CS/MA 109 – Fall 2015

Wayne Snyder
Computer Science Department
Boston University

Today: Data Compression: Run-Length Encoding and Huffman Encoding
Next: Huffman Encoding continued; Practical consequences of compression
Next Next: Error-detecting and error-correcting codes

Next week:
   Algorithms: What are they, how do they process digital information, how do we study them?

Manipulating Digital Information

Digital Information can be manipulated by many different algorithms to improve its properties; we will look at two different aspects of this in the next two lectures:

Today: Data Compression (lossy and lossless): Compressed data takes less storage space and less time to transmit; lossless data compression does this without any loss of information. We will look at two different lossless methods:
   o Run-Length Encoding
   o Huffman Encoding

Rest of the Week: Error detecting and correcting codes: Data can be encoded in such a way that errors (scratched disks, errors in transmission, computer glitches) can be detected or even corrected
Data Compression: It’s Theoretically Impossible!

First of all, let’s examine the mathematical problem, and show that it is impossible to solve in general!

Suppose we have the following letters: A, B, D, E, F, M, N & _ (blank).

These 8 letters would take 3 bits to encode, e.g,

```
A  000   F  100
B  001   M  101
D  010   N  110
E  011       _  111
```

So we could encode the message “BE_A_MAN” as

```
001011111001111101000110
B E _ A _ M A N
```

That’s 8 characters and 3*8 = 24 bits.

How many possible messages of 24 bits could we have?

Well, how many 24-bit sequences are there?
Data Compression: It’s Theoretically Impossible!

So we could encode the message “BE_A_MAN” as

001011111000111010001110

B E _ A _ M A N

That’s 8 characters and 3*8 = 24 bits.

How many possible messages of 24 bits could we have?

Well, how many 24-bit sequences are there? $2^{24} = 16,777,216$

Any such sequence is a possible message, e.g.,

000000000000000000000000 = AAAAAAAA

Data Compression: It’s Theoretically Impossible!

How many 24-bit sequences are there? $2^{24} = 16,777,216$

Any such sequence is a possible message, e.g.,

000000000000000000000000 = AAAAAAAA
000000000000000000000001 = AAAAAAAB
000000000000000000000010 = AAAAAAAC
010101010101010101010101 = BMBMBMBM
0101110110111010000101 = D_ED_MAM (random series of bits)
111111111111111111111110 = _______N (7 blanks and N)
111111111111111111111111 = ________ (8 blanks)

To encode $2^{24}$ random possibilities in bits takes a minimum of 24 bits! If we use only 23 bits, then we have only $2^{23}$ possibilities, and some would be left out! PUNCHLINE: You can't compress ALL possible messages.
Data Compression: It’s Practically Possible!

BUT, we don’t have to represent all possible messages, just the ones we are interested in. The number of possible meaningful messages with the code shown at right is far less than $2^{24}$. This is a relative statement, but true in practice!

Basic Idea of All Compression Algorithms: Interesting data has **redundancies**, and are not random; we can take advantage of the patterns to reduce the total number of bits.

Music file:

```
0 (silence)
0
0
0
0
0
0
...  
```

Code:

- A  000
- B  001
- D  010
- E  011
- F  100
- M  101
- N  110
- _  111

Data Compression: Run-Length Encoding

The simplest compression method is called RLE: “Run-Length Encoding”: The idea is to count the number of repeated symbols, and alternate counts and symbols:

Instead of

```
AAAAABBAADDDFFFFF (would require 45 bits)
```

we would write:

```
6A2B2A3D5F
```

We could alternate binary numbers for the count and binary numbers for the symbols:

```
1100000100010100000011010110100     (30 bits or 33% smaller, or an average of 2 bits/letter)
6 A 2 B 2 A 3 D 5 F
```
Data Compression: Run-Length Encoding

One problem is that the counts can not be bigger than 3 bits, or 7. We could just live with this, e.g.,

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.....
```

is encoded as

```
7A7A7A7A.....
```

or we could make a rule that whenever the count field is 000 (not useful in general), the next 6 bits are the actual count; this gives us counts as high as $2^6 - 1 = 63$.

Suppose we have 50 As and 3 Bs:  

```
000110010000011001  
50  A   3   B
```

This seems like a good idea, so let’s use it in the homework: We'll call this method RLE109.

From last time:

**Digital Information** can be manipulated in many useful ways, e.g., it can be compressed to take up less space and be transmitted more rapidly.

You can’t compress all files all the time, but you can compress most realistic kinds of data, especially text and media.

Compression takes advantage of redundancies and patterns in the data; random data can not be compressed, but most files contain data, which is inherently non-random.

**Run-Length Encoding** is a lossless compression technique which works well for repeated symbols, such as occur often in media files.
Probabilistic Compression: Huffman Encoding

The fundamental characteristic of data is that it is non-random, and compression relies on finding patterns and encoding them more compactly.

We can use the principles of Probability Theory to design other kinds of lossless compression algorithms.

Huffman Encoding is one such approach, and it relies on the probability distribution of the symbols in the data.

Let’s focus on text files, and look at three different files, with different probabilities for the occurrence of the letters.....

File 1: Random sequence of the letters A...Z:

YBJOJNQTCTAAYTCJZKUMOEUOKACDWDAXGMZJUFSDTOKIRMZFT
CZLBIXDOLGIEWABJFUMLLJHONYGDIBPERZJJUAFNTNOGCSFTWISF
KYKMNFOQLYJEVSZYECDNFLHYCEZYYCMZECVBJUVJGFGYHYODU
JDKVGMONVHSVQLJCALCJHRPCZNCNFXMCDVQMKRISBKHNYMM
DMALJDBOUIRDKJQBJJYURDZDWPQUNENOWFNWOWDKVNNSYLMMD
KOECLXEEFSXASYNBOPNPCVLVBAOJTJPKZJUKICOJEULJGFMGMXPM
VIHUYOJKWIXRFORKWNPHTIDGDTCJHCXKQPKXVOPPCCJVLMBVJAHMUY
GEZAPRLLUVRHSADHRUCSBDQHLWJMAHLJYYRHIWVCLSGHTSUIPJ
NMOUKBBBPCDVKFDAGWQDFPINGNWNCBESCQDFOFGBVIBXXHYUJRXXLZ
YWGXUIWPZNCQDQAYAEORQNZJRHZGBNJBWHLPPYFMWUQOAATZJUPZX
SIHBZJRCYVLMSXUMRAFXOZMICPEPDDYRBICCDOYHQZIWGHMRNW
WCEQDQDOUJNYZRSIJUHSCFDVRBPPLLYPNNVNDWJDHZLTGRFYGME
CQXSMMSZGCVMNVUKISXHHTHFCFQMLPVPJUEGWLUYMEHIHSWSYVEASM
NJLKMGOEKVREWLYBZDNGKYFYUMAWVPMBDQPUJOVWNTGNSDQIUYDJK
JLJQPKNGNBBPQOZHYWHBZQAHKMIAKJGKVEDRFRUFZRYEARMOYOF
OWUAIJIOCJTPEZEUUMDQOGNDQOBLWQVAOTQDMXFXVEIWKZCUPJ
SVBYHFPZGFRACQINJXKOCBPSJZFQZITYRNCLELVGSQHXDUUIM
MRLVXISAFUPEQDDWVSIVOMABMYYRSELMNBMSJUCYGCWRELK
HSSDQZNFRRAJFQOOUJKFLJULYLXDFPAPBMCKREXAZDPKAMUDJKRGS
VMSUXMYCKXPNHYSMPSQRMDZNDCGQGQETAKTBABWOWQHYWHRNTLFJ
JQOBTDEXNLXUHYWQDKFAKXZOMAMJEDTUGUPCTAHRLCIQROGSDAHVM
TFVADARCONYSOTTCQYJIPBDJWUNAYAGACULWLEZQOCTAKIQKHYSM
NTGKYMDIERPOFDRRSLFSEWHRUXRTXNMRLXLVUTUSQSMPEZNYJUM
KNSBZUDUHDXZKOEVALBVJLZLJXIEYNQAMFIUVOZQYJDDZVZIWN
HMWWLTLKDJZPPXTGFQYEXLEHISEAJZDPNCHDSEFMWJDKZIBWYZ..

What is the probability that any particular letter is A?

Since all letters occur with equal probability, and there are 26 letters, the probability of A is 1/26 = 0.3846...

Each of the letters has the same probability.
Probabilistic Compression: Huffman Encoding

File 2: Non-random sequence of letters

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...}

What is the probability that any particular letter is A?

Um.... 1.0

Probabilistic Compression: Huffman Encoding

File 3: Another non-random sequence of letters

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
it was the epoch of belief,
it was the epoch of incredulity,
it was the season of Light,
it was the season of Darkness,
it was the spring of hope,
it was the winter of despair,
we had everything before us,
we had nothing before us,
we were all going direct to Heaven,
we were all going direct the other way--
in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever.

What is the probability that any particular letter is A?

We could count the letters in "A Tale of Two Cities" but for an good estimate, we could use statistics about the distribution of letters in the English language....

http://en.wikipedia.org/wiki/Letter_frequency

The probability in English of A is 8.167%.
Probabilistic Compression: Huffman Encoding

Why does this matter?

If all symbols occur with equal frequency, then we have little choice but to simply figure out how many possibilities there are (= the number of letters), and invent a binary code with enough bits to cover all possibilities.

26 English letters:

\[2^4 = 16\] Not enough!

\[2^5 = 32\] Enough and room to spare.

So we need a minimum of 5 bits per letter to store random sequences.

When you have random data, this is the best you can do.....

But, what if the probabilities are not equal?

How many bits does it take to indicate an A if the text file contains only As?

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA....
Probabilistic Compression: Huffman Encoding

But, what if the probabilities are not equal?

How many bits does it take to indicate an A if the text file contains only As?

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA....

ONLY ONE!

111111111111111111111111111111111
111111111111111111111111111111111
111111111111111111111111111111111
11111111111111111111111111111111111

This is the absolute minimum number of bits for a code! Ok, this is silly, but let's consider a more realistic example.....

Suppose I have three letters: A, B, C

Normally you would need how many bits to encode messages with these three letters?
Probabilistic Compression: Huffman Encoding

Suppose I have three letters: A, B, C

Normally you would need how many bits to encode messages with these three letters?

Two bits: \( 2^2 = 4 \)

BUT: Suppose we know that the probabilities of the letters are not equal, e.g.,

A occurs with probability .8
B occurs with probability .1
C occurs with probability .1

AABACAAAAAABAAACAAAAABACAAAAAAA 00 00 01
ABAAACAAAAABAAAAAACAABAAACAAAA 00.....
AACAAABAAABAAACAAAA......

Then the following Huffman Code is more efficient:

A  0
B  10
C  11

Let's try the message AABACAAAAA with these two codes

Basic Code: 00 00 01 00 10 00 00 00 00 00 (20 bits)
Huffman Code: 0 0 10 0 11 0 0 0 0 0 (12 bits)

That is 60% compression..... Instead of each letter taking 2 bits, each on average uses

\[ 0.8 \times 1 + 0.1 \times 2 + 0.1 \times 2 = 1.2 \text{ bits} \]
This is the basic idea of Huffman Encoding:

Use variable length codes, and encode more frequent letters with shorter codes.

This is actually an old idea, and was used in designing Morse Code for telegraph operators to most quickly transmit messages.

It also used, more or less, to figure out the points for letters in Scrabble: