

## 61A Extra Lecture 4

---

# Announcements

## Encoding Strings

## Representing Strings: UTF-8 Encoding

---

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0-255.



## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0-255.

bytes

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0-255.

bytes

integers

## Representing Strings: UTF-8 Encoding

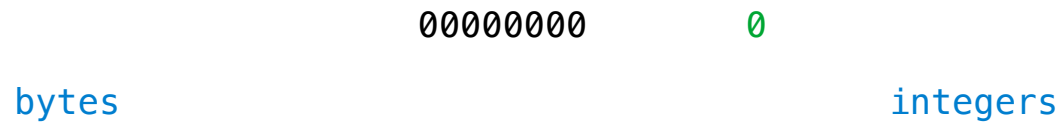
---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0-255.



## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0-255.

	00000000	0	
bytes	00000001	1	integers

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

	00000000	0	
bytes	00000001	1	integers
	00000010	2	

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

	00000000	0	
bytes	00000001	1	integers
	00000010	2	
	00000011	3	

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

	00000000	0	
bytes	00000001	1	integers
	00000010	2	
	00000011	3	

Variable-length encoding: integers vary in the number of bytes required to encode them.

## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

	00000000	0	
bytes	00000001	1	integers
	00000010	2	
	00000011	3	

Variable-length encoding: integers vary in the number of bytes required to encode them.

In Python: `string` length is measured in characters, `bytes` length in bytes.



## Representing Strings: UTF-8 Encoding

---

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

	00000000	0	
bytes	00000001	1	integers
	00000010	2	
	00000011	3	

Variable-length encoding: integers vary in the number of bytes required to encode them.

In Python: `string` length is measured in characters, `bytes` length in bytes.

(Demo)

## Fixed-Length Encodings

## 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

## Decoding

---

## Decoding

---

- An encoding without a deterministic decoding procedure is not very useful

## Decoding

---

- An encoding without a deterministic decoding procedure is not very useful
- How many bits do we need to encode each letter uniquely?



## Decoding

---

- An encoding without a deterministic decoding procedure is not very useful
- How many bits do we need to encode each letter uniquely?
  - lowercase alphabet

## Decoding

---

- An encoding without a deterministic decoding procedure is not very useful
- How many bits do we need to encode each letter uniquely?
  - 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 deterministic

## Analysis

---

### Pros

- Encoding was easy
- Decoding was deterministic

### Cons

## Analysis

---

### **Pros**

- Encoding was easy
- Decoding was deterministic

### **Cons**

- Takes more space...

## Analysis

---

### Pros

- Encoding was easy
- Decoding was deterministic

### Cons

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

## Analysis

---

### Pros

- Encoding was easy
- Decoding was deterministic

### Cons

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

## Variable-Length Encodings

## Variable Length Encoding

---

## Variable Length Encoding

---

- Encoding Candidate 1: A: 1, B:01, C: 10, D: 11, E: 100, F: 101, ...



## Variable Length Encoding

---

- Encoding Candidate 1: A: 1, B:01, C: 10, D: 11, E: 100, F: 101, ...
  - What does 01111 encode?

## Variable Length Encoding

---

- Encoding Candidate 1: A: 1, B:01, C: 10, D: 11, E: 100, F: 101, ...
  - What does 01111 encode?
- Encoding Candidate 2: A: 00, B: 01, C: 100, D: 101, E: 1100, F: 1101, ...

## Variable Length Encoding

---

- Encoding Candidate 1: A: 1, B: 01, C: 10, D: 11, E: 100, F: 101, ...
  - What does 01111 encode?
- Encoding Candidate 2: A: 00, B: 01, C: 100, D: 101, E: 1100, F: 1101, ...
  - What does 0100101 encode? How about 10111001101001001100?

## Variable Length Encoding

---

- Encoding Candidate 1: A: 1, B: 01, C: 10, D: 11, E: 100, F: 101, ...
  - What does 01111 encode?
- Encoding Candidate 2: A: 00, B: 01, C: 100, D: 101, E: 1100, F: 1101, ...
  - What does 0100101 encode? How about 10111001101001001100?
- Deterministic decoding from left to right is possible if the encoding of one character is **never** a proper prefix of the decoding of another character.

## Deterministic Codes Have a Tree Structure

---

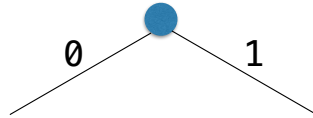
## Deterministic Codes Have a Tree Structure

---

Letter	Binary
A	00
B	01
C	1

## Deterministic Codes Have a Tree Structure

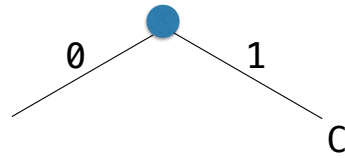
---



Letter	Binary
A	00
B	01
C	1

## Deterministic Codes Have a Tree Structure

---

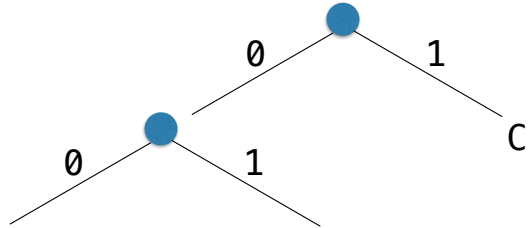


Letter	Binary
A	00
B	01
C	1



## Deterministic Codes Have a Tree Structure

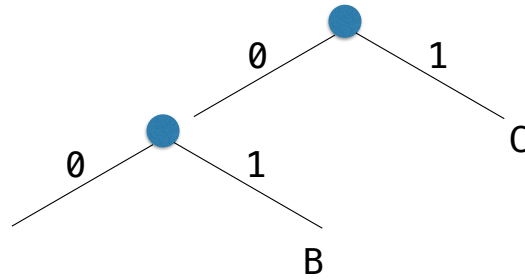
---



Letter	Binary
A	00
B	01
C	1

## Deterministic Codes Have a Tree Structure

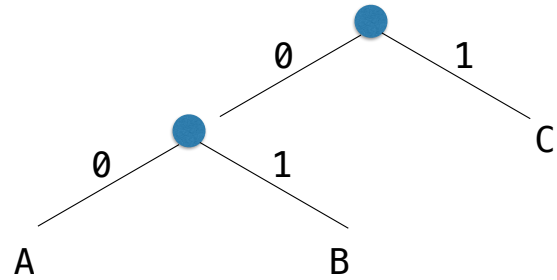
---



Letter	Binary
A	00
B	01
C	1

## Deterministic Codes Have a Tree Structure

---



Letter	Binary
A	00
B	01
C	1

# 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:
  - AAAAAAAAAABBBBBCCCCCCDDDDDDDD

## Huffman Encoding

---

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

## Huffman Encoding

---

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



## Huffman Encoding

---

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

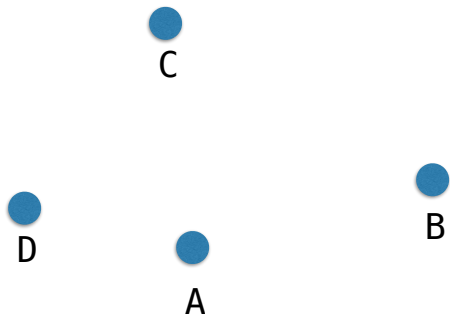
---

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

## Huffman Encoding

---

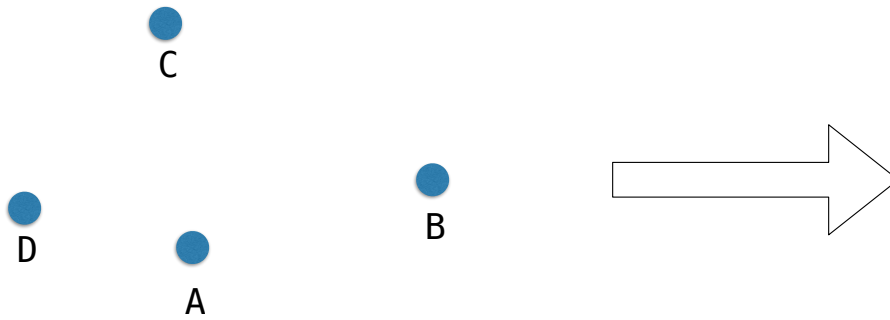
- Start with the two smallest frequencies
  - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times



## Huffman Encoding

---

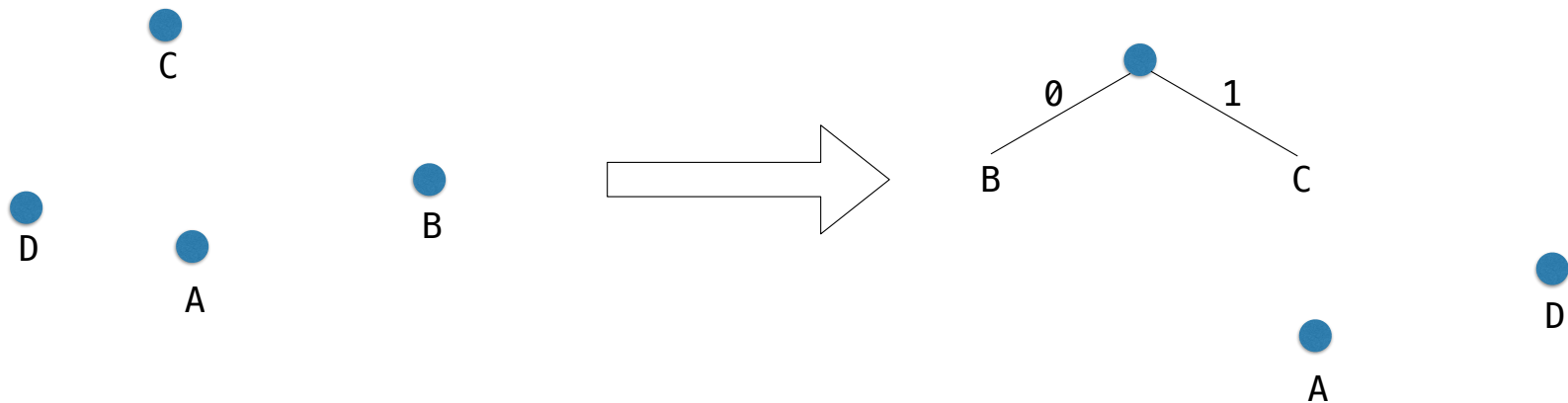
- Start with the two smallest frequencies
  - A appears 10 times, B appears 5 times, C appears 7 times, D appears 9 times



## 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

---

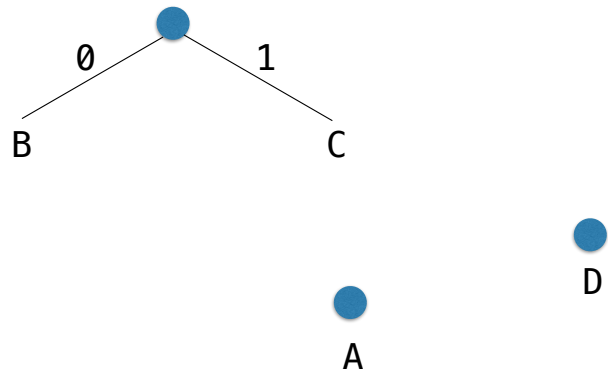
- Continue...
- A appears 10 times, B & C appear a combined 12 times, D appears 9 times

## Huffman Encoding

---

- Continue...

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

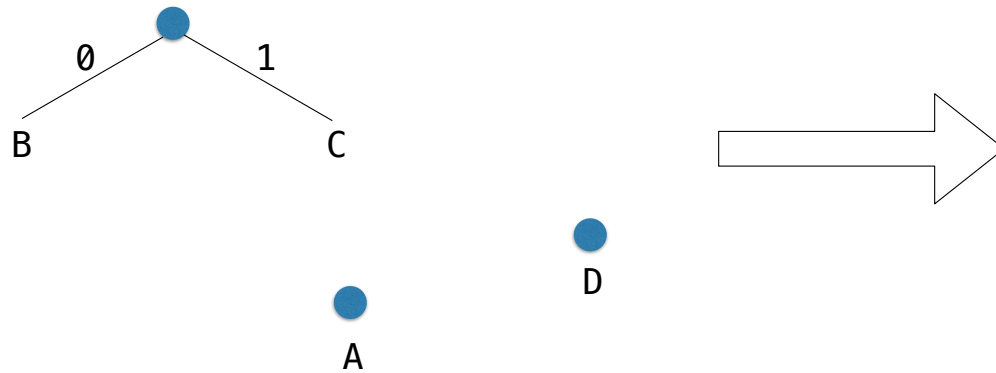


## Huffman Encoding

---

- Continue...

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

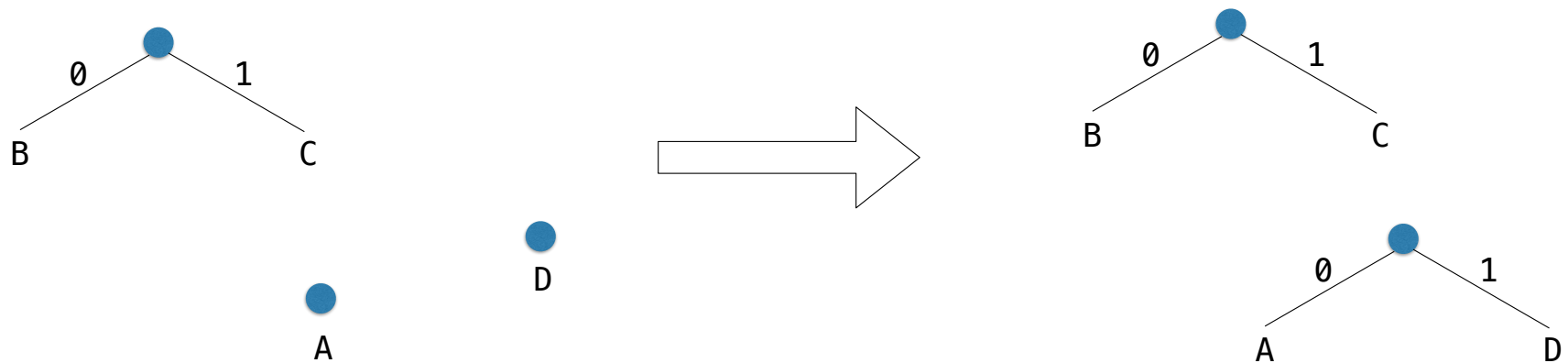


## Huffman Encoding

---

- Continue...

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



# Huffman Encoding

---

## Huffman Encoding

---

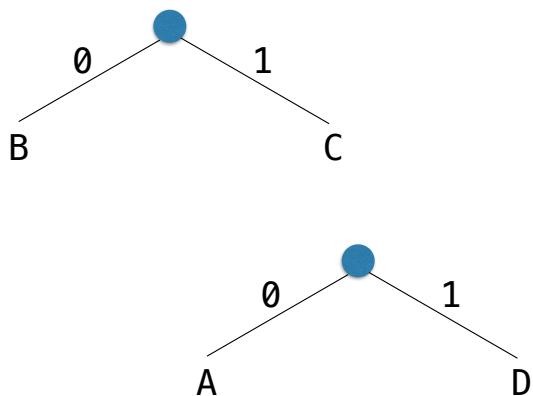
- And finally...



## Huffman Encoding

---

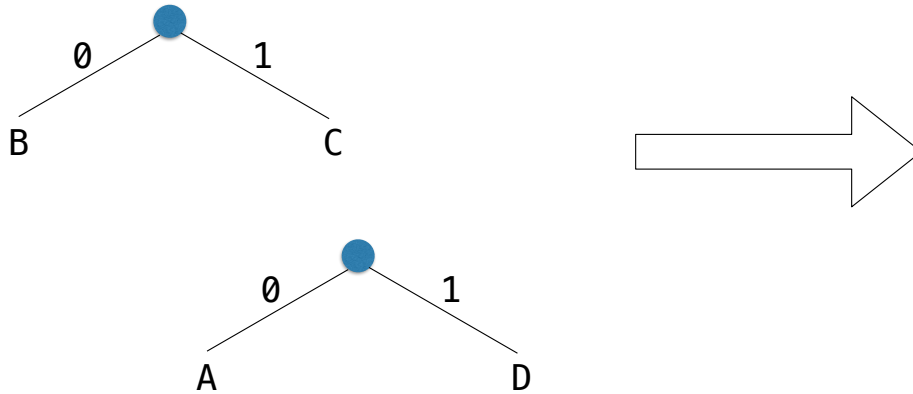
- And finally...



## Huffman Encoding

---

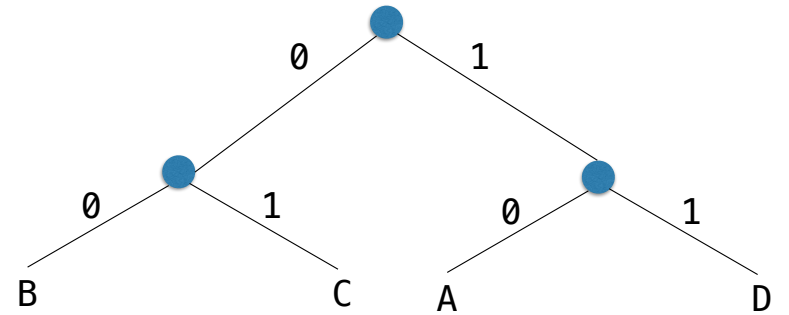
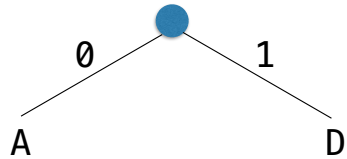
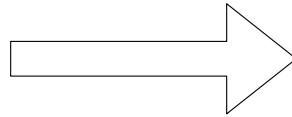
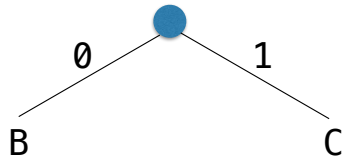
- And finally...



## 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

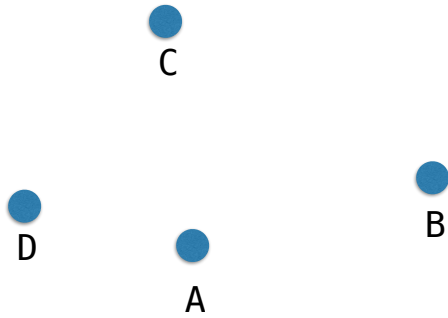
---

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

## Huffman Encoding

---

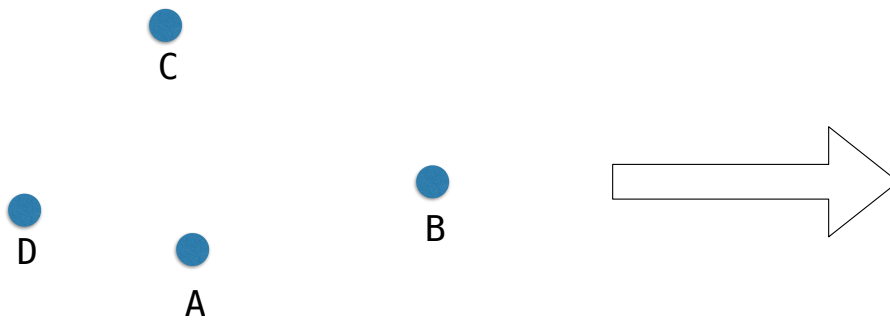
- Start with the two smallest frequencies
  - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time



## Huffman Encoding

---

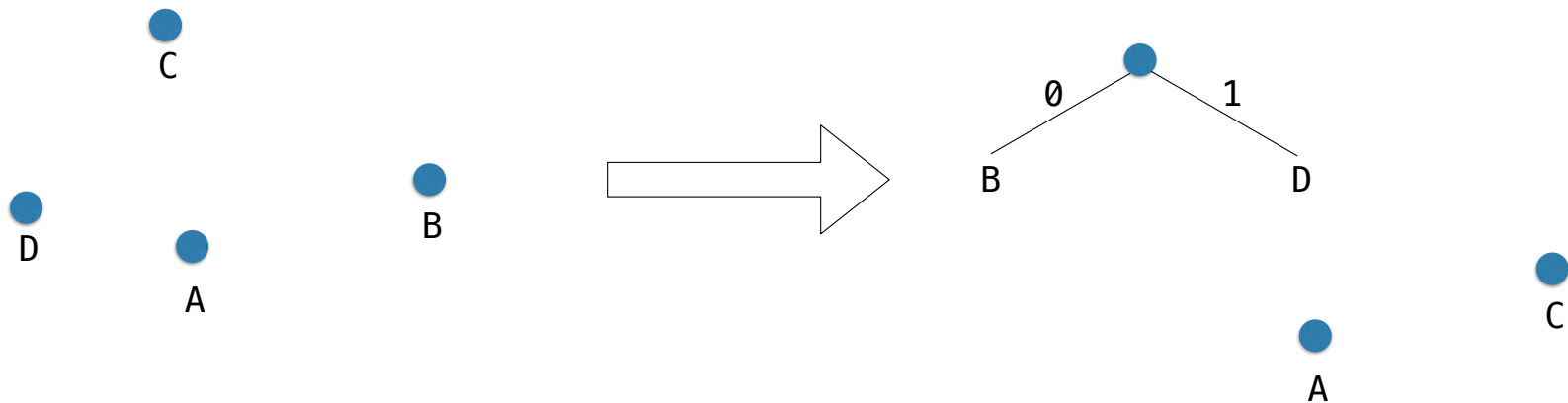
- Start with the two smallest frequencies
  - A appears 10 times, B appears 1 time, C appears 2 times, D appears 1 time



## 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

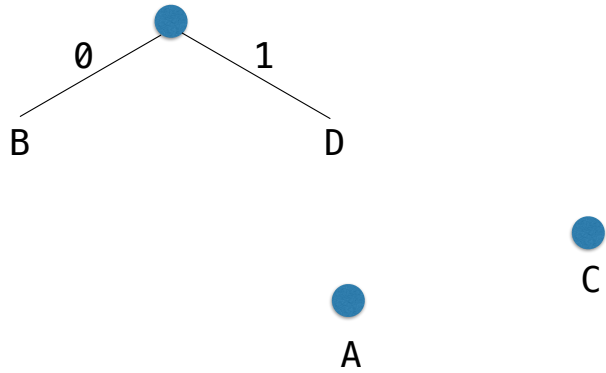
---

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

## Huffman Encoding

---

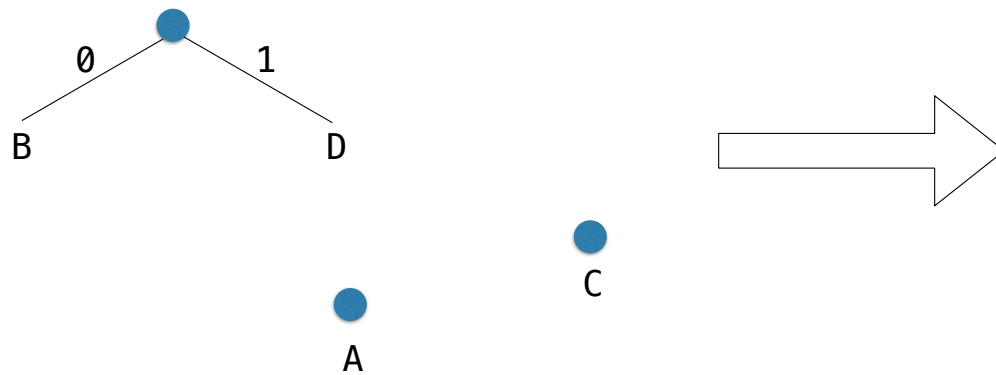
- Start with the two smallest frequencies
  - A appears 10 times, B & D appear a combined 2 times, C appears 2 times



## Huffman Encoding

---

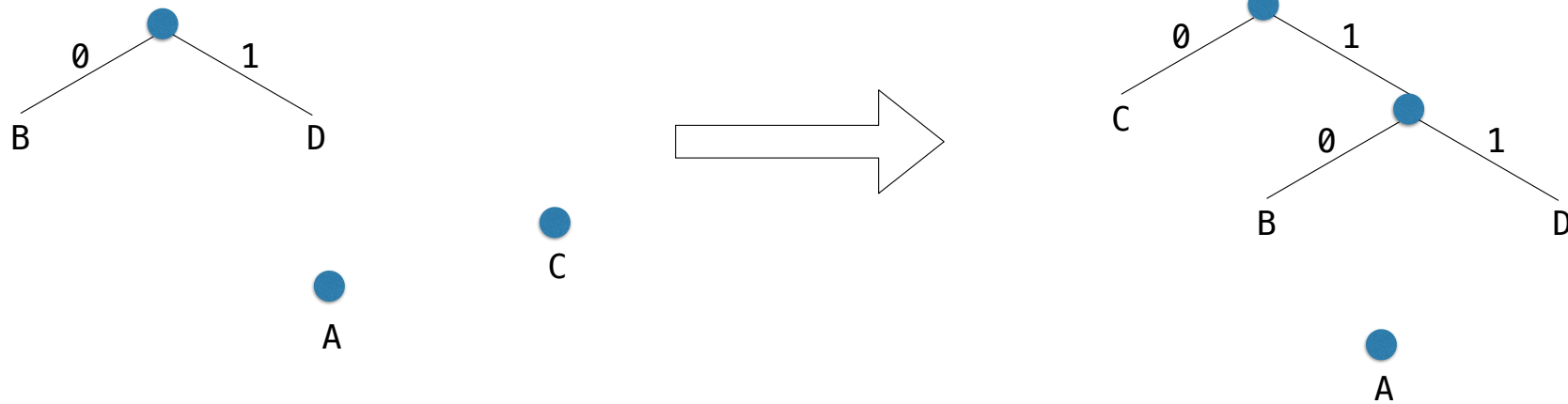
- Start with the two smallest frequencies
  - A appears 10 times, B & D appear a combined 2 times, C appears 2 times



## 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

---

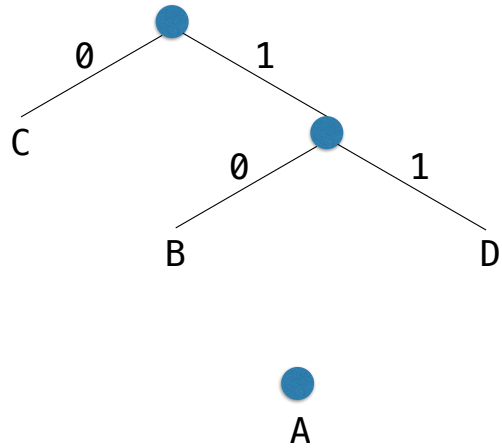
- And finally...



## Huffman Encoding

---

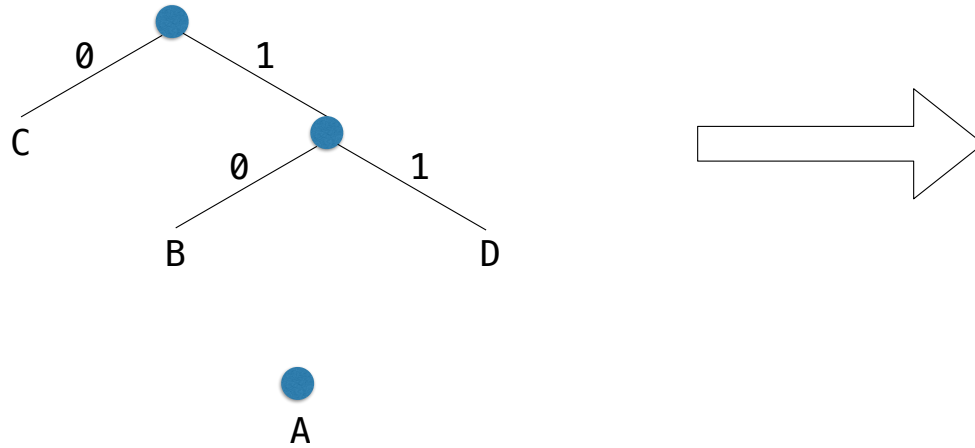
- And finally...



## Huffman Encoding

---

- And finally...



## Huffman Encoding

---

- And finally...

