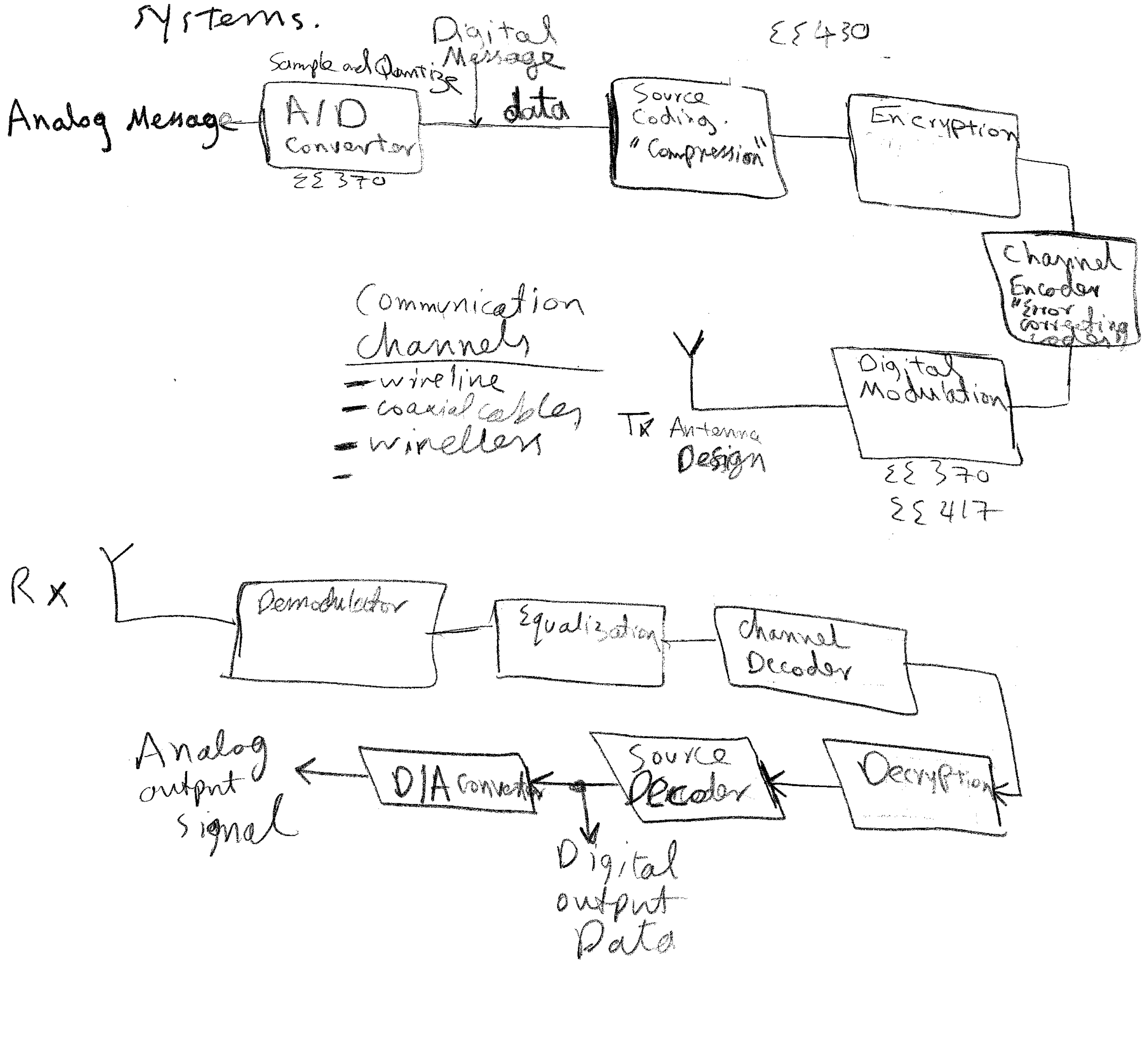
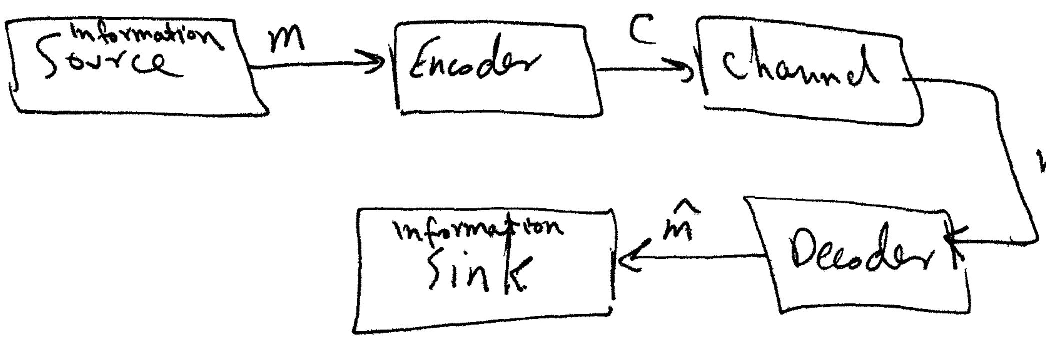
Introduction Class

— Let take a quick Review A Digital Communication systems.



Chapter 1

Discrete Sources and Entropy



1.2 Source Alphabets and Entropy.

Def. — An information source outputs a set of symbols.

— A finite discrete source ~~is one~~ output ~~is one~~ belongs to a finite discrete set of symbols. The symbol set is called the source alphabet.

— $A = \{a_0, a_1, \ldots, a_{M-1}\}$ is a source alphabet (Symbol set) of cardinality $M = |A|$.
(size) or (number of elements)

— The source outputs symbols in a time sequence represented by the notation $\bar{a} = (s_0 s_1 \ldots s_t \ldots)$
where $s_t \in A$

— The probability that the source emits symbol a_m is written as $P_m = Pr(a_m)$.

— The set of probabilities for the source alphabet is

$$P_A = \{P_0, P_1, \ldots, P_{M-1}\}$$

— Information Theory makes an important distinction between data and information. Not all data carries information. Information is the part that adds ~~to~~ knowledge.

~~Given a set of data~~

Example — Assume that we have a source that emits only one symbol. ~~and with~~ A set of data from this source will have zero information content since the receiver knows that the source only send one unique symbol all the time. Thus, we have a set of data but no information content.

— From the above example, we can see that the information content of the data increases with uncertainty of transmitted symbols. —

— Entropy

Information theory provides a measure of the average amount of information conveyed per source symbol. This measure is called the entropy of the source and is defined as:

$$H(A) = \sum_{m=0}^{M-1} P_m \log_2\left(\frac{1}{P_m}\right)$$

; where P_m is the probability that symbol m was transmitted. Notice that the Logarithm is to the base 2.

the units of H(A) is bits. It tells us How much information carried by each symbol.

To help in calculation, Recall that $\log_2(x) = \frac{\ln(x)}{\ln(2)}$

Also, $\lim_{x \to 0} x \log(x) = 0$

Notice that the information per symbol is $I(A) = \log_2\left(\frac{1}{P_m}\right)$ and Entropy is the average information per

Example 1.2.1

What is the ~~in~~average amount of information conveyed per source symbol of a 4-ary source having probabilities

$$P_A = \{0.5, 0.3, 0.15, 0.05\} \quad ?$$

Solution:

$$H(A) = 0.5\log_2(2) + 0.3\log_2(10/3) + 0.15\log_2\left(\frac{100}{15}\right) + 0.05\log_2(20)$$
$$= 1.6477 \text{ bits.}$$

So, each symbol carries 1.6477 bits of information. while it carries 2 bits of data.

Therefore, it is possible to use some data compression techniques to compress the above source and use fewer bits ~~on~~ on average.

Def: Information efficiency $= \dfrac{\text{The entropy of the source}}{\text{Average number of bits used to represent the source data}}$.

For the above example,

the information efficiency is $= \dfrac{1.6477}{2} = 82.387\%$

this means that approximately 17.6% of the bits are redundant and carry no information.

Example; Find the entropy ~~[illegible]~~ of the 4-ary signal if all symbols are equally Probable. i.e $P_m = \frac{1}{4}$

Lemma: Consider an M-ary source A, the maximum entropy of this source is $\log_2 M$ and it happens when all symbols are equally probable $\Rightarrow P_m = \frac{1}{M}$ for all $m \in A$

See example 1.2.2 for proof.

This result makes sense intuitively. If every symbol in A is equally probable, an observer would have no idea what symbol will be emitted next by the source. Thus, each symbol carries the maximum surprise value and the average amount of information is maximized.

1.2.2 Joint and Conditional Entropy

Most communication systems are designed to be used by a large number of users. The designers of such a system are concerned ~~by~~ with maximizing the total information carrying capacity of the system.

~~The~~ Information theory gives ~~us~~ also tools to measure the joint and conditional entropy ~~of~~ ~~more~~ for more than one source.

Consider a situation where we have two information sources, A and B. Let $|A| = M_A$ and $|B| = M_B$.

The joint probability that A sends symbol a_i and B sends symbol b_j is $P_{ij} = Pr(a_i, b_j)$

If the two symbol are statistically independent, then

Statistically independent case

If sources A and B are statistically independent, the total entropy of this system will be

$$\underset{\text{joint entropy}}{\underset{\uparrow}{H(A,B)}} = \underset{\text{Entropy for A}}{\underset{\uparrow}{H(A)}} + \underset{\text{Entropy for B}}{\underset{\uparrow}{H(B)}}$$

If the two sources are <u>dependent</u>, what do you expect will happen to the joint entropy? will it increase or decrease? The answer is that it will decrease. There is a fundamental theorem in information theory that says "Side information never increases entropy". The joint entropy

$$H(A,B) \leqslant H(A) + H(B)$$

with equality if and only if A and B are statistically independent.

Statiscally dependent Case

Denote the combined emission of a_i and b_j as a compound symbol $c_{i,j} \equiv \langle a_i, b_j \rangle$. let the probability of emiting $c_{ij} = P_{ij}$.

The entropy of C is

joint entropy $\longrightarrow$
$$H(C) = \sum_{c_{ij} \in C} P_{ij} \log_2\left(\frac{1}{P_{ij}}\right)$$
$$= \sum_{i=0}^{M_A - 1} \sum_{j=0}^{M_B - 1} P_{ij} \log_2\left(\frac{1}{P_{ij}}\right)$$

The joint probability P_{ij} may be written in terms of a conditional probability $[P_{j|i} = Pr(b_j | a_i)]$

as $$P_{ij} \equiv P_{j|i} \cdot P_i$$

$$\therefore H(C) = \sum_i \sum_j P_{ij} \log_2\left(\frac{1}{P_{j|i} P_i}\right)$$
$$= \sum_i \sum_j P_{ij} \log_2\left(\frac{1}{P_i}\right) + \sum_i \sum_j P_{ij} \log_2\left(\frac{1}{P_{j|i}}\right)$$

Notice that, $$\sum_i \sum_j P_{ij} \log_2\left(\frac{1}{P_i}\right) = \sum_i \sum_j P_{j|i} P_i \log_2\left(\frac{1}{P_i}\right)$$
$$= \sum_i P_i \log_2\left(\frac{1}{P_i}\right) \underbrace{\sum_j P_{j|i}}_{=1}$$

$$\therefore H(C) = \sum_i P_i \log_2\left(\frac{1}{P_i}\right) + \sum_i \sum_j P_{ij} \log_2\left(\frac{1}{P_{j|i}}\right)$$
$$H(C) = H(A,B) = H(A) + H(B|A)$$

Similarly, $$H(A,B) = H(B) + H(A|B)$$

$H(B/A)$ and $H(A/B)$ are conditional Entropy.

$$H(B/A) = \sum_i \sum_j P_{ij} \log\left(\frac{1}{P_{j|i}}\right)$$

$$H(A/B) = \sum_i \sum_j P_{ij} \log\left(\frac{1}{P_{i|j}}\right)$$

Also, $H(B/A) < H(B)$

and $H(A/B) < H(A)$

The end result is that the joint entropy ~~of~~ two sources A and B is

$$H(A,B) = H(A) + H(B/A)$$
$$= H(B) + H(A/B)$$

and

$$H(A,B) \leq H(A) + H(B)$$

with equality if and only if A and B are independent.

continue 1.2.2

— Joint and conditional Entropy.

$$H(C) = \underset{\text{joint Entropy.}}{H(A,B)} = H(A) + \underset{\text{conditional Entropy.}}{H(B|A)}$$
$$= H(B) + H(A/B)$$
$$H(A,B) \leq H(A) + H(B)$$

and equality happens when A and B are independent.

Example 1.2.3

Error detection using parity bits

parity bits are added to the transmitted bits to detect errors. Let A be an information source with alphabet $A = \{0, 1, 2, 3\}$. Let each symbol a be equally probable and Let $B = \{0, 1\}$ be a parity generator with

$$b_j = \begin{cases} 0 & \text{if } a = 0 \text{ or } a = 3 \\ 1 & \text{if } a = 1 \text{ or } a = 2 \end{cases}$$

What are $H(A)$, $H(B)$ and $H(A, B)$?

$$H(A) = 4 \cdot \tfrac{1}{4} \log_2(4) = 2 \text{ bit}$$

Likewise $H(B) = 2 \cdot \tfrac{1}{2} \log_2(2) = 1$ bit

} This is because each symbol is equally probable.

The conditional probabilities $Pr(b/a)$ are

$Pr(0|0) = 1$, $Pr(1/0) = 0$ $\rightarrow \sum_j Pr(b_j/0) = 1$

$Pr(0|1) = 0$, $Pr(1/1) = 1$

$Pr(0/2) = 0$, $Pr(1/2) = 1$

$Pr(0/3) = 1$, $Pr(1/3) = 0$

Therefore,
$$H(B/A) = \sum_{i=0}^{3} P_i \sum_{j=0}^{1} P_{j|i} \log_2\left(\frac{1}{P_{j|i}}\right)$$
$$= 4 \cdot \tfrac{1}{4} (1 \log_2 1 - 0 \log_2 0)$$

This says that B is completely determined by A.

$$H(A,B) = H(A) + H(B|A) = 2+0 = 2$$

$\therefore$ Source B contributes no information to the compound signal.

1.7.3 Entropy of Symbol Blocks and the Chain Rule.

What is the information content of a block of symbols?

A block of symbols is a sequence of n symbols $(s_0, s_1, \dots, s_{n-1})$ produced by source A. $\{s_t \in A\}$.

The entropy of this block is denoted as

Joint Entropy $\rightarrow H(A_0, A_1, \dots, A_{n-1})$.

We can use the joint entropy result to express the entropy as

$$H(A_0, A_1, \dots, A_{n-1}) = H(A_0) + H(A_1, A_2, \dots, A_{n-1}|A_0)$$

This term can also be written as $\longrightarrow$ $H(A_1, A_2, \dots, A_{n-1}|A_0) = H(A_1|A_0) + H(A_2, \dots, A_{n-1}|A_0, A_1)$

Repeating this argument inductively, we get

$$H(A_0, A_1, \dots, A_{n-1}) = H(A_0) + H(A_1|A_0) + H(A_2|A_0, A_1) + \dots H(A_{n-1}|A_0, \dots, A_{n-2})$$

This is called the chain rule for entropy.

Since $H(B|A) \leq H(B)$

The upper bound on the Entropy of symbol blocks is

$$H(A_0, \cdots, A_{n-1}) \leq \sum_{i=0}^{n-1} H(A_i)$$

with equality if and only if all ~~of~~ the symbols in the equence are <u>statistically independent</u>.

<u>Memoryless Source</u>

A source that emits statistically independent symbols is called a memoryless source. That is because it doesn't remember from one time to the next what symbols it has previously emitted. So, there is no correlation between Symbols.

If the information source is the same ~~for~~ all the time and the probabilities of the source don't change over time, then we have the joint entropy to be.

$$H(A_0, \cdots, A_{n-1}) \leq n \cdot H(A)$$

~~See~~ <u>Examples 1.2.4</u>

Suppose a memoryless source with $A = \{0, 1\}$ having equal symbol probabilities emits a sequence of six symbols. Following the sixth symbol, a seven symbol is added which is ~~the~~ sum modulo 2 of the six previous symbols. What is the entropy of this sequence?

Solution

Let $b = \sum_{t=0}^{5} \oplus s_t$, where $\sum \oplus$ is just the summation Modulo 2 and $s_t \in A$.

$$H(A_0, A_1, \cdots, A_5, b) = H(A_0) + H(A_1|A_0) + \cdots + H(b|A_0, \cdots, A_5)$$

since the First six symbols are statistically independent, we ~~[illegible]~~ get

$$H(A_0, A_1, \cdots, A_5, b) = 6\, H(A) + H(b|A_0, \cdots, A_5)$$

since b is completly determined by b, it has no new information and $H(b/A_0, \cdots, A_5) = 0$

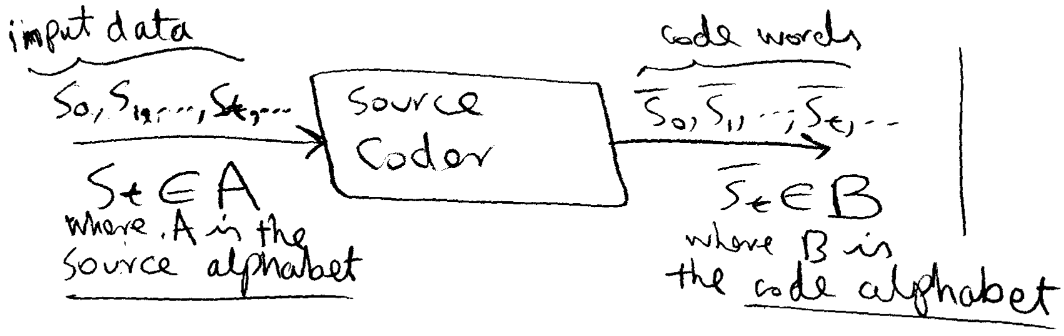
$\Rightarrow$ The ~~joint~~ entropy of the ~~[illegible]~~ block is

$$H(A_0, A_1, \cdots, A_5, b) = 6\, H(A)$$
$$= 6$$

since the symbols of A are equally probable.

1.3 Source Coding

1.3.1 Mapping Functions and Efficiency


input data
$S_0, S_1, \ldots, S_t, \ldots$
Source Coder
code words
$\bar{S}_0, \bar{S}_1, \ldots, \bar{S}_t, \ldots$
$S_t \in A$
where A is the source alphabet
$\bar{S}_t \in B$
where B is the code alphabet

Why we use source coder? maximum entropy happens when the symbols from A are equally probable.

If the source entropy is $H(A) < \log_2(|A|)$, then we can make the represention of the source symbols more efficient and close to $H(A)$ by using a Source coder. Thus, instead of transmitting $\log_2(|A|)$ bits per symbol, we can transmit $H(A)$ bits per symbol on average.

Example 1.3.1

Let A be 4-ary source. Let C be an encoder which maps the symbols in A into binary digits as follows:

$$
\left.\begin{array}{ll}
P_0 = 0.5 & C(a_0) \rightarrow 0 \\
P_1 = 0.3 & C(a_1) \rightarrow 10 \\
P_2 = 0.15 & C(a_2) \rightarrow 110 \\
P_3 = 0.05 & C(a_3) \rightarrow 111
\end{array}\right\} \text{There are called codewords}
$$

So The averge number of bits tranmitted per codeword is

$$\bar{L} = \sum_{m=0}^{3} P_m L_m = 0.5(1) + 0.3(2) + 0.15(3) + 0.05(3)$$
$$= 1.70 \text{ bits per symbol.}$$

Compared to $\frac{H(A)}{2} = 82.385\%$,
we see that the efficiency increased 14%.

- The above example is ~~stagll~~ A Huffman code ~~, we will study more systematic way to implement a source coder.~~

Mathmatically, a source encoder is a function C that maps the source alphabet A to a code alphabet B.

$$C : A \longrightarrow B$$

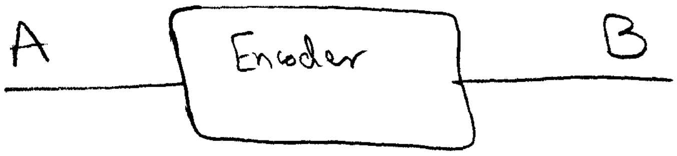
Since the encoder must be decoded at the receiver, the function C must be invertible. The inverse of C is called C^{-1} such that if $C(a) \rightarrow b$, then $C^{-1}(b) \rightarrow a$.

$\Longrightarrow$ The function C is one-to-one and onto.

Solve example 1.3.2.

In this example, by grouping the symbols emitted by source A into ordered pairs $\langle a_i, a_j \rangle$, the encoder applied in this example got more information efficiency.

1.3.2 Mutual Information



The Mutual information $I(B;A)$ is the amount of reduction in uncertainty about B when we observe A.

$$I(B;A) = H(B) - H(B/A)$$
$$= \sum_{b\in B}\sum_{a\in A} P_{ba} \log_2\left(\frac{P_{ba}}{P_b P_a}\right)$$

Also, $I(A;B) = H(A) - H(A/B)$
$$= \sum_{b\in B}\sum_{a\in A} P_{ba} \log_2\left(\frac{P_{ba}}{P_b P_a}\right)$$

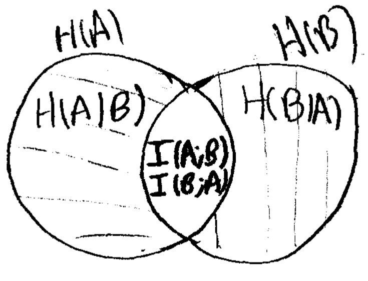
Notice that $I(A;B) = I(B;A)$

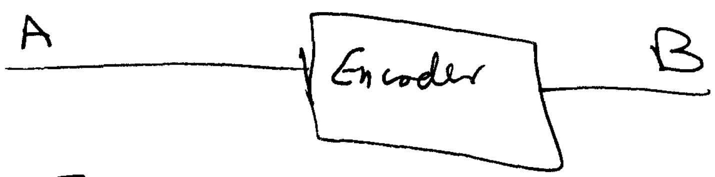
In other words, the Mutual information $I(B;A)$ ~~is~~ the amount of information we can tell about B from observing A.

Graphical Illustration

~~Let~~

now





The relation between $H(A)$ and $H(B)$ defines three kinds of Encoders.

1- Lossless Encoder

No information is lost. In this case,

$H(A) = H(B)$ and $H(B|A) = 0$

and $I(B;A) = H(B)$

$\Rightarrow$ The two circles are on top of each other and the information content of A and B are the same.

2- Lossy Encoder

Some information is Lost.

$\Rightarrow$ $H(B) < H(A)$

Also, $I(A;B) < H(A)$

$\Rightarrow I(B;A) < H(A)$

3- Encryption or Channel Coding

In this case, the information content in B is increased ~~in~~ In order to improve performance over noisy channels "channel coding" or to increase security "Encryption".

$H(B) > H(A)$

The extra amount of information is redundant. It doesn't add more knowledge to A. ~~But~~

This added information makes it hard for

1.4 Huffman Coding

Prefix codes

- Huffman codes are lossless data compression codes.
- Huffman codes are optimal in the sense that they can deliver codeword sequences which asymptotically approach the source entropy.
- Huffman codes are variable-length code words.
- Huffman codes are prefix codes. This kind of codes have the property of being self-punctuating. That means that the decoder can easily know the begining and ending of each code word. That makes decoding possible. In order to have that, the design must satisfy the following rule:

  No code word is a prefix of some other code word.

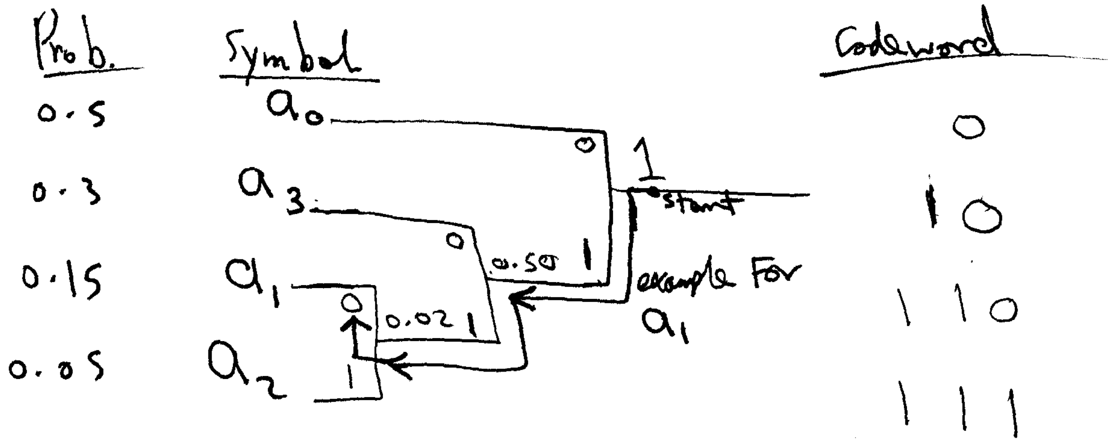
Such codes are prefix codes and instantaneously decodable.

Huffman coding Construction

Example $A = \{a_0, a_1, a_2, a_3\}$

$P_A = \{0.5, 0.15, 0.05, \cancel{0.05}\ 0.3\}$

First Arrange Symbols from High Probability to Low:



Prob.	Symbol	Codeword
0.5	a_0	0
0.3	a_3	1 0
0.15	a_1	1 1 0
0.05	a_2	1 1 1

Second Combine the probability of symbols ~~B~~ starting from the Lowest probability and Lable the top one with "0" and the lower one with "1".

Third Codewords are assigned by reaching the tree From right to left back to the original symbol. For example, for a_1, when we go from right to left, we get 1 1 0

Huffman Decoding

Example Assume we received the following sequence from left to right

100111100010110

The decoder ~~has~~ knows the codeword table.

~~First~~ The valid codewords in the sequence are

10, 0, 111, 10, 0, 10, 110

decoded sequence $\rightarrow$ a_3 a_0 a_2 a_3 a_0 a_3 a_1

1.5 Dictionary Codes and Lempel-Ziv Coding

— Huffman codes require us ~~to~~ find or estimate the probabilities of the source symbols. This might be practical for some applications. However, real-time applications, such as compressing computer files, ~~[illegible]~~ are not tolerant to delays.

— Dictionary codes are compression codes that dynamically construct their coding and decoding tables "On the Fly" by looking at the data stream itself. No calculations of probabilities are required.

— Disadvantage; dictionary codes are efficient only for long files or messages. Short files can result in the transmission of more bits, on average, than the original message.

— Lempel-Ziv (LZ) codes

- A class ~~of~~ dictionary codes

Advantages:

— Asymptotically approach the Source entropy for long messages.

- The decoder doesn't require prior knowledge ~~of~~ the coding table constructed by the transmitter.

— The receiver construct its own decoding table on the fly from the received code words.

— ~~These~~ codes initially expand rather than compress the data since extra information is sent to aid the receiver. ~~That is the reason~~ Because of that, ~~the~~ these codes don't work well for short files and we might get larger size than before compression.

— However, For long Files, As time progress, less data is sent and compression occurs.

— LZ codes have no decoding delays

1.5.2 A Linked-List LZ Algorithm

— Assume the source alphabet is $A = \{a_0, a_1, \cdots, a_{M-1}\}$ of cardinality M. The algorithm is initialized by constructing the first $M+1$ entries in the dictionary as follows:

	address	dictionary entry
$M+1$ entries	0	0 , null
	1	0, a_0
	2	0, a_1
	$\vdots$	$\vdots$
	m	0, a_{m-1}
	$\vdots$	$\vdots$
	M	0 , a_{M-1}

— Initialize pointer variable $n = 0$ ~~, address pointer~~

Address pointer $m = M+1$ $\rightarrow$ next blank location in the dictionary.

- Algorithm

```
1. Fetch next source symbol $a$
2. If the ordered pair $<n,a>$ is already in the dictionary then
      — $n$ = dictionary address of entry $<n,a>$
   else
      — transmit $n$
      — create new dictionary entry $<n,a>$
           at dictionary address $m$.
      — $m = m+1$
      — $n$ = dictionary address of entry $<0,a>$
3. Return to step 1.
```

Example 1.5.1

A binary source emits the sequence of symbols

110 001 011 001 011 100 011 11

See the example in Textbook Page 24

~~[illegible]~~

1.5.3 Decoding Process of LZ codes

- At ~~the~~ receiver, the decoder will construct a dictionary identical to ~~the~~ one at the transmitter
- it starts by the initial dictionary "root symbols" and build on it by examining the received pointer n.
- Operation of the decoder is governed by the following Rules:
  1. Reception of any code word means that a new dictionary entry must be constructed.
  2. Pointer n for this new dictionary entry is the same as the received code word n.
  3. Source symbol a for this entry is not yet known since it is the root symbol of the next string.

Example 1.5.2 : Read the explaination in Text book P.26

Received (n)	decoded symbols	
2	- we received $n=2$, add a new entry $<2,?>$, still we don't know the source symbol a for this entry. However, we can find the source symbol of the previous source, we link the pointer 2 to the root symbols, we get symbol "1".	1
2	- we receive $n=2$, link it to root symbol, we find the previous symbol to be "1".	1
1	- $n=1$, see text book	0
5	- $n=5$; link to address 5 we have $<1,?>$, link it again to address 1 we get $a=0$. So the previous symbols are "00". ~~So~~ we decoded two symbols at the same time since we did two linking steps.	0 0
4	- Read the explaination for the rest in the text book.	1 0
3		1 1
		0 0 1

m	Entry	Links
0	0, null	
1	0, 0	
2	0, 1	
3	2, 1	$\to <0,1>$ (was transmited)
4	2, 0	$\to <0,1>$
5	1, 0	$\to <0,0>$
6	5, 1	$\to <1,0> \to <0,0>$ "00"
7	4, 1	$\to <2,0> \to <0,1>$ "10"
8	3, 0	$\to <2,1> \to <0,1>$ "11"
9	6,	$\to <5,1> \to <1,0> \to <0,0>$ "001"

— Notice in ~~example~~ 1.5.1, the dictionary has more than 16 entries, ~~therefore~~, n must consist ~~of~~ at least five bits. Thus, more bits are being transmitted than were in the source message up to that point.

— So, where is the compression?

Compression will happen when we process large amount of data.

Example of Huffman Coding ~~with~~

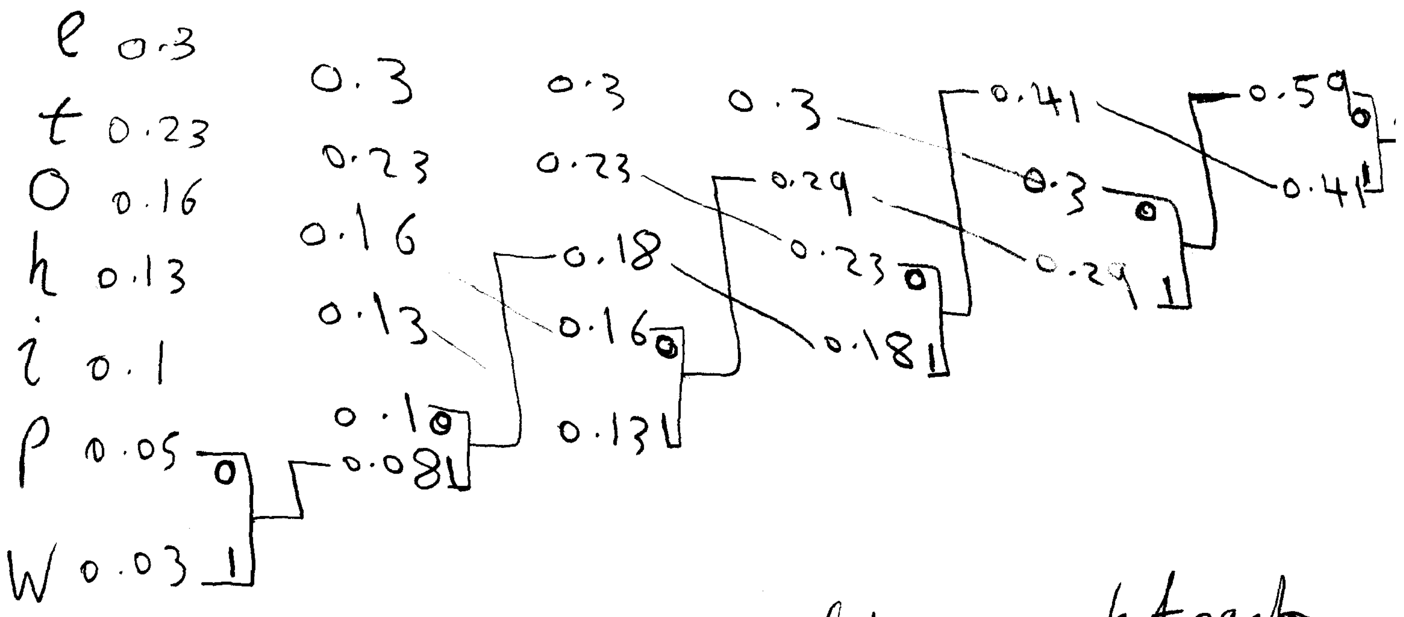
When we sum the Lowest two probabilities, we might get a sum greater than some other symbol probability. in this case, we have to sort the probabilities again from Max to min and keep tracking the changes.

The following example will illustrate this ~~proc~~

Assume we have seven letters;

$$A = \{e, h, i, o, p, t, w\}$$

with Prob. $PA = \{0.3, 0.13, 0.1, 0.16, 0.05, 0.23, 0.03\}$



Remember to sort every time and keep track of each symbol. Then trace the code of each symbol, we will get

e →	00
t →	10
O →	010
h →	011
i →	110
P →	1110
w →	1111