03.02.2020

$H(S) \leq \bar{l} < H(S)+1$ — we can come up with a code whose average length $< H(S)+1$

Let a source 'S' with an alphabet $A = \{a_1, \ldots a_m\}$ and a prob. model $P(a_1), P(a_2) \ldots, P(a_m)$.

$$l_i = \left\lceil \log_2 \frac{1}{P(a_i)} \right\rceil \quad \text{for } i = 1, 2, \ldots m$$

$$\log_2 \frac{1}{P(a_i)} \leq l_i < \log_2 \frac{1}{P(a_i)} + 1$$

$$-\log_2 P(a_i) \leq l_i < -\log_2 P(a_i) + 1$$

Multiply by '-1'

$$\log_2 P(a_i) \geq -l_i$$

Raise to the power '2'

$$2^{\log_2 P(a_i)} \geq 2^{-l_i}$$

$$P(a_i) \geq 2^{-l_i}$$

$$\sum_{i=1}^{m} P(a_i) \geq \sum_{i=1}^{m} 2^{-l_i}$$

sum of probabilities is '1'

$$\sum_{i=1}^{m} 2^{-l_i} \leq 1$$

By Kraft McMillan inequality, there exists a prefix code with codeword lengths $l_1, l_2, \ldots l_m$

$$\rightarrow l_i < \log_2 \frac{1}{P(a_i)} + 1$$

$$\sum_{i=1}^{m} P(a_i) l_i < \sum_{i=1}^{m} P(a_i) \left[\log_2 \frac{1}{P(a_i)} + 1 \right]$$

$$= \underbrace{\sum_{i=1}^{m} P(a_i) \cdot \log_2 \frac{1}{P(a_i)}}_{H(S)} + \underbrace{\sum_{i=1}^{m} P(a_i)}_{1}$$

Extended Huffman codes

Ex: $A = \{a_1, a_2, a_3\}$

$P(a_1) = 0.8$, $P(a_2) = 0.02$, $P(a_3) = 0.18$

$H = 0.816$ bits/symbol

$C(a_1) = 0$

$C(a_2) = 11$

$C(a_3) = 10$

Average length of the code $= 0.8 \times 1 + 0.02 \times 2 + 0.18 \times 2$

$= 1.2$ bits/symbol

Redundancy $= 1.2 - 0.816$

$= 0.384$ bits/symbol

47% of the entropy

This is optimum and this itself has 50% of wastage.

So, we can sometimes reduce the coding rate by blocking more than one symbol at a time, together.

→ Consider a source 'S', $A = \{a_1, a_2, \ldots a_m\}$

Each element of the sequence emited by 'S' is independent

$$H(S) = -\sum_{i=1}^{m} P(a_i) \log_2 P(a_i)$$

We have just shown that, we can generate a Huffman code (whose average length satisfies) for this source with rate 'R' such that,

$$H(S) \leq R < H(S) + 1 \quad \text{——} (1)$$

Suppose, we now encode the sequence by generating one codewood for every 'n' symbols.

As there are m^n combinations of 'n' symbols, we will need m^n codewords in our huffman code.

We could generate this code by viewing the m^n symbols as letters of an extended alphabet.

$$A^{(n)} = \{\underbrace{a_1, \ldots a_1}_{n \text{ times}}, \underbrace{a_1, a_1 \ldots a_1 a_2}_{n-1 \text{ times}}, \ldots\ldots \underbrace{a_m a_m, \ldots a_m}_{n \text{ times}}\}$$

For the same source with an extended alphabet, we get,

$S^{(n)}$ $\quad H(S^{(n)}) \leq R^{(n)} < H(S^{(n)}) + 1$ ——— (2)

$$\frac{H(S^{(n)})}{n} \leq \frac{R^{(n)}}{n} < \frac{H(S^{(n)})}{n} + \frac{1}{n}$$

$$H(S^{(n)}) = -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1}a_{i_2}\cdots a_{i_n}) \log_2 P(a_{i_1}a_{i_2}\cdots a_{i_n})$$

Independent event

$$= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} \left[P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n})\right] \log_2 \left[P(a_{i_1}).P(a_{i_2})\cdots P(a_{i_n})\right]$$

$$= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} \left[P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n})\right].\left[\sum_{k=1}^{n} \log_2 P(a_{i_k})\right]$$

$$\left[\log_2 P(a_{i_1}) + \log_2 P(a_{i_2}) + \cdots \log_2 P(a_{i_n})\right]$$

$$= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} \left[P(a_{i_1}).P(a_{i_2})\cdots P(a_{i_n})\right] \log_2 P(a_{i_1}) -$$

$$\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1}).P(a_{i_2})\cdots P(a_{i_n}) \log_2 P(a_{i_2}) -$$

$$\cdots\cdots\cdots\cdots$$

$$\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n}) \log_2 P(a_{i_n})$$

$$= -\sum_{i_1=1}^{m} P(a_{i_1}) \log_2 P(a_{i_1}) \underbrace{\sum_{i_2=1}^{m} P(a_{i_2})}_{1} \cdots \underbrace{\sum_{i_n=1}^{m} P(a_{i_n})}_{1} -$$

$$\sum_{i_1=1}^{m} P(a_{i_1}) \sum_{i_2=1}^{m} P(a_{i_2}) \log_2 P(a_{i_2}) \cdots \sum_{i_n=1}^{m} P(a_{i_n})$$

$$\cdots\cdots -$$

$$\sum_{i_1=1}^{m} P(a_{i_1}) \sum_{i_2=1}^{m} P(a_{i_2}) \cdots \sum_{i_n=1}^{m} P(a_{i_n}) \log_2 P(a_{i_n})$$

$$= -\sum_{i_1=1}^{m} P(a_{i_1}) \log_2 P(a_{i_1}) - \sum_{i_2=1}^{m} P(a_{i_2}) \log_2 P(a_{i_2}) \cdots\cdots - \sum_{i_n=1}^{m} P(a_{i_n}) \log_2 P(a_{i_n})$$

$$= -\sum_{i=1}^{m} P(a_i) \log_2 P(a_i) - \sum_{i=1}^{m} P(a_i) \log_2 P(a_i) \cdots\cdots$$

$$= -n \sum_{i=1}^{m} P(a_i) \log_2 P(a_i)$$

$$= n \cdot H(S)$$

$$\frac{n \cdot H(S)}{n} \leq \frac{R^{(n)}}{n} < \frac{n H(S)}{n} + \frac{1}{n}$$

$$H(S) \leq \frac{R^{(n)}}{n} < H(S) + \frac{1}{n}$$

blocking of letters

Adaptive Huffman coding

Suppose there is live show going on, we can't do the compression, analyze the data being sent and etc,. So, we use sth called as adaptive HC.

dynamically does with the data.

Associated with each node we are going to assign two numbers

1. weight of a node
2. Node number

- Weight of a leaf node is equal to the number of times the symbol corresponding to that leaf has occured.

- Weight of an internal node is equal to the sum of the weights of its children.

Node number is a unique number that is assigned to each node of the tree.

10.02.2020

Adaptive Huffman coding

Huffman code can be described interms of a binary tree called Huffman tree → Weight of a node
→ Node number

- Root node is given the highest node number.
- NYT node is given the least node number.

'N' is the number of external nodes, then the total no. of nodes is '2n-1'.

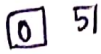
Transmitter

- Start with a single node

NYT (not yet transmitted)

Receiver

- Start with a single node



NYT

As transmission progresses, nodes corresponding to symbols transmitted will be added to the tree and the tree is reconfigured using an UPDATE procedure.

Before beginning of transmission, a fixed code for each symbol is agreed upon between transmitter and receiver.

Fixed code:

If the source has an alphabet $(a_1, a_2, a_3, \ldots a_m)$ of size 'm', then pick 'e' and 'r' such that,

$$m = 2^e + r \quad 0 \le r < 2^e$$

The letter a_k is encoded as (e+1) bit binary representation of k-1 if $1 \le k \le 2r$, else a_k is encoded as the e-bit binary representation of k-r-1.

For example, $m = 26$ $(a_1, a_2, a_3, \ldots\ldots a_{26})$

$26 = 2^4 + 10$ $\quad r = 10, e = 4$ $\qquad 21 - 10 - 1 = 10$?

$a_1 \longleftrightarrow$ 00000

$a_2 \longleftrightarrow$ 00001

$a_3 \longleftrightarrow$ 00010

$\vdots$

$a_{20} \longleftrightarrow$ 10011

$a_{21} \longleftrightarrow$ 1010

$a_{22} \longleftrightarrow$ 1011

$\vdots$

When a symbol is encountered for the first time, the code for the NYT node is transmitted, followed by the fixed code for the symbol.

A node for the symbol is then created and the symbol is taken out from the NYT list.

The numbers from the NYT node to the root of the tree is assigned in increasing order from left to right and lower level to upper level.

* The set of nodes with this same weight makes up a block.

After a symbol has been encoded or decoded, the external node corresponding to the symbol is examined to see if it has the largest node number in its block.

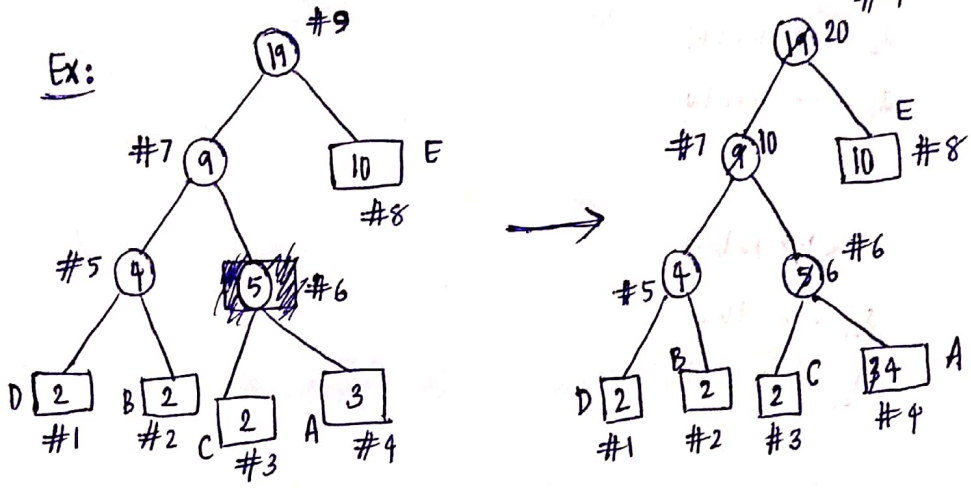
If the external node doesn't have the largest node number, it is exchanged with the node that has the largest node number, in the block as long as the node with the highest number is not the parent of the node being updated.

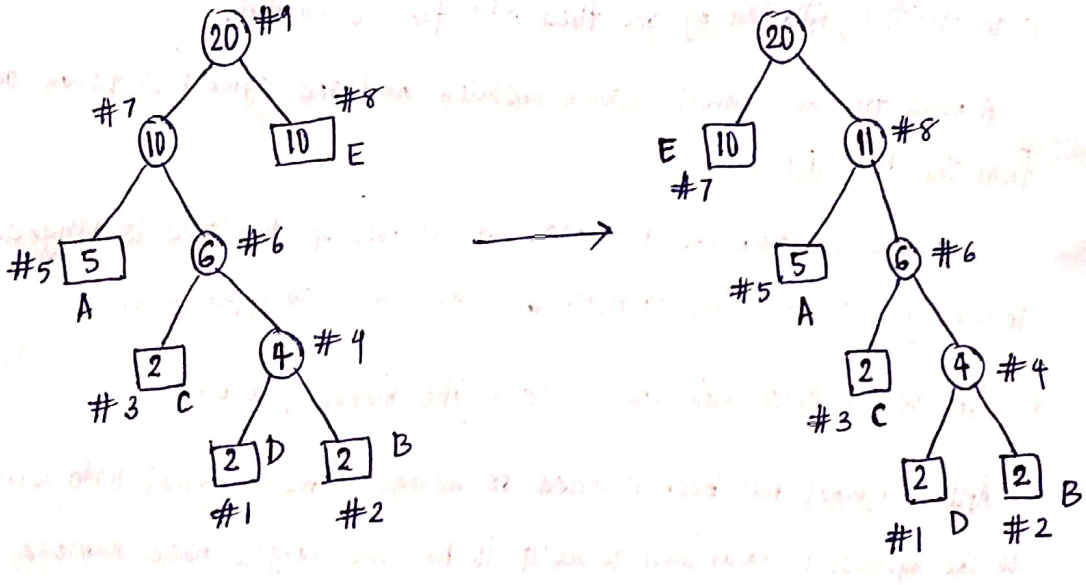
The weight of the external node is then incremented.

This is repeated.

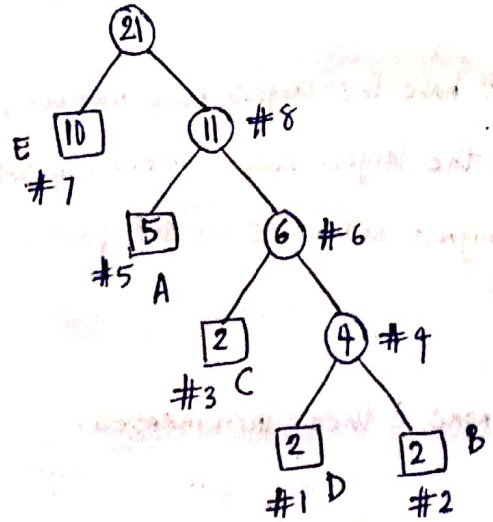
If the symbol to be encoded or decoded has occured for the first time, a new external node is assigned to the symbol and a new NYT node is appended to the tree.

Ex:





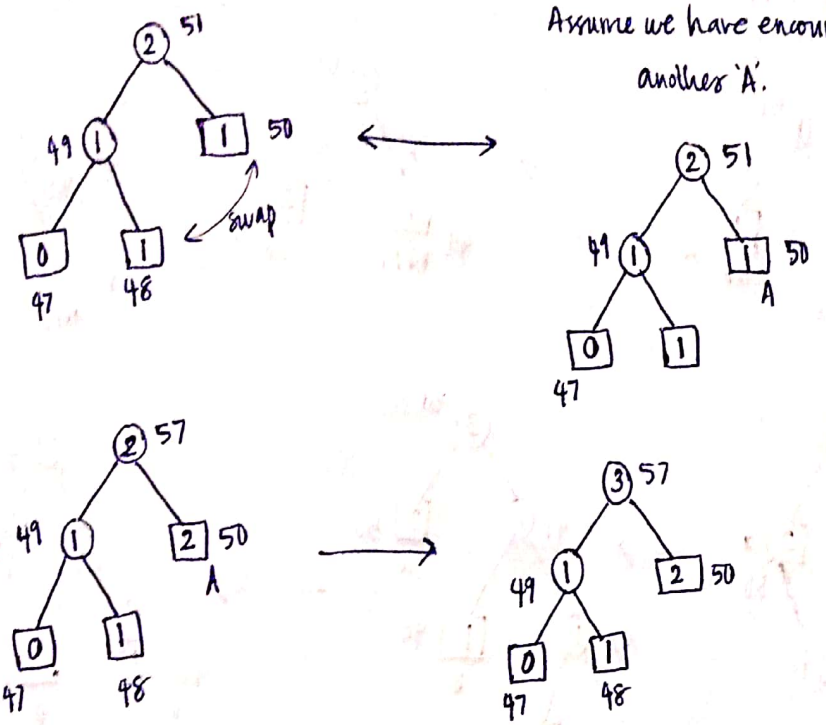
?



13.02.2020

Adaptive Huffman Coding

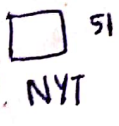
Ex:



Complete example

Assume that we are encoding the message "a a r d v a r k", where our alphabet consists of the 26 lowercase letters of the English alphabet.

Encoding procedure



$26 = 2^4 + 10$, $e = 4$, $r = 10$

$a_k \leftrightarrow$ e+1 bit representation of k-1 if $1 \le k \le 2r$

e-bit representation of k-r-1 if $k > 2r$

a - 00000
b - 00001
c - 00010
:
t = 10011
u - 1010
v - 1011
w - 1100

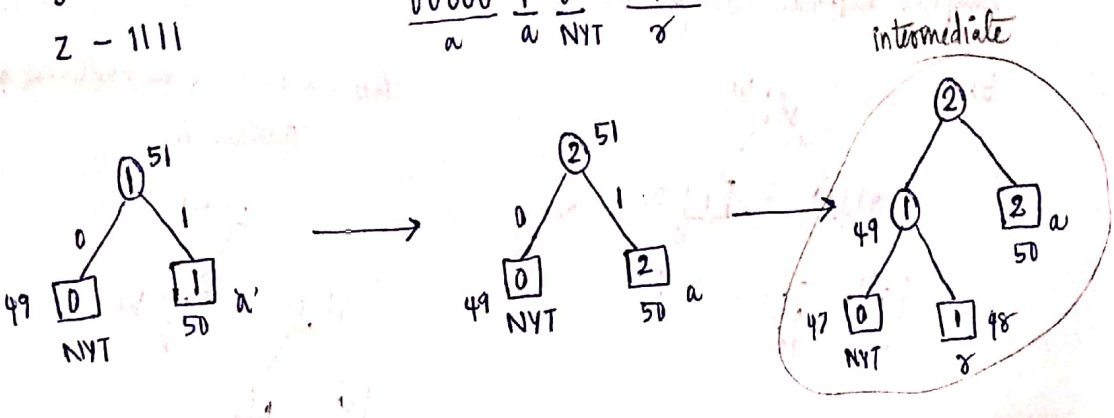
x - 1101

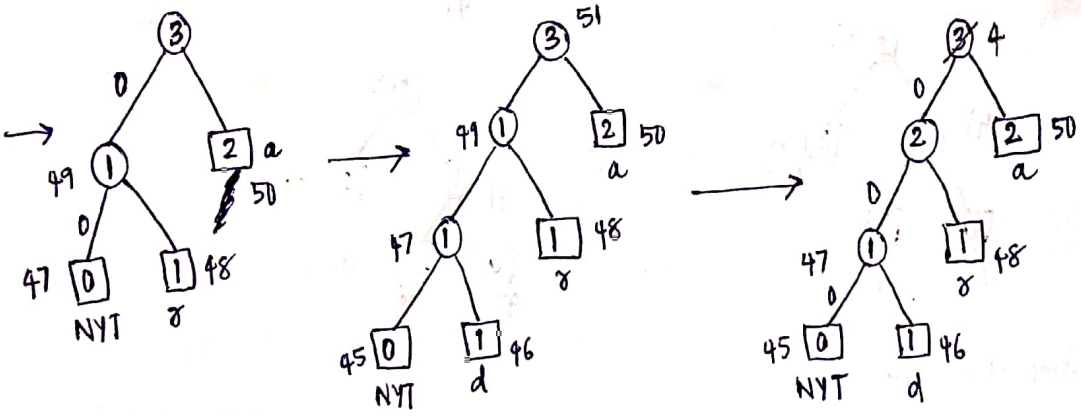
y - 1110

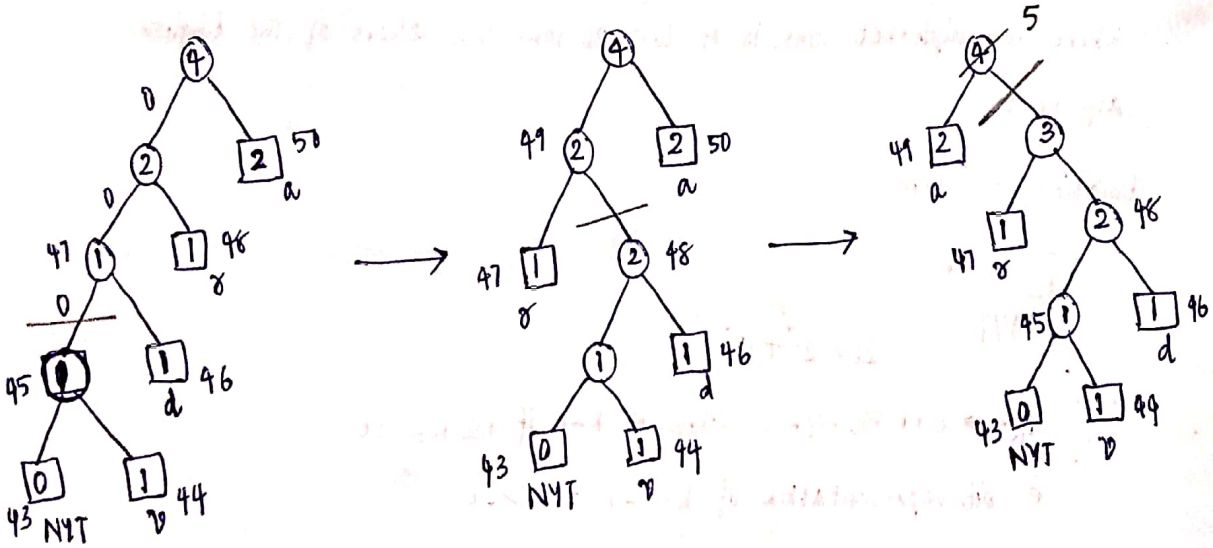
z - 1111

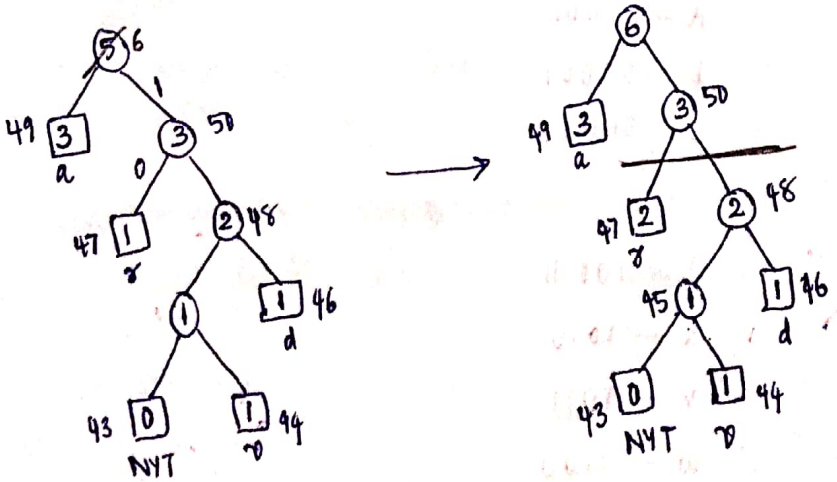
00000 1 0 10001
a a NYT r

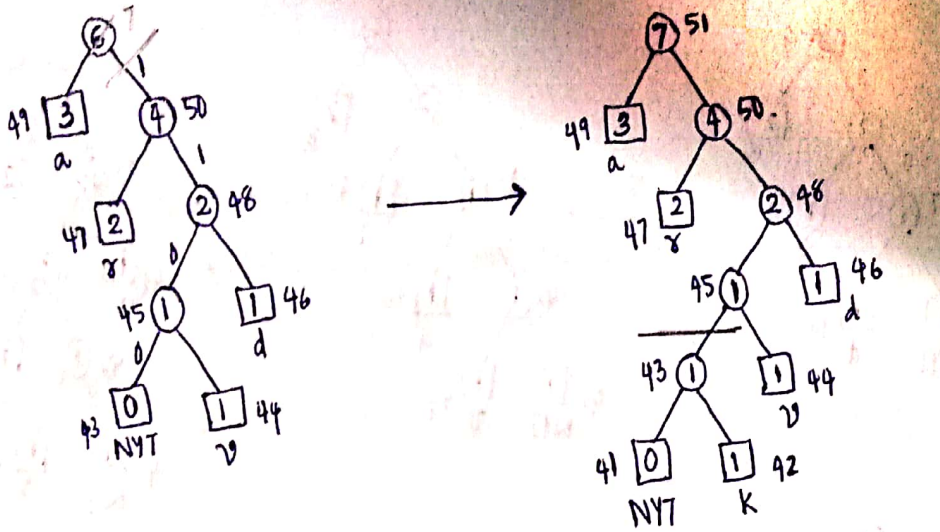
intermediate



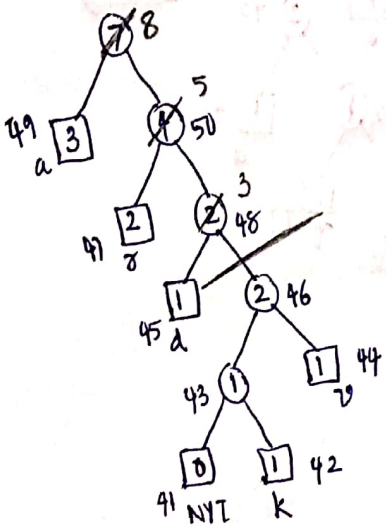






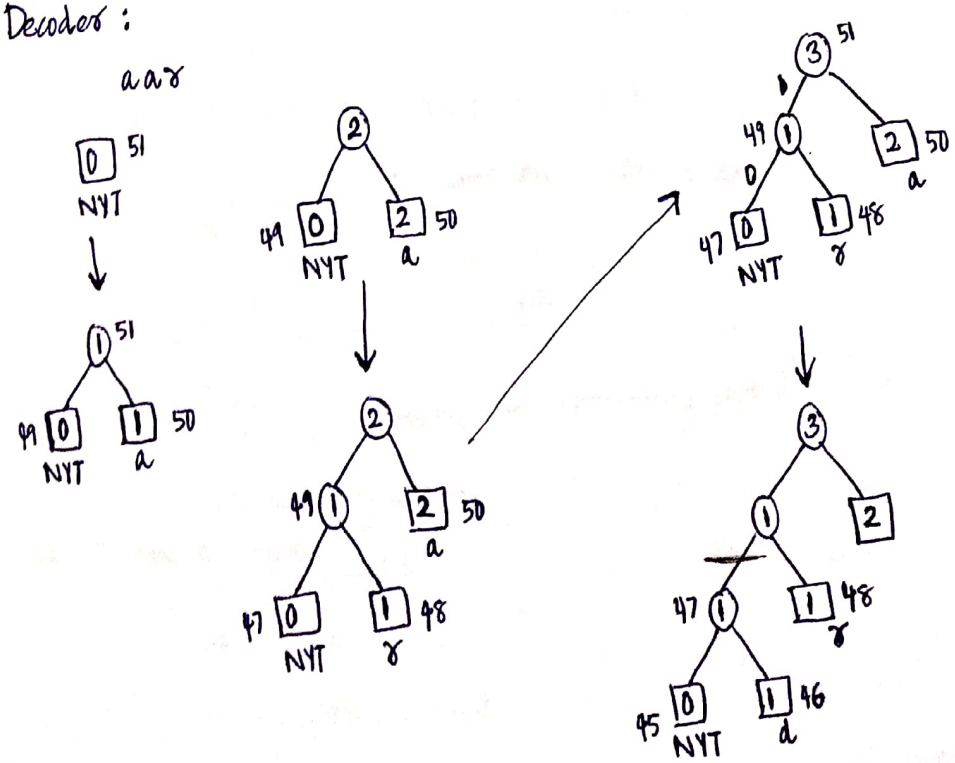


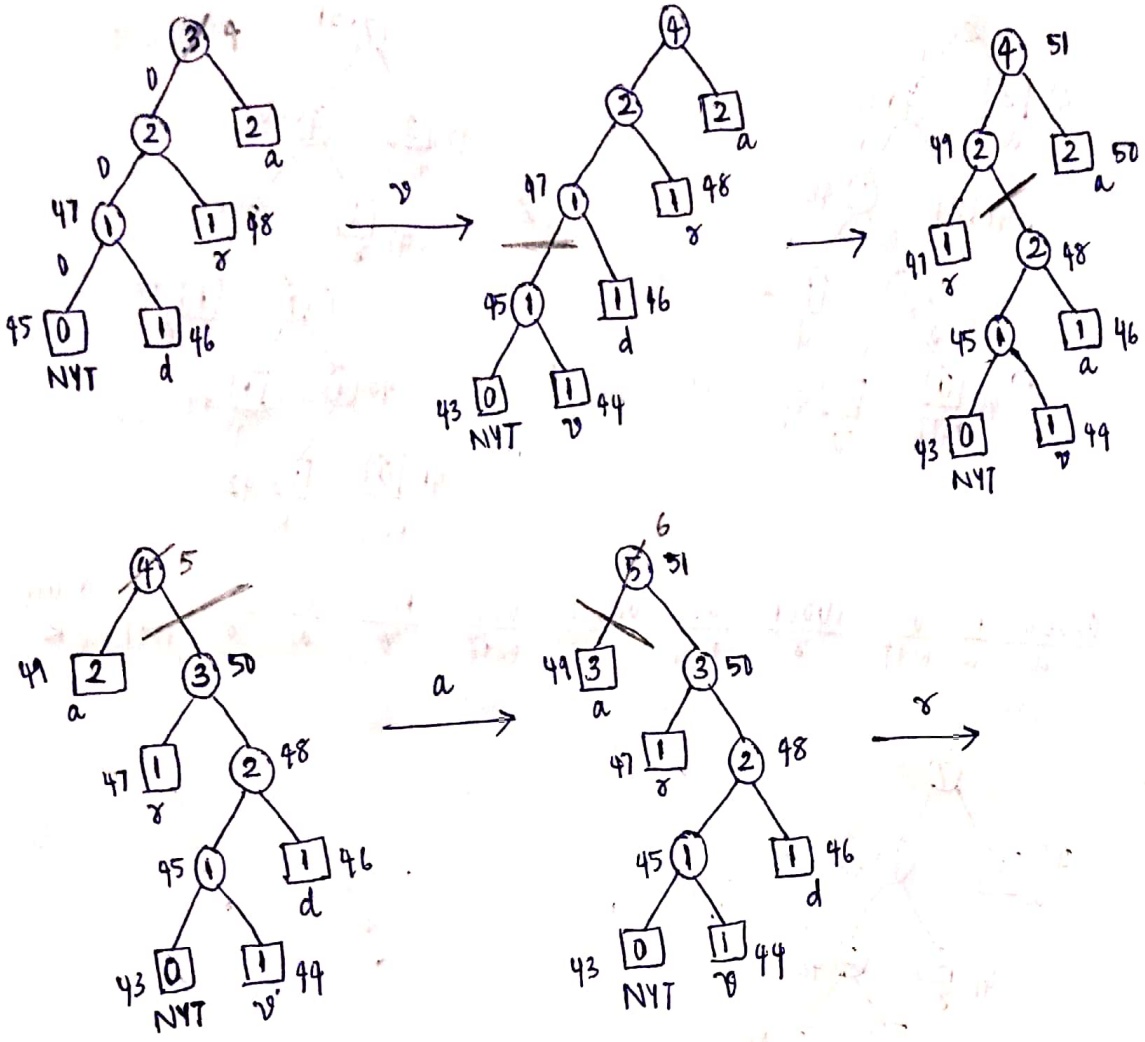
00000 a | 1 a | 0 NYT | 10001 r | 00 NYT | 00011 d | 000 NYT | 1011 v | 0 a | 10 r | 1100 NYT | 01010 K



Decoder :

aar





Golomb code

Unary code -

These codes are usually meant for integers.

Assumption : The bigger the integer, the less the probability with which it will be encoded.

'n' integer $\longrightarrow$ $\underbrace{1111\ldots10}_{n\ 1's}$

3 - 1110
4 - 11110
⋮

It becomes a huffman code, if the prob. model is,

$$P(k) = \frac{1}{2^k}$$

Golomb code is a family of codes parameterized by a positive integer 'm'.

Fix 'm', then

codeword for integer 'n' is,

$n = mq + r \quad 0 \le r < m$ (remainder)

q - unary code

r - some other code for 'r'

If $m = 2^k$, then no. of bits of 'r' $\log_2 m = \log_2 2^k = k$

$m \neq 2^k$, $\lceil \log_2 m \rceil$

But this ceil can be a waste

So, he used ceil for some and floor for some other values of 'r'.

$\lceil \log_2 m \rceil$ bits for some values of r.

$\lfloor \log_2 m \rfloor$ bits for some other values of r.

$0 \le r \le 2^{\lceil \log_2 m \rceil} - m \rightarrow \lfloor \log_2 m \rfloor$ bits

$r > 2^{\lceil \log_2 m \rceil} - m \rightarrow \lceil \log_2 m \rceil$ bits representing $r + 2^{\lceil \log_2 m \rceil} - m$

Ex: Design a Golomb code for $m = 5$

$\lfloor \log_2 5 \rfloor = 2 \quad \lceil \log_2 5 \rceil = 3$

$r \in \{0, 1, 2, 3, 4\}$

$2^{\lceil \log_2 5 \rceil} - 5 = 2^3 - 5 = 3$ values of $r = \{0, 1, 2\}$

$r \in \{3, 4\} \rightarrow$ 3 bit representation

$(r + 2^{\lceil \log_2 m \rceil} - m)$

$3 + 3 = 6$, $3 + 4 = 7$

$m = 5$

quotient is taken.

n	q	r	codeword
0	0	0	000
1	0	1	001
2	0	2	010
3	0	3	0110
4	0	4	0111
5	1	0	1000
6	1	1	1001
7	1	2	1010
8	1	3	10110
9	1	4	10111
10	2	0	11000
11	2	1	11001
12	2	2	11010
13	2	3	110110
14	2	4	110111
15	3	0	111000
16	3	1	111001
17	3	2	111010
18	3	3	1110110
19	3	4	1110111
20	4	0	1111000

21	4	1	11110 01
22	4	2	11110 10
23	4	3	11110 110
24	4	4	11110 111
25	5	0	111110 00

Tunstall code

All the codewords are of the same length, but the no. of symbols are different.

Sequence	codeword
AAA	00
AAB	01
AB	10
B	11

Lets encode the sequence

AAA B AAB AAB AAB AAA

00 11 01 01 01 00

1. We should be able to parse a source output sequence into sequences of symbols that appear in the code block.
2. We should maximize the average number of source symbols represented by each codeword.

Sequence	codeword
AAA	00
ABA	01
AB	10
B	11

AAA B AAB AAB AAA

00 11

Tunstall gave an algorithm that fulfils these two conditions.

17.02.2020

Suppose we want an n-bit Tunstall code for a source that generates iid letters from an alphabet of size N.

Number of codewords possible is 2^n.

Ex: 3-bit Tunstall code

$A = \{A, B, C\}$

$P(A) = 0.6$, $P(B) = 0.3$, $P(C) = 0.1$

Letter	Probability	Codeword
→ A	0.6	
B	0.3	
C	0.1	

Take the highest prob. symbol

Letter	Prob.	Codeword
→ AA	0.36	
AB	0.18	
AC	0.06	
B	0.3	
C	0.1	

We can multiply the probabilities as they are independent.

Letter	Prob.	Codeword
→ AAA	0.216	000
AAB	0.108	001
AAC	0.036	010
AB	0.18	011
AC	0.06	100
B	0.3	101
C	0.1	110

AAAA
AAAB
AAAC
AAB
AAC
AB
AC
B
C

We cannot anymore concatenate the letters as it will exceed the no. of codewords possible i.e, 8.

$N + K(N-1) \leq 2^n$

$3 + 2k \leq 8$

$2k \leq 5$

$k \leq 2$

Arithmetic code

Ex: Consider a source that puts out independent & identically distributed (iid) letters from the alphabet, $A = \{a_1, a_2, a_3\}$ with the probability model

$P(a_1) = 0.95$, $P(a_2) = 0.02$ and $P(a_3) = 0.03$

Entropy = 0.335 bits/symbol

Letter	prob.	codeword
a_1	0.95	$c(a_1)$
a_3	0.03	$c(a_3)$
a_2	0.02	$c(a_2)$

$c(a_2) = \alpha_1 * 1 = 11$

$c(a_3) = \alpha_1 * 0 = 10$

Letter	prob.	codeword
a_1	0.95	$c(a_1) = 0$
a_2'	0.06	$c(a_2') = 1 = \alpha_1$

$c(a_1) = 0$, $c(a_2) = 11$, $c(a_3) = 10$

Average length $= 0.95 \times 1 + 0.02 \times 2 + 0.03 \times 2$

$= 1.05$ bits/symbol $\simeq$ 3 x the no. of bits being used.

Block-2 symbol

$a_1 a_1$

$a_1 a_2$

$a_1 a_3$

$a_2 a_1$

$a_2 a_2$

$a_2 a_3$

$a_3 a_1$

$a_3 a_2$

$a_3 a_3$

$\frac{1.222 \text{ bits/symbol}}{2}$

$= 0.611$ bits/symbol

$\simeq$ 2 x the number of bits required

If you want to bring the redundancy to acceptable levels, we need to block 8 symbols.

Then, the size of the codeblock $= 3^8 = 6581$

Here, blocking is definitely done, but the size of the codeblock increasing with blocking has to be addressed. (Consumes more memory).

Arithmetic code is a solution to this problem.

→ Design of an arithmetic code can be explained by splitting the procedure into two parts.

(i) Compute a unique tag for a given sequence. → $[0,1]$

(ii) Assign a unique codeword for every tag.

Let the sequence be of the symbols of any block-size, diff. sequences should still have unique tags for each of the symbols.

Random variable maps the probabilities of events onto the set of real numbers.

Quite often, the random variable takes a_i and maps it to real number 'i'.

$A = \{a_1, a_2, \ldots a_n\}$

$X(a_i) = i \quad \forall\ i = 1, 2, \ldots m$

Given the prob. model, we can talk about prob. density function of the corresponding random variable.

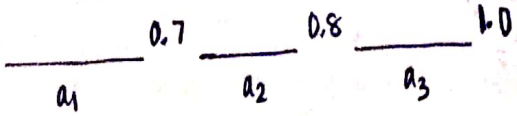
$$P(X=1) = P(a_i)$$

pdf of 'X'

→ Cumulative density function is defined as,

$$F_X(i) = \sum_{j=1}^{i} P(X=j)$$

Ex: Consider a 3-letter alphabet $A = \{a_1, a_2, a_3\}$ with $P(a_1) = 0.7$, $P(a_2) = 0.1$ and $P(a_3) = 0.2$



$F_x(0) = 0$;

$F_x(1) = 0.7$, $F_x(2) = 0.8$

$F_x(3) = 1.0$

- Any number in the interval $[0, 0.7)$ is a unique tag for the symbol a_1
- Any number in the interval $[0.7, 0.8)$ is a unique tag for the symbol a_2.
- Any number in the interval $[0.8, 1.0)$ is a unique tag for the symbol a_3

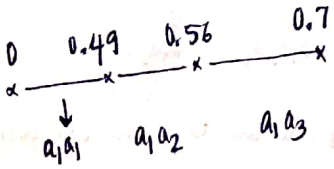
Not only for symbol a_i but for any ~~symbol~~ sequence starting with symbol a_1, the tag lies in the interval $[0, 0.07)$

If the sequence starts with a_2, then the unique tag of the sequence lies in the interval $[0.7, 0.8)$

Suppose the first symbol of the sequence is a_1.

So, the unique tag lies in the interval $[0, 0.7)$

Partition this into, equal probabilities



If the sequence length is 2, the unique tag for a_1a_1 lies in the interval $[0, 0.49)$ and can be any number.

Same with a_1a_2 & a_1a_3.

With arithmetic coding, the intention was to find the codeword for any we want, without having to find for all of the rest of symbols like in extended huffman code.

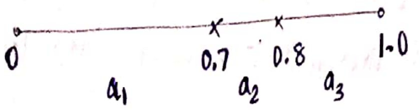
⇒ But the problem now is there is still so much calculations that is to be done for the codeword of a particular sequence.

20.02.2020

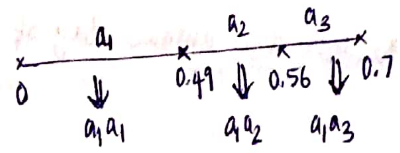
Example: $A = \{a_1, a_2, a_3\}$

$P(a_1) = 0.7$, $P(a_2) = 0.1$, $P(a_3) = 0.2$ $X(a_i) = i$

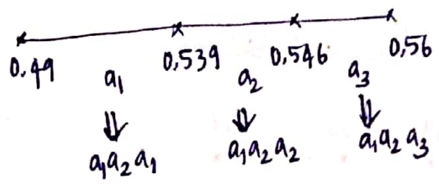
$F_X(1) = 0.7$ $F_X(2) = 0.8$ $F_X(3) = 1.0$



Suppose the first symbol of the sequence is a_1, the tag interval gets confined to $[0\ 0.7)$



Suppose my second symbol is a_2, then the tag interval corresponding to the sequence a_1a_2 is $[0.49\ 0.56)$



Suppose third symbol is a_3, so the tag interval corresponding to the sequence $a_1a_2a_3$ is $[0.546, 0.56)$.

Tag for $a_1a_2a_3 = 0.546$ or midpoint of tag interval 0.553

$[F_X(k-1)\ \ F_X(k))$

Tag generation process (mathematically)

Generate tags of sequences of length one $[0\ \ F_X(1))$ $[F_X(1)\ \ F_X(2))$ . . .
(a_1) (a_2)

$A = \{a_1, a_2, \ldots, a_m\}$

$X(a_i) = i$

$\bar{T}_X(a_i)$ denotes the unique tag of the symbol (sequence) a_i

$$\bar{T}_X(a_i) = F_X(i-1) + \frac{1}{2} P(X=i)$$

$$= \sum_{k=1}^{i-1} P(X=k) + \frac{1}{2} P(X=i)$$

Scanned by CamScanner

Eg: Fair dice

$P(X=k) = \frac{1}{6}$ for $k=1,2,\ldots,6$

$\bar{F}_X(2) = F_X(1) + \frac{1}{2}P(X=2)$

$= P(X=1) + \frac{1}{2}P(X=2)$

$= \frac{1}{6} + \frac{1}{12}$

$= \frac{3}{12} = \frac{1}{4} = 0.25$

$\bar{F}_X(5) = F_X(4) + \frac{1}{2}P(X=5)$

$= P(X=1) + P(X=2) + P(X=3) + P(X=4) + \frac{1}{2}P(X=5)$

$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{2}\cdot\frac{1}{6}$

$= \frac{9}{12} = \frac{3}{4} = 0.75$

$\bar{F}_X(1) = \frac{1}{2}P(X=1) = \frac{1}{12}$

$\bar{F}_X(3) = F_X(2) + \frac{1}{2}P(X=3)$

$= \frac{1}{6} + \frac{1}{6} + \frac{1}{12}$

$= \frac{5}{12}$

$\bar{F}_X(4) = F_X(3) + \frac{1}{2}P(X=4)$

$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{12}$

$= \frac{7}{12}$

$\bar{F}_X(6) = F_X(5) + \frac{1}{2}P(X=6)$

$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{12}$

$= \frac{11}{12}$

We want to extend this procedure for sequances of length '2' and more.

Lexicographic ordering is an ordering induced by the ordering of the letters.

$$\bar{F}_X(x) = \sum_{y<x} P(X=y) + \frac{1}{2}P(X=x)$$

'x' is a sequence of length more than or equal to one ordering among $\underset{y \prec x}{}$ the sequences.

⇒ Two rolls of a dice

11, 12, ----16

21, 22, ----26

31, 32, ----

61, 62, ----66

$$\bar{F}_X(13) = P(X=11) + P(X=12) + \frac{1}{2}P(X=13)$$

$$= \frac{1}{36} + \frac{1}{36} + \frac{1}{72}$$

$$= \frac{5}{72}$$

We don't want to compute probabilities of all the sequences less than the required sequence.

→ Three rolls of a dice

$\bar{F}_X(322)$ lower limit Upper limit

$l^{(0)} = 0$ $u^{(0)} = 1$

$l^{(1)} = F_X^{(1)}(2)$ $u^{(1)} = F_X^{(1)}(3)$

$l^{(2)} = F_X^{(2)}(31)$ $u^{(2)} = F_X^{(2)}(32)$

$l^{(3)} = F_X^{(3)}(321)$ $u^{(3)} = F_X^{(3)}(322)$

$$F_X^{(2)}(32) = P(X=11) + P(X=12) + \cdots P(X=16) + P(X=21) + \cdots + P(X=26) + P(X=31) + P(X=32)$$

$$= P(x_1=1\ x_2=1) + P(x_1=1\ x_2=2) + \cdots$$

$$= \sum_{j=1}^{6} P(x_1=1\ x_2=j) + \sum_{j=1}^{6} P(x_1=2\ x_2=j) + P(x_1=3\ x_2=1) + P(x_1=3\ x_2=2)$$

$$= \sum_{j=1}^{6} P(x_1=1)\,P(x_2=j) + \sum_{j=1}^{6} P(x_1=2)\,P(x_2=j) + P(x_1=3)\,P(x_2=1) + P(x_1=3).P(x_2=2)$$

$$= P(x_1=1)\underbrace{\sum_{j=1}^{6} P(x_2=j)}_{=1} + P(x_1=2)\underbrace{\sum_{j=1}^{6} P(x_2=j)}_{=1} + P(x_1=3)[P(x_2=1) + P(x_2=2)]$$

$$= P(x_1=1) + P(x_2=2) + P(x_1=3)[P(x_2=1) + P(x_2=2)]$$

$$= F_X(2) + P(x_1=3)[F_X(2)]$$

$$= F_X(2) + [F_X(3) - F_X(2)]\,F_X(2)$$

$$u^{(2)} = F_x^{(2)}(32) = F_x(2) + (F_x(3) - F_x(2))\, F_x(2)$$

$$= l^{(1)} + (u^{(1)} - l^{(1)})\, F_x(2)$$

$$l^{(2)} = F_x^{(2)}(31) = l^{(1)} + (u^{(1)} - l^{(1)})\, F_x(1)$$

$$u^{(3)} = F_x^{(3)}(322) = l^{(2)} + (u^{(2)} - l^{(2)})\, F_x(2)$$

$$l^{(3)} = F_x^{(3)}(321) = l^{(2)} + (u^{(2)} - l^{(2)})\, F_x(1)$$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})\, F_x(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})\, F_x(x_n - 1)$$

$$\bar{T}_x(n) = \frac{l^{(n)} + u^{(n)}}{2}$$

Eg: $A = \{a_1, a_2, a_3\}$

$P(a_1) = 0.8$, $P(a_2) = 0.02$, $P(a_3) = 0.18$

$\bar{T}_x(a_1 a_3 a_2 a_1)$

$X(a_i) = i$

$F_x(0) = 0$, $F_x(1) = 0.8$, $F_x(2) = 0.82$, $F_x(3) = 10$

$l^{(0)} = 0$ $u^{(0)} = 1$

$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) \cdot F_x(0) = 0$

$u^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) \cdot F_x(1) = 0.8$

$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) \cdot F_x(2)$

$= 0 + (0.8 - 0) \times 0.82$

$= 0.656$

$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})\, F_x(3)$

$= 0 + (0.8 - 0) \times 1 = 0.8$

$l^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})\, F_x(1)$

$= 0.656 + (0.8 - 0.656) \times 0.8$

$= 0.7712$

$$u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)}) F_X(2)$$
$$= 0.656 + (0.8 - 0.656) \times 0.82$$
$$= 0.77408$$

$$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)}) F_X(0)$$
$$= 0.7712 + (0.77408 - 0.7712) \times 0$$
$$= 0.7712$$

$$u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)}) F_X(1)$$
$$= 0.7712 + (0.77408 - 0.7712) \times 0.8$$
$$= 0.773504$$

$$\bar{T}_X(1321) = \frac{0.7712 + 0.773504}{2}$$
$$= 0.772352$$

Given the tag value, how does we get back the sequence?

$F_X(0) = 0 \quad F_X(1) = 0.8 \quad F_X(2) = 0.82 \quad F_X(3) = 1.0$

$l^{(0)} = 0 \quad u^{(0)} = 1$

$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) F_X(x_1 - 1) = 0 + (1-0) \cdot F_X(x_1 - 1) = F_X(x_1 - 1)$

$u^{(1)} = l^{(1)} + (u^{(0)} - l^{(0)}) F_X(x_1) = 0 + (1-0) F_X(x_1) = F_X(x_1)$

What is x_1 so that,

$$F_X(x_1 - 1) \leq 0.772352 < F_X(x_1)$$

$l^{(1)} = 0 \quad u^{(1)} = 0.8$

$x_1 = 1 \quad 0 \leq 0.772352 < 0.8$ ✓

$x_1 = 2 \quad 0.8 \leq 0.772352 < 0.82$ ✗

$x_1 = 3 \quad 0.82 \leq 0.77352 < 1.0$ ✗

$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) \cdot F_X(x_2 - 1) = 0 + (0.8 - 0) F_X(x_2 - 1)$

$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) \cdot F_X(x_2) = 0 + (0.8 - 0) \cdot F_X(x_2)$

$$0.8 F_X(x_2 - 1) \leq 0.772352 < 0.8 F_X(x_2)$$

What is x_2 which satisfies this inequality?

$x_2 = 1 \quad 0 \leq 0.772352 < 0.64$ ✗

$x_2 = 2 \quad 0.64 \leq 0.772352 < 0.656$ ✗

$x_3 = 3 \quad 0.656 \leq 0.772352 < 0.8$ ✓

$l^{(2)} = 0.656 \quad u^{(2)} = 0.8$

$l^{(3)} = l^{(2)} + (l^{(2)} - l^{(2)})\, F_X(x_3 - 1) = 0.656 + (0.8 - 0.656)\, F_X(x_3 - 1)$

$u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})\, F_X(x_3) = 0.656 + (0.8 - 0.656)\, F_X(x_3)$

$l^{(3)} = 0.656 + 0.144\, F_X(x_3 - 1)$

$u^{(3)} = 0.656 + 0.144\, F_X(x_3)$

Which x_3 satisfies the inequality,

$0.656 + 0.144\, F_X(x_3 - 1) \leq 0.772352 < 0.656 + 0.144\, F_X(x_3)$

$x_3 = 1$ $0.656 \leq 0.772352 < 0.656 + 0.144 \times 0.8 = 0.7712$ ✗

$x_3 = 2$ $0.7712 \leq 0.772352 \leq 0.656 + 0.144 \times 0.82 = 0.77408$ ✓

$x_3 = 3$ $0.77408 \leq 0.772352 < 0.8$ ✗

$l^{(3)} = 0.7712$ $u^{(3)} = 0.77408$

$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})\, F_X(x_4 - 1) = 0.7712 + (0.77408 - 0.7712)\, F_X(x_4 - 1)$

$u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})\, F_X(x_4) = 0.7712 + (0.77408 - 0.7712)\, F_X(x_4)$

$l^{(4)} = 0.7712 + 0.00288\, F_X(x_4 - 1)$

$u^{(4)} = 0.7712 + 0.00288\, F_X(x_4)$

For what value of x_4, the inequality satisfies

$0.7712 + 0.00288\, F_X(x_4 - 1) \leq 0.772352 \leq 0.7712 + 0.00288\, F_X(x_4)$

$x_4 = 1$ $0.7712 + 0.00288 \times 0 \leq 0.772312 < 0.7712 + 0.00288 \times 0.8 = 0.773504$ ✓

Steps

1. Initialize $l^{(0)} = 0$ and $u^{(0)} = 1$
2. For each 'k' find $t^* = (t - l^{(k-1)}) / (u^{(k-1)} - l^{(k-1)})$
3. Find the value of x_k for which $F_X(x_k - 1) \leq t^* < F_X(x_k)$
4. Update $u^{(k)}$ and $l^{(k)}$
5. Continue until the entire sequence has been decoded.

24.02.2020

Arithmetic code

Generating a binary code

Given a sequence — tag — unique identifier

tag → given sequence

Given a sequence - codeword

Given a sequence - tag → codeword ?

Take the binary representation if tag = codeword

↳ truncate it → may not be unique

$$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$$

we'll truncate till $l(x)$ bits

Eg: $A = \{a_1, a_2, a_3, a_4\}$

$P(a_1) = \frac{1}{2}$, $P(a_2) = \frac{1}{4}$, $P(a_3) = P(a_4) = \frac{1}{8}$

Symbol	F_x	$\bar{T}_x$	In binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	codeword
1	$P(a_2)$ 0.5 $+0.25$	0.25	$0.\underline{010}\ldots0$	2	01
2	$P(a_3)$ 0.75 $+0.125$	$(0.5+0.75)/2$ 0.625	$0.\underline{1010}\ldots0$	3	101
3	0.875 $P(a_4)$ $+0.125$	$(0.75+0.875)/2$ 0.8125	$0.\underline{11010}.0$	4	1101
4	1	$(0.875+1)/2$ 0.9375	$0.\underline{11110}\ldots0$	4	1111

$$\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16} = \frac{15}{16} = 0.9375$$

Tag after truncation is still a unique identifier of the given sequence.

$$[F_x(x-1) \quad F_x(x))$$

↳ unique identifier of the sequence x.

$$\lfloor \bar{T}_x(x) \rfloor_{l(x)} \in [F_x(x-1) \quad F_x(x))$$

$$F_x(x-1) \leq \lfloor \bar{T}_x(x) \rfloor_{l(x)} < F_x(x)$$

$$F_x(x-1) \leq \bar{T}_x(x) < F_x(x) \quad \text{——} (1)$$

We know that,

$$\left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)} \leq \bar{T}_x(x) \quad \text{—②}$$

From ① and ②,

we have $\left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)} \leq \bar{T}_x(x) < F_x(x)$ —③

$$\bar{T}_x(x) - \left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)} \leq \frac{1}{2^{l(x)}} \quad \text{—④}$$

↳ max. difference

$0.\underbrace{xx\ldots x}\,\underbrace{(\;)}_{\frac{1}{2^{l(x)+1}}}\,\underbrace{(\;)}_{\frac{1}{2^{l(x)+2}}}\ldots$

$$\Rightarrow \quad \frac{1}{2^{l(x)+1}}\left[1+\frac{1}{2}+\ldots\right] = \frac{1}{2^{l(x)+1}}\left[\frac{1}{1-1/2}\right]$$

sum of infinite geometric series

$$= \frac{1}{2^{l(x)+1}} \times 2$$

Consider, $\frac{1}{2^{l(x)}} = \frac{1}{2^{\lceil \log_2 1/p(x) \rceil + 1}}$

$$= \frac{1}{2.2^{\lceil \log_2 1/p(x) \rceil}} \leq \frac{1}{2.2^{\log_2 \frac{1}{p(x)}}}$$

$$= \frac{1}{2.\frac{1}{p(x)}} = \frac{p(x)}{2} \quad \text{—⑤}$$

Consider, $\bar{T}_x(x) - F_x(x-1) = \frac{p(x)}{2} \geqslant \frac{1}{2^{l(x)}} > \bar{T}_x(x) - \left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)}$

$$\Downarrow$$

$$\cancel{\bar{T}_x(x)} - F_x(x-1) > \cancel{\bar{T}_x(x)} - \left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)}$$

$$\left\lfloor \bar{T}_x(x) \right\rfloor_{l(x)} > F_x(x-1) \quad \text{—⑥}$$

Truncated tag value is still a member of the tag interval and hence it is a unique identifier of the given sequence.

The code that we designed is uniquely decodable?

↳ The code that we constructed is a prefix code.

should be proved ↳ No codeword is a prefix of any other codeword.

Let 'x' and 'y' be two distinct sequences of same length.

$$\bar{T}_x(x) \in \left[F_x(x-1)\ F_x(x)\right)$$

$\bar{T}_x(y) \in [F_x(y-1)\ \ F_x(y))$

Since 'x' and 'y' are distinct we have,

$[F_x(x-1)\ \ F_x(x)) \cap [F_x(y-1)\ \ F_x(y)) = \emptyset$ (empty set)

proof of contradiction: Assume that $\lfloor \bar{T}_x(x) \rfloor_{l(x)}$ (A) is a prefix of $\lfloor \bar{T}_x(y) \rfloor_{l(y)}$ (B)

That is A is a prefix of B

$A = a_1 \cdots a_m$, $B = b_1 b_2 \cdots b_n$ $\quad n \geq m$ $\quad l(y)\ \ l(x)$

$B = a_1 \cdots a_m\ b_{m+1}\ b_n$

$A \in [F_x(x-1)\ \ F_x(x))$, $B \in [F_x(y-1)\ \ F_x(y))$

$B \leq A + \frac{1}{2^{l(x)}}$ $\qquad A = a_1 a_2 \cdots a_m$

$A' = a_1 a_2 \cdots a_m 111 \cdots 1$

$A' = A + \frac{1}{2^{l(x)}}$

If I show $A + \frac{1}{2^{l(x)}} < F_x(x)$, $F_x(x-1) \leq A + \frac{1}{2^{l(x)}} < F_x(x)$

Consider, $F_x(x) - \bar{T}_x(x) = \frac{P(x)}{2} > \frac{1}{2^{l(x)}}$

$\bar{T}_x(x) + \frac{1}{2^{l(x)}} < \frac{P(x)}{2} + F_x(x) = F_x(x)$

$\bar{T}_x(x) + \frac{1}{2^{l(x)}} < F_x(x) \Rightarrow \lfloor \bar{T}_x(x) \rfloor_{l(x)} + \frac{1}{2^{l(x)}} < F_x(x)$

27.02.2020

Arithmetic code

How efficient is it?

A = alphabet of the original source

$A^{(m)}$ = alphabet of the source where in sequences of length 'm' of the original alphabet.

$\bar{l}_{A^{(m)}}$ – Average length of sequences of length 'm' in arithmetic code

$H(X^{(m)})$ – entropy of the source that puts out letters from the alphabet, $A^{(m)}$

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

$A^{(2)} = \{a_1a_1, a_1a_2, a_1a_3, a_2a_1, a_2a_2, a_2a_3, a_3a_1, a_3a_2, a_3a_3\}$

$$H(X^{(m)}) \leq \bar{l}_{A^{(m)}} < H(X^{(m)}) + 2$$

Consider,

$$\bar{l}_{A^{(m)}} = \sum_{x \in A^{(m)}} p(x) \left(\left\lceil \log_2 \frac{1}{p(x)} \right\rceil + 1 \right)$$

$$\leq \sum_{x \in A^{(m)}} p(x) \left[\log_2 \frac{1}{p(x)} + 1 + 1 \right]$$

$$= \sum_{x \in A^{(m)}} p(x) \log_2 \frac{1}{p(x)} + 2 \underbrace{\sum_{x \in A^{(m)}} p(x)}_{=1}$$

$$\leq H(X^{(m)}) + 2$$

$$H(X^{(m)}) = m \cdot H(x)$$

$$\boxed{H(X^{(m)}) \leq \bar{l}_{A^{(m)}} \leq H(X^{(m)}) + 2}$$

$$m \cdot H(x) \leq \bar{l}_{A^{(m)}} \leq m \cdot H(x) + 2$$

$$H(x) \leq \frac{\bar{l}_{A^{(m)}}}{m} \leq H(x) + \frac{2}{m}$$

Extended huffman code

$$H(x) \leq \frac{\bar{l}^{(m)}}{m} \leq H(x) + \frac{1}{m}$$

In theory, we can find the code to any lengths of sequences but practically its not possible.

How to overcome finite precision effects?

Also we want to have incremental encoding

1. Tag interval is confined to the lower half of the unit interval.
2. Tag interval is confined to the upper half of the unit interval.
3. Tag interval straddles the mid-point of the unit interval.

Suppose, tag interval $\leq [0 \ 0.5)$ (or) tag interval $\leq [0.5 \ 1.0)$

$l^{(n)} \geq 0 \quad u^{(n)} < 0.5$

$\Rightarrow 0 \leq l^{(n)} < 0.5$

$l^{(n)} < u^{(n)} < 0.5$

MSB of $l^{(n)}$ & $u^{(n)}$ are both '0'.

$$[l^{(n+1)} \ u^{(n+1)}) \leq [l^{(n)} \ u^{(n)}) \leq [0 \ 0.5)$$

Once the tag interval is confined to the lower-half of the unit interval, it remains so for all the succeeding tag intervals and hence the MSB of tag is '0'.

Similarly, argument is true when the tag interval is confined to the upper half of the unit interval. Here, the MSB of tag is '1'.

Once, the tag interval is continued to the lower half of the unit interval, we can forgo the upper half of the unit interval from consideration.

$E_1(x) = 2x$

$[0 \ 0.5) \longrightarrow [0 \ 1.0)$

$E_2(x) = 2(x - 0.5)$

$[0.5 \ 1.0) \longrightarrow [0 \ 1.0)$

Eg: $A = \{a_1, a_2, a_3\}$

$P(a_1) = 0.8$, $P(a_2) = 0.02$, $P(a_3) = 0.18$

Sequence $a_1 a_3 a_2 a_1$

$F_X(0) = 0$, $F_X(1) = 0.8$, $F_X(2) = 0.82$, $F_X(3) = 1.0$

$l^{(0)} = 0$, $u^{(0)} = 1$

$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) \cdot F_X(0) = 0$

$u^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) \cdot F_X(1) = 0.8$

Is the tag interval confined to the lower half or upper half of the interval?

$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) \cdot F_X(2) = 0.8 \times 0.82 = 0.656$

$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) \cdot F_X(3) = 0.8 \times 1 = 0.8$

Send 1 Apply E_2

$l^{(2)} = 2(0.656 - 0.5) = 0.312$

$u^{(2)} = 2(0.8 - 0.5) = 0.6$

$l^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)}) \cdot F_X(1) = 0.656 + 0.288 \times 0.8 = 0.5424$

$u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)}) \cdot F_X(2)$

$= 0.656 + 0.288 \times 0.82 = 0.59816$

SEND 1 Apply E_2

$l^{(3)} = 2(0.5424 - 0.5) = 0.0848$

$u^{(3)} = 2(0.59816 - 0.5) = 0.09632$

SEND 0 Apply E_1

$l^{(3)} = 2 \times 0.0848 = 0.1696$

$u^{(3)} = 2 \times 0.09632 = 0.19264$

SEND 0 Apply E_1

$l^{(3)} = 2 \times 0.1656 = 0.3392$

$u^{(3)} = 2 \times 0.19264 = 0.38528$

SEND 0 Apply E_1

$l^{(3)} = 2 \times 0.3392 = 0.6784$

$u^{(3)} = 2 \times 0.38528 = 0.77056$

SEND 1 Apply E_2

$l^{(3)} = 2(0.6784 - 0.5) = 0.3568$

$u^{(3)} = 2(0.77056 - 0.5) = 0.5412$

$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)}) \cdot F_X(0) = 0.3568$

$u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)}) \cdot F_X(1) = 0.504256$

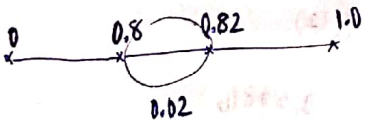
$l^{(4)} = 0.3568 \quad | \quad u^{(4)} = 0.504256$

$\rightarrow 0.5$

11000110 0

Tag $= \frac{1}{2} + \frac{1}{4} + \frac{1}{64} + \frac{1}{128} = \frac{64+32+2+1}{128}$

$= \frac{99}{128} = 0.7734375$



Decoder should collect atleast 'k' bits before starting to decode, where 'k' be such that,

$2^k \leq 0.02$

0.5, 0.25, 0.125, 0.0625, 0.03125, 0.0105125

11000 110 . . . 0 $\quad k \geq 6$

Tag $= .110001 = \frac{49}{64} = 0.765625$

$l^{(0)} = 0, \; u^{(0)} = 1$

$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) F_X(x-1) = F_X(x-1)$

$u^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)}) F_X(x) = F_X(x)$

What is x such that,

$$F_X(x-1) \leq 0.765625 < F_X(x)$$

$x=1$, $0 \leq 0.765625 < 0.8$ ✓

$x=2$, $0.8 \leq 0.765625 < 0.82$ ✗

$x=3$, $0.82 \leq 0.765625 < 1.0$ ✗

$l^{(1)}=0$, $u^{(1)}=0.8$

$$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) F_X(x-1) = 0 + (0.8-0) F_X(x-1) = 0.8 . F_X(x-1)$$

$$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)}) . F_X(x) = 0.8 F_X(x)$$

What is x such that,

$$0.8 F_X(x-1) \leq 0.765625 < 0.8 F_X(x)$$

$x=1$, $0 \leq 0.765625 < 0.64$ ✗

$x=2$, $0.64 \leq 0.765625 < 0.656$ ✗

$x=3$, $0.656 \leq 0.765625 < 0.8$ ✓

Tag = ~~X000~~ 11 $= \frac{32+2+1}{64} = \frac{35}{64} = 0.546875$

$l^{(2)}=0.656$ $u^{(2)}=0.8$

Apply E_2 mapping,

$l^{(2)} = 2(0.656-0.5) = 0.312$

$u^{(2)} = 2(0.8-0.5) = 0.6$

$$l^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)}) . F_X(x-1) = 0.312 + 0.288 F_X(x-1)$$

$$u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)}) . F_X(x) = 0.312 + 0.288 F_X(x)$$

What is x such that,

$$0.312 + 0.288 F_X(x-1) \leq 0.546875 < 0.312 + 0.288 F_X(x)$$

$x=1$, $0.312 \leq 0.546875 < 0.312 + 0.288 \times 0.8 = 0.5426$ ✗

$x=2$, $0.5424 \leq 0.546875 < 0.312 + 0.288 \times 0.82 = 0.54816$ ✓

$x=3$,

$0.59816 \leq 0.546875 < 0.6$ x

$Tag = \not{0}00110 = \frac{3}{32} = 0.09375$

$l^{(3)} = 0.5424 \quad u^{(3)} = 0.59816$

$l^{(3)} = 2(0.5424 - 0.5) = 0.0848$

$u^{(3)} = 2(0.59816 - 0.5) = 0.9632$

$l^{(3)} = 0.0848$, $u^{(3)} = 0.09632$

$l^{(3)} = 2 \times 0.0848 = 0.1696$

$u^{(3)} = 2 \times 0.9632 = 0.19264$

$Tag = \not{0}01100 = \frac{3}{16}$

$l^{(3)} = 0.1696$, $u^{(3)} = 0.19264$

$l^{(3)} = 2 \times 0.1696 = 0.3392$

$u^{(3)} = 2 \times 0.19264 = 0.38528$

$Tag = \not{0}11000 = \frac{3}{8}$

$l^{(3)} = 0.3392$, $u^{(3)} = 0.38528$

$l^{(3)} = 2 \times 0.3392 = 0.6784$

$u^{(3)} = 2 \times 0.38528 = 0.77056$

$l^{(3)} = 0.6784 \quad u^{(3)} = 0.77056$

$l^{(3)} = 2(0.6784 - 0.5) = 0.3568$

$u^{(3)} = 2(0.77056 - 0.5) = 0.54112$

$Tag = \not{1}0000 = 3/4$

$l^{(3)} = 0.3568$, $u^{(3)} = 0.54112$

$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)}) F_X(x-1)$

$= 0.3568 + 0.1843 \times F_X(x-1)$

$u^{(4)} = 0.3568 + 0.1843 \times F_X(x)$

What is x such that?

$$0.3568 + 0.1843 \times F_X(x-1) \leq 0.5 < 0.3568 + 0.1843 F_X(x)$$

$x=1$, $0.3568 \leq 0.5 < 0.3568 + 0.1843 \times 0.8 = 0.50424$

$x=2$, $0.50424 < 0.5 < 0.3568 + 0.1843 \times 0.82 = x$

$x=3$

×

If $0.25 \leq l < u < 0.75 \longrightarrow E_3$ mapping

$$E_3(x) = 2(x-0.25)$$

Don't send anything, but record the fact that we have applied E_3 mapping.

Suppose immediately after appling E_3 mapping, we have come across E_2 mapping.

$u < 0.75 \quad l \geq 0.25$ $\qquad$ $l \geq 0.5 \quad u < 1.0$

E_3 mapping

$$\left.\begin{array}{l} u < 2 \times 0.5 \\ l > 0 \end{array}\right] \text{—①}$$

$$l \geq 0.5 \quad u < 1.0 \text{]} \text{—②}$$

From ① and ②, $l \geq 0.5$

$$2(l-0.25) \geq 0.5$$

$$2l - 0.5 \geq 0.5$$

$$l \geq 0.5$$

$l \geq 0.5 \quad u < 0.75$

E_2 mapping

$$2(0.5-0.5) \geq 0$$

$$2(0.75-0.5) < 0.5$$

$$E_3 E_2 = E_2 E_1$$