

Static Huffman Code

- Static 2-pass
 - Pass 1: read all input to compute Huffman code
 - Pass 2: re-read input and use code to compress
 - good method when
 - full input is provided, and
 - there are no time constraints
- Static 1-pass
 - Pass 1: compress input using a previously developed code

Applications Motivating *1-Pass* Huffman

VERY
LARGE
FILE

2 passes may be too slow

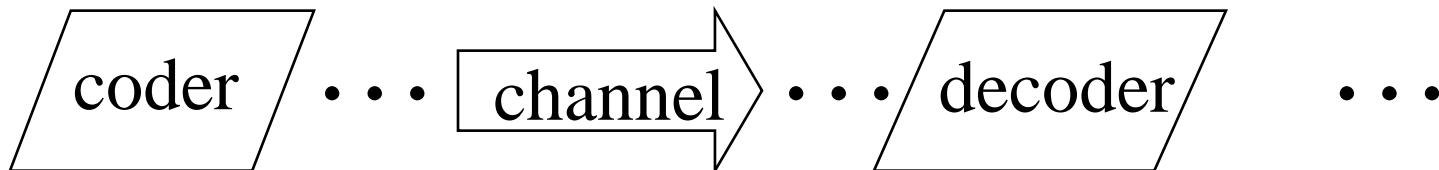
Real time

Source
(input)

Sender

Receiver

(output)



2 passes are not possible!

BUT ... what if input distribution is unknown?

1 pass *static* is not possible!

Adaptive Huffman Code

- 1-pass adaptive
 - build code and compress input simultaneously
 - code is built **dynamically** at both coder and decoder
 - code changes with each character encoded !

ENCODER

```
Initialize_model();
while ((c = getc (input)) != eof)
{
    encode (c, output);
    update_model (c);
}
```

DECODER

```
Initialize_model();
while ((c = decode (input)) != eof)
{
    putc (c, output);
    update_model (c);
}
```

**Encoder and decoder must use identical
Initialize_model and update_model routines**

Adaptive Huffman Code - encoding

Time = 0

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 0

Input: a

Output: 001100001

Time = 1

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 0
 (1) a 1

a

1

Time = 2

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

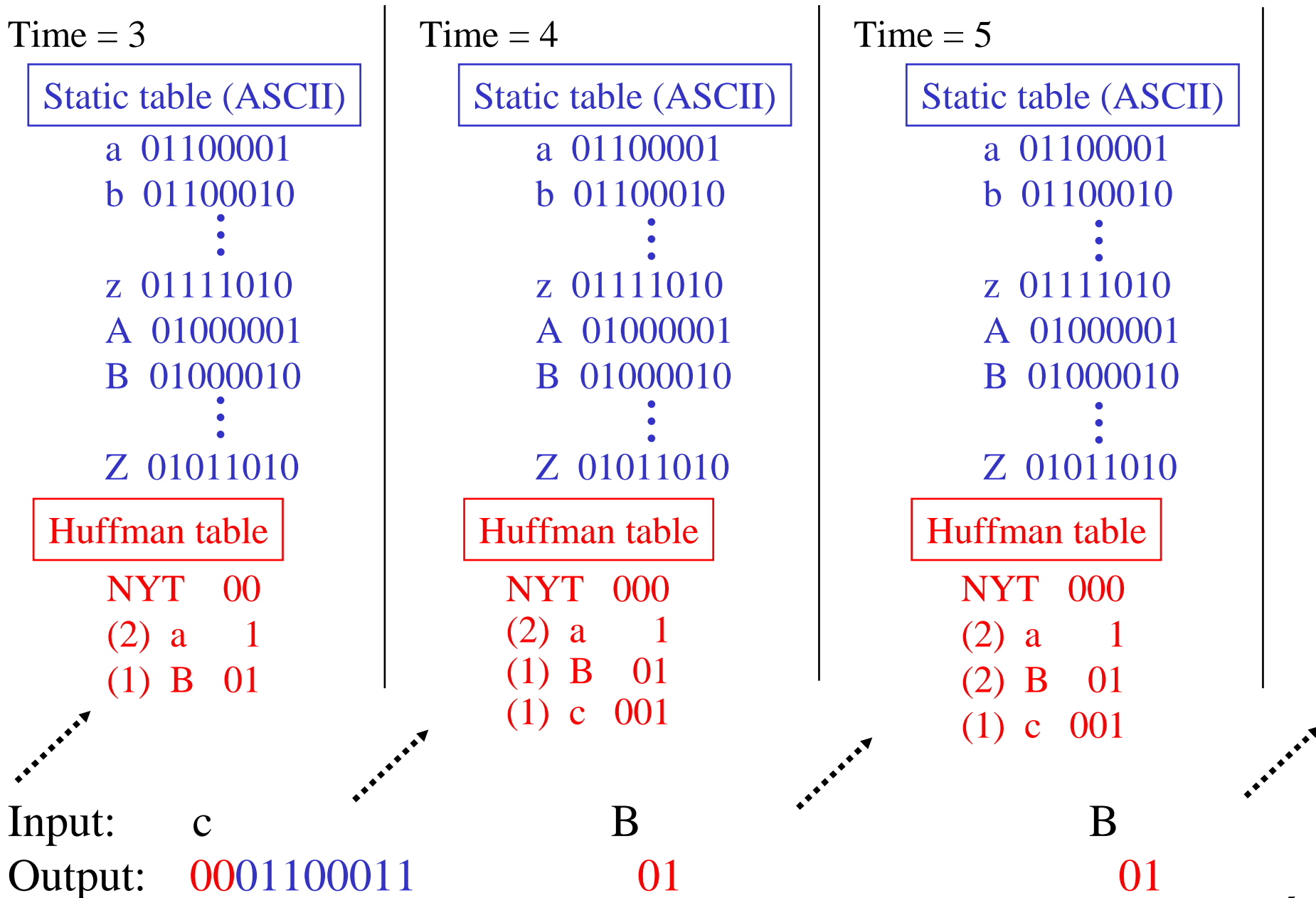
NYT 0
 (2) a 1

B

001000010

NYT = Not Yet in Table

Adaptive Huffman Code – encoding (cont'd)



Adaptive Huffman Code – encoding (cont'd)

Time = 6

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 01
 (3) B 1
 (1) c 001

Input: c

Output: 001

Time = 7

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 01
 (3) B 1
 (2) c 001

c

001

Time = 8

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 001
 (3) B 1
 (3) c 01

c

01

Adaptive Huffman Code – encoding (cont'd)

Time = 9

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 001
 (3) B 01
 (4) c 1

Time = n

~~Static table (ASCII)~~

Huffman table

~~NYT 0000000000000000~~
 (233) a 001
 (128) b 0110
 ⋮
 (12) z 011110100001
 (35) A 01010
 (17) B 01000010
 ⋮
 (2) Z 0101101000011

...

...

Input: c

Output: 1

Adaptive Huffman Code – encoding (cont'd)

Result:

a	a	B	c	B	B	c	c	c	c
001100001	1	001000010	0001100011	01	01	001	001	01	1

Adaptive Huffman Code - decoding

Time = 0

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 0

Input: 001100001
 Output: a

Time = 1

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 0
 (1) a 1

1
 a

Time = 2

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 0
 (2) a 1

001000010
 B

NYT = Not Yet in Table

Adaptive Huffman Code – decoding (cont'd)

Time = 3

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 00
 (2) a 1
 (1) B 01

Input: 0001100011
 Output: c

Time = 4

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 1
 (1) B 01
 (1) c 001

01
 B

Time = 5

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 1
 (2) B 01
 (1) c 001

01
 B

Adaptive Huffman Code – decoding (cont'd)

Time = 6

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 01
 (3) B 1
 (1) c 001

Input: 001

Output: c

Time = 7

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 01
 (3) B 1
 (2) c 001

Input: 001

Output: c

Time = 8

Static table (ASCII)

a 01100001
 b 01100010
 ⋮
 z 01111010
 A 01000001
 B 01000010
 ⋮
 Z 01011010

Huffman table

NYT 000
 (2) a 001
 (3) B 1
 (3) c 01

Input: 01

Output: c

Variable Length Codes (cont'd)

Summary

- disadvantages
 - less tolerant to bit errors
 - requires accurate knowledge of underlying distribution of characters
- advantage
 - on average, uses fewer bits/character