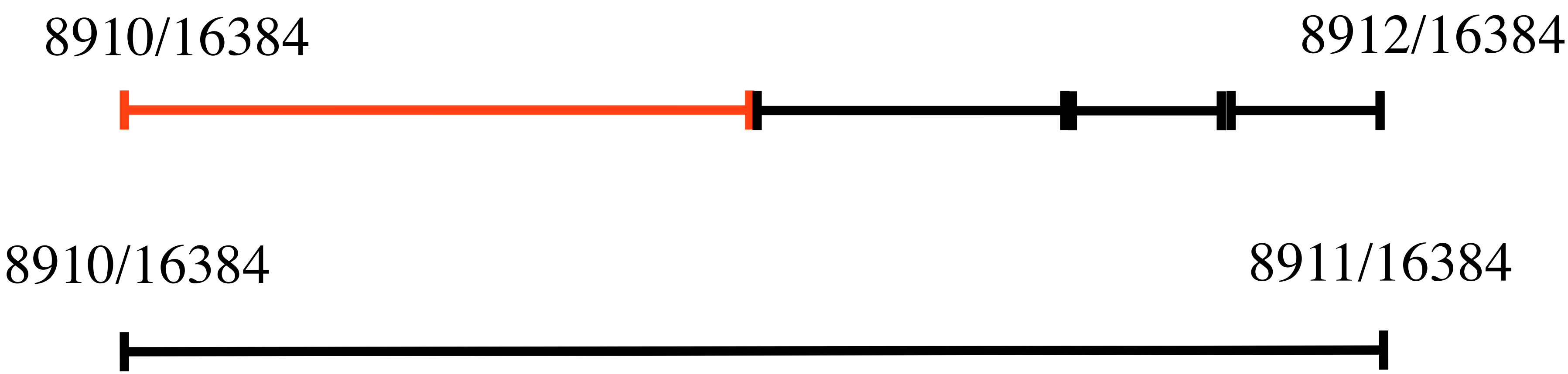
~~BAABCAD~~A



Arithmetic Encoding

Encode message

~~BAABCADA~~

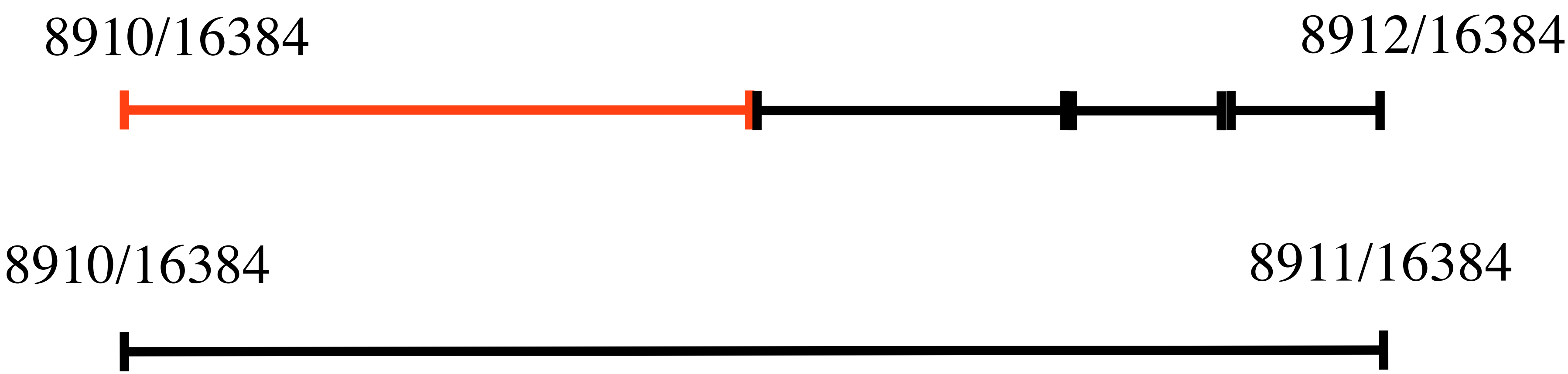


Final message can be encoded as any value in the final interval.

Arithmetic Encoding

Encode message

~~BAABCADA~~



8910/16384 in binary is 0.1000101100111

Arithmetic Encoding

Encode message

~~BAABCADA~~

8910/16384 in binary is 0.10001011100111

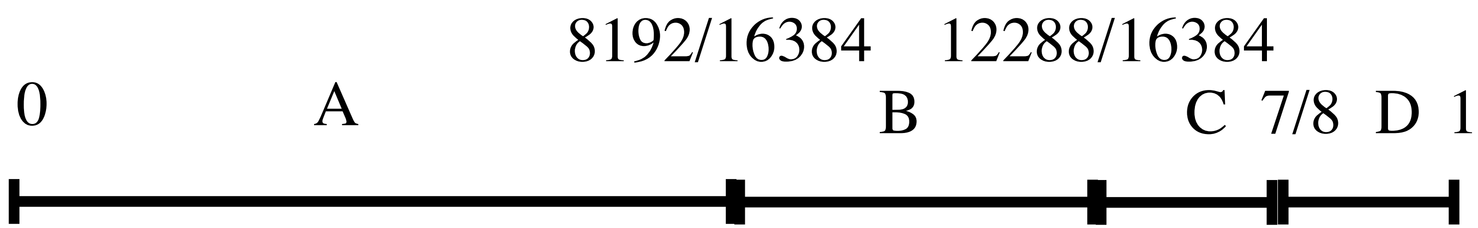
In general* the number of binary bits needed to encode a message using arithmetic encoding is $-\log_2(r)$ where r is the interval of the final range.

* there are issues of arithmetic that have to be resolved in a real world application. In practice this statement is true whenever messages are sufficiently long.

Arithmetic Decoding

Decoding an arithmetic encoded message simply reverses the encoding process.

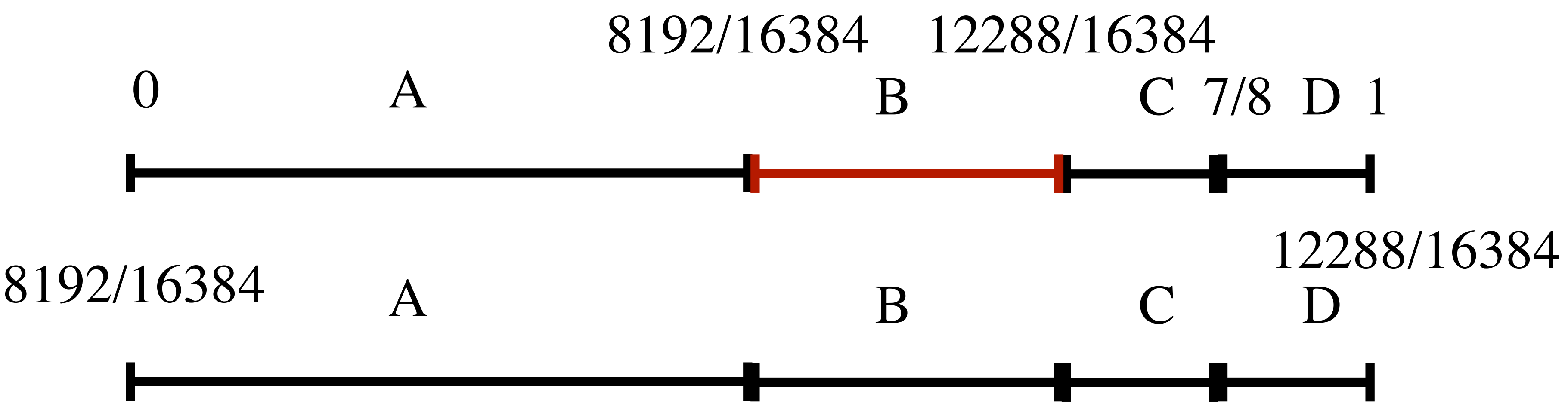
The value 8910/16384 is in B's interval so B must be the first symbol in the message



Arithmetic Decoding

Decoding an arithmetic encoded message simply reverses the encoding process.

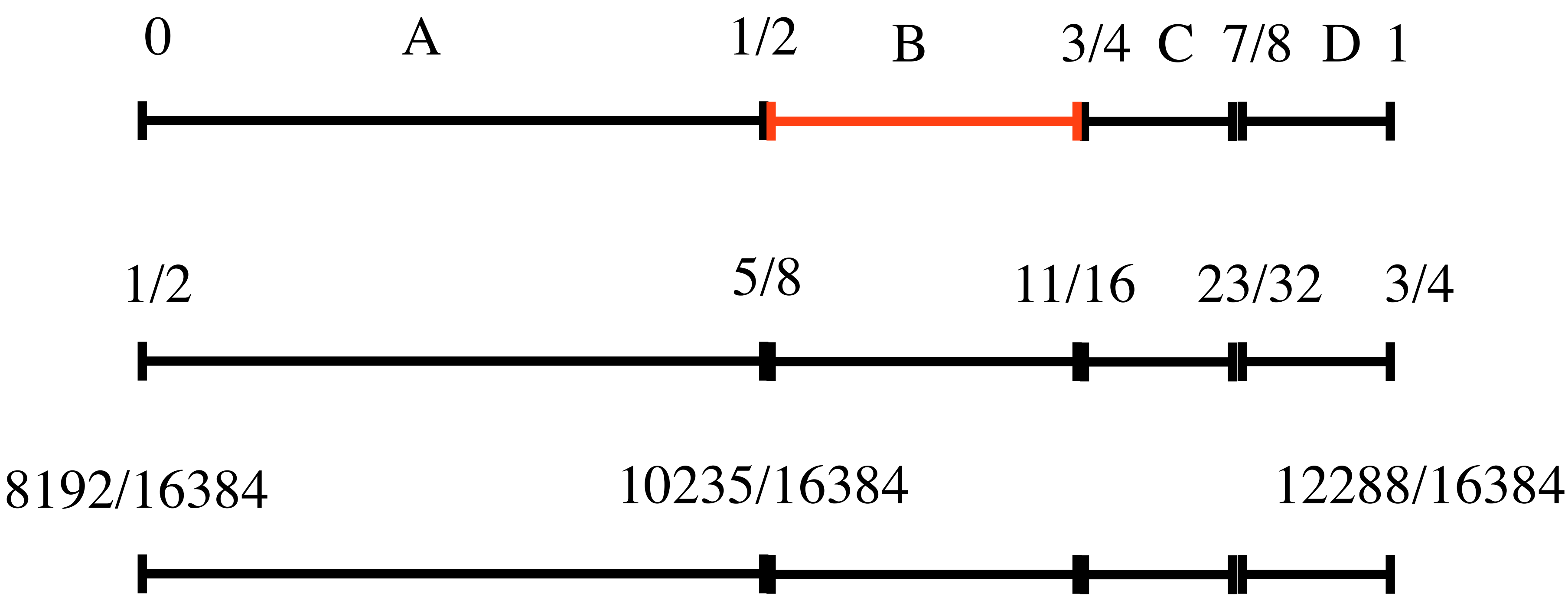
The value 8910/16384 is in B's interval so B must be the first symbol in the message



Arithmetic Decoding

Encode message

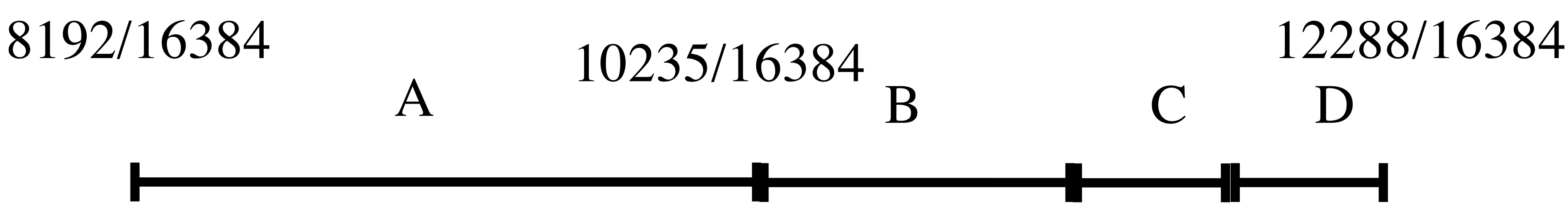
~~B~~AABCADA



Arithmetic Decoding

Decoding an arithmetic encoded message simply reverses the encoding process.

The value 8910/16384 is now in A's interval so A must be the next symbol in the message.



Arithmetic Decoding

```
lo = 0;
r = 1;
val = number received
while symbols decoded less than total number of symbols
    find symbol s such that
    s_lo ≤ val /r  < s_lo + s_r
    \\adjust range
    lo = lo  + r * s_lo
    r = r * s_r
```

Performance

Let p_k denote the probability for symbol s_k.

Let f_k denote the frequency for symbol s_k.

For a message of length n, $f_k = n \times p_k$.

Let m denote the size of the alphabet.

So the range of the final interval is:

$$\prod_{k=1}^{n} p_k = \prod_{k=1}^{m} p_k^{f_k}$$

Performance

We have:

$$-\log_2 \prod p_k^{f_k} = -n \sum p_k \log_2 p_k$$

expressing the number of binary bits needed to represent a message (of n symbols). So the average number of bits per symbol matches the entropy bound exactly.

Arithmetic Encoding

Very efficient and effective implementations of arithmetic encoding and variants of the algorithm have been developed. In general arithmetic encoding can achieve compression rates superior to Huffman encoding.

Epilogue

- Claude Shannon, Robert Fano, Peter Elias professors at MIT during ~ 1950 - 1970
- Claude Shannon invented the field of information theory and concepts such as "entropy"
- Shannon-Fano algorithm
- Shannon-Fano-Elias algorithm (Arithmetic Encoding)

Epilogue

- David Huffman student at MIT early 50s
- Huffman’s algorithm was developed as a project. (Fano was Huffman’s professor who gave students a choice between writing a final exam or doing a project.)

Epilogue

- Many modern compressors use Huffman encoding e.g. PKZIP, JPEG and mp3.
- Very efficient versions of arithmetic encoding have been developed and implemented. Although the original JPEG specification had an option to use arithmetic encoding, Huffman is used due to patent issues. (It has been shown that arithmetic encoding can reduce a Huffman JPEG up to 25%!)

Epilogue

- These methods assume that every symbol comes from a sequence of independent and identically distributed random variables
- In English that is clearly not the case, *e.g.* queen, inquiry, queue, cheque, question
- There are various compression schemes that can adapt to the highly non-independent sequence of symbols in messages. *e.g.*, LZW encoding (gif, UNIX compress).

Resources

- **Data Compression**
  http://www.ics.uci.edu/~dan/pubs/DataCompression.html
  See sections

  1.1 Definitions

  3.1 Shannon-Fano Coding

  3.2 Static Huffman Coding

  3.4 Arithmetic Coding

Resources

- People
  - http://en.wikipedia.org/wiki/Claude_Shannon
  - http://en.wikipedia.org/wiki/David_A._Huffman
  - http://en.wikipedia.org/wiki/Robert_Fano
  - http://en.wikipedia.org/wiki/Peter_Elias