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

Dict [index]

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

Dict [index]

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

Dict [index]

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

Dict [index]

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

```
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

Dict [index]

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

```
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

Dict [index]

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			

```
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

Dict [index]

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			

```
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

Dict [index]

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

```
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

Dict [index]

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		

```
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

Dict [index]

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**		

```
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	Dict [index] index	symbol
	a	a		

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	Dict [index] index	Dict [index] symbol
	a	a		
a	b	b	256	ab

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	Dict [index] index	Dict [index] symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba

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	Dict [index] index	Dict [index] symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba
<256>	<258>	WHAT???		

Encoding

Input: abababab

Dict [index]

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		

```
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	Dict [index] index	Dict [index] symbol
	a	a		
a	b	b	256	ab
b	<256>	ab	257	ba
<256>	<258>	**ab**a	258	aba

Decoding

Input: ab<256><258>b

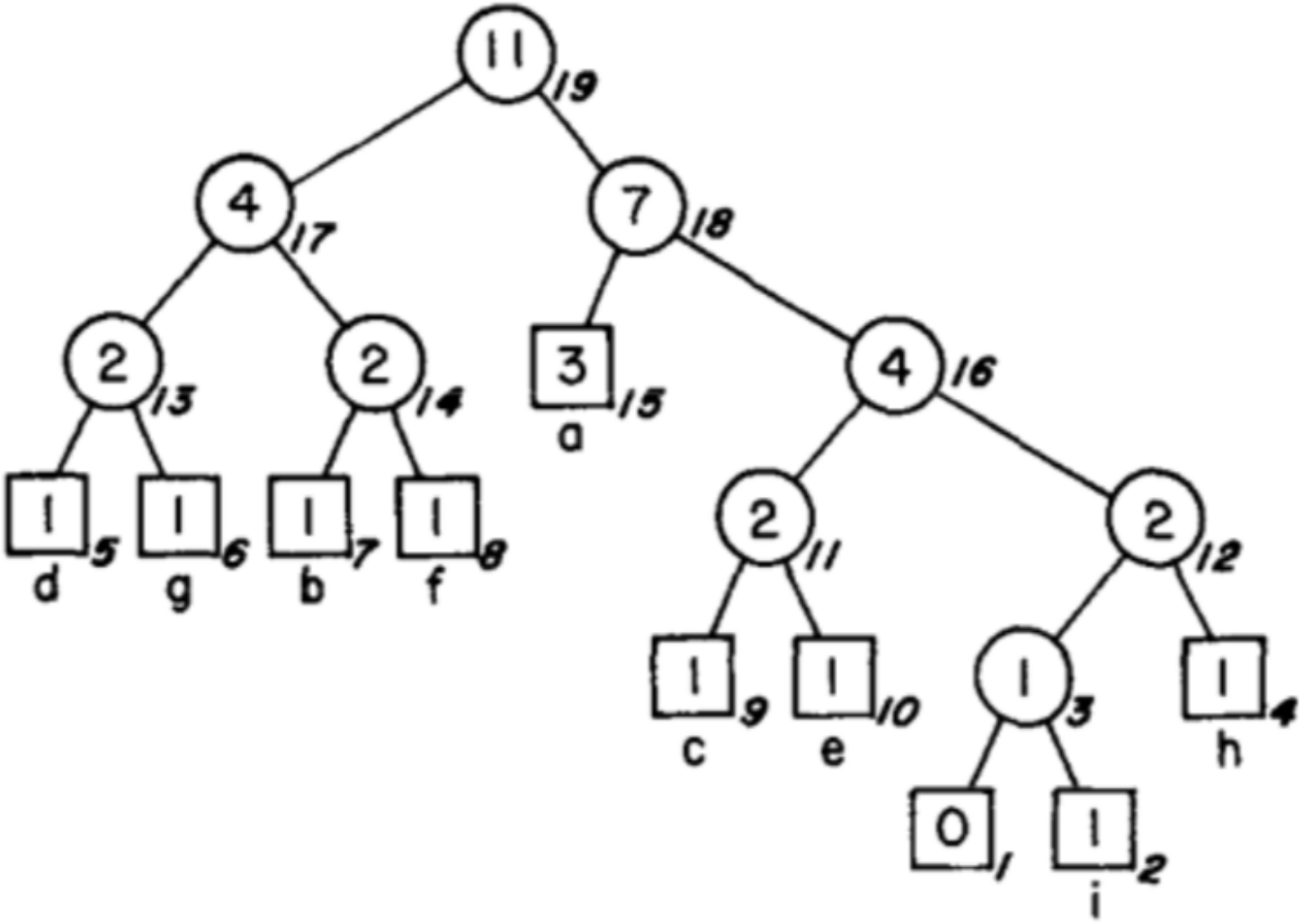
```
read(c);   // c is likely to be > 8 bits
output c;
p = c;
while read(c):
    output Dict[c];
    add p + Dict[c][0] to Dict;
    p = Dict[c];
```

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

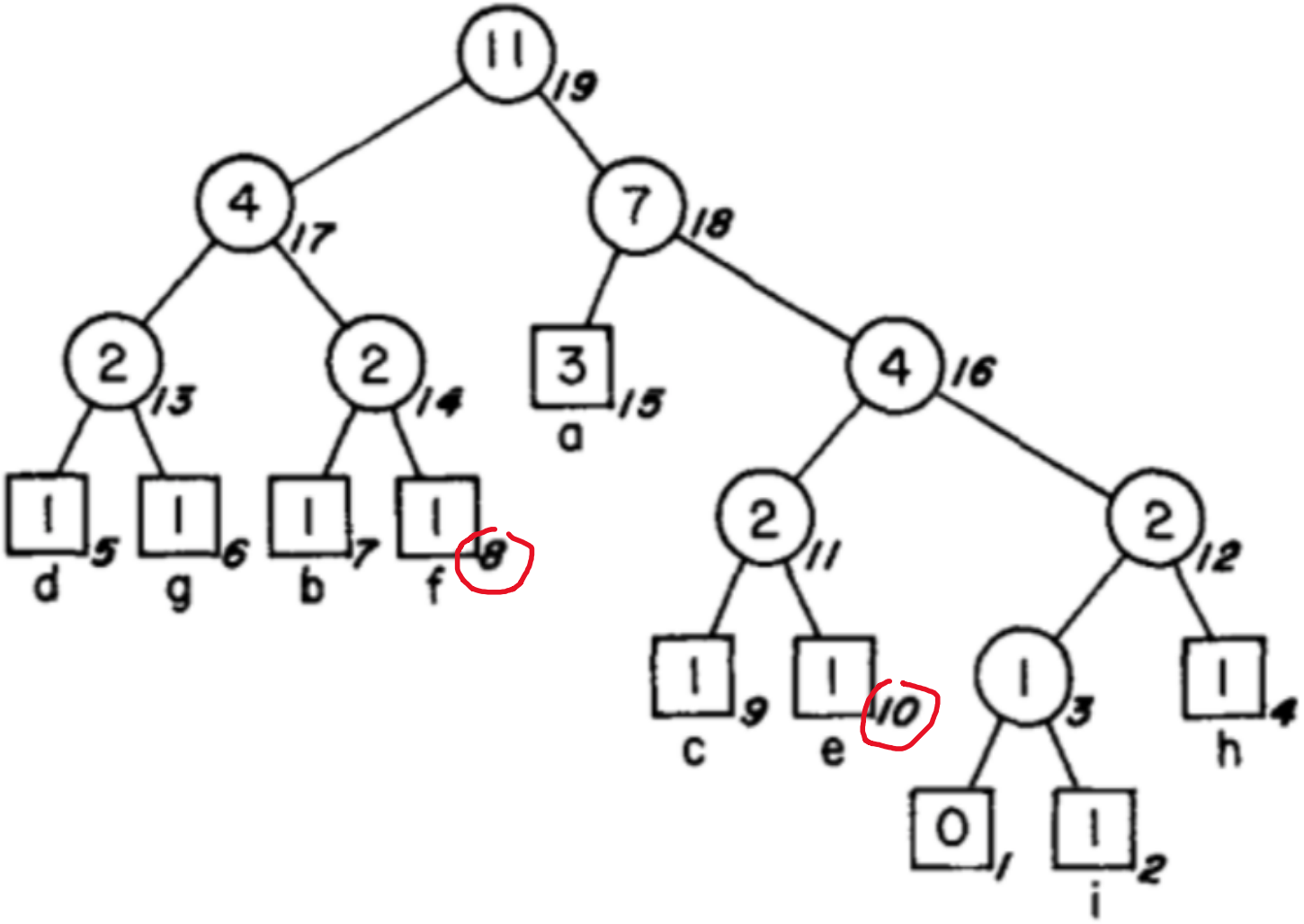
The FGK example notes

(that I said I would address it later after the lecture)

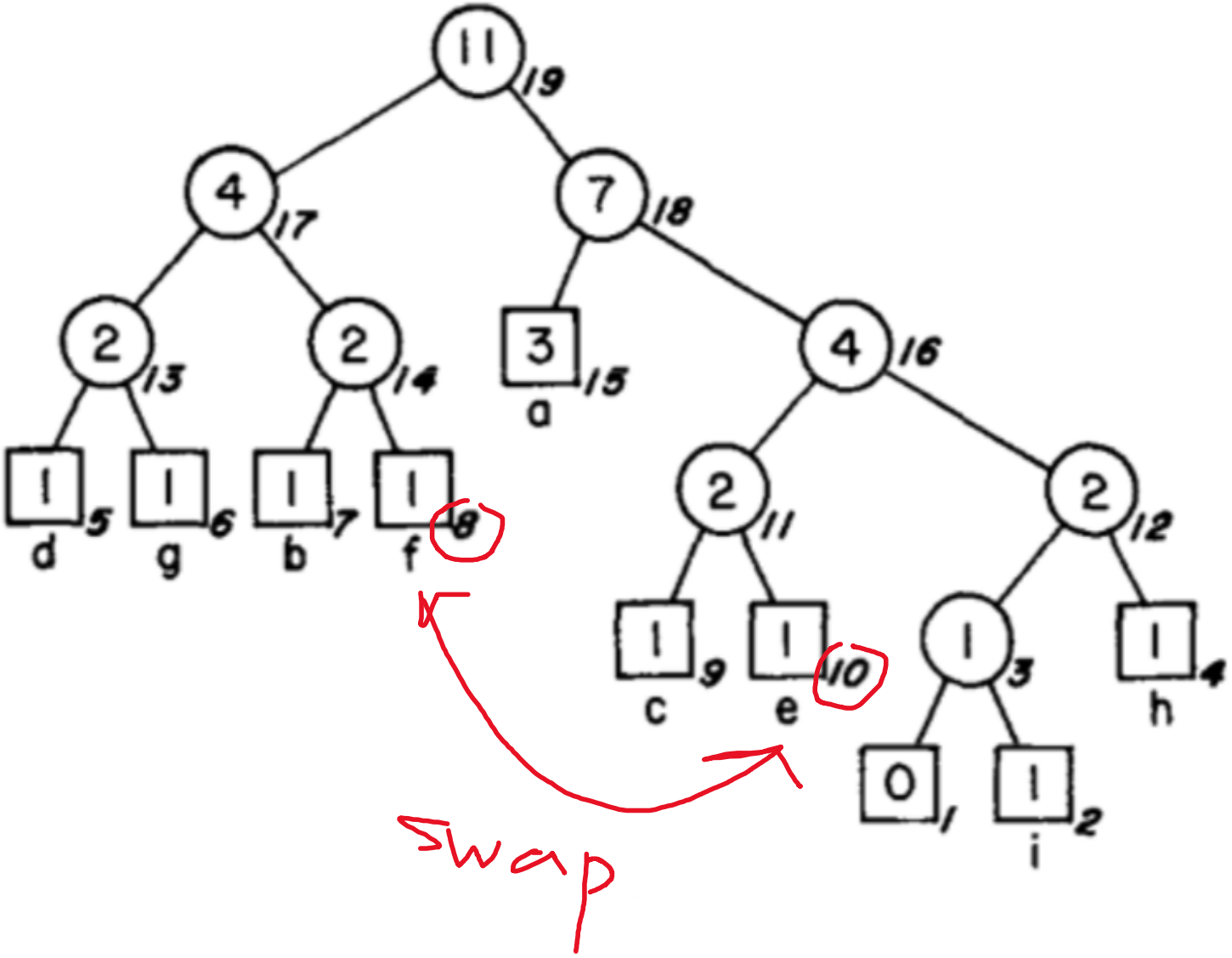
Adaptive Huffman (FGK)



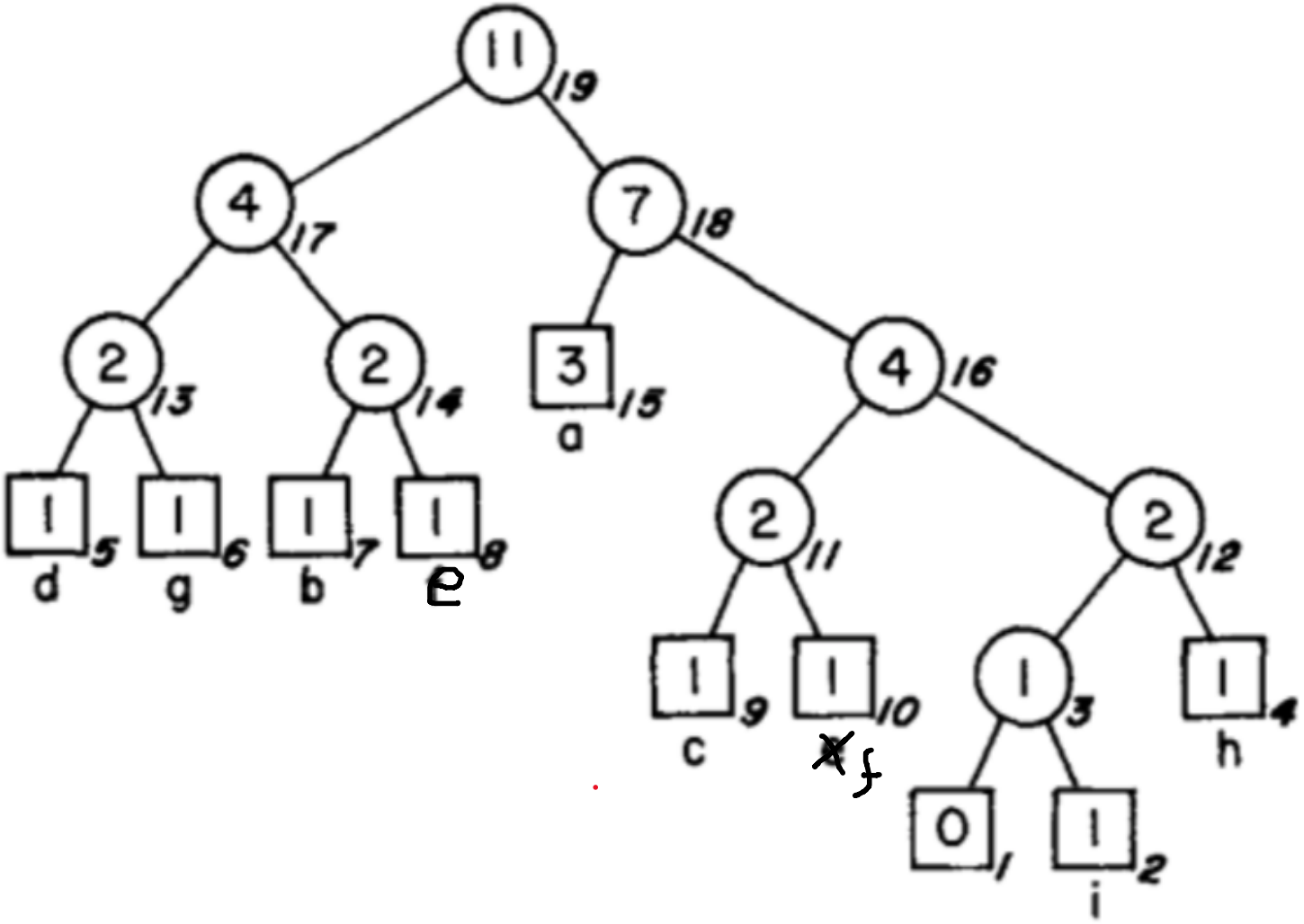
Adaptive Huffman (FGK)



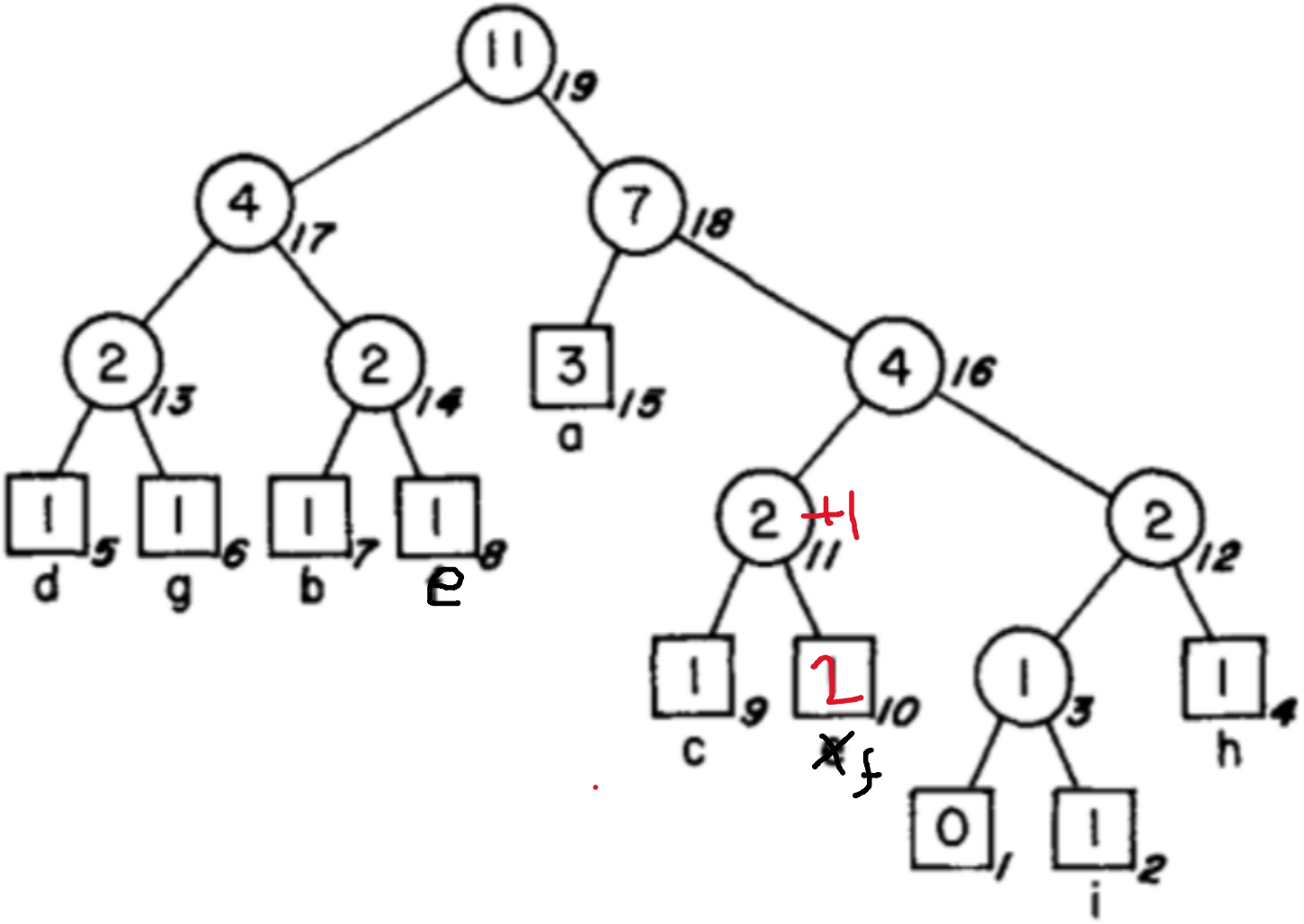
Adaptive Huffman (FGK)



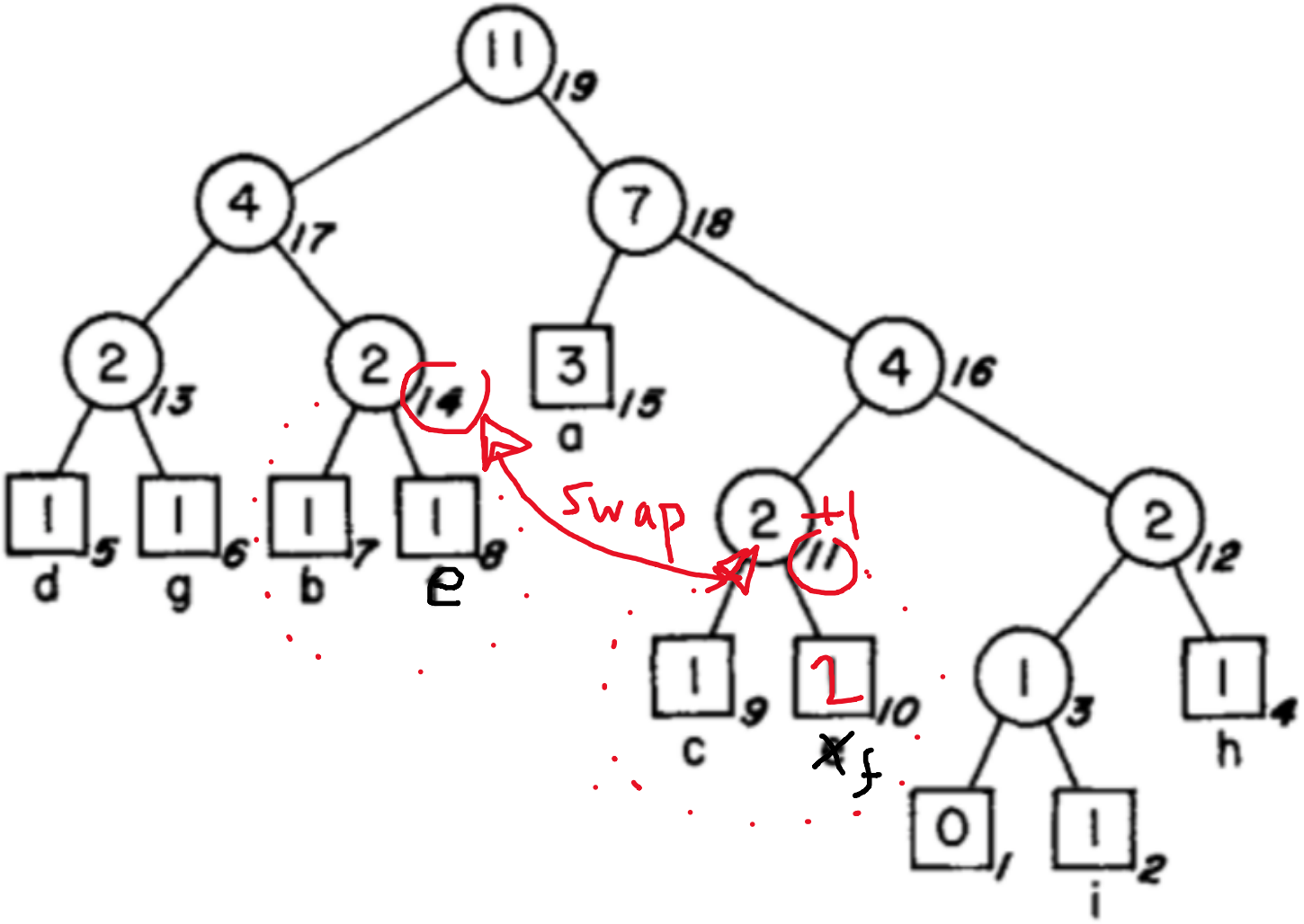
Adaptive Huffman (FGK)



Adaptive Huffman (FGK)



Adaptive Huffman (FGK)



Adaptive Huffman (FGK): when f is inserted