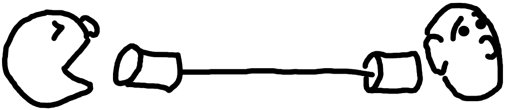
CPSC 420 Lecture 15: Today's announcements:

- Examlet 2 on Feb 17 in class. Closed book & no notes
- Reading: Shor's notes on Lempel-Ziv compression `https://math.mit.edu/~shor/PAM/lempel_ziv_notes.pdf`

Today's Plan

- Compression
  - Huffman Coding
  - Lempel-Ziv Compression

Information Theory [Shannon 1948]



The **information** (measured in bits) contained in a message i that has probability p_i of being sent is

average information

$$\log_2 \frac{1}{p_i}$$

info measure

Entropy is the average information content of a message X:

$$H(X) = \sum_i p_i \log_2 \frac{1}{p_i} = -\sum_i p_i \log_2 p_i$$

Source Coding Theorem m i.i.d. random variables each with entropy $H(X)$ can be compressed into more than $mH(X)$ bits with negligible risk of information loss but using less than $mH(X)$ bits results almost certainly in information loss. [Wikipedia-ish]

Huffman Coding [Huffman 1952]

Given a set of characters $a_1, a_2, \ldots, a_\alpha$ and a probability p_i for each a_i ($\sum_i p_i = 1$)
Construct an encoding (sequence of bits) c_i for each character a_i so that the expected length of an encoded message is minimized.

Idea Higher probability characters get shorter codes.

A	B	C	D	E	F	G	H
.2	.05	.15	.1	.3	.03	.1	.07

Huffman Coding [Huffman 1952]

Given a set of characters $a_1, a_2, \ldots, a_\alpha$ and a probability p_i for each a_i ($\sum_i p_i = 1$)
Construct an encoding (sequence of bits) c_i for each character a_i so that the expected length of an encoded message is minimized.

Idea Higher probability characters get shorter codes.

A	B	C	D	E	F	G	H
.2	.05	.15	.1	.3	.03	.1	.07

minPQ:

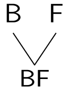
E	A	C	G	D	H	B	F
.3	.2	.15	.1	.1	.07	.05	.03

Huffman Coding [Huffman 1952]

Given a set of characters $a_1, a_2, \ldots, a_\alpha$ and a probability p_i for each a_i ($\sum_i p_i = 1$)
Construct an encoding (sequence of bits) c_i for each character a_i so that the expected length of an encoded message is minimized.

Idea Higher probability characters get shorter codes.

A	B	C	D	E	F	G	H
.2	.05	.15	.1	.3	.03	.1	.07



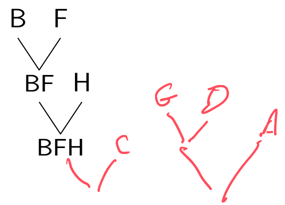
minPQ:	E	A	C	G	D	BF	H
	.3	.2	.15	.1	.1	.08	.07

Huffman Coding [Huffman 1952]

Given a set of characters $a_1, a_2, \ldots, a_\alpha$ and a probability p_i for each a_i ($\sum_i p_i = 1$)
Construct an encoding (sequence of bits) c_i for each character a_i so that the expected length of an encoded message is minimized.

Idea Higher probability characters get shorter codes.

A	B	C	D	E	F	G	H
.2	.05	.15	.1	.3	.03	.1	.07



minPQ:

E	A	~~BFH~~	~~C~~	~~G~~	~~D~~
.3	.2	~~.15~~	~~.15~~	~~.1~~	~~.1~~

B..C
.3

.2
GD

.4

Huffman Coding [Huffman 1952]

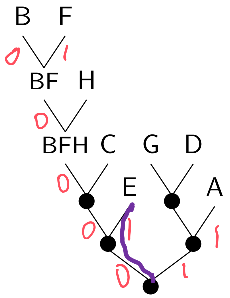
Given a set of characters $a_1, a_2, \ldots, a_\alpha$ and a probability p_i for each a_i ($\sum_i p_i = 1$)
Construct an encoding (sequence of bits) c_i for each character a_i so that the expected length of an encoded message is minimized.

Idea Higher probability characters get shorter codes.

A	B	C	D	E	F	G	H
.2	.05	.15	.1	.3	.03	.1	.07

$\underbrace{01}_{E}\,\underbrace{01}_{E}\,\underbrace{100}_{G}$

A: 11
B: 00000
C: 001
D: 101
E: 01
F: 00001
G: 100
H: 0001



Decoding

	A	B	C	D
not unique	0	010	01	10
not prefix-free	00	10	11	110
prefix-free	0	10	110	111

bad

Decode 10101100010

D

B B D A B

Decode 010

AD

B

B B A

Decoding

	A	B	C	D
not unique	0	010	01	10
not prefix-free	00	10	11	110
prefix-free	0	10	110	111

Decode 10101100010

No (unique) code can compress all inputs of length n.

There are 2^n different inputs.
There are $\sum_{i=1}^{n-1} 2^i = 2^n - 2$ codes of length $< n$.
Pigeonhole.

Compression [Lempel & Ziv 1978]

$$AAABABBBBAABBBB$$

- Parse input into distinct phrases reading from left to right.
  Each phrase is the shortest string not already a phrase.
  The 0th phrase is $\emptyset$.
- Output $i\,c$ for each phrase w, where c is the last character of w and i is the index of phrase u where $w = u \circ c$

Let $c(n)$ be the number of phrases created from input of length n.
Let α be the size of the alphabet of characters in input.
Length of output is $c(n)(\log_2 c(n) + \log_2 \alpha)$ bits.

Compression [Lempel & Ziv 1978]

A 0
B 1

$$|A|AA|B|AB|BB|BA|ABB|BB$$

- Parse input into distinct phrases reading from left to right.
  Each phrase is the shortest string not already a phrase.
  The 0th phrase is $\emptyset$.
- Output $i\ c$ for each phrase w, where c is the last character of w and i is the index of phrase u where $w = u \circ c$

	1	2	3	4	5	6	7	8
	A	AA	B	AB	BB	BA	ABB	BB
code	$\cancel{0}A$	$1A$	$0B$	$1B$	$3B$	$3A$	$4B$	~~3B~~ 5
code in bits	00	10	$\boxed{001}$	011	0111	0110	1001	111

Let $c(n)$ be the number of phrases created from input of length n.
Let α be the size of the alphabet of characters in input.
Length of output is $c(n)(\log_2 c(n) + \log_2 \alpha)$ bits.

Compression [Lempel & Ziv 1978]

How do we show LZ78 is a good compressor?

Empirical Try it on lots of inputs.

Worst case? Average case? Something else?

Worst case

$c(n) = \#\text{phrases}$ $\qquad \alpha = |\text{alphabet}|$ (assume $\alpha = 2$)

Maximize $c(n)$ (make many small phrases)

$$A|B|AA|AB|BA|BB|\ldots$$

The smallest input with **all** phrases of lengths $1, 2, \ldots, k$ has length

$$n_k = \sum_{j=1}^{k} j2^j = (k-1)2^{k+1} + 2$$

For such an input, $c(n_k) = \sum_{i=1}^{k} 2^i = 2^{k+1} - 2$, so $c(n_k) \leq \frac{n_k}{k-1}$.

Compression [Lempel & Ziv 1978]

Worst case continued

In fact, for all n between n_k and n_{k+1},

$$c(n) \overset{❶}{\leq} \frac{n_k}{k-1} + \frac{n-n_k}{k+1} \leq \frac{n}{k-1} \overset{❷}{\leq} \frac{n}{\log_2 c(n) - 3}$$

since ❶ to maximize $c(n)$, the first n_k input bits make $\leq c(n_k)$ phrases and the rest make phrases of length $k+1$, and
❷ $c(n) \leq c(n_{k+1}) = 2^{k+2} - 2$.

As we saw, LZ78 compresses inputs of length n to about

$$c(n) \log_2 c(n) + c(n) \leq n + 4c(n) = n + O\left(\frac{n}{\log_2 n}\right) \text{ bits.}$$

Is this good?

Compression [Lempel & Ziv 1978]

Average case

$$\text{Alphabet} = \{a_1, a_2, \dots, a_\alpha\}$$

Create input $x = a_{x(1)} a_{x(2)} \dots a_{x(n)}$ by choosing n characters at random, where a_i is chosen with probability p_i.

Let $Q(x) = \prod_{i=1}^{n} p_{x(i)}$ be the probability of input x.

If LZ78 breaks x into distinct phrases $x = y_1 y_2 \dots y_{c(n)}$ then

$$Q(x) = \prod_{j=1}^{c(n)} Q(y_j) = \prod_{\ell} \prod_{|y_i|=\ell} Q(y_i)$$

Let c_ℓ be the number of phrases of length ℓ.
Since the y_i's with length ℓ are distinct $\sum_{|y_i|=\ell} Q(y_i) \le 1$ and

$$\prod_{|y_i|=\ell} Q(y_i) \le \left(\frac{1}{c_\ell}\right)^{c_\ell} \quad \begin{array}{l}\text{take log}\\ \text{sum over } \ell\end{array} \quad -\log_2 Q(x) \ge \sum_{\ell} c_\ell \log_2 c_\ell$$

Compression [Lempel & Ziv 1978]

Average case continued

$$\sum_{\ell} c_\ell \log_2 c_\ell \leq -\log_2 Q(x) \approx np_i \log_2(1/p_i) = nH(X)$$

where X is a random character.

Recall, LZ78 compresses to approx. $c(n) \log_2 c(n)$ bits. If this is approx. $\sum_\ell c_\ell \log_2 c_\ell$ then LZ78 compresses (nearly) optimally [Source Coding Theorem].
In fact,

$$nH(X) \geq -\log_2 Q(x) \geq c(n) \log_2 c(n) - O(\log_2(\frac{n}{c(n)}))$$