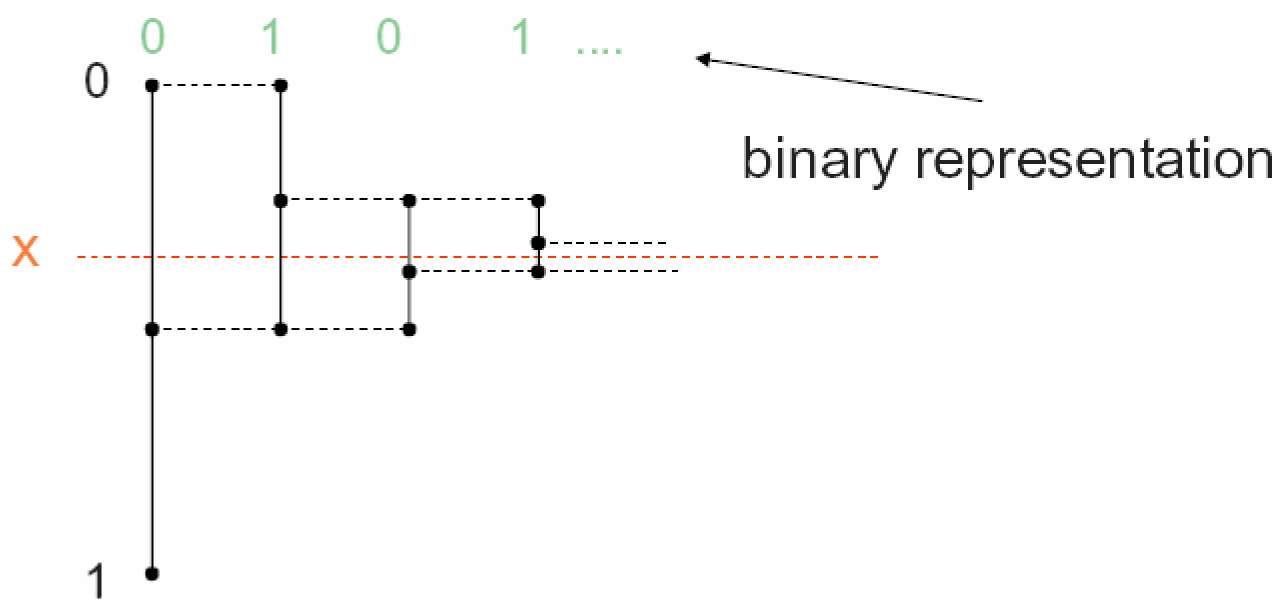
Lecture 8:
Arithmetic Coding

Thinh Nguyen
Oregon State University

Representation of Real Number in Binary

- Any real number x in the interval [0,1) can be represented in binary as $.b_1b_2...$ where b_i is a bit.



Real-to-Binary Conversion Algorithm

```
L := 0; R :=1; i := 1
while x > L *
    if x < (L+R)/2 then b_i := 0 ; R := (L+R)/2;
    if x ≥ (L+R)/2 then b_i := 1 ; L := (L+R)/2;
    i := i + 1
end{while}
b_j := 0 for all j ≥ i
```

* Invariant: x is always in the interval [L,R)

Arithmetic Coding

- Basic idea in arithmetic coding (Shannon-Fano-Elias):

  - Represent each string x of length n by a unique interval [L,R) in [0,1).

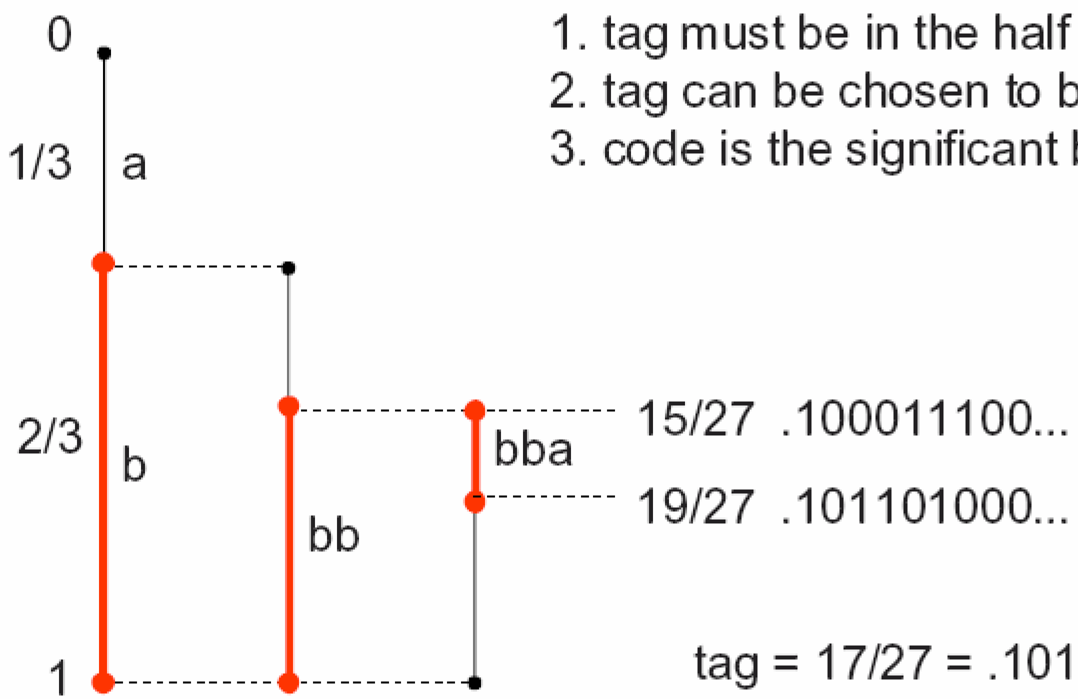
  - The width r-l of the interval [L,R) represents the probability of x occurring.

  - The interval [L,R) can itself be represented by any number, called a tag, within the half open interval.

  - The k significant bits of the tag $.t_1t_2t_3...$ is the code of x. That is, . $.t_1t_2t_3...t_k000...$ is in the interval [L,R).

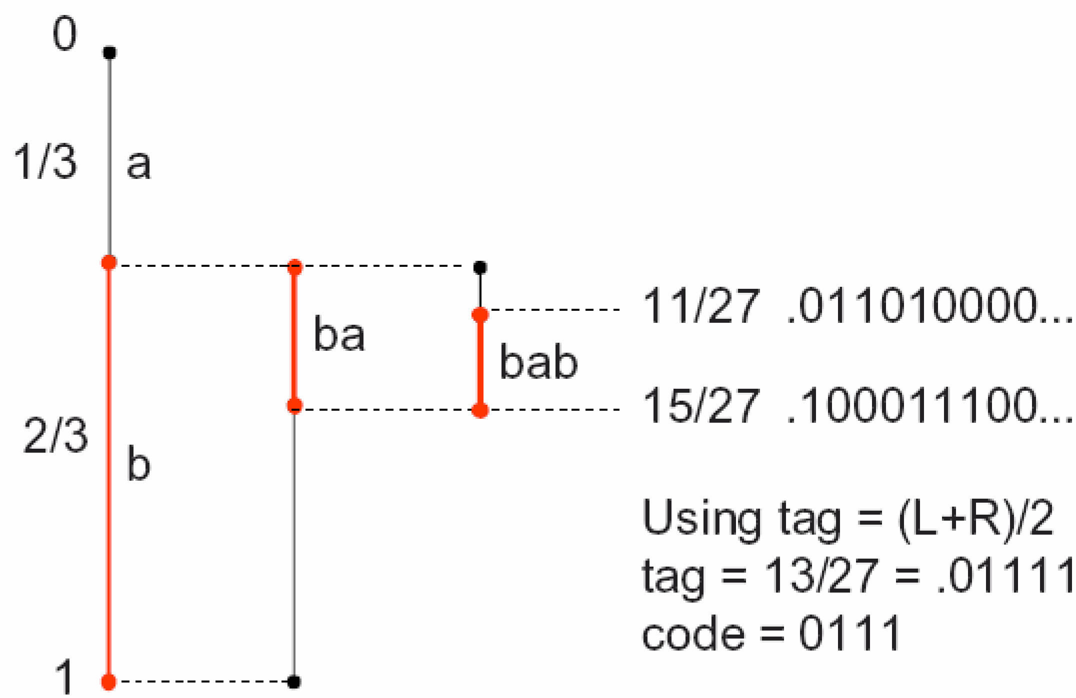
Example of Arithmetic Coding

1. tag must be in the half open interval.
2. tag can be chosen to be (L+R)/2.
3. code is the significant bits of the tag.



tag = 17/27 = .101000010...
code = 101

Some Tags are better than others

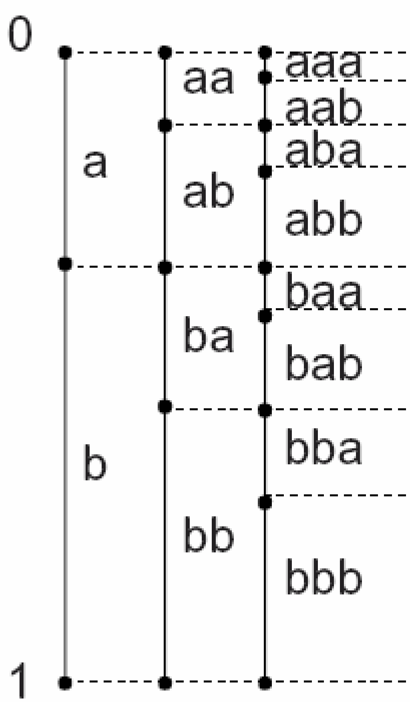


Using tag = (L+R)/2
tag = 13/27 = .011110110...
code = 0111

Alternative tag = **14/27** = .100001001...
code = 1

Examples

- P(a) = 1/3, P(b) = 2/3.



0/27	.000000000...
1/27	.000010010...
3/27	.000111000...
5/27	.001011110...
9/27	.010101010...
11/27	.011010000...
15/27	.100011100...
19/27	.101101000...
27/27	.111111111...

tag = (L+R)/2	code	
.000001001...	0	aaa
.000100110...	0001	aab
.001001100...	001	aba
.010000101...	01	abb
.010111110...	01011	baa
.011110111...	0111	bab
.101000010...	101	bba
.110110100...	11	bbb

.95 bits/symbol

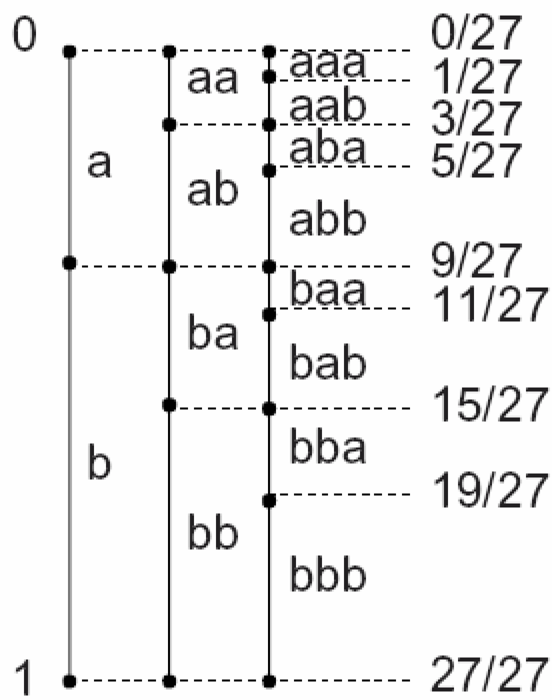
.92 entropy lower bound

Code Generation from Tags

- If binary tag is $.t_1t_2t_3... = (L+R)/2$ in $[L,R)$ then we want to choose k to form the code $t_1t_2...t_k$.
- Short code:
  - choose k to be as small as possible so that $L \leq .t_1t_2...t_k000... < R$.
- Guaranteed code:
  - choose $k = \lceil \log_2 (1/(R-L)) \rceil + 1$
  - $L \leq .t_1t_2...t_kb_1b_2b_3... < R$ for any bits $b_1b_2b_3...$
  - for fixed length strings provides a good prefix code.
  - example: [.000000000..., .000010010...), tag = .000001001...
    Short code: 0
    Guaranteed code: 000001

Guaranteed Code Example

- P(a) = 1/3, P(b) = 2/3.



tag = (L+R)/2	short code	Prefix code	
.000001001...	0	0000	aaa
.000100110...	0001	0001	aab
.001001100...	001	001	aba
.010000101...	01	0100	abb
.010111110...	01011	01011	baa
.011110111...	0111	0111	bab
.101000010...	101	101	bba
.110110100...	11	11	bbb

Arithmetic Coding Algorithm

$$C(x_i) = P(x_0) + P(x_1) + \dots P(x_i)$$

Initialize L: = 0 and R: = 1;

For i = 1 to n do

W:= R – L;

L: = L + W*C(x_{i-1});

R := L + W*C(x_i);

T: = (L+R)/2;

Choose code for the tag

Example

P(A) = ¼, P(b) = ½ , P(c) = ¼

C(a) = ¼, C(b) = ¾, C(c) = 1

abca

W := R - L;
L := L + W C(x);
R := L + W P(x)

symbol	W	L	R
		0	1
a	1	0	1/4
b	1/4	1/16	3/16
c	1/8	5/32	6/32
a	1/32	5/32	21/128

tag = (5/32 + 21/128)/2 = 41/256 = .001010010...
L = .001010000...
R = .001010100...
code = 00101
prefix code = 00101001

Example

P(A) = ¼, P(b) = ½ , P(c) = ¼

C(a) = ¼, C(b) = ¾, C(c) = 1

bbbb

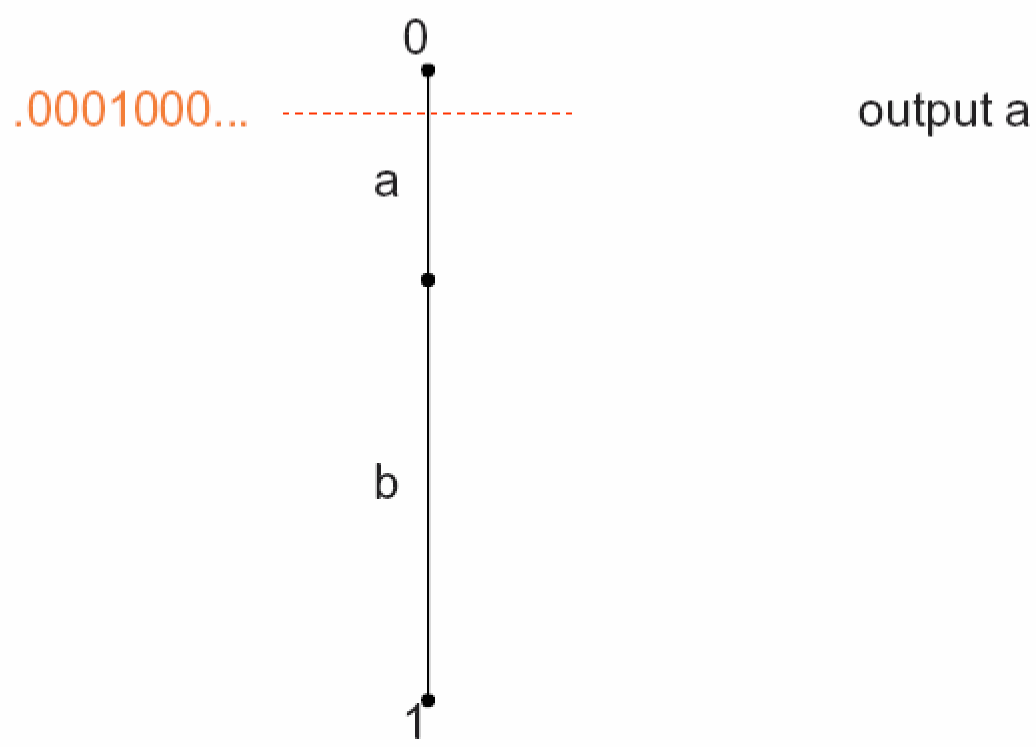
```
W := R - L;
L := L + W C(x);
R := L + W P(x)
```

symbol	W	L	R
		0	1
b	1		
b			
b			
b			

tag =
L =
R =
code =
prefix code =

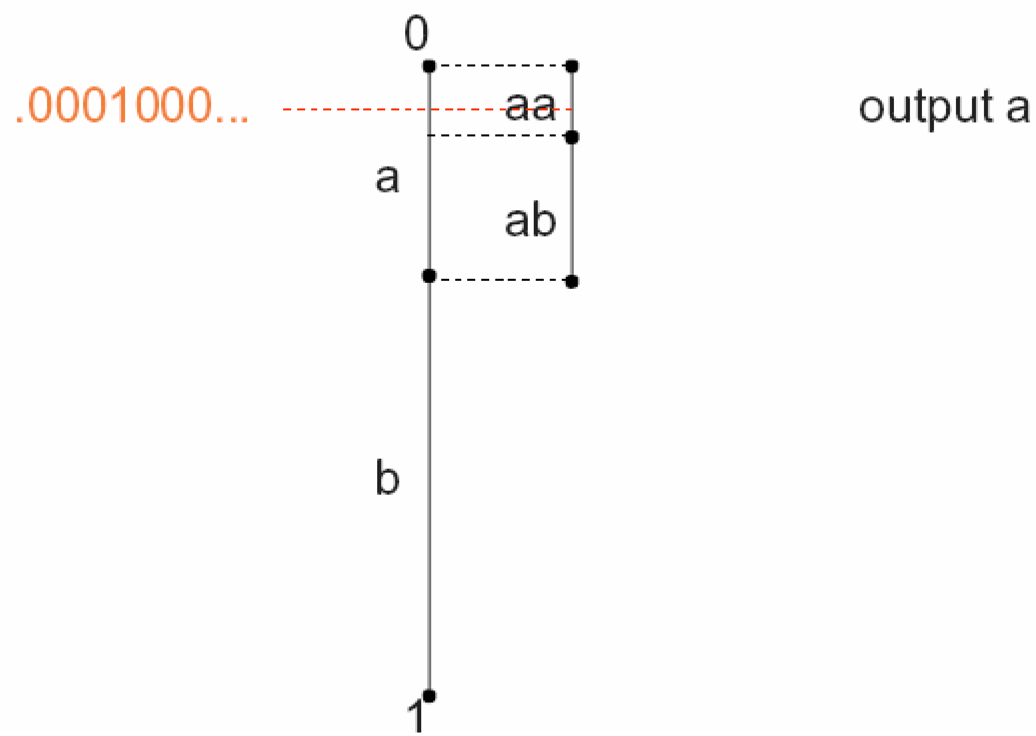
Decoding

- Assume the length is known to be 3.
- 0001 which converts to the tag .0001000...



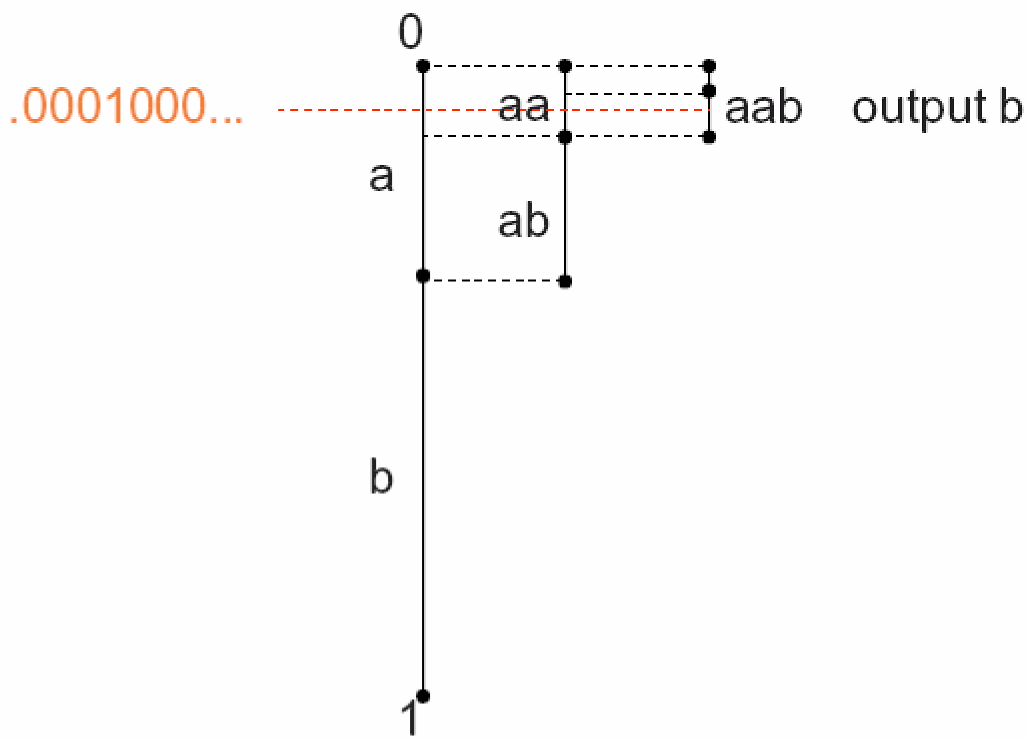
Decoding

- Assume the length is known to be 3.
- 0001 which converts to the tag .0001000...



Decoding

- Assume the length is known to be 3.
- 0001 which converts to the tag .0001000...



Arithmetic Decoding Algorithm

$C(x_i) = P(x_0) + P(x_1) + \dots P(x_i)$

Decode $b_1 b_2 \dots b_m$, the number of symbols in n

```
Initialize L:= 0 and R := 1;
t:= .b1b2...bm
For i = 1 to n do
    W := R-L;
    Find j such that L + W*C(x_{j-1}) <= t < L + W*C(x_j)
    Output x_j;
    L := L + W*C(x_{j-1});
    R := L + W*C(x_j);
```

Decoding Example

P(a) = ¼, P(b) = ½ , P(c) = ¼

C(a) = 0, C(b) = 1/4, C(c) = 3/4

- 00101

tag = .00101000... = 5/32

W	L	R	output
	0	1	
1	0	1/4	a
1/4	1/16	3/16	b
1/8	5/32	6/32	c
1/32	5/32	21/128	a

Decoding Issues

- There are two ways for the decoder to know when to stop decoding.

1. Transmit the length of the string
2. Transmit a unique end of string symbol

Practical Arithmetic Coding

- Scaling:
  - By scaling we can keep L and R in a reasonable range of values so that W = R - L does not underflow.
  - The code can be produced progressively, not at the end.
  - Complicates decoding some.
- Integer arithmetic coding avoids floating point altogether.

Uniqueness and Efficiency of Arithmetic Code

- Uniqueness:

Proof:

- Efficiency:

Proof: