COMP9319 Web Data Compression and Search

LZW,
Adaptive Huffman,
(live lecture)

Note: LZW decoding

- There is one special case that the LZW decoding pseudocode presented is unable to handle.
- This is your exercise to find out in what situation that happens, and how to deal with it.
- I'll go through this at the live lecture.

LZW Notes
(Handling the special case)

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			

```
p = nil;    // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```


Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab

```
p = nil;    // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```


Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba

```
p = nil;    // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

7

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	a	256	258	aba

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

8

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	a	256	258	aba
a	b			

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

9

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	a	256	258	aba
a	b			
ab	a			

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

10

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	a	256	258	aba
a	b			
ab	a			
aba	b	258	259	abab

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

11

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	a	256	258	aba
a	b			
ab	a			
aba	b	258	259	abab
b	EOF	b		

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

12

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	**a**	256	ab
b	a	**b**	257	ba
a	b			
ab	a	**256**	258	aba
a	b			
ab	a			
aba	b	**258**	259	abab
b	EOF	**b**		

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```


Decoding

Input: ab<256><258>b

```
read(c);   // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

p	c	output	index	symbol
	a	a		

Dict [index]

Decoding

Input: ab<256><258>b

```
read(c);   // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

p	c	output	index	symbol
	a	a		
a	b	b	256	ab

Dict [index]

Decoding

Input: ab<256><258>b

```
read(c);   // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

p	c	output	index	symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba

Dict [index]

Decoding

Input: ab<256><258>b

```
read(c);   // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

p	c	output	index	symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba
<256>	<258>	⟵ WHAT???		

Dict [index]

Encoding

Input: abababab

p	c	output	index	symbol
nil	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	**a**	256	258	aba
a	b			
ab	a			
aba	b	258	259	abab
b	EOF	b		

Dict [index]

```
p = nil;   // p for prev char
while read(c):
    if pc ∈ Dict:
        p = pc;
    else:
        add pc to Dict;
        output code(p);
        p = c;
```

Decoding

Input: ab<256><258>b

```
read(c);  // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

Dict [index]

p	c	output	index	symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba
<256>	<258>	aba	258	aba

Decoding

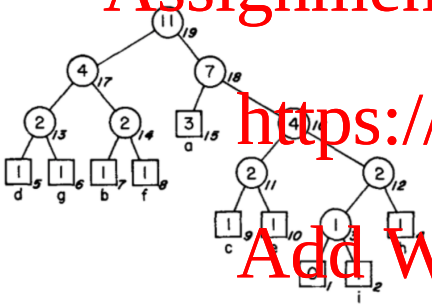
Input: ab<256><258>b

```
read(c);  // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

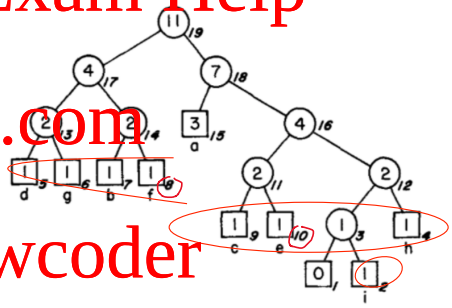
Dict [index]

p	c	output	index	symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba
<256>	<258>	aba	258	aba
<258>	b	b	259	abab

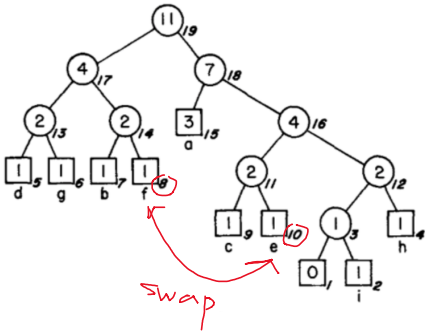
Adaptive Huffman (FGK)



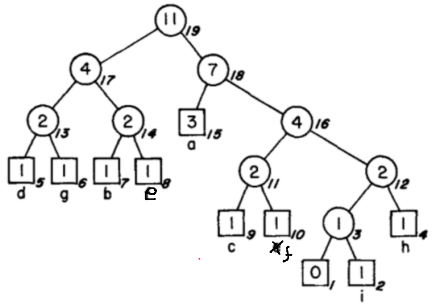
Adaptive Huffman (FGK)



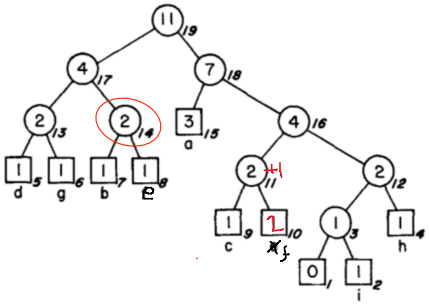
Adaptive Huffman (FGK)



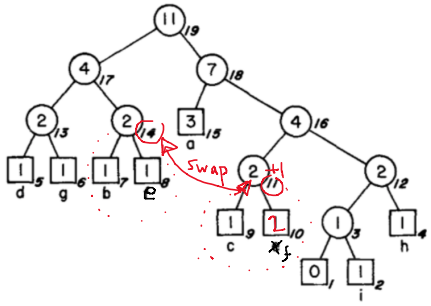
Adaptive Huffman (FGK)



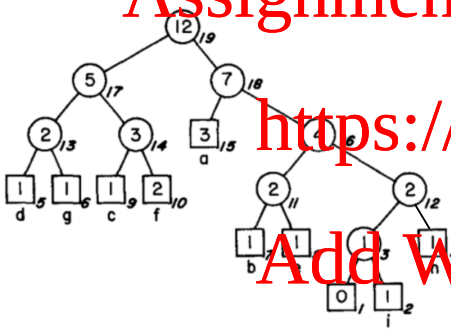
Adaptive Huffman (FGK)



Adaptive Huffman (FGK)



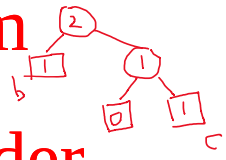
Adaptive Huffman (FGK): when f is inserted



Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

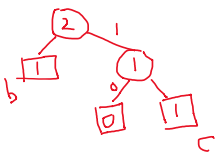


01100010 001100011

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

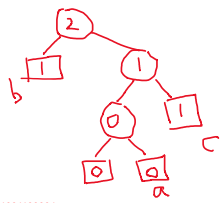


01100010 001100011 1001100001

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

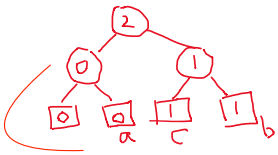


01100010 001100011 1001100001

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

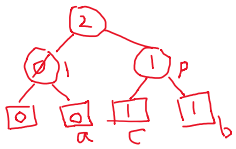


01100010 001100011 1001100001

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



01100010 001100011 1001100001

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

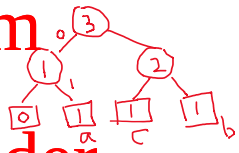
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

01100010 001100011 1001100001

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

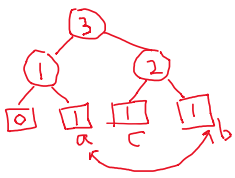


01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

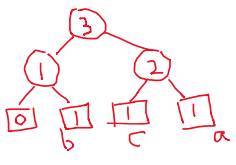


01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

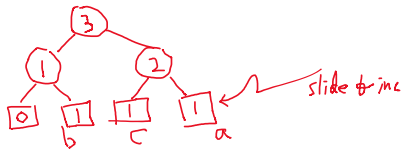


01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

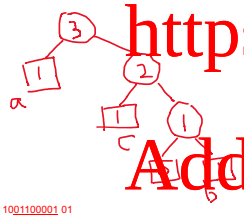
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

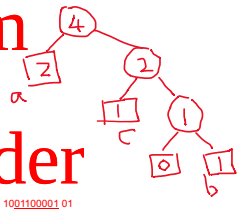


01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

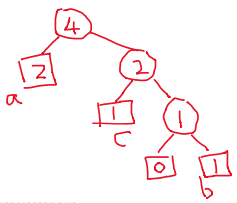


01100010 001100011 1001100001 01

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

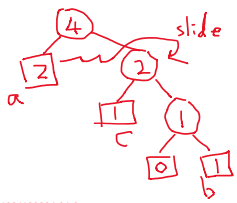


01100010 001100011 1001100001 01 0

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

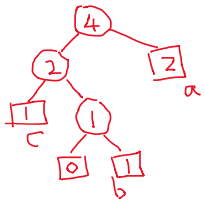


01100010 001100011 1001100001 01 0

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

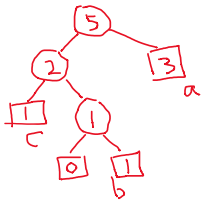


01100010 001100011 1001100001 01 0

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

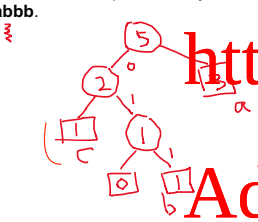


01100010 001100011 1001100001 01 0

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

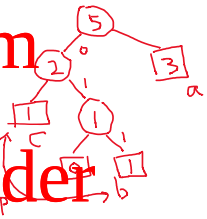


01100010 001100011 1001100001 01 0 011

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

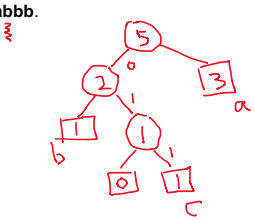


01100010 001100011 1001100001 01 0 011

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

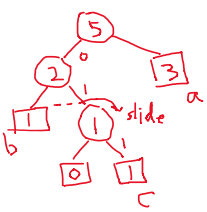


01100010 001100011 1001100001 01 0 011

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

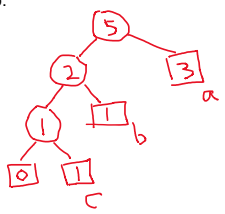


01100010 001100011 1001100001 01 0 011

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

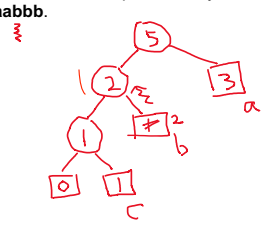
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



Adaptive Huffman (Ex2, Q2)

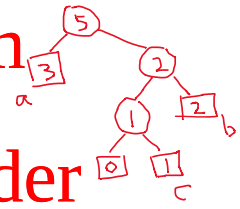
The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

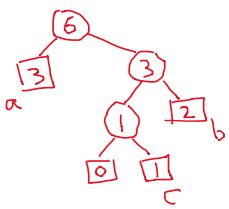
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

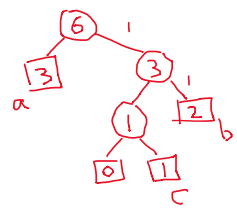
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

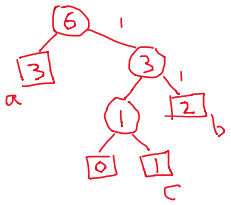
Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

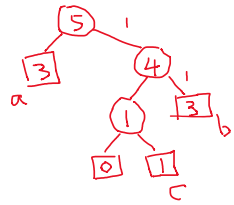


01100010 001100011 1001100001 01 0 011 11

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.

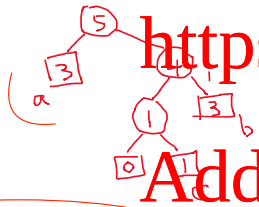


01100010 001100011 1001100001 01 0 011 11

Adaptive Huffman (Ex2, Q2)

The initial coding before any transmission is: a=01100001, b=01100010, c=01100011.

Derive the encoded bitstream produced by the encoder for the string **bcaaabbb**.



01100010 001100011 1001100001 01 0 011 11 11