

61A Extra Lecture 4

Thursday, February 19

0100010101101110011000110110111101100100011010010110111001100111

0100010101101110011000110110111101100100011010010110111001100111

(Encoding)

What's the point?

What's the point?

- Why do we encode things?

What's the point?

- Why do we encode things?
 - You don't speak binary

What's the point?

- Why do we encode things?
 - You don't speak binary
 - Computers don't speak English

What's the point?

- Why do we encode things?
 - You don't speak binary
 - Computers don't speak English



©Cory Thoman * illustrationsOf.com/215196

A First Attempt

A First Attempt

- Let's use an encoding

A First Attempt

- Let's use an encoding

Letter	Binary	Letter	Binary
a	0	n	1
b	1	o	0
c	0	p	1
d	1	q	1
e	1	r	0
f	0	s	1
g	0	t	0
h	1	u	0
i	1	v	1
j	1	w	1
k	0	x	1
l	1	y	0
m	1	z	0

Analysis

Analysis

Pros

Analysis

Pros

- Encoding was easy

Analysis

Pros

- Encoding was easy
- Took a very small amount of space

Analysis

Pros

- Encoding was easy
- Took a very small amount of space

Cons

Analysis

Pros

- Encoding was easy
- Took a very small amount of space

Cons

- Decoding it was impossible

Decoding

Decoding

- Encoding by itself is **useless**

Decoding

- Encoding by itself is **useless**
- **Decoding** is also necessary

Decoding

- Encoding by itself is **useless**
- **Decoding** is also necessary
- So... we need more bits

Decoding

- Encoding by itself is **useless**
- **Decoding** is also necessary
- So... we need more bits
- How many bits do we need?

Decoding

- Encoding by itself is **useless**
- **Decoding** is also necessary
- So... we need more bits
- How many bits do we need?
 - lowercase alphabet

Decoding

- Encoding by itself is **useless**
- **Decoding** is also necessary
- So... we need more bits
- How many bits do we need?
 - lowercase alphabet
 - 5 bits

A Second Attempt

A Second Attempt

- Let's try another encoding

A Second Attempt

- Let's try another encoding

Letter	Binary	Letter	Binary
a	00000	n	01101
b	00001	o	01110
c	00010	p	01111
d	00011	q	10000
e	00100	r	10001
f	00101	s	10010
g	00110	t	10011
h	00111	u	10100
i	01000	v	10101
j	01001	w	10110
k	01010	x	10111
l	01011	y	11000
m	01100	z	11001

Analysis

Analysis

Pros

Analysis

Pros

- Encoding was easy

Analysis

Pros

- Encoding was easy
- Decoding was possible

Analysis

Pros

- Encoding was easy
- Decoding was possible

Cons

Analysis

Pros

- Encoding was easy
- Decoding was possible

Cons

- Takes more space...

Analysis

Pros

- Encoding was easy
- Decoding was possible

Cons

- Takes more space...
- What restriction did we place that's unnecessary?

Analysis

Pros

- Encoding was easy
- Decoding was possible

Cons

- Takes more space...
- What restriction did we place that's unnecessary?
 - Fixed length

Variable Length Encoding

Variable Length Encoding

- Problems?

Variable Length Encoding

- Problems?
 - When do we start and stop?

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA
 - A – 00, B – 0

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA
 - A – 00, B – 0
 - Encode ABA: 00000

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA
 - A – 00, B – 0
 - Encode ABA: 00000
 - Decode 00000:

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA
 - A – 00, B – 0
 - Encode ABA: 00000
 - Decode 00000:
 - ABA, AAB, BAA?

Variable Length Encoding

- Problems?
 - When do we start and stop?
 - String of As and Bs: ABA
 - A – 00, B – 0
 - Encode ABA: 00000
 - Decode 00000:
 - ABA, AAB, BAA?
 - What lengths do we use?

A Second Look at Fixed Length

A Second Look at Fixed Length

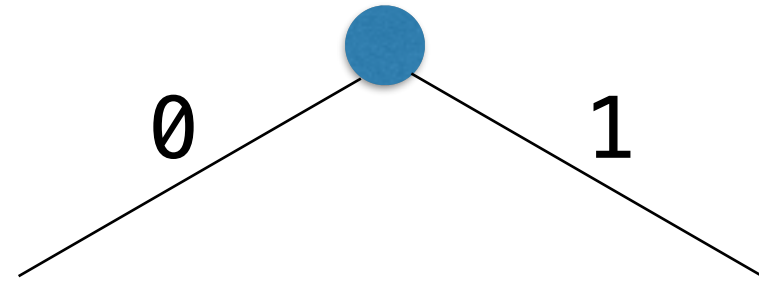
Letter	Binary	Letter	Binary
a	00000	n	01101
b	00001	o	01110
c	00010	p	01111
d	00011	q	10000
e	00100	r	10001
f	00101	s	10010
g	00110	t	10011
h	00111	u	10100
i	01000	v	10101
j	01001	w	10110
k	01010	x	10111
l	01011	y	11000
m	01100	z	11001

Trees!

Trees!

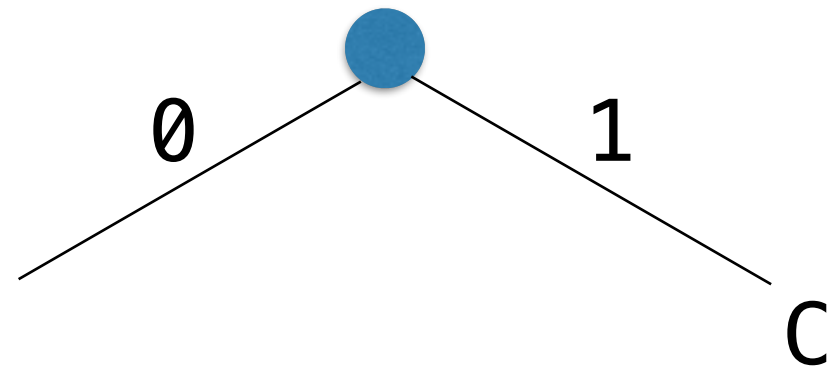
Letter	Binary
A	00
B	01
C	1

Trees!



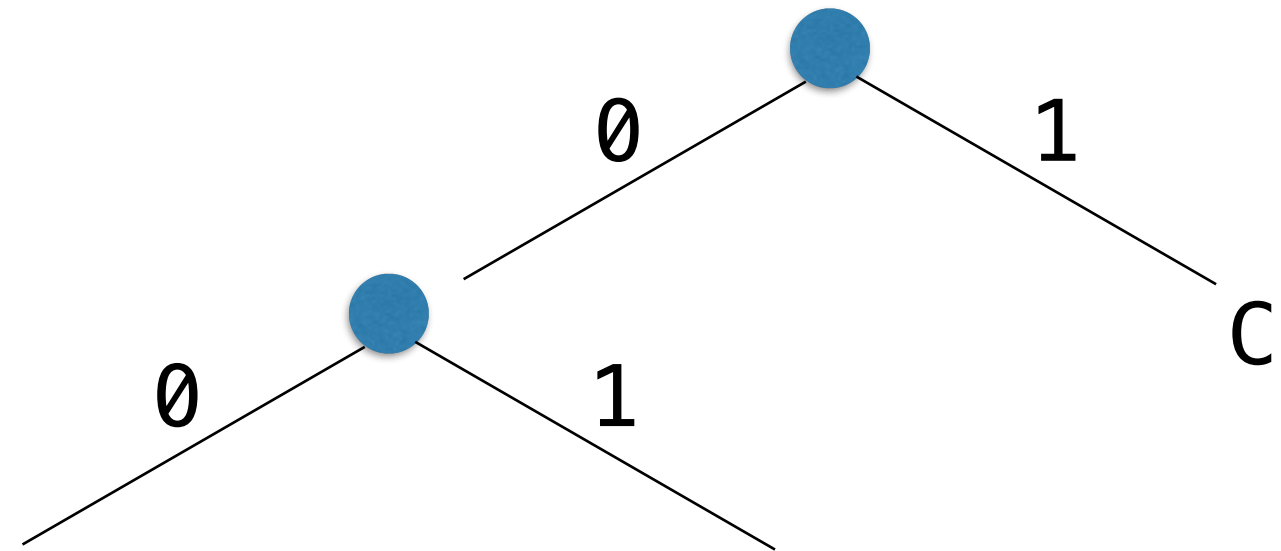
Letter	Binary
A	00
B	01
C	1

Trees!



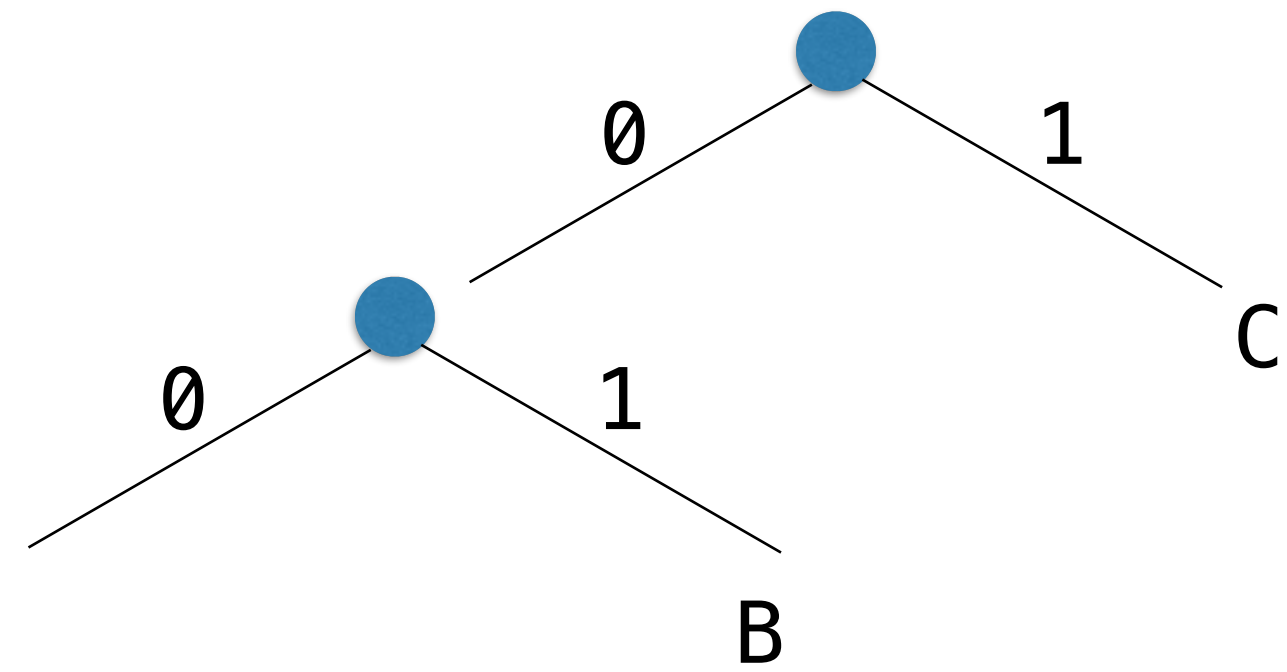
Letter	Binary
A	00
B	01
C	1

Trees!



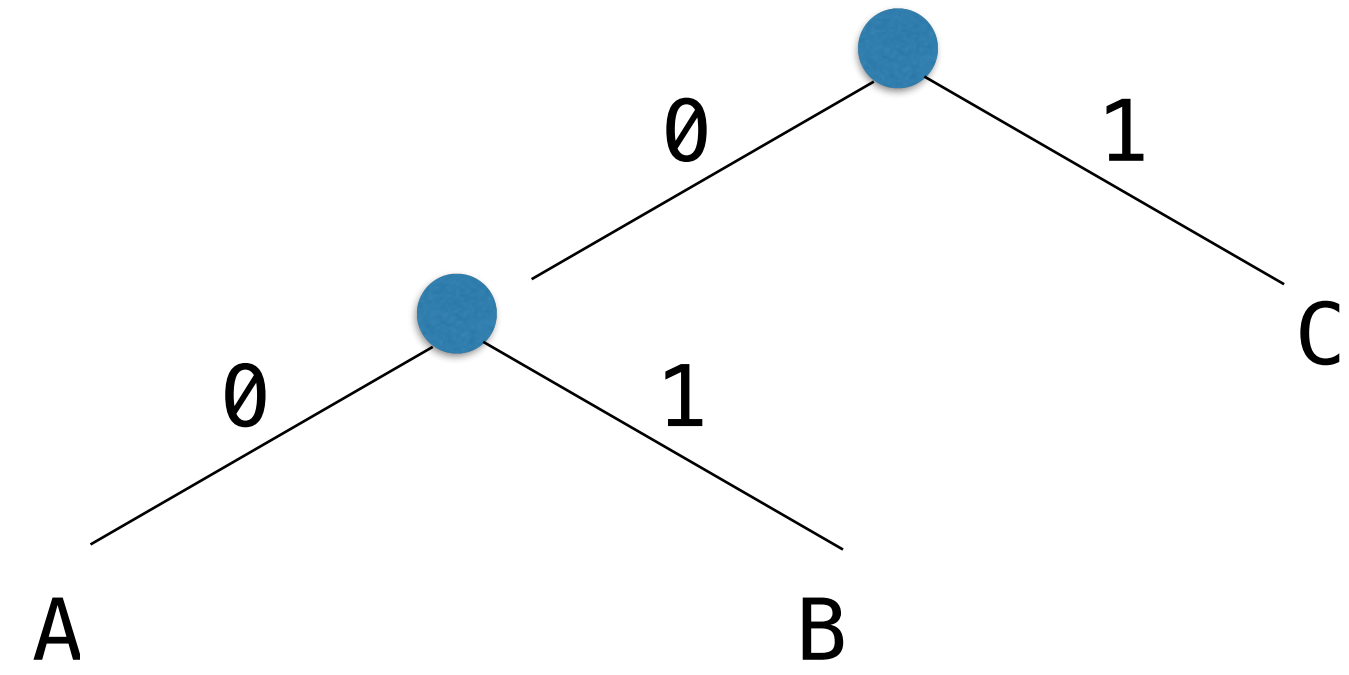
Letter	Binary
A	00
B	01
C	1

Trees!



Letter	Binary
A	00
B	01
C	1

Trees!



Letter	Binary
A	00
B	01
C	1

What happens when...?

Letter	Binary	Letter	Binary
a	0	n	1
b	1	o	0
c	0	p	1
d	1	q	1
e	1	r	0
f	0	s	1
g	0	t	0
h	1	u	0
i	1	v	1
j	1	w	1
k	0	x	1
l	1	y	0
m	1	z	0

What happens when...?

- Rule 1: Each leaf only has 1 label

Letter	Binary	Letter	Binary
a	0	n	1
b	1	o	0
c	0	p	1
d	1	q	1
e	1	r	0
f	0	s	1
g	0	t	0
h	1	u	0
i	1	v	1
j	1	w	1
k	0	x	1
l	1	y	0
m	1	z	0

What happens when...?

Letter	Binary
A	00
B	0

What happens when...?

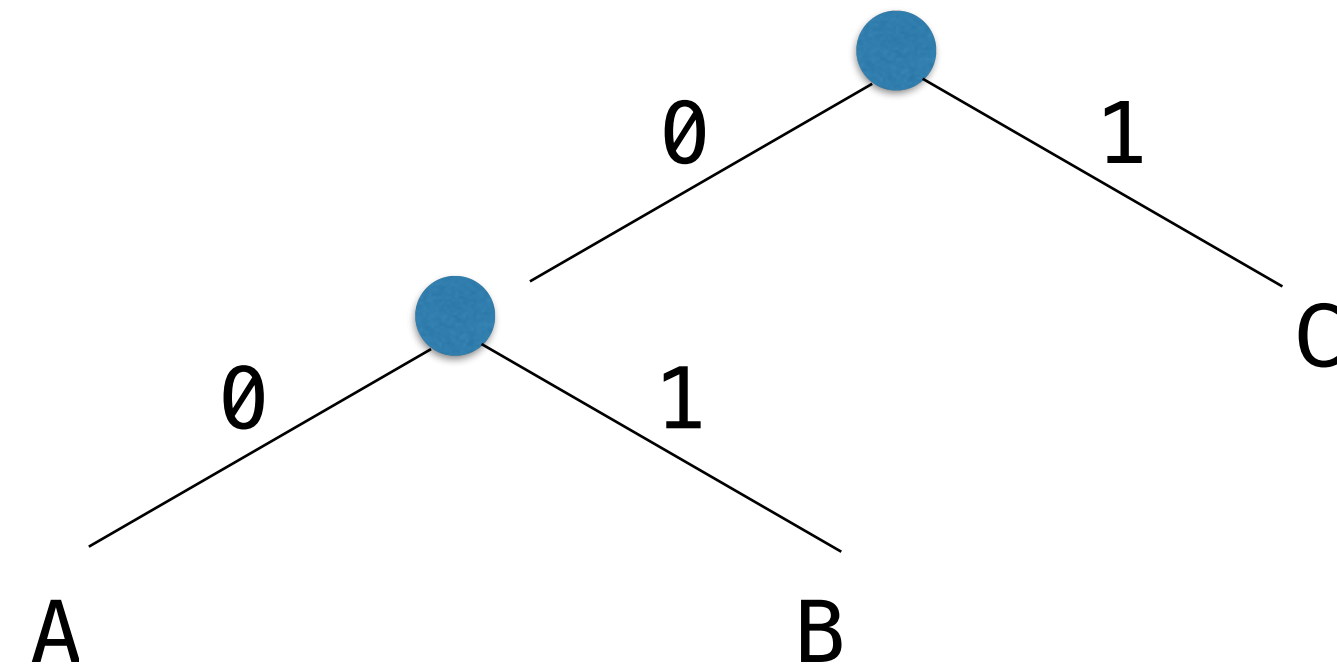
- Rule 2: Only leaves get labels

Letter	Binary
A	00
B	0

An Optimal Encoding

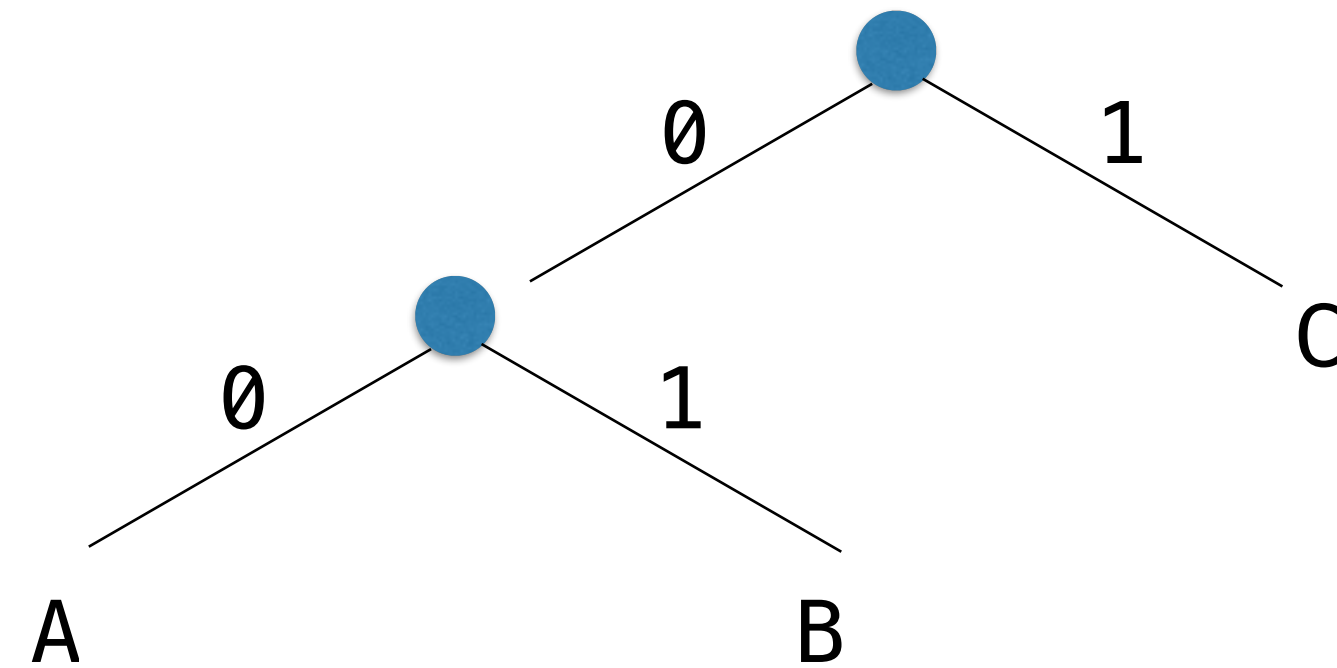
An Optimal Encoding

- Start with a tree



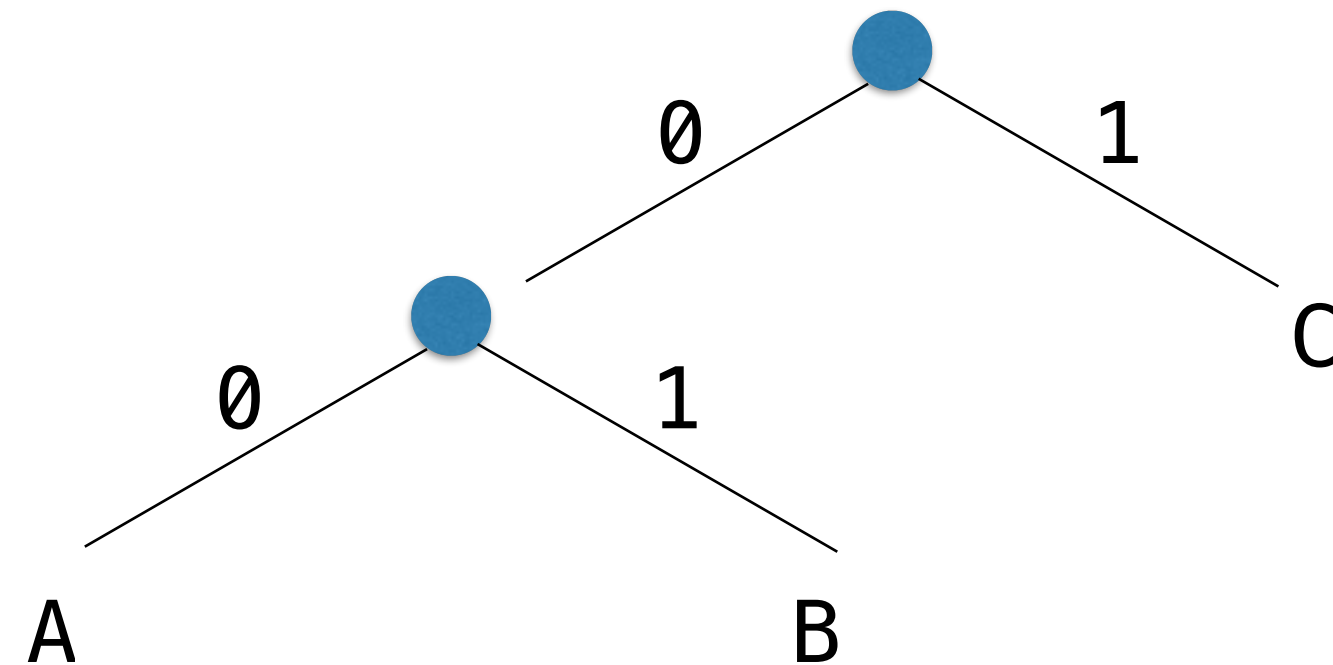
An Optimal Encoding

- Start with a tree
- What kinds of things do we want to encode with this?



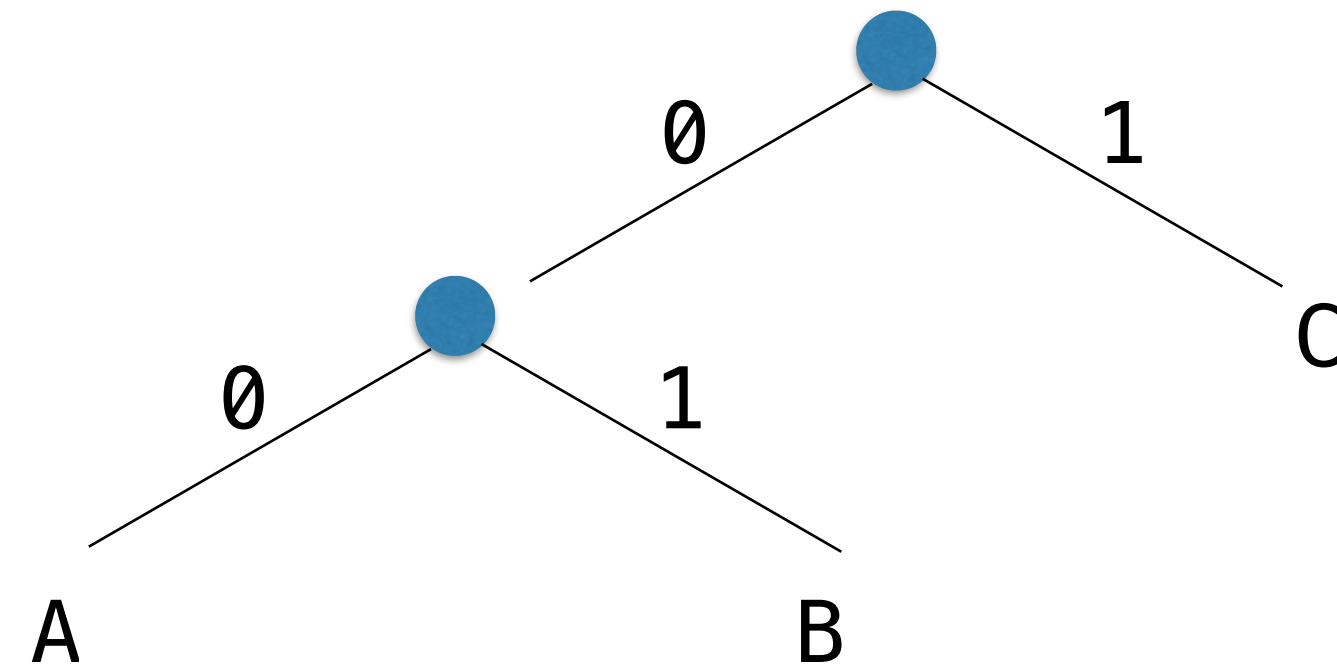
An Optimal Encoding

- Start with a tree
- What kinds of things do we want to encode with this?
- What letter do we want to appear the most?



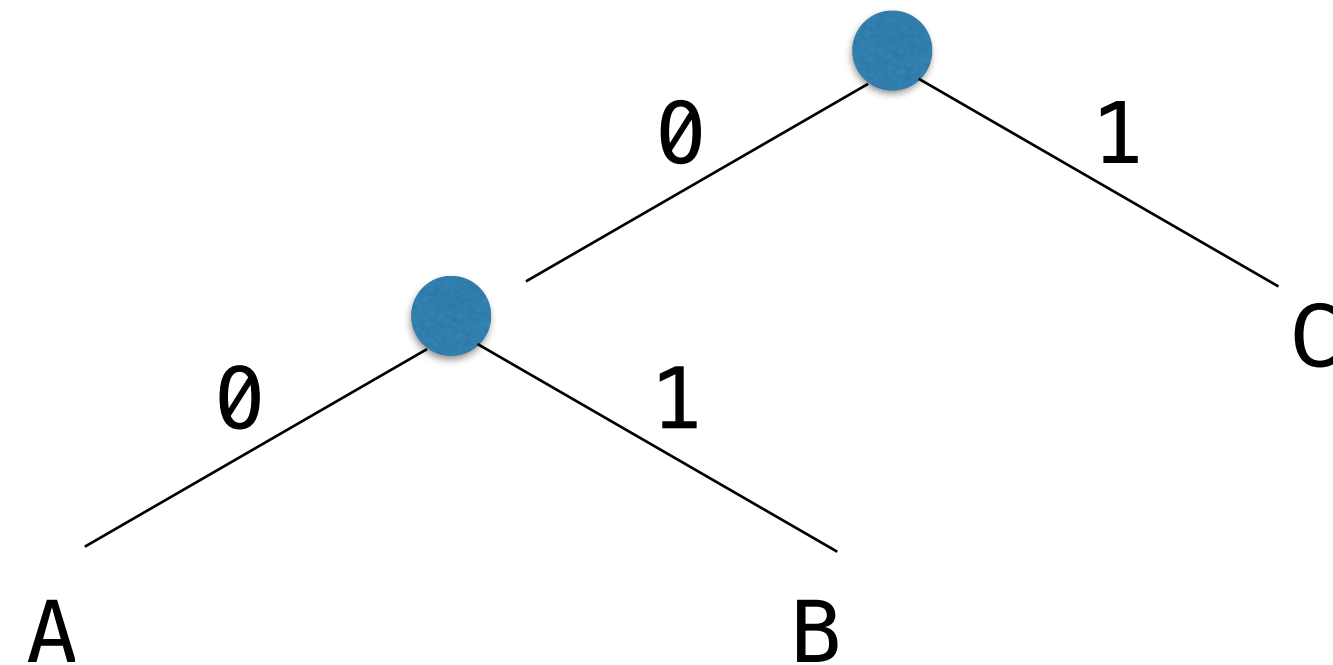
An Optimal Encoding

- Start with a tree
- What kinds of things do we want to encode with this?
- What letter do we want to appear the most?
- How about the least?



An Optimal Encoding

- Start with a tree
- What kinds of things do we want to encode with this?
- What letter do we want to appear the most?
- How about the least?
- This is called a **Huffman Encoding**



Huffman Encoding

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:
 - AAAAAAAAAABBBBBCCCCCCCDDDDDDDDDD

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:
 - AAAAAAAAAABBBBBCCCCCCCDDDDDDDDDD
 - A appears 10 times

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:
 - AAAAAAAAAAABBBBBBCCCCCCCCDDDDDDDDDD
 - A appears 10 times
 - B appears 5 times

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:
 - AAAAAAAAAABBBBBCCCCCCCCDDDDDDDDDD
 - A appears 10 times
 - B appears 5 times
 - C appears 7 times

Huffman Encoding

- Let's pretend we want to come up with the optimal encoding:
 - AAAAAAAAAAABBBBBBCCCCCCCCDDDDDDDDDD
 - A appears 10 times
 - B appears 5 times
 - C appears 7 times
 - D appears 9 times

Huffman Encoding

Huffman Encoding

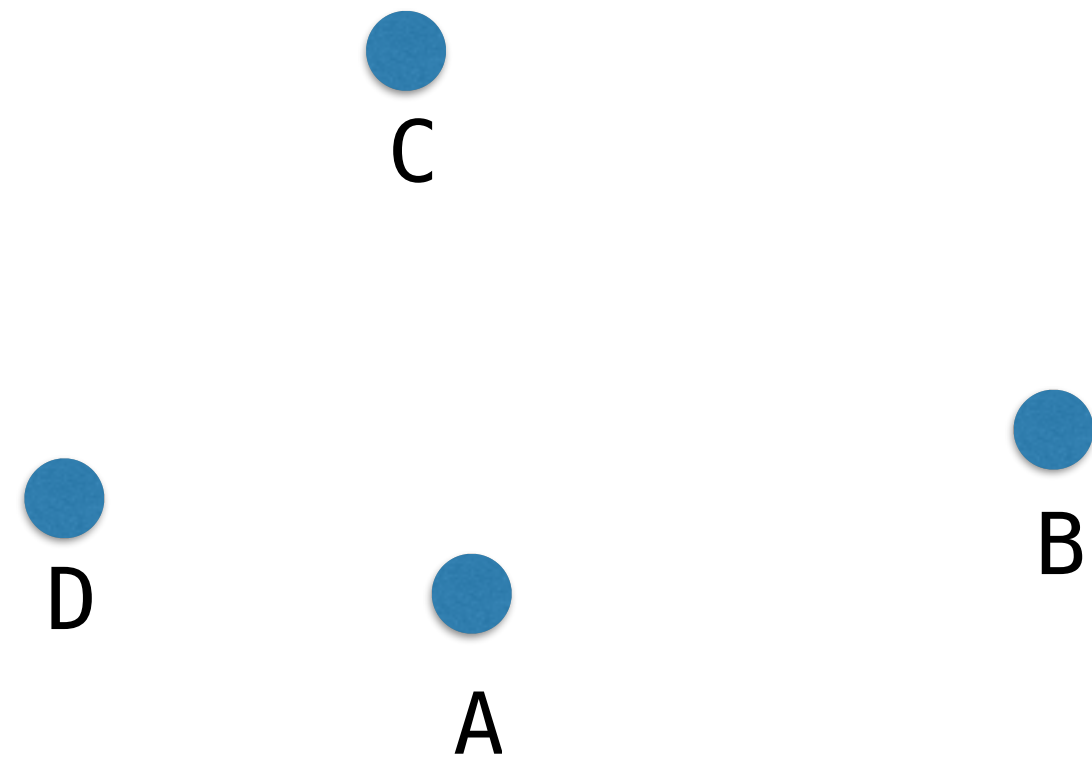
- Start with the two smallest frequencies

Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times

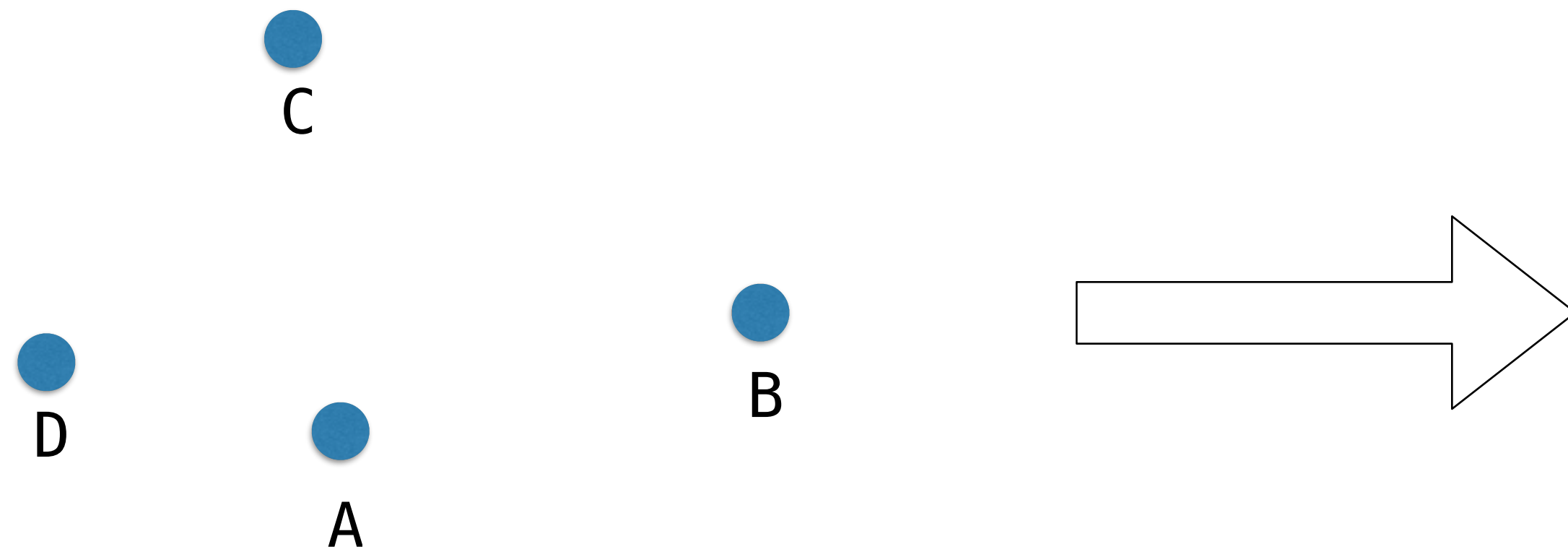
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times



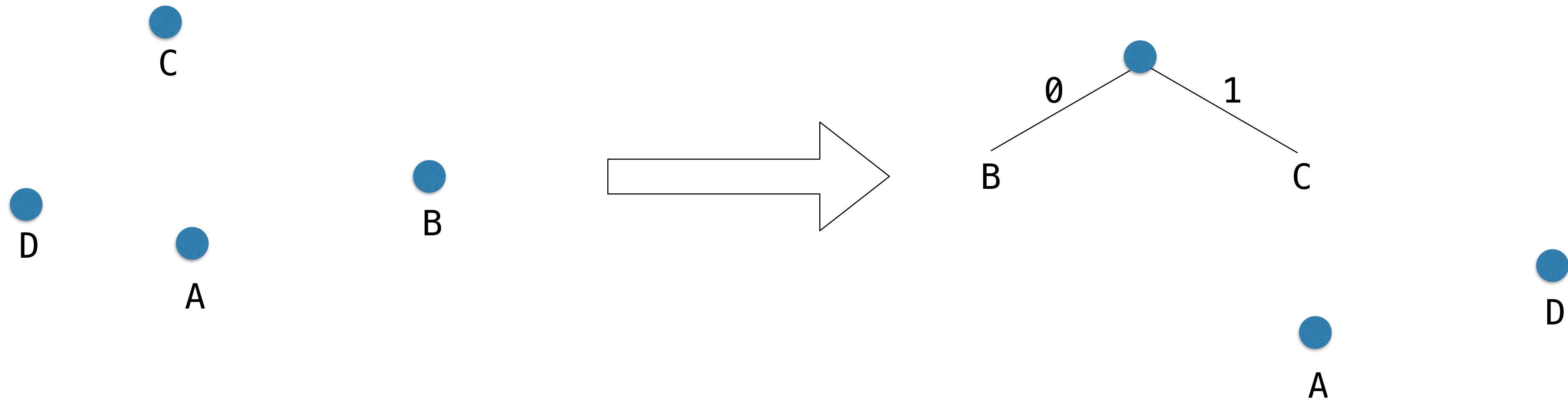
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times



Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times



Huffman Encoding

Huffman Encoding

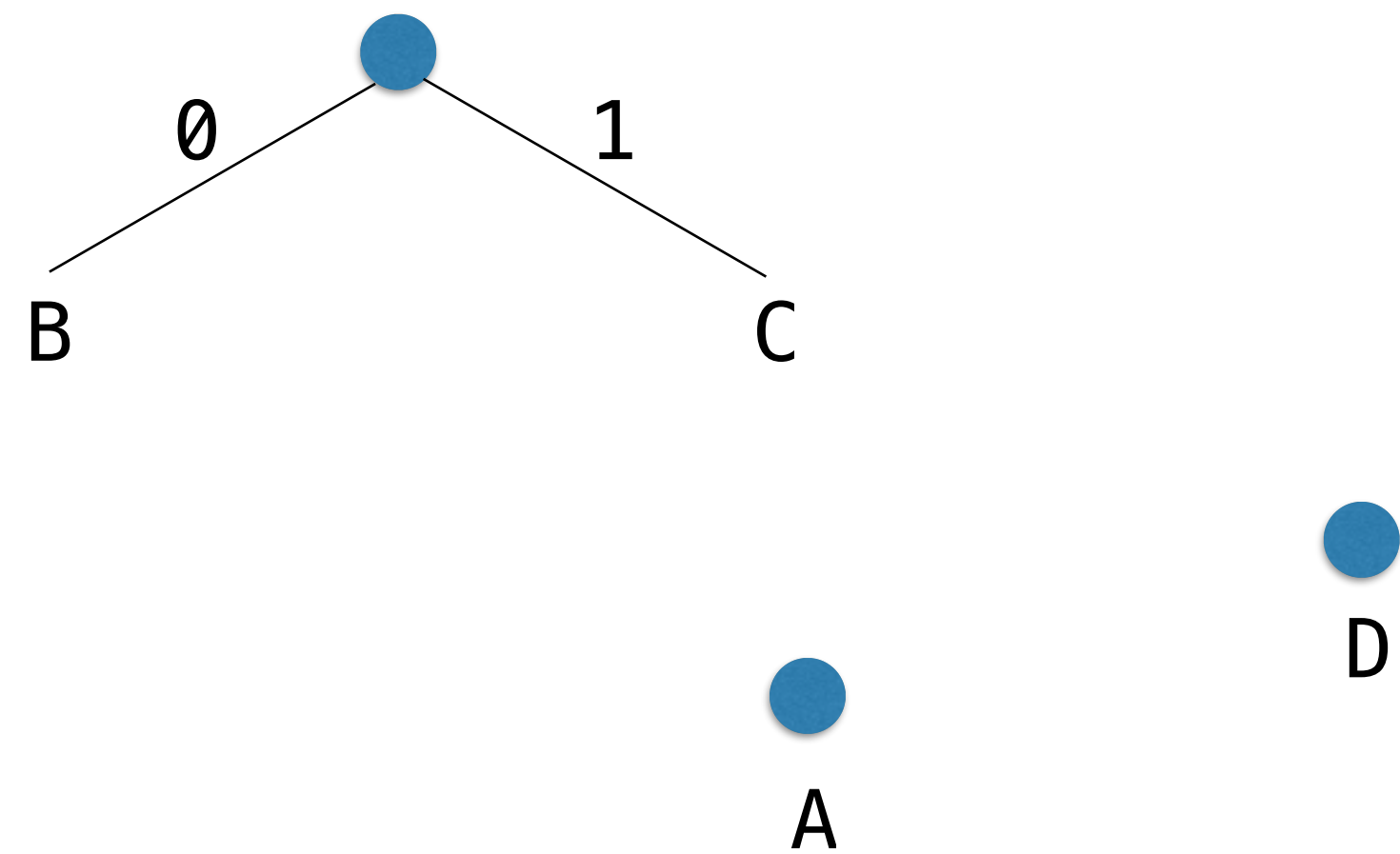
- Continue...

Huffman Encoding

- Continue...
 - A appears 10 times, B & C appear a combined 12 times, D appears 9 times

Huffman Encoding

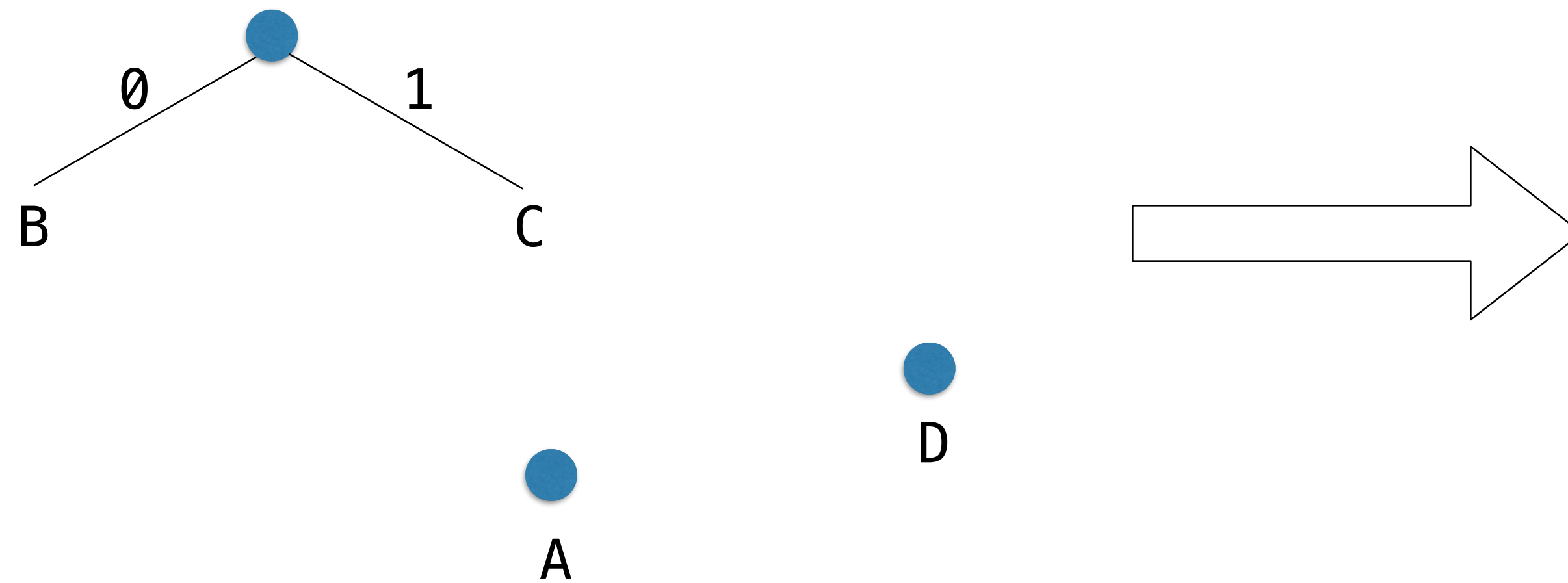
- Continue...
 - A appears 10 times, B & C appear a combined 12 times, D appears 9 times



Huffman Encoding

- Continue...

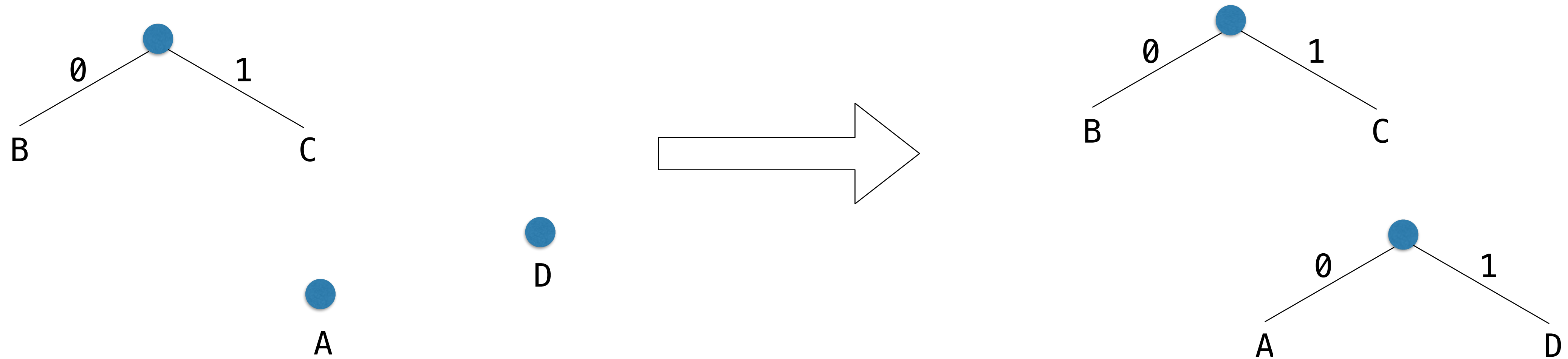
- A appears 10 times, B & C appear a combined 12 times, D appears 9 times



Huffman Encoding

- Continue...

- A appears 10 times, B & C appear a combined 12 times, D appears 9 times



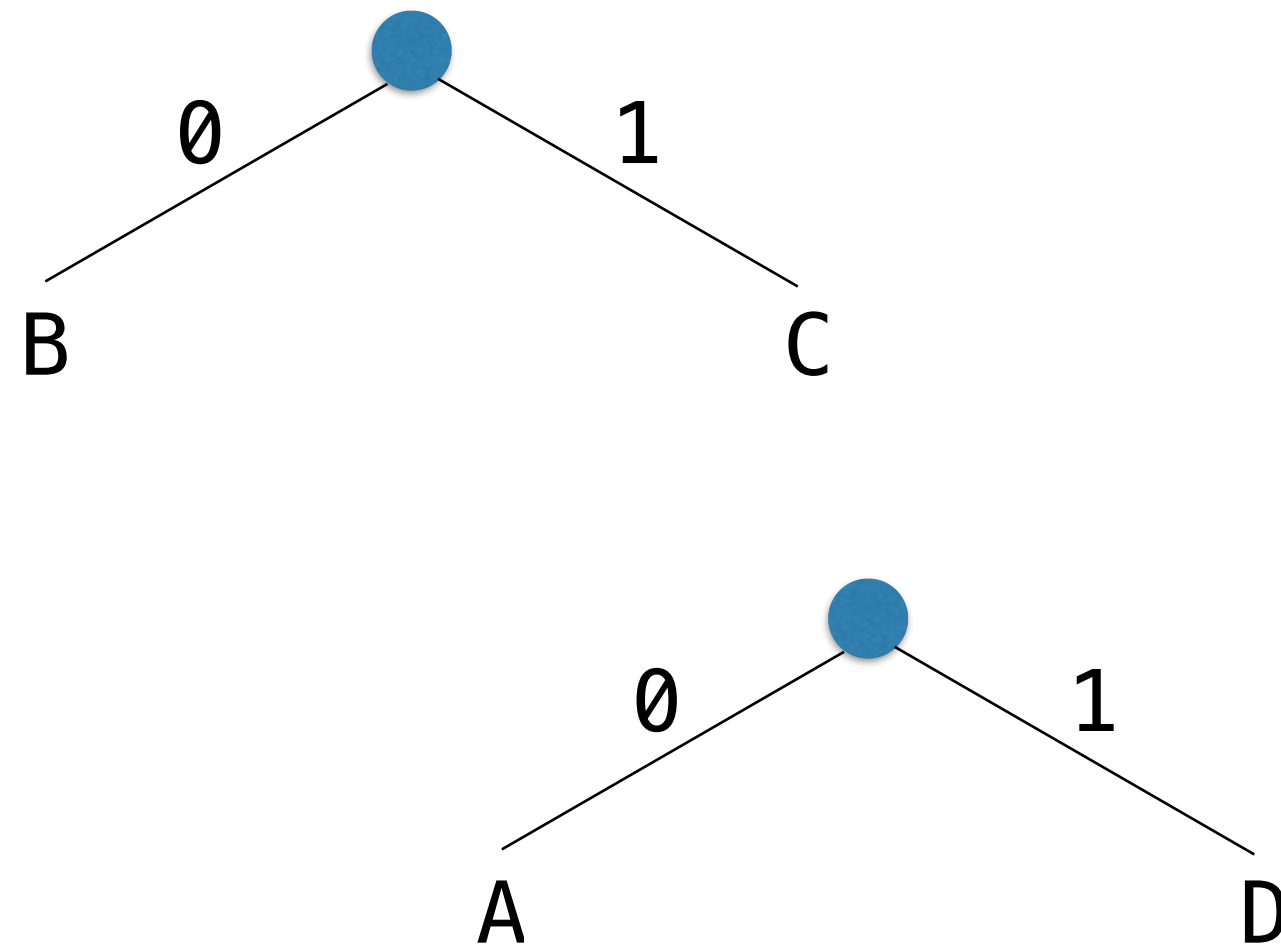
Huffman Encoding

Huffman Encoding

- And finally...

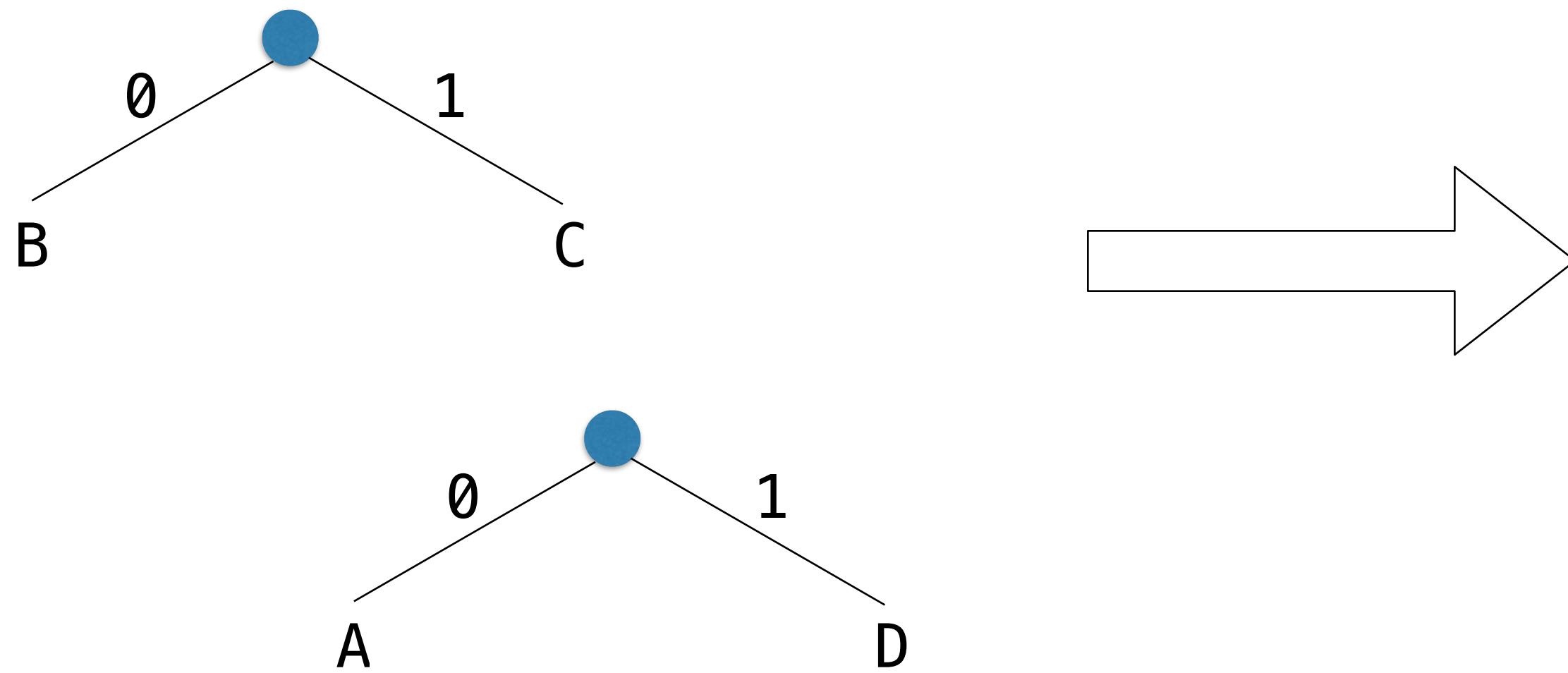
Huffman Encoding

- And finally...



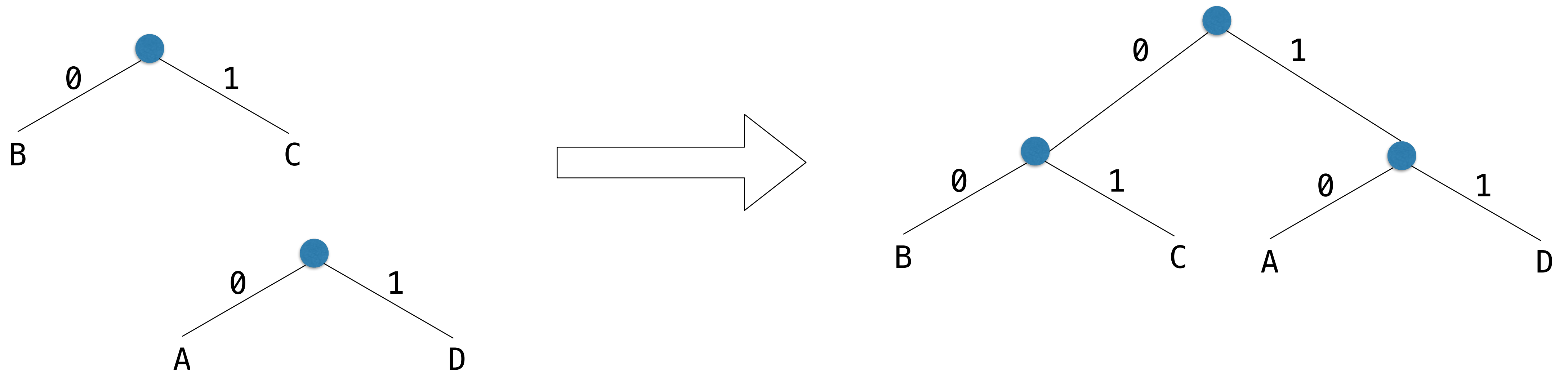
Huffman Encoding

- And finally...



Huffman Encoding

- And finally...



Huffman Encoding

Huffman Encoding

- Another example...

Huffman Encoding

- Another example...
 - AAAAAAAAAABCCD

Huffman Encoding

- Another example...
 - AAAAAAAAAABCCD
 - A appears 10 times

Huffman Encoding

- Another example...
 - AAAAAAAAAABCCD
 - A appears 10 times
 - B appears 1 time

Huffman Encoding

- Another example...
 - AAAAAAAAAABCCD
 - A appears 10 times
 - B appears 1 time
 - C appears 2 times

Huffman Encoding

- Another example...
 - AAAAAAAAAABCCD
 - A appears 10 times
 - B appears 1 time
 - C appears 2 times
 - D appears 1 time

Huffman Encoding

Huffman Encoding

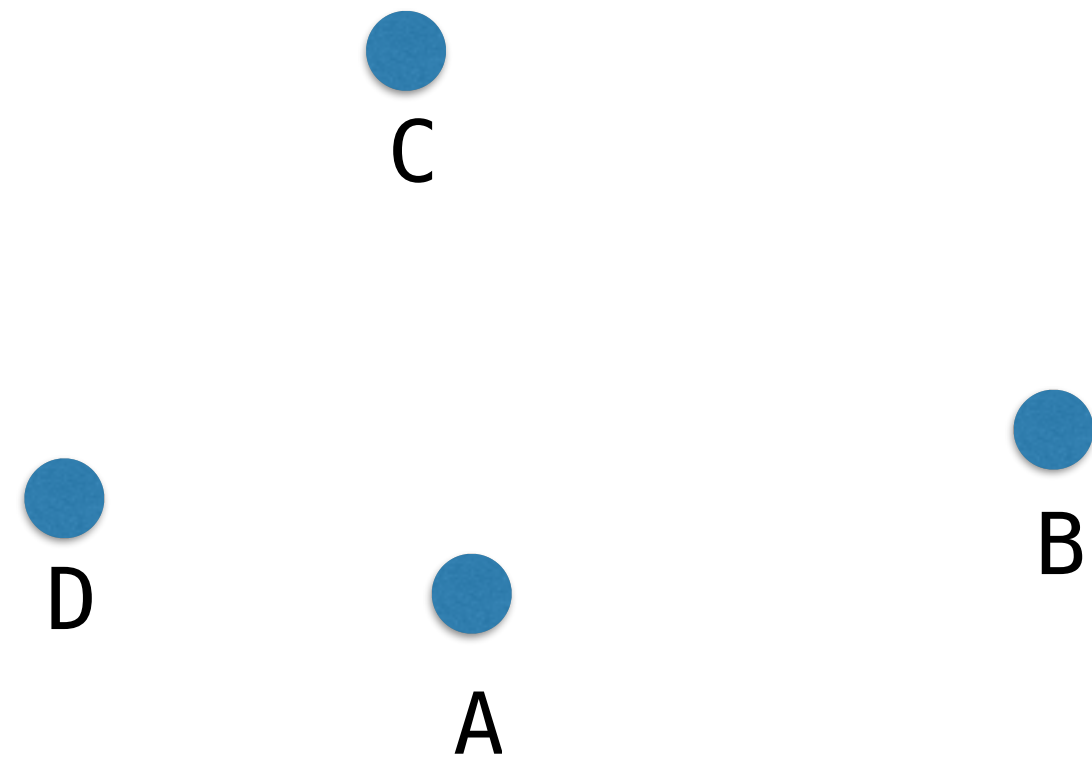
- Start with the two smallest frequencies

Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time

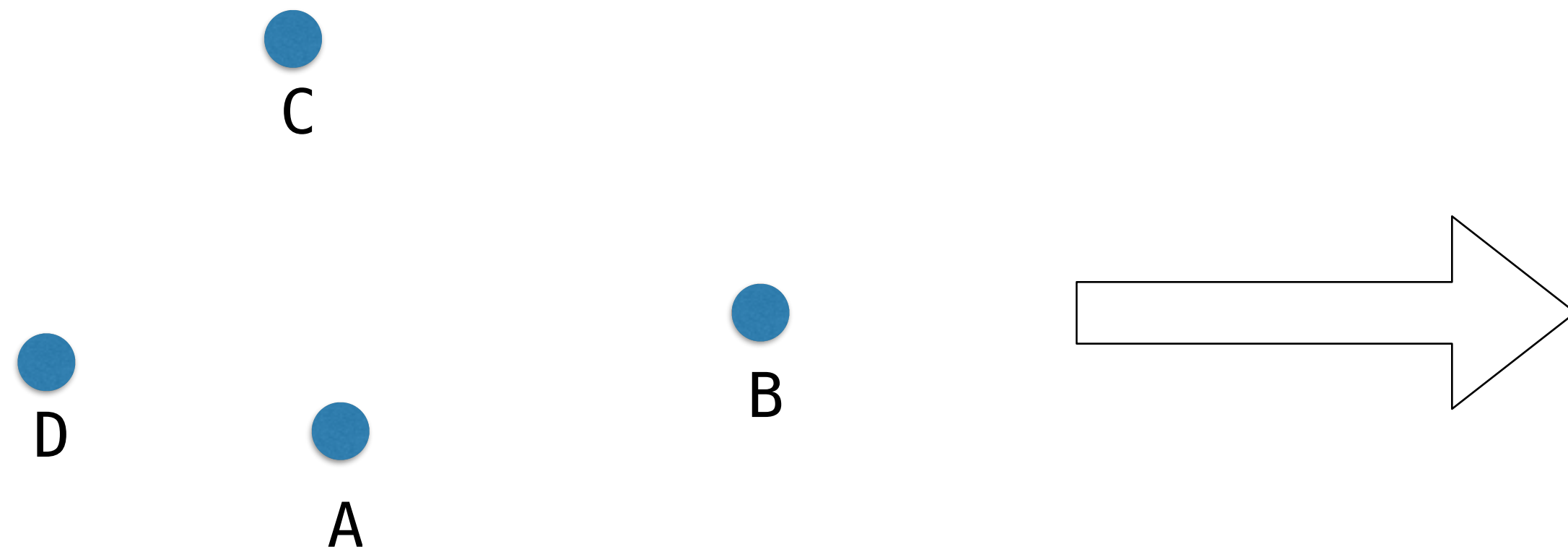
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time



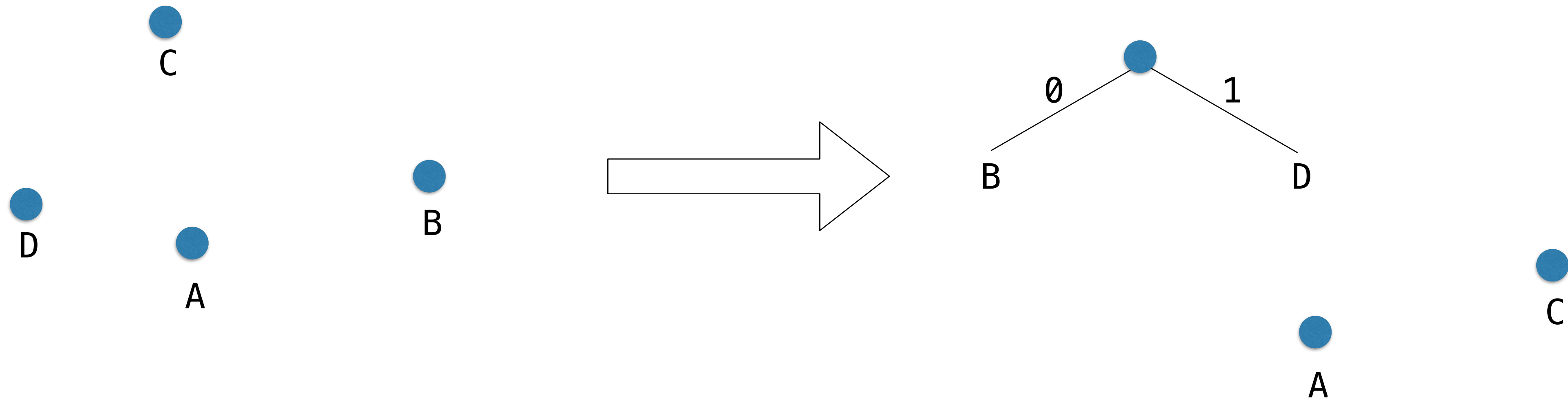
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time



Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time



Huffman Encoding

Huffman Encoding

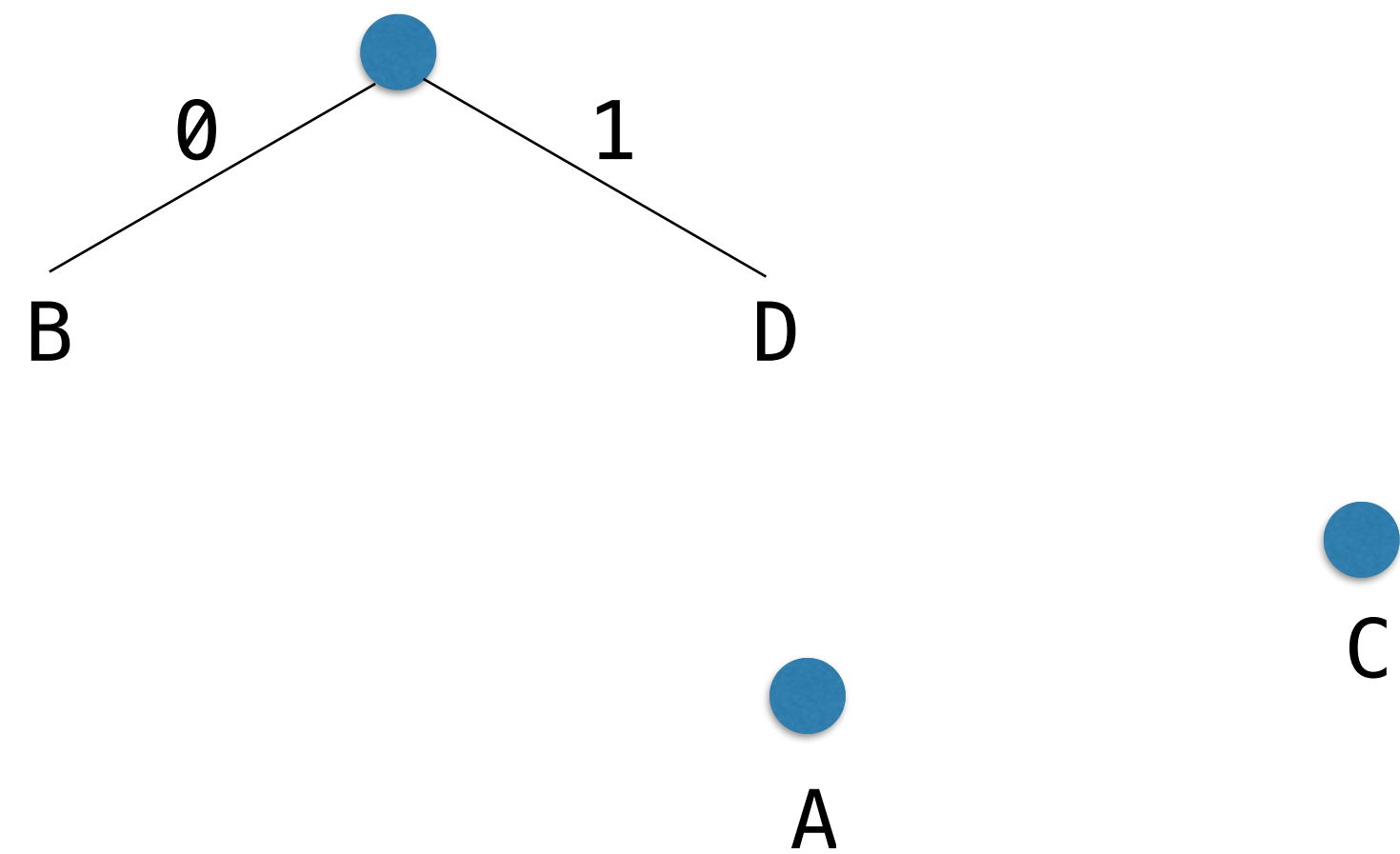
- Start with the two smallest frequencies

Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B & D appear a combined 2 times, C appears 2 times

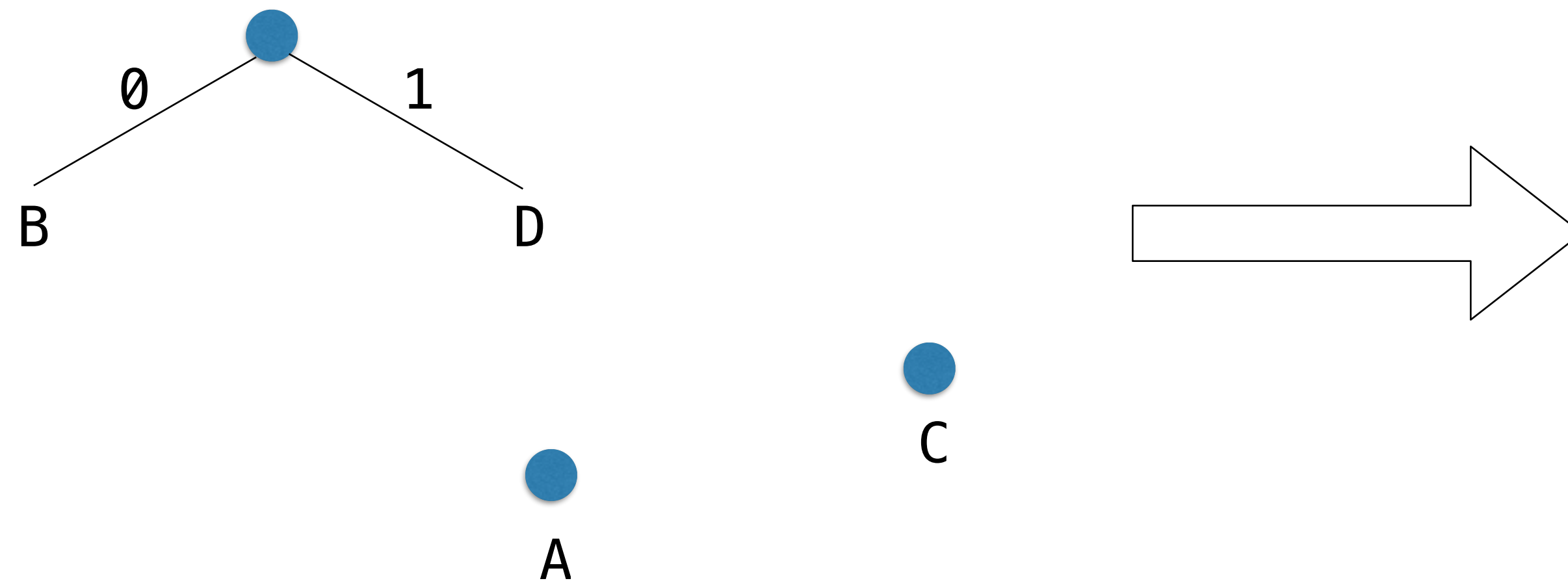
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B & D appear a combined 2 times, C appears 2 times



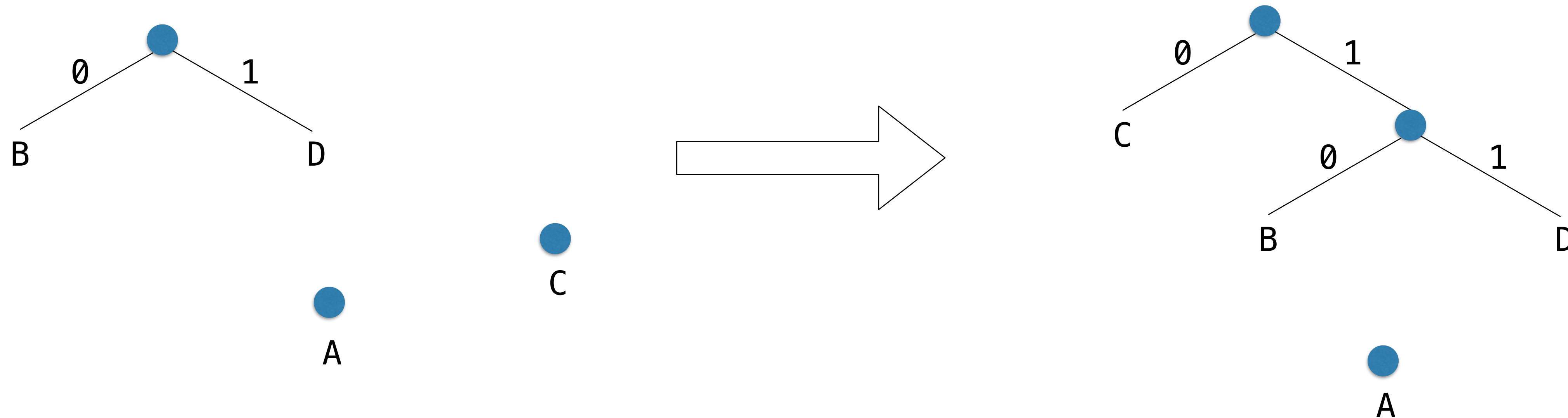
Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B & D appear a combined 2 times, C appears 2 times



Huffman Encoding

- Start with the two smallest frequencies
 - A appears 10 times, B & D appear a combined 2 times, C appears 2 times



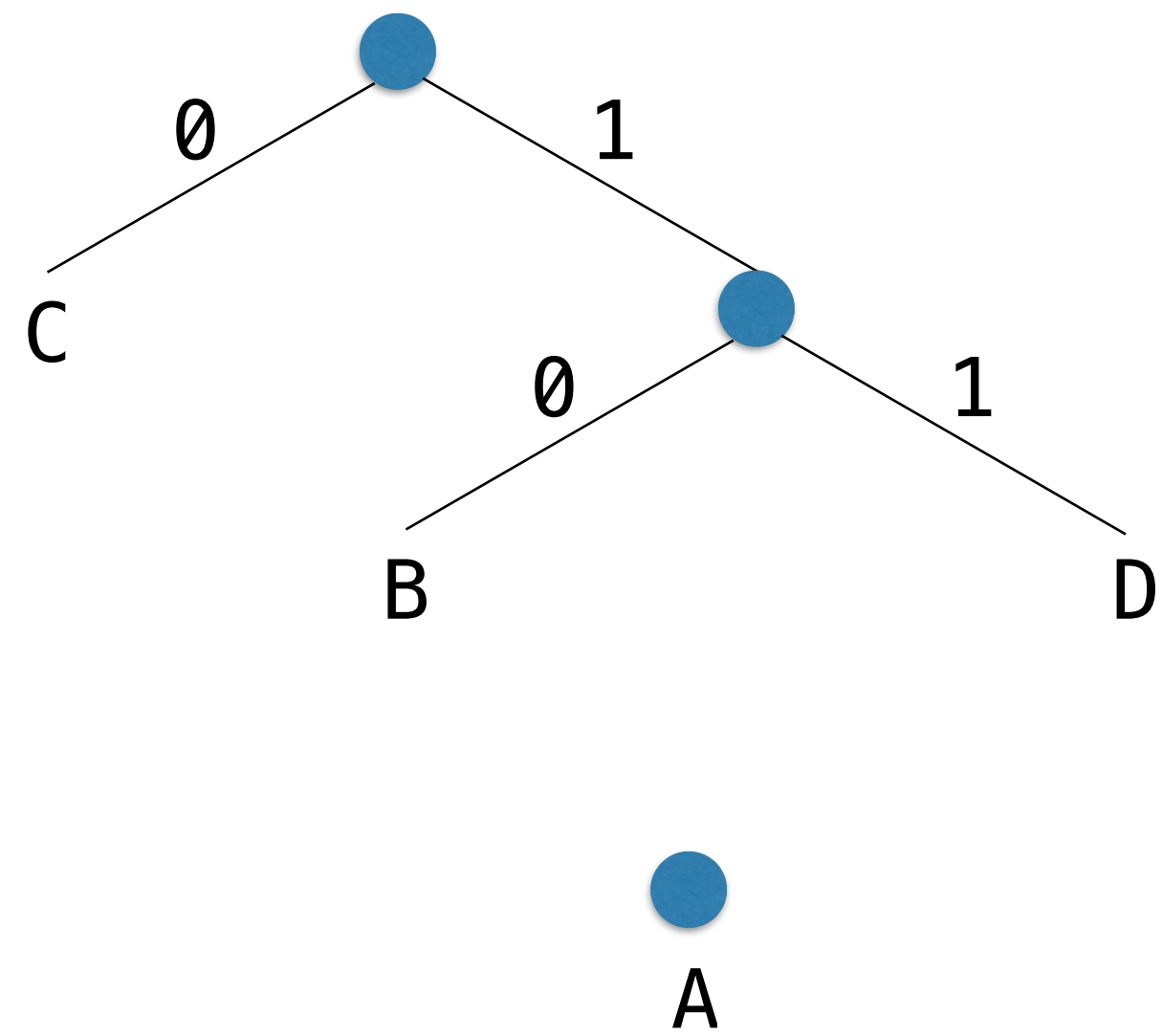
Huffman Encoding

Huffman Encoding

- And finally...

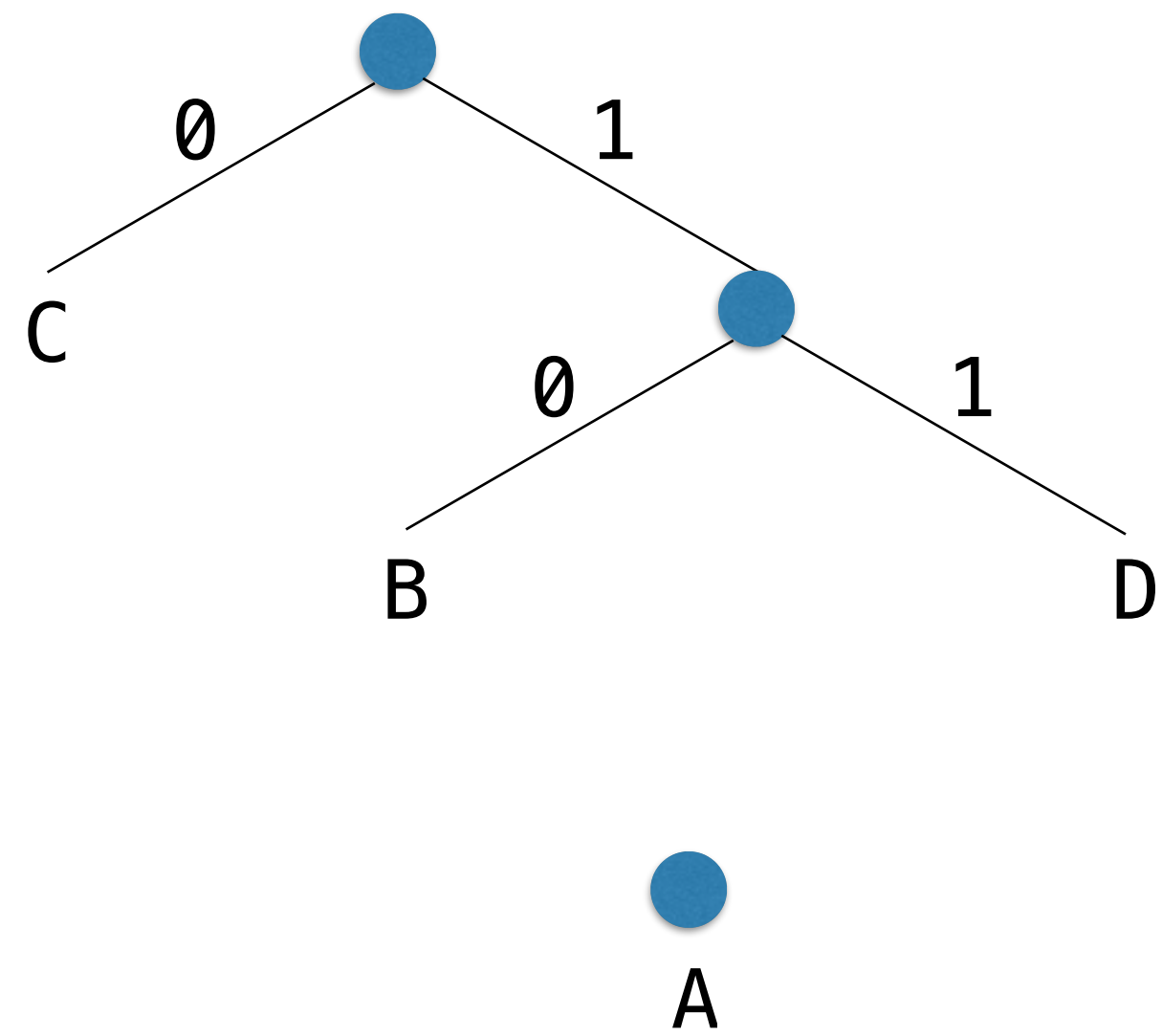
Huffman Encoding

- And finally...



Huffman Encoding

- And finally...



Huffman Encoding

- And finally...

