CS 5112
Algorithms for Applications

Interval Scheduling (continued)
Huffman Codes

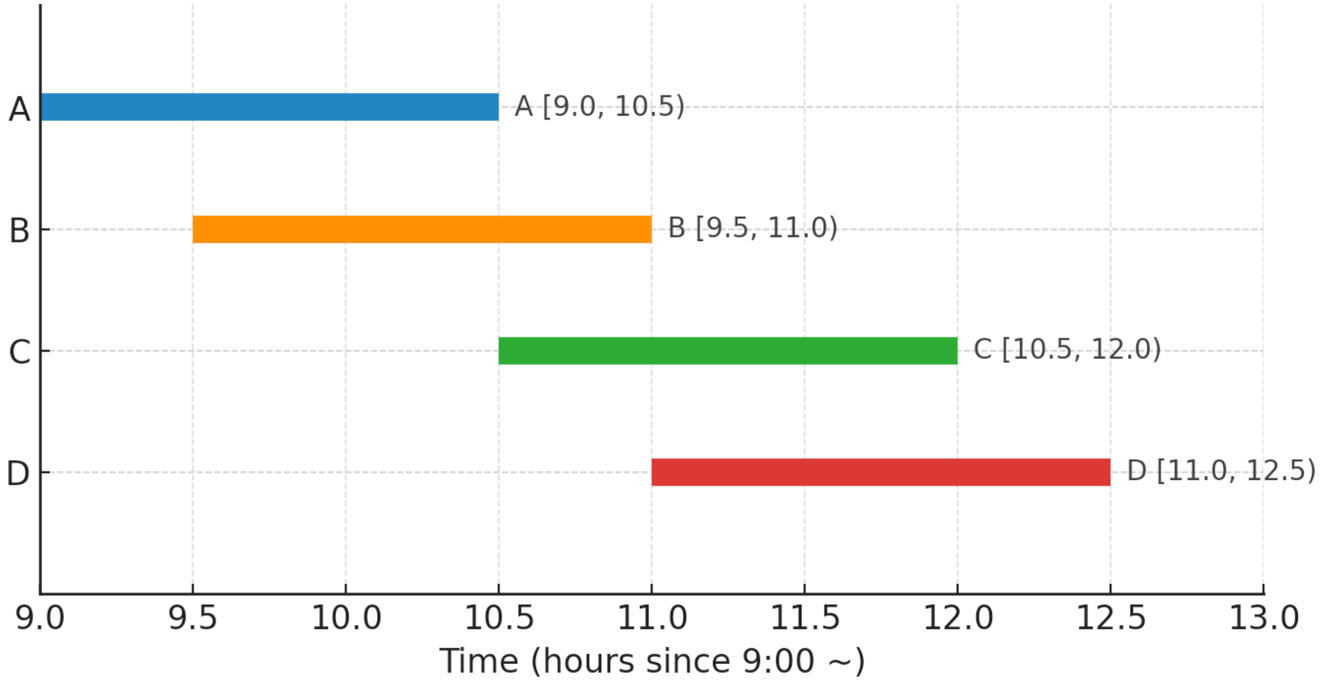
Interval Scheduling Problem

You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

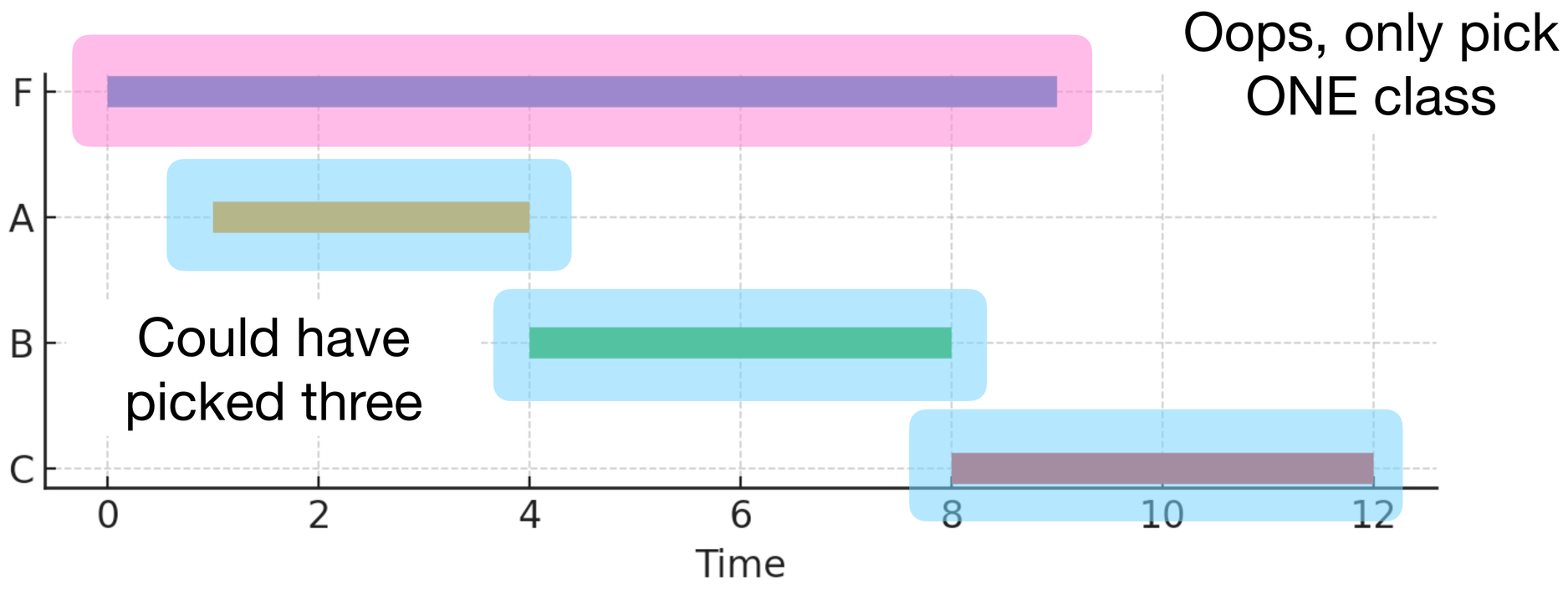
How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.



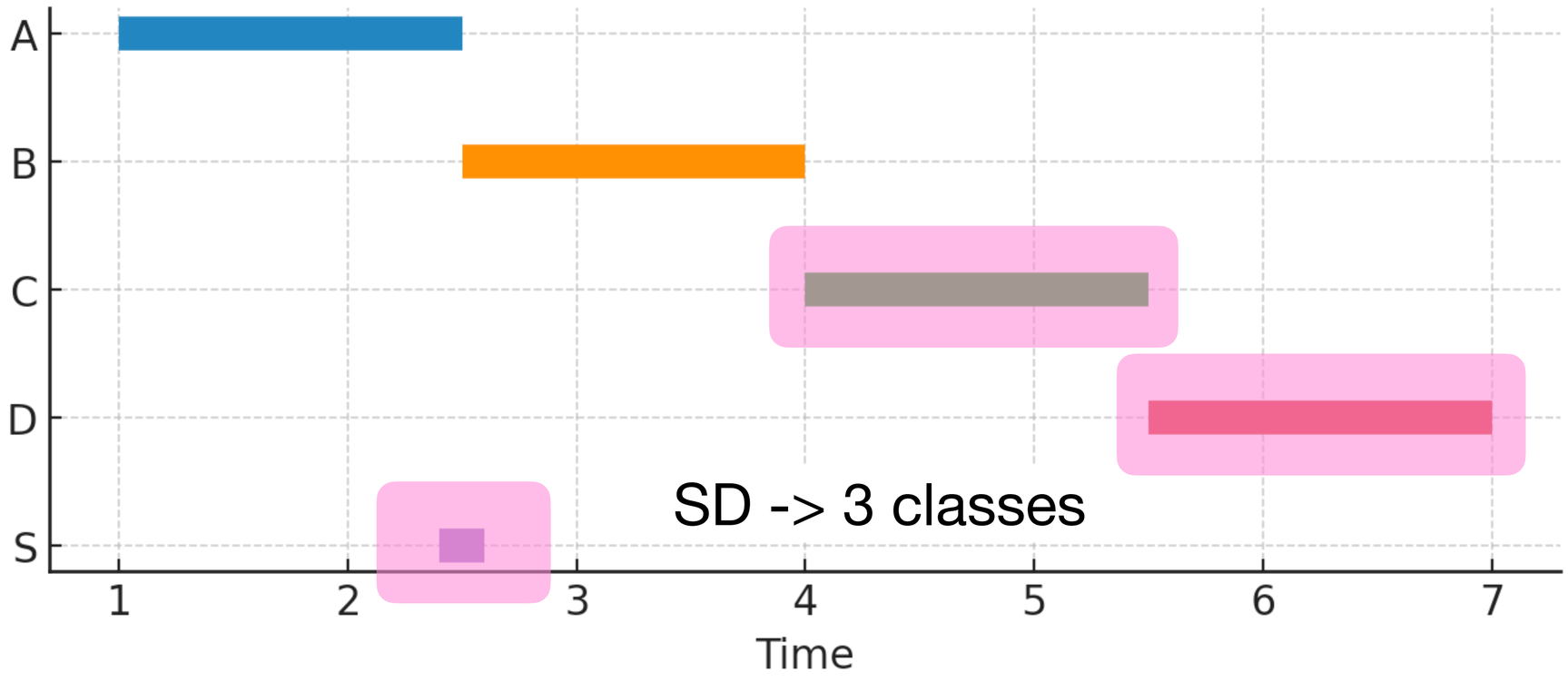
Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)



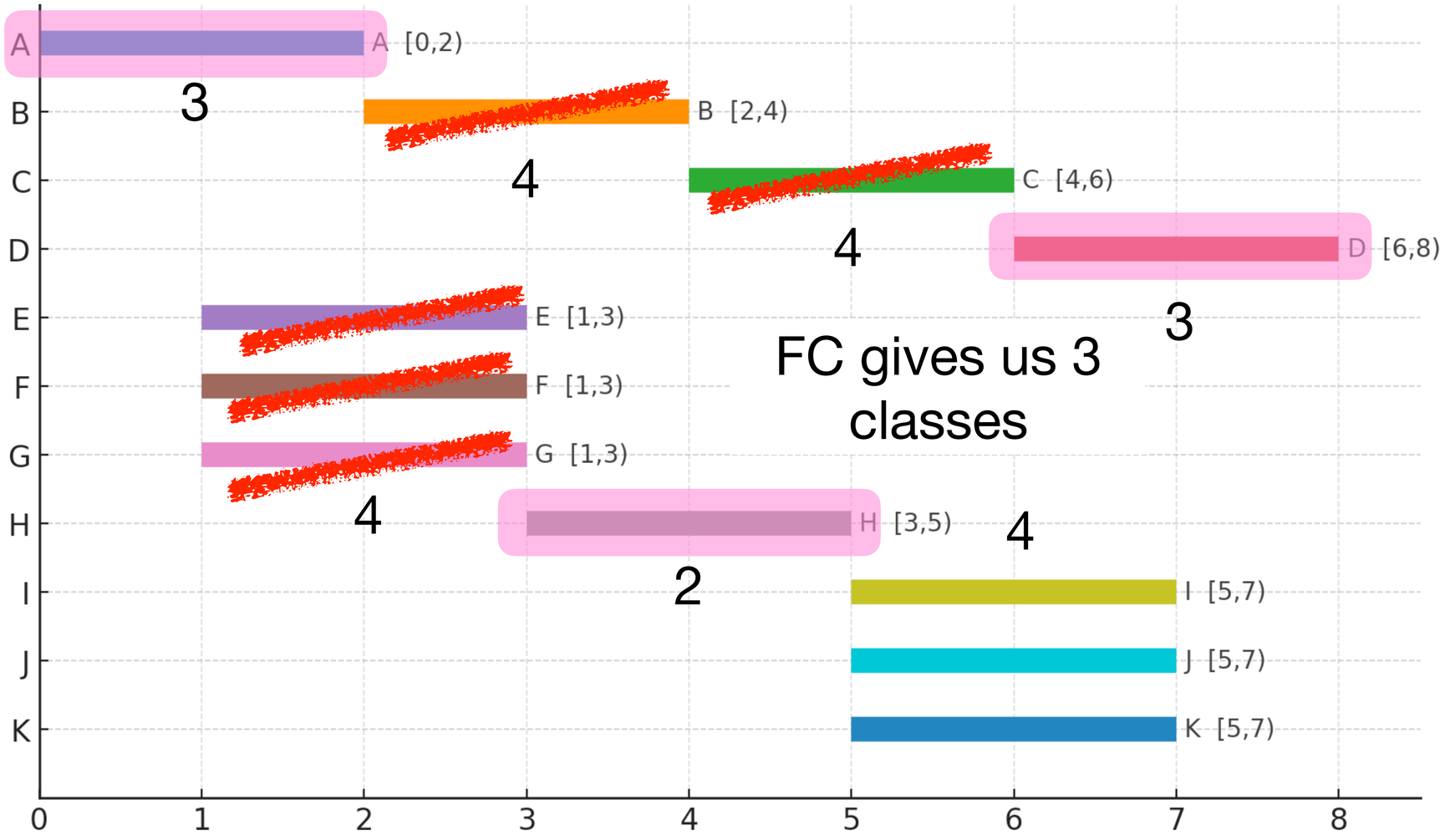
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



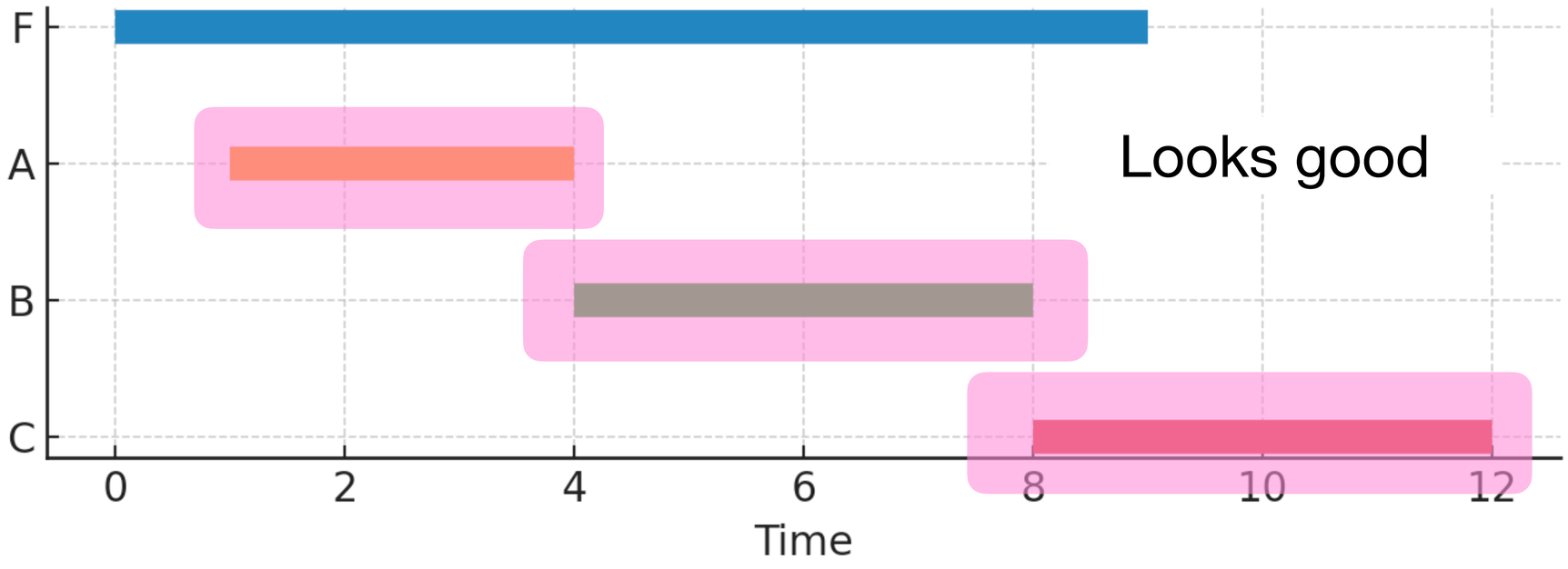
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



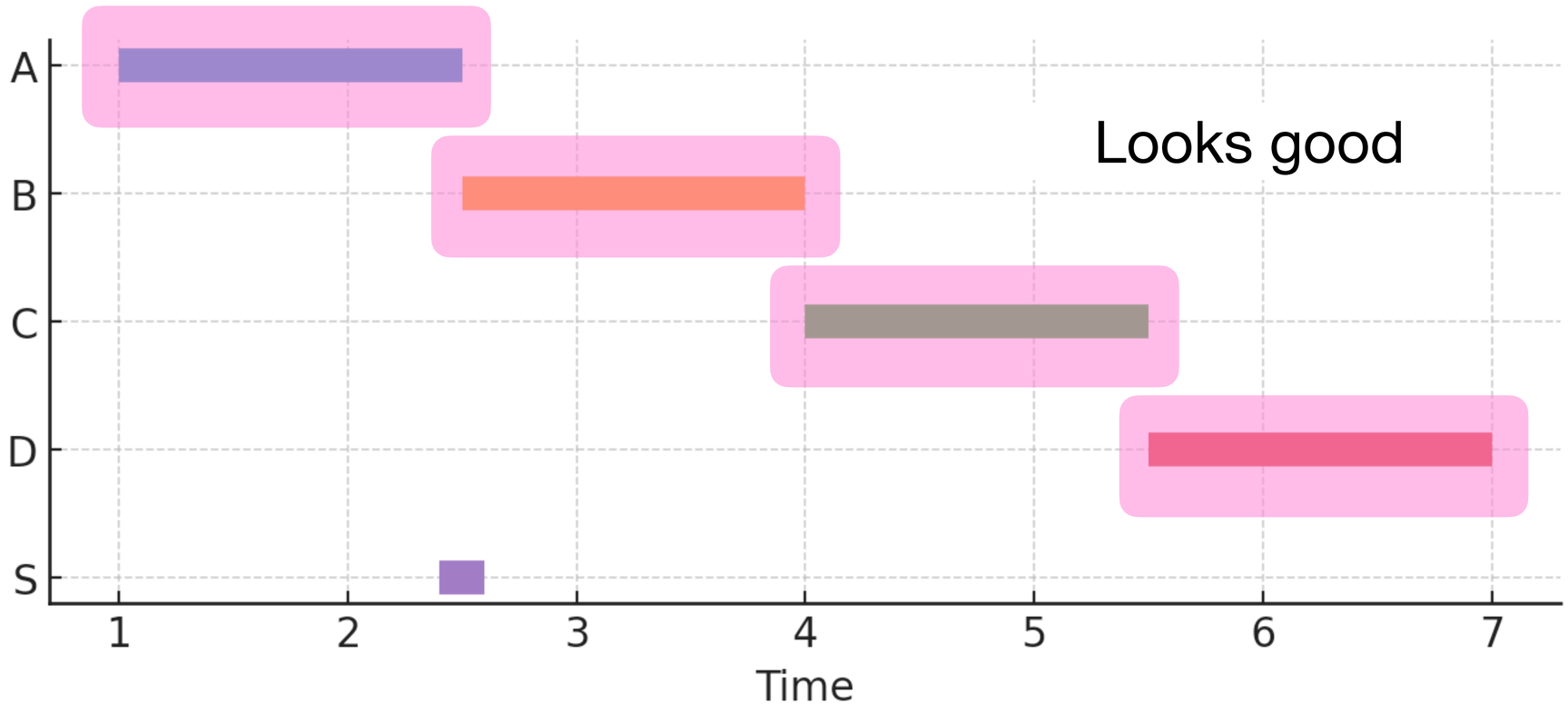
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



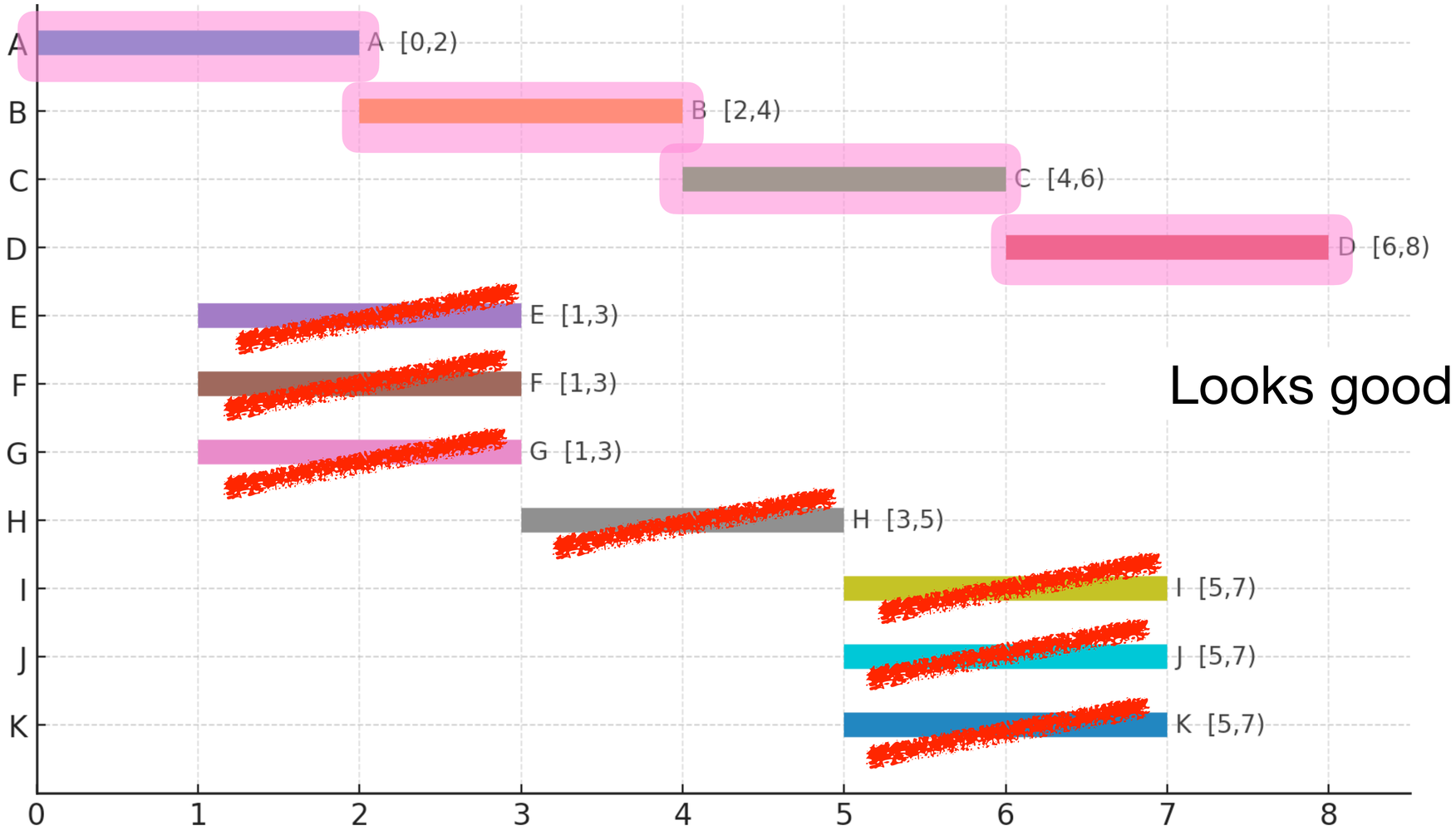
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



Looks good

Huffman Codes

Secret Message?

Secret Message?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

Secret Message?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

ASCII

Char	Binary
[space]	00100000
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101
F	01000110
G	01000111
H	01001000
I	01001001
J	01001010
K	01001011
L	01001100
M	01001101
N	01001110
O	01001111
P	01010000
Q	01010001
R	01010010
S	01010011
T	01010100
U	01010101
V	01010110
W	01010111
X	01011000
Y	01011001
Z	01011010
.	00101110
?	00111111
!	00100001

Secret Message?

01001001 001 01 01001110

00100000 010 11 00100000

01000010 0100 0 00111111

I CAN HAS BITZ?

ASCII

Char	Binary
[space]	00100000
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101
F	01000110
G	01000111
H	01001000
I	01001001
J	01001010
K	01001011
L	01001100
M	01001101
N	01001110
O	01001111
P	01010000
Q	01010001
R	01010010
S	01010011
T	01010100
U	01010101
V	01010110
W	01010111
X	01011000
Y	01011001
Z	01011010
.	00101110
?	00111111
!	00100001

Secret Message?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

It takes lots of bits to write this short message!

ASCII

Char	Binary
[space]	00100000
	01000001
	01000010

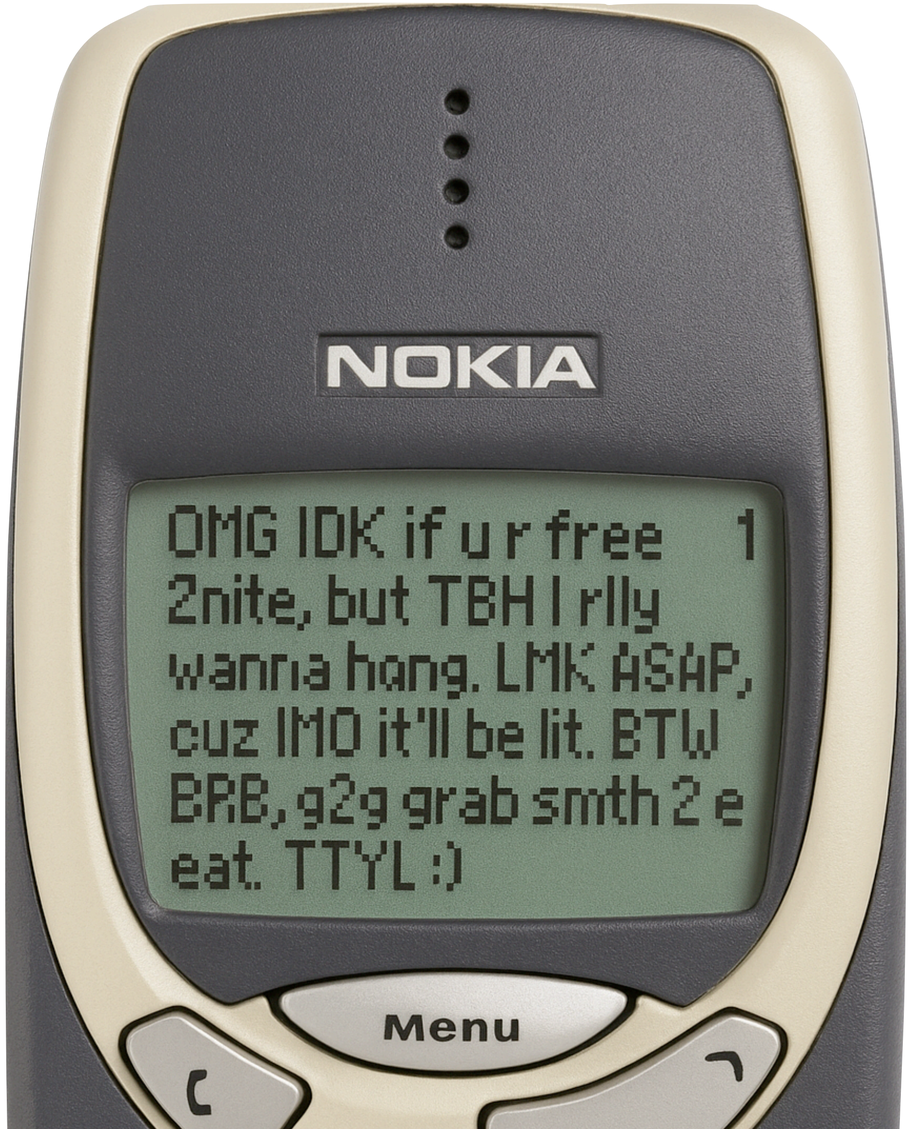
That’s like so long…

That’s like so long…

That’s like so long…

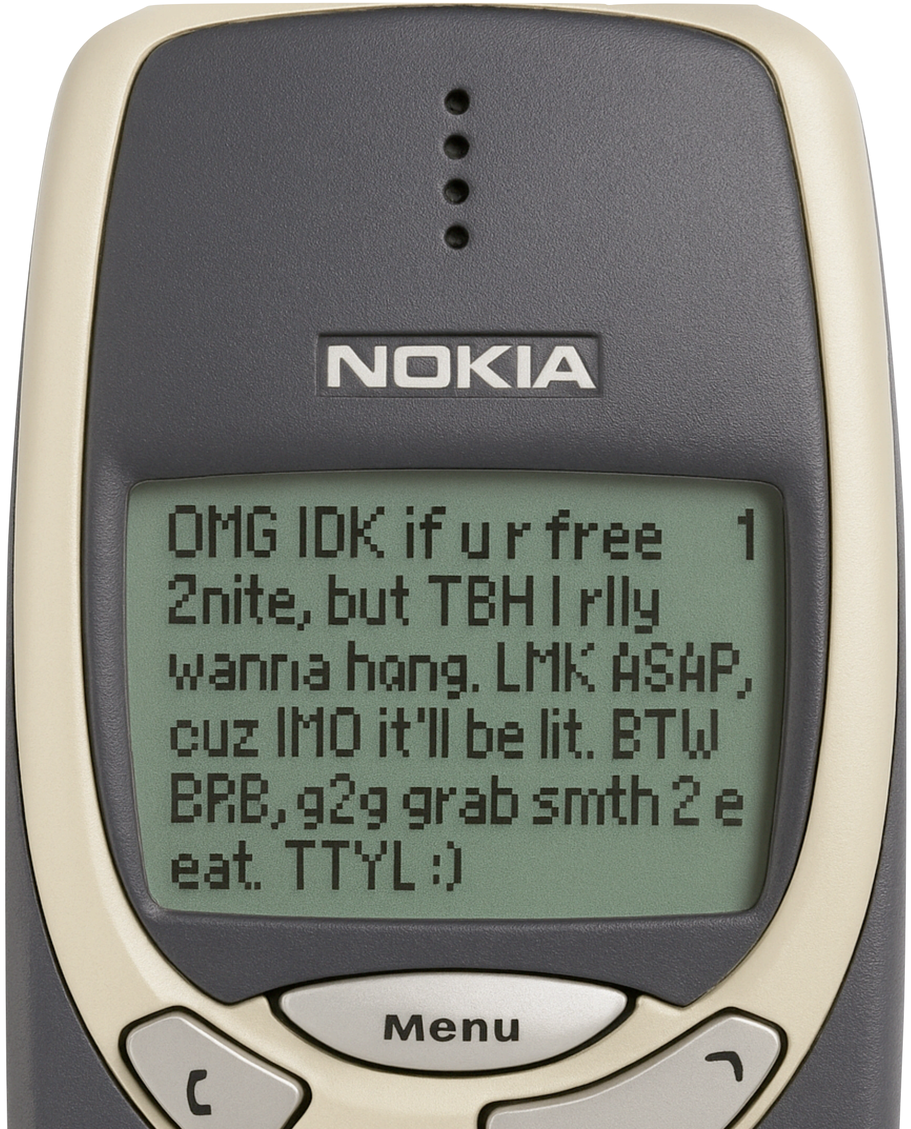
Sometimes we really care about how long messages are

Texts used to be expensive



In 1999…

Texts used to be expensive



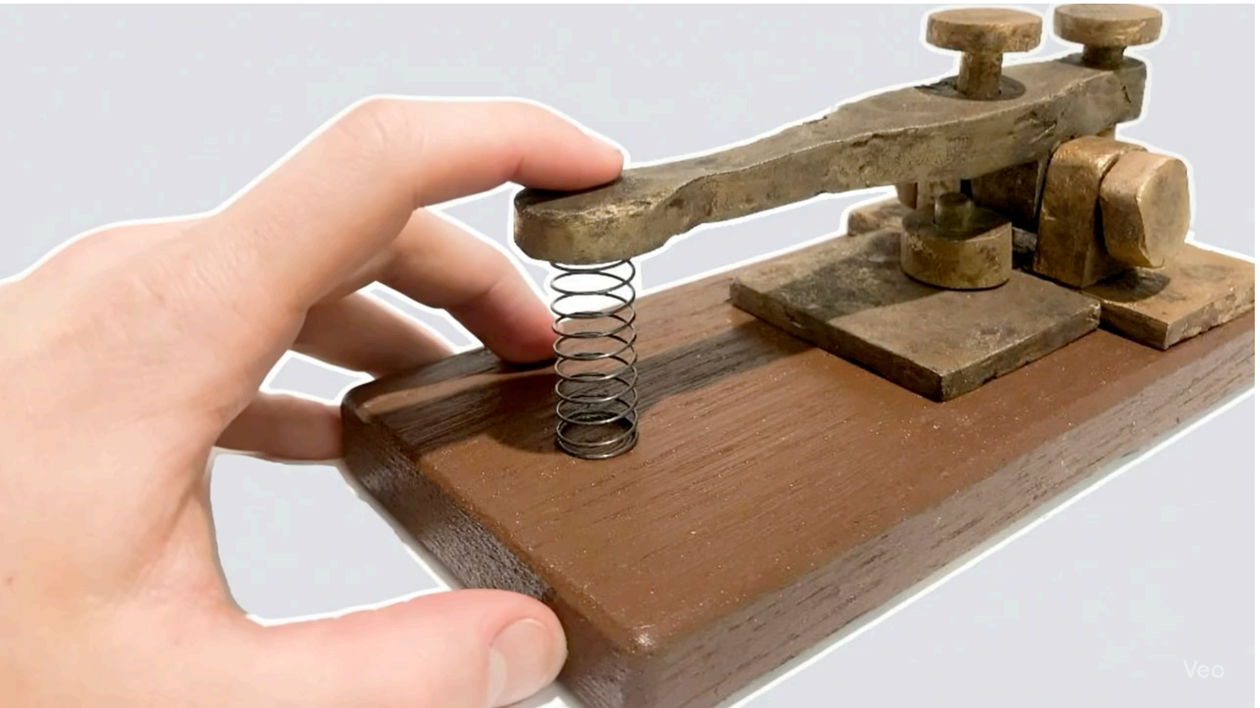
In 1999…

Each text was 160 characters

10¢ to send

10¢ to receive

Telegrams were worse…

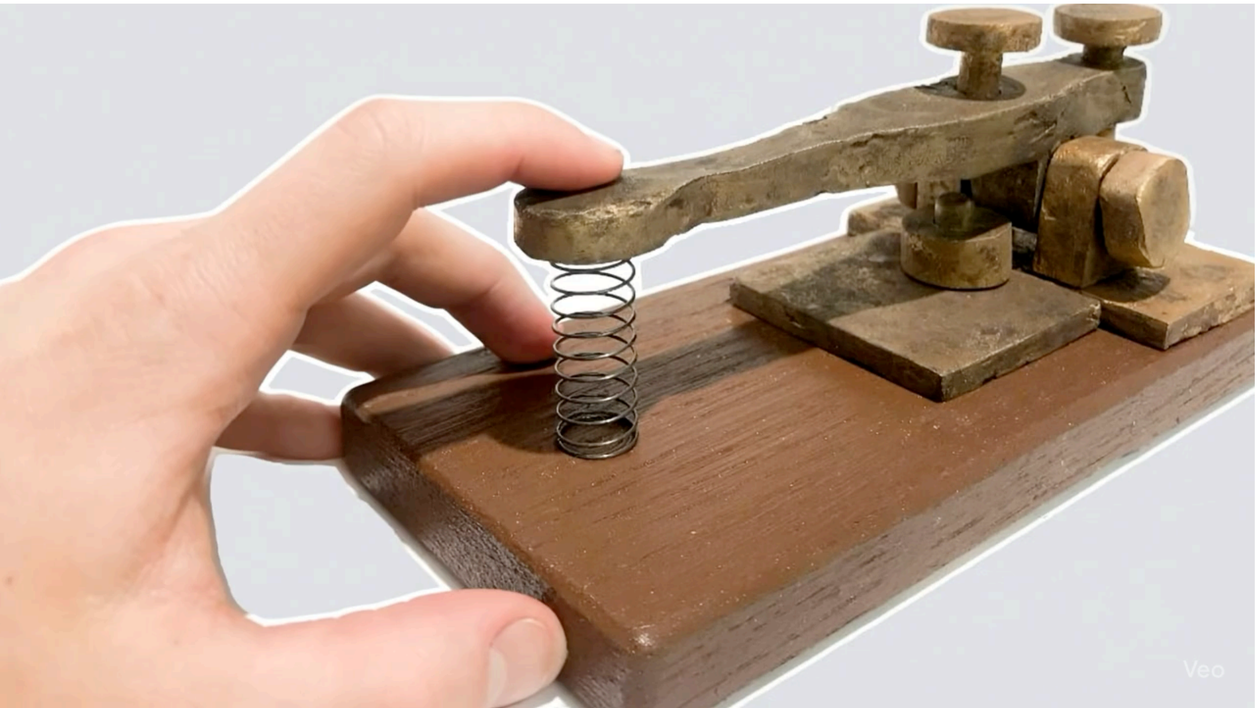


The first transatlantic telegram was $10/word

A word was defined as 5 consecutive characters

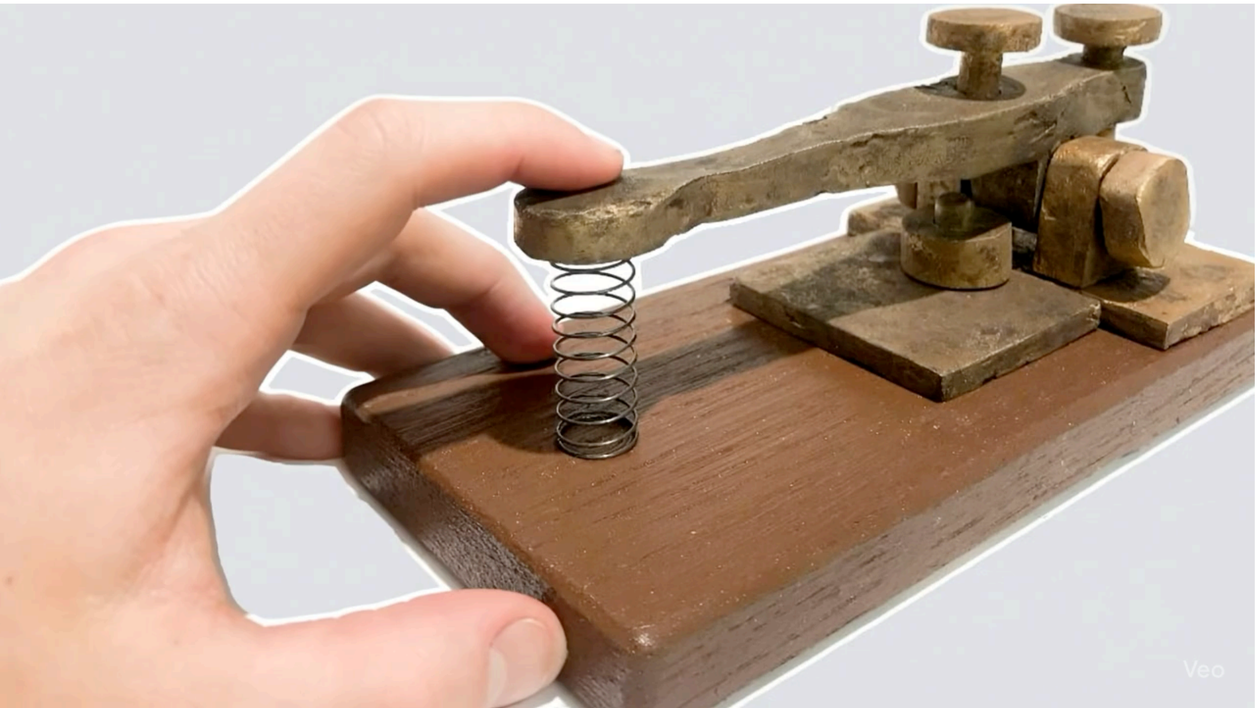
Morse Code

Telegrams operate by sending tones, and characters were encoded using a mix of short and long tones



Morse Code

Telegrams operate by sending tones, and characters were encoded using a mix of short and long tones



Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Morse Code?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

I CAN HAS BITZ?

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Morse Code?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

•• / –•–• •– –• / •••• •– ••• / –••• •• – ––•• ••––••

I CAN HAS BITZ?

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

•• / –•–• •– –• / •••• •– ••• / –••• •• – ––•• ••––••

I CAN HAS BITZ?

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

•• / –•–• •– –• / •••• •– ••• / –••• •• – ––•• ••––••

I CAN HAS BITZ?

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

Common letters have short encodings

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

Common letters have short encodings

Less common letters have longer encodings

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

Common letters have short encodings

Less common letters have longer encodings

Punctuation was super long and rarely used

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Why is Morse Code so much shorter?

Common letters have short encodings

Less common letters have longer encodings

Punctuation was super long and rarely used

Could you do better?

Char	Morse Code
[space]	[space]
A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
?	••––••
!	–•–•––

Efficient Text Encoding Problem

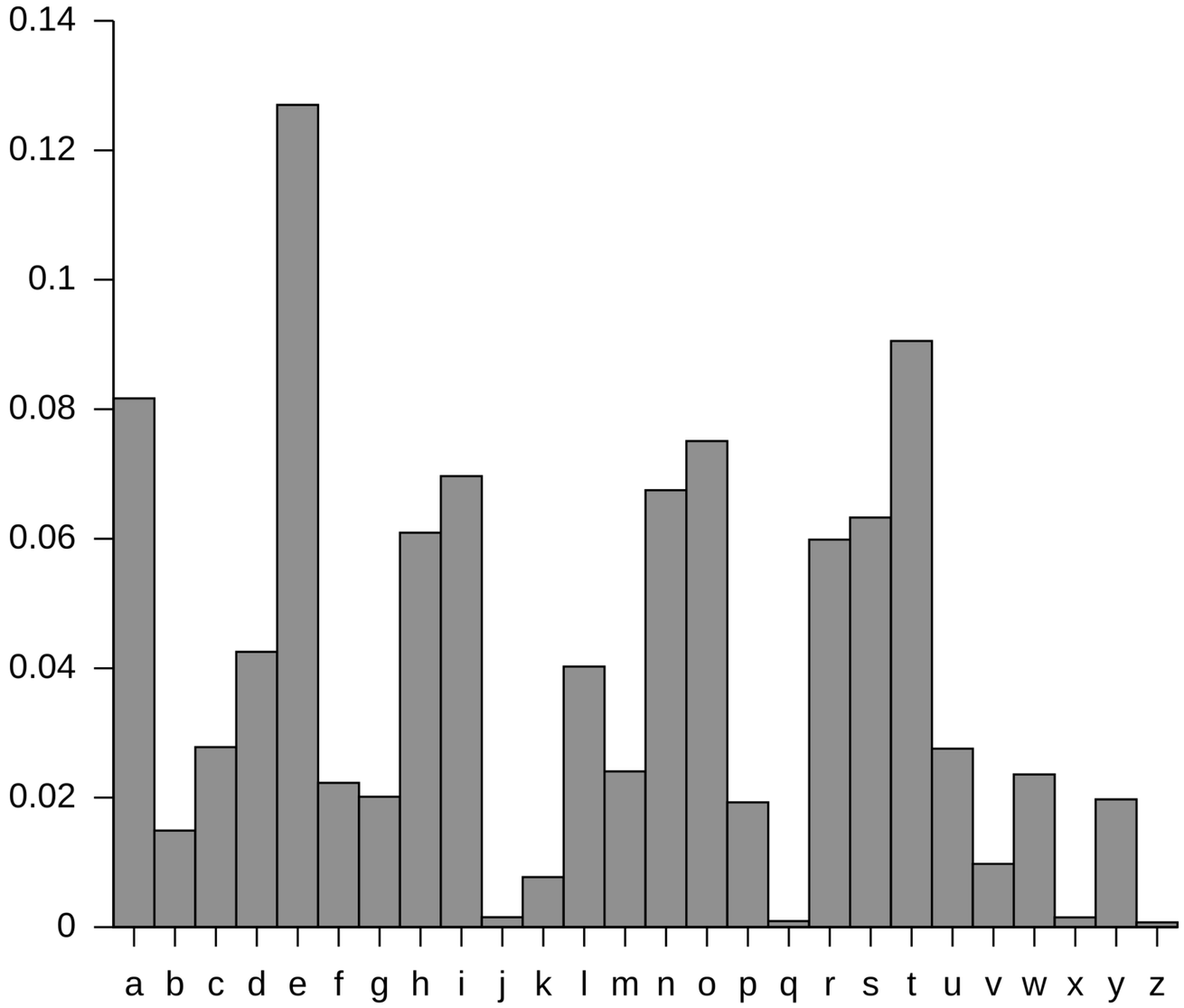
Want to encode text or other symbols
for storage, transmission, etc.

Input:
Set of symbols with
known frequencies

Output:
Assign a variable-length
codeword to each symbol

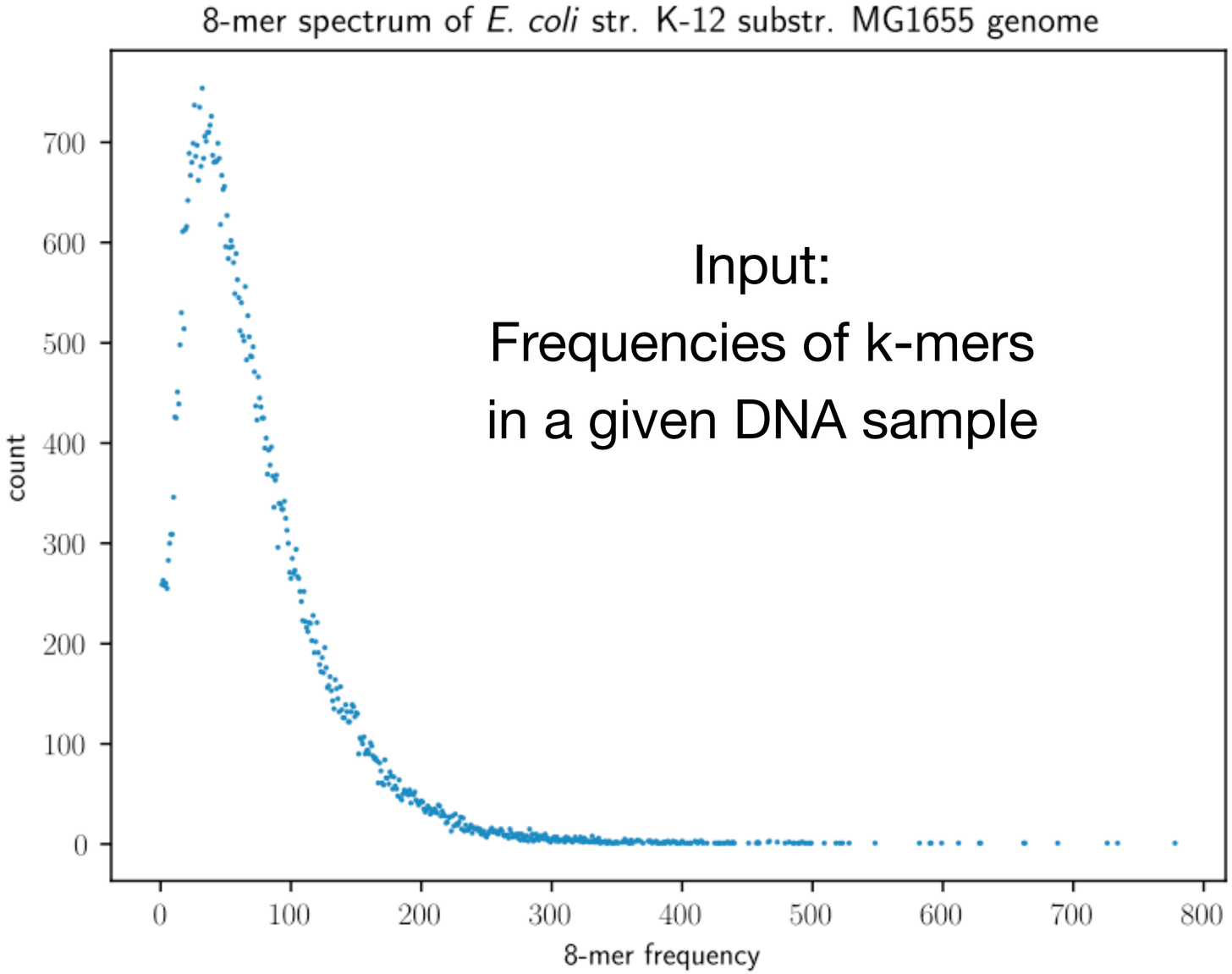
Goal:
Minimize the average
message length

Example: English Encoding



Input:
Frequencies of each letter in common English use

Example: Encoding DNA



Input:
Frequencies of k-mers
in a given DNA sample

DNA consists of 4 nucleotides: A, T, G, C

It is often represented as k-mers: k-length strings of nucleotides

Some 8-mers:
GTAGAGCT, TAGAGCTG, AGAGCTGT

Prefix Codes

How do you tell when a codeword ends?

01001001 00100000 01000011 01000001 01001110

00100000 01001000 01000001 01010011 00100000

01000010 01001001 01010100 01011010 00111111

ASCII is fixed-length, so it’s easy, but can’t take advantage of frequencies

•• / –•–• •– –• / •••• •– ••• / –••• •• – ––•• ••––••

Morse needs a special character, the pause

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Alex: This example is broken (not a prefix code!)

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011110110111101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011110110111101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111011011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

00011010110111011011101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101101110111101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111011011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111011011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111011011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101110101011111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

A prefix code is a code where no codeword is a prefix of another

HELLO WORLD

0001101011011101101111010111111

Alex: This example is broken (not a prefix code!)

H → 00
E → 01
L → 10
O → 110
W → 1110
R → 11110
D → 11111
(space) → 111110

Prefix Codes

How do you tell when a codeword ends?

What happens if we don't use a prefix code?

HELLO WORLD

0011010111101111110101111

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Prefix Codes

How do you tell when a codeword ends?

What happens if we don't use a prefix code?

HELLO WORLD

0011010111101111110101111

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Prefix Codes

How do you tell when a codeword ends?

What happens if we don't use a prefix code?

HELLO WORLD

0011010111101111110101111

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Prefix Codes

How do you tell when a codeword ends?

What happens if we don't use a prefix code?

HELLO WORLD

0011010111101111110101111

01

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Prefix Codes

How do you tell when a codeword ends?

What happens if we don't use a prefix code?

HELLO WORLD

0011010111101111110101111

01

HH or HE ??

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Efficient Text Encoding Problem

Want to encode text or other symbols
for storage, transmission, etc.

Input:
Set of symbols with
known frequencies

Output:
Assign a ~~variable-length~~
~~codeword~~ to each symbol

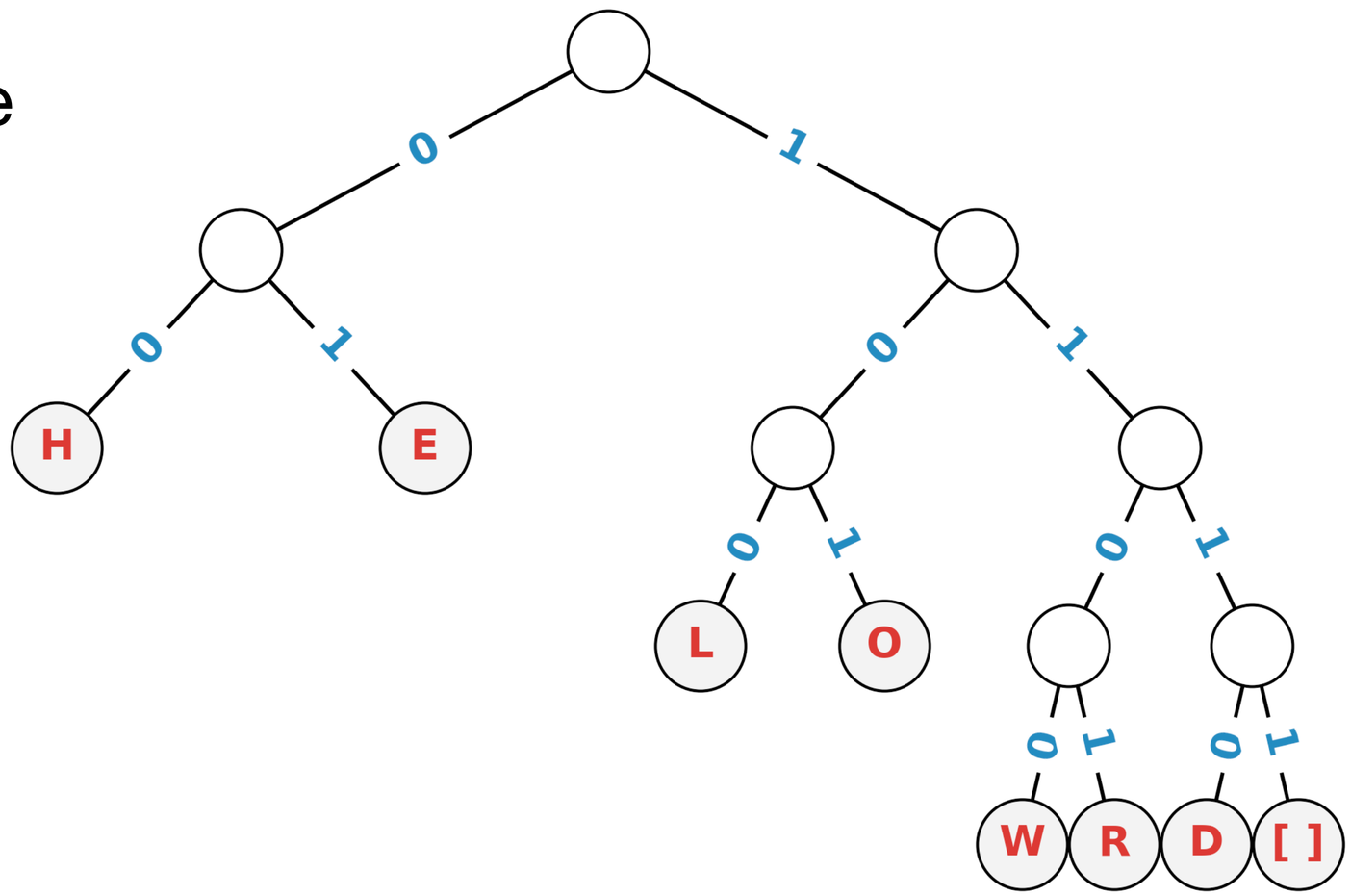
A prefix code

Goal:
Minimize the average
message length

Codes and Binary Trees

Given a code, can construct a binary tree

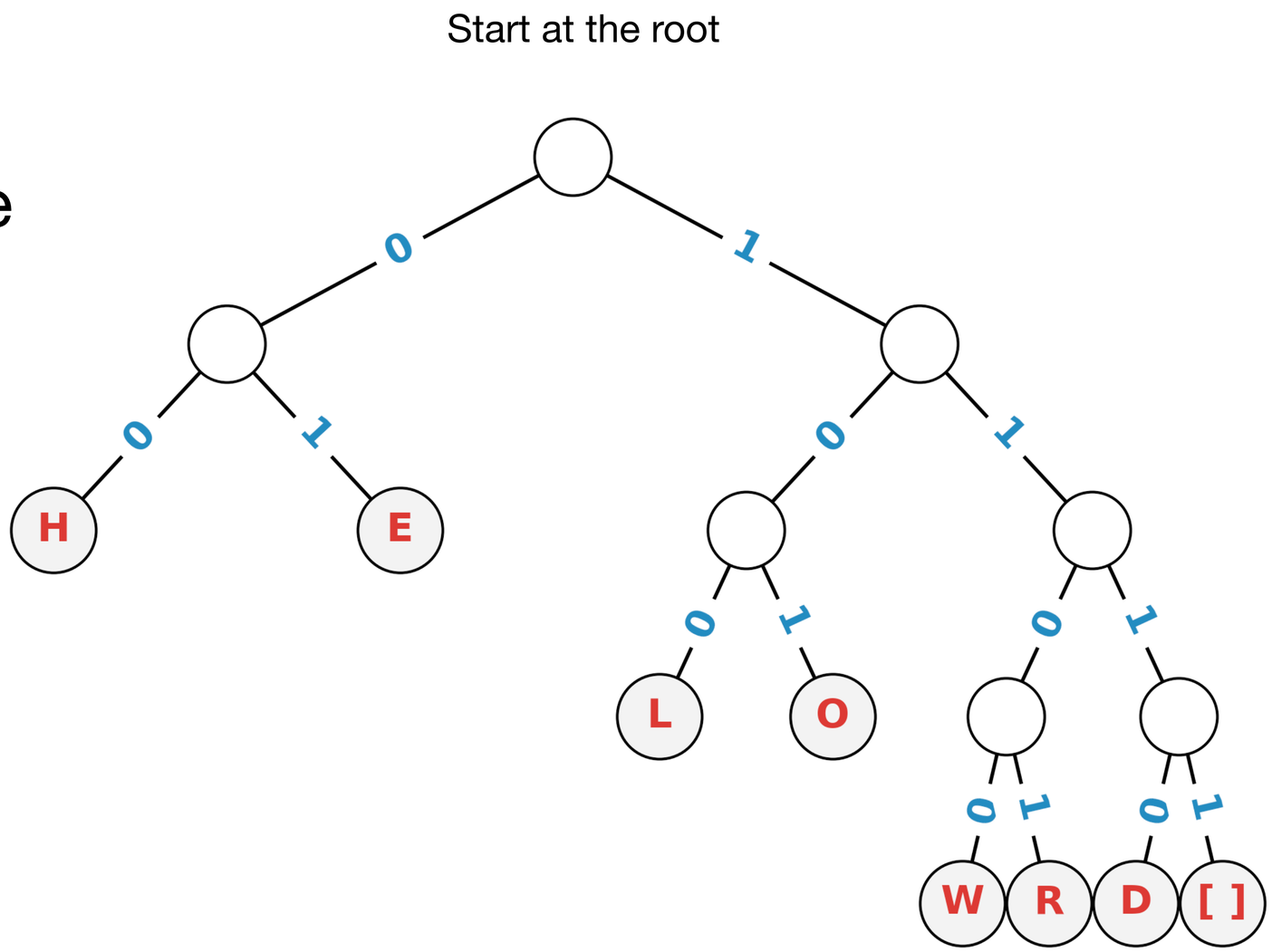
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

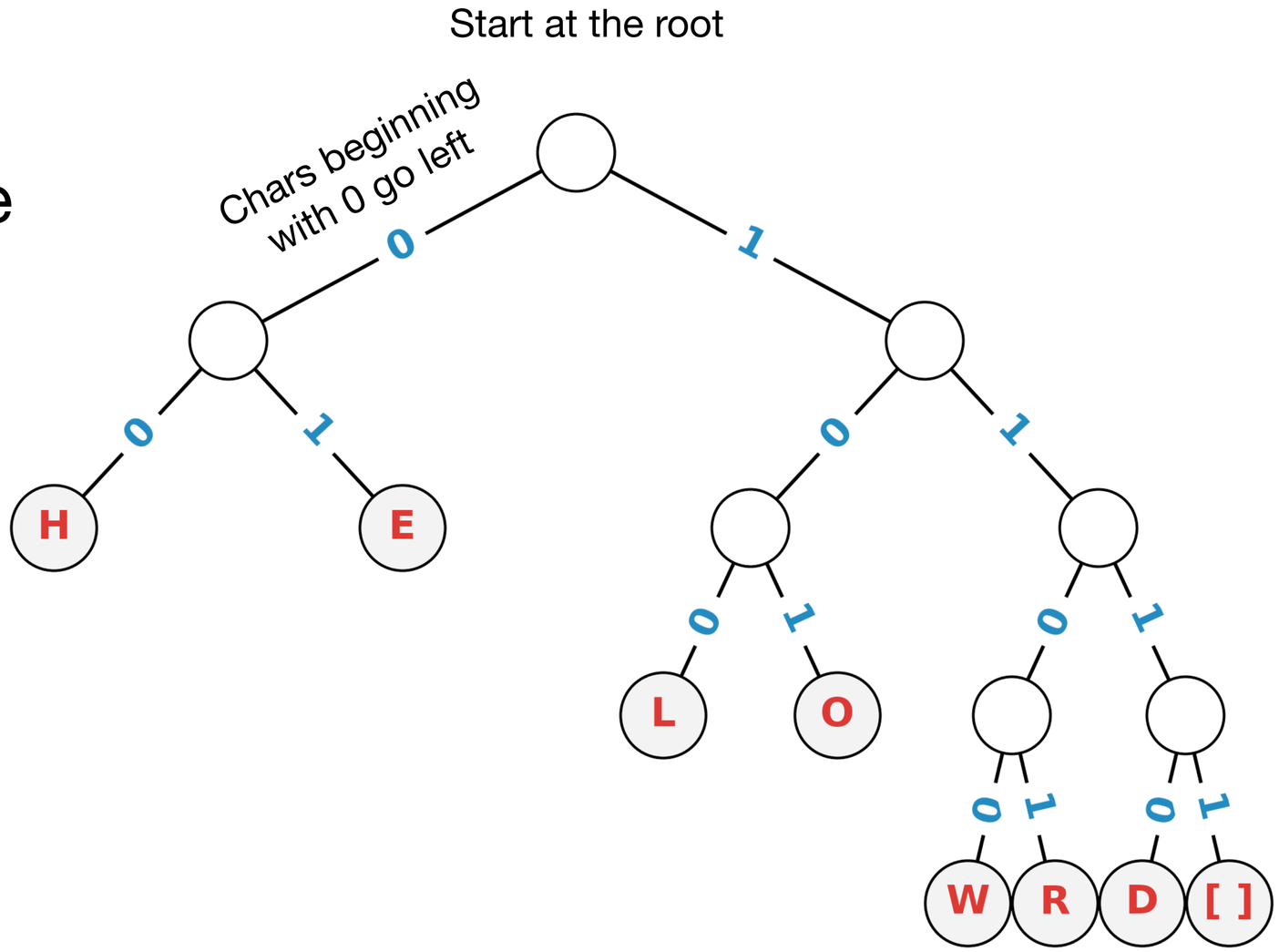
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

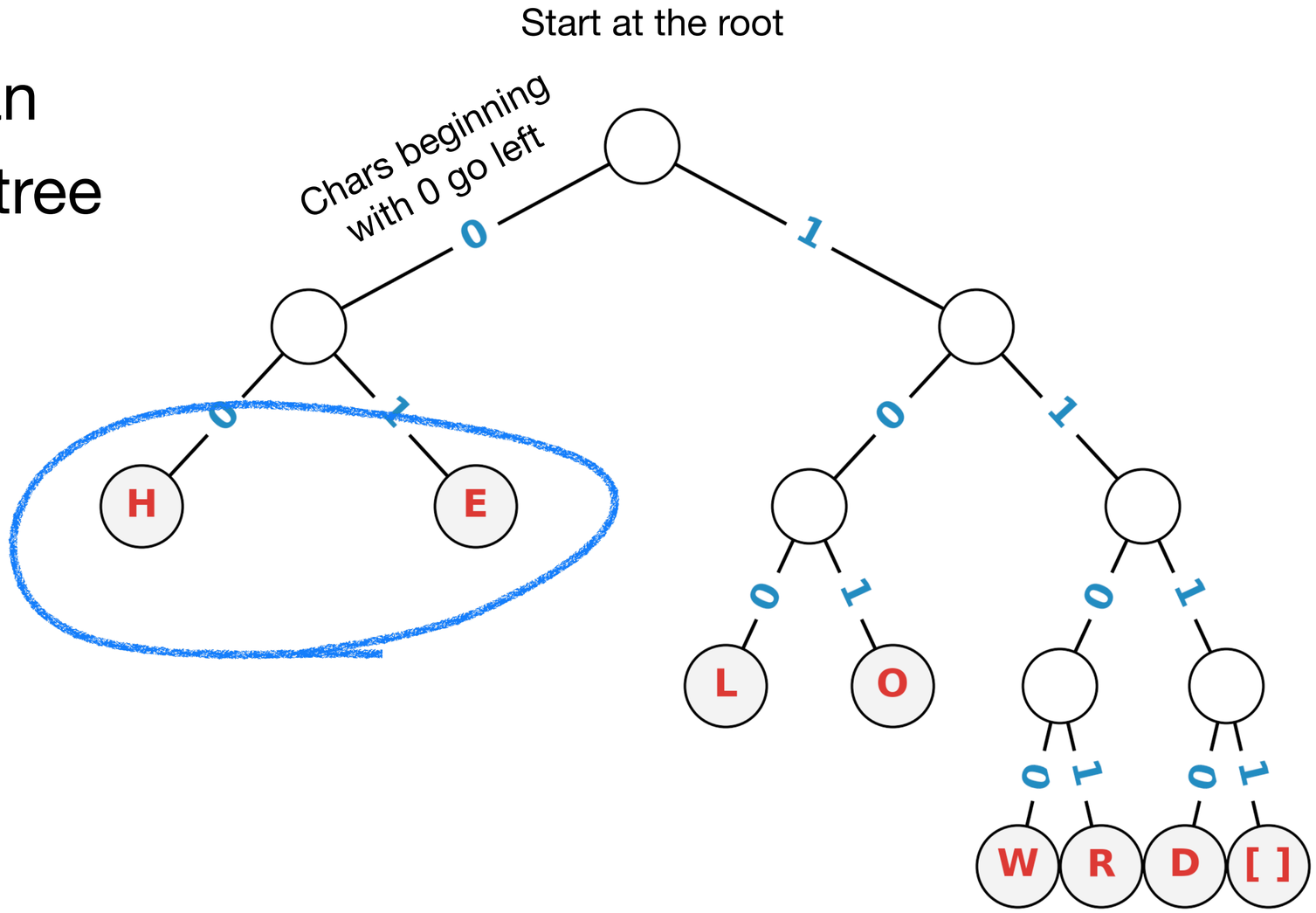
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

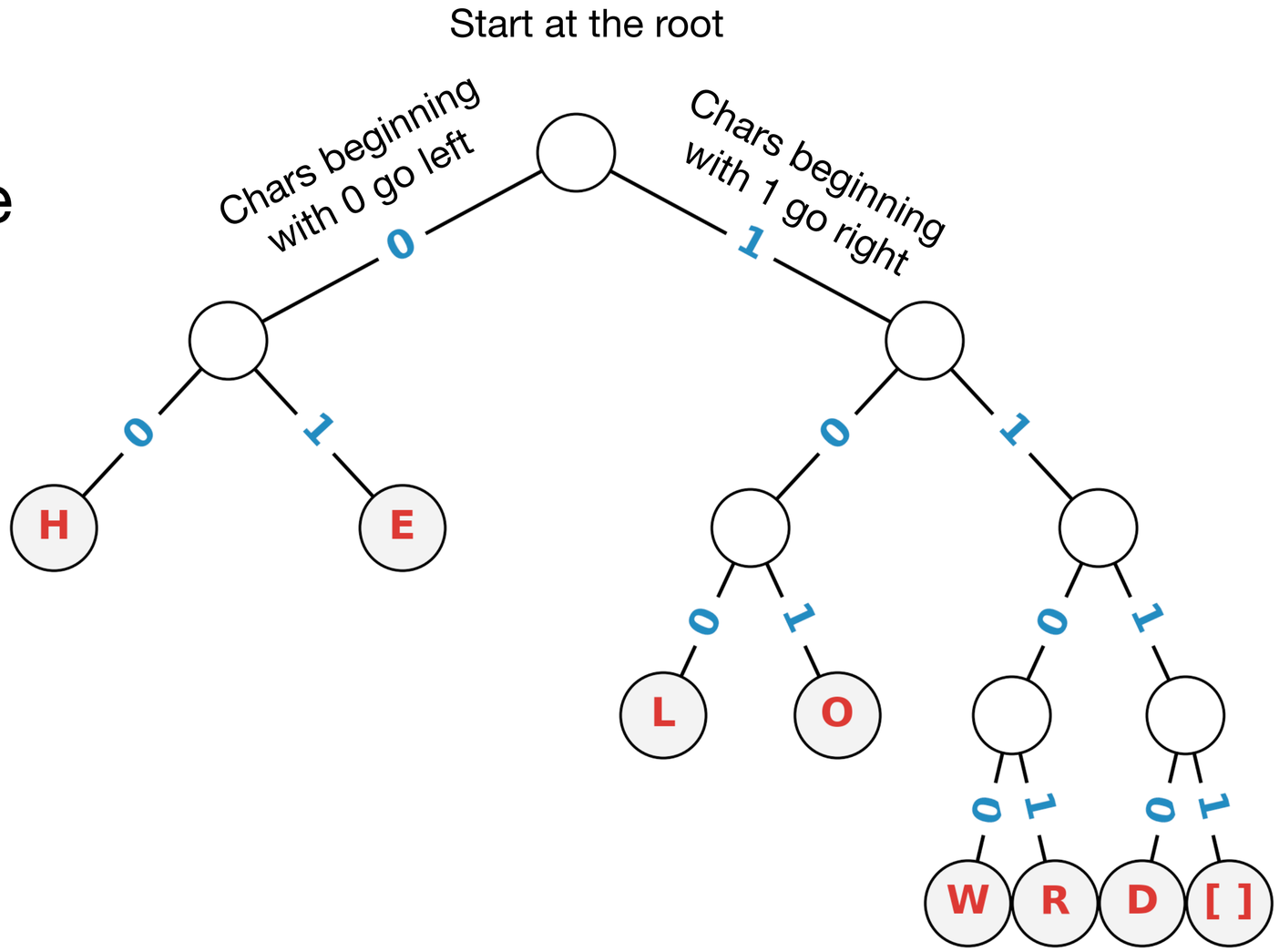
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

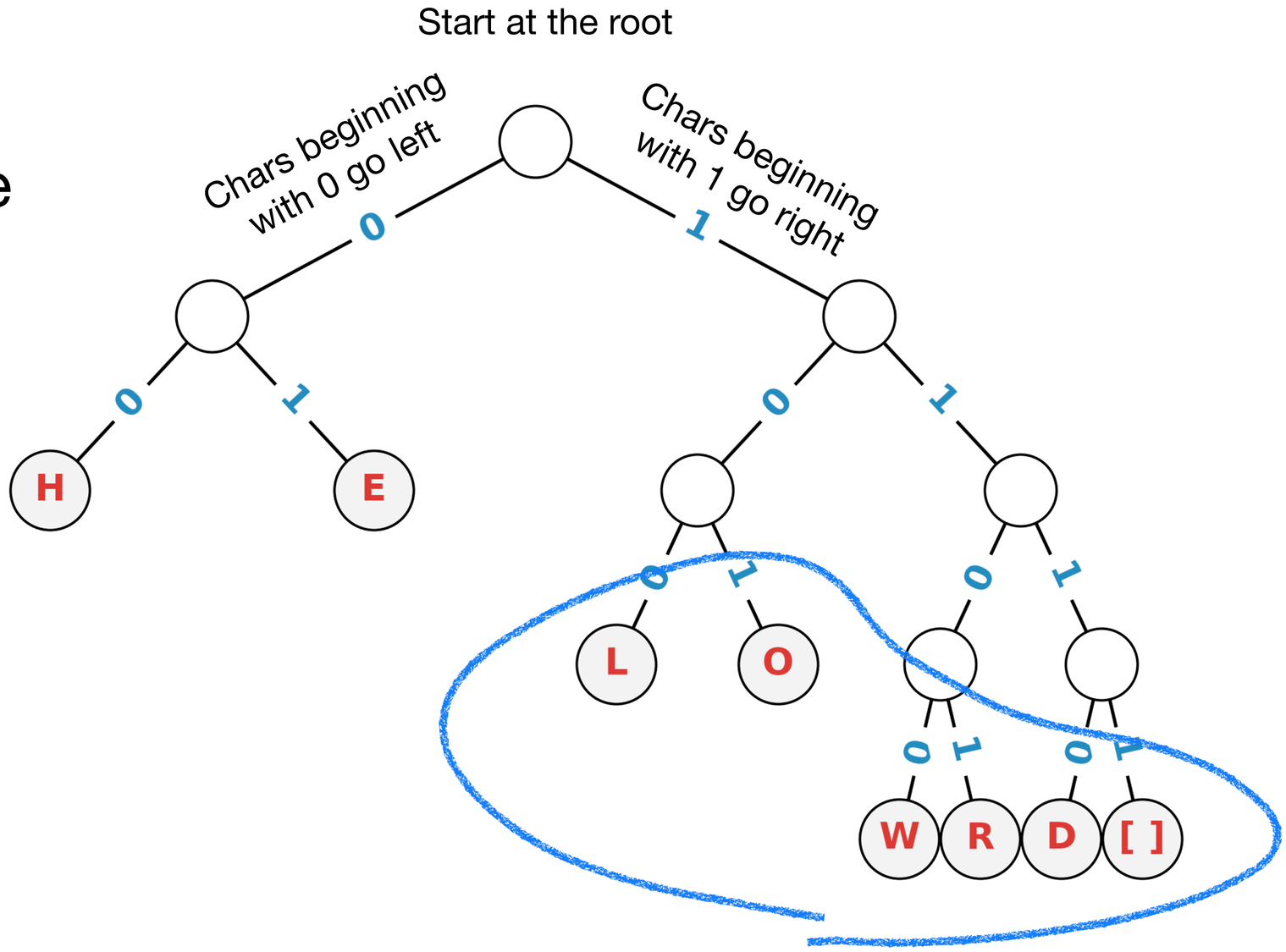
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

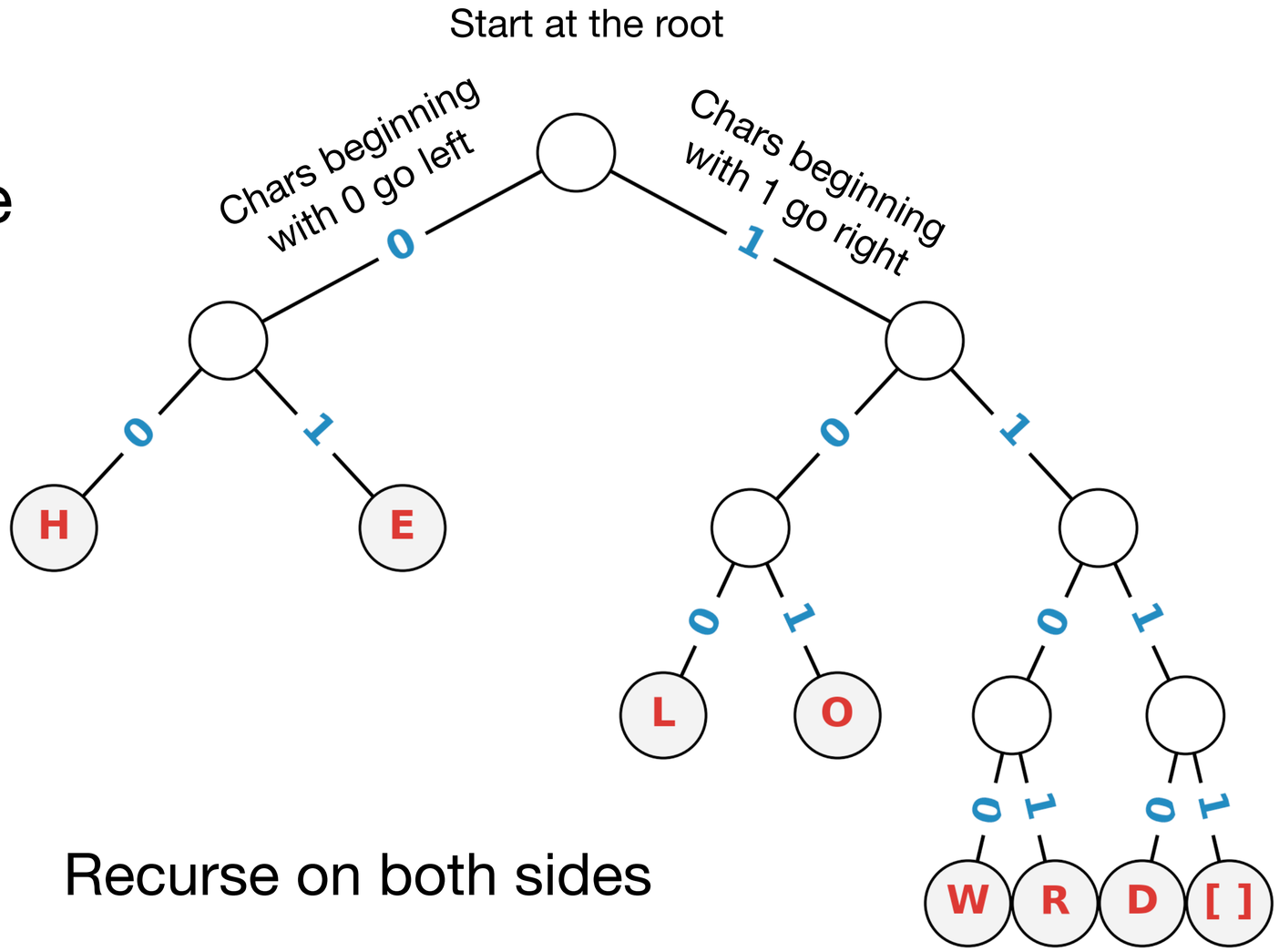
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

Given a code, can construct a binary tree

H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111

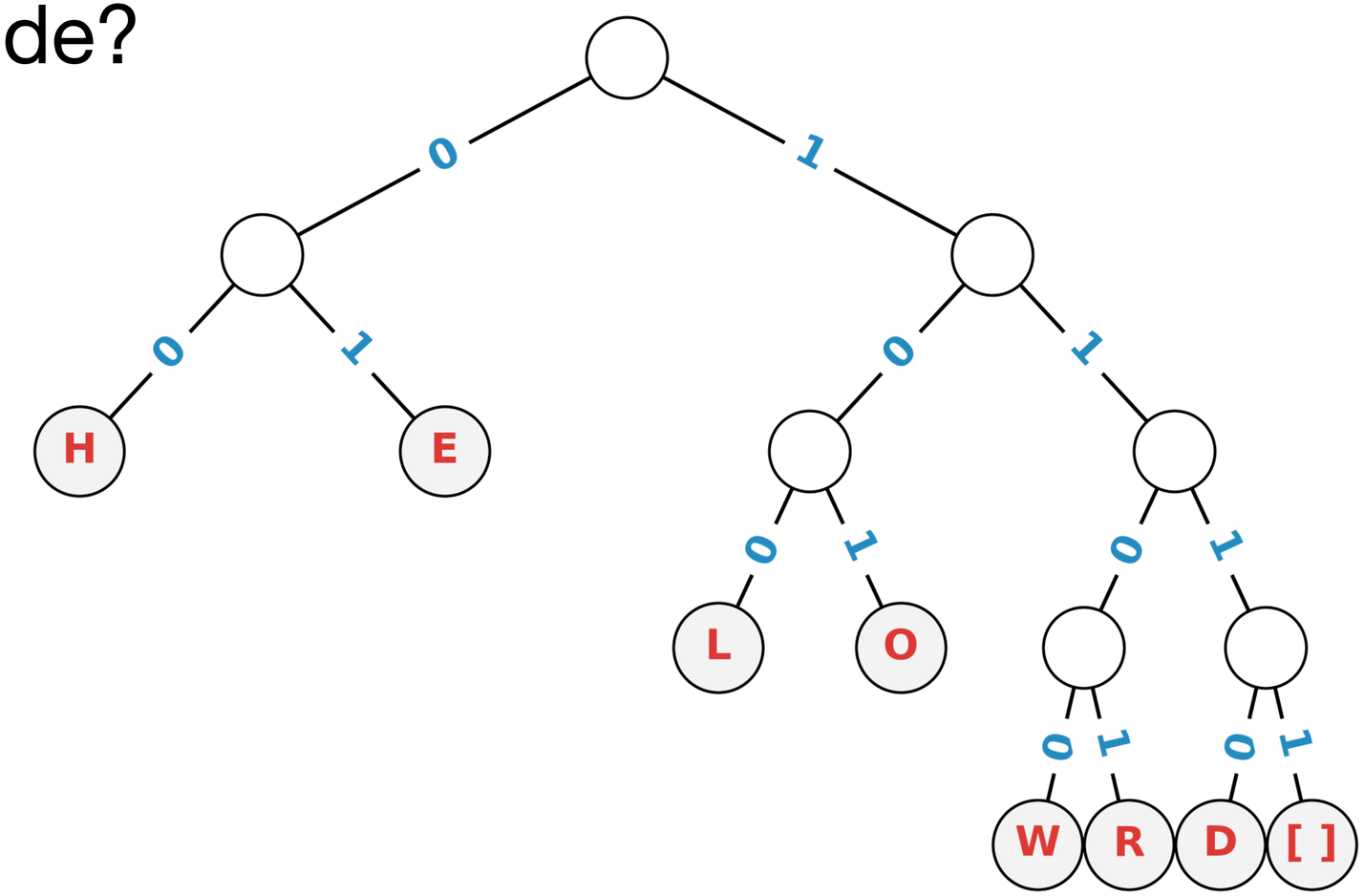


Recurse on both sides

Codes and Binary Trees

Is this a prefix code?

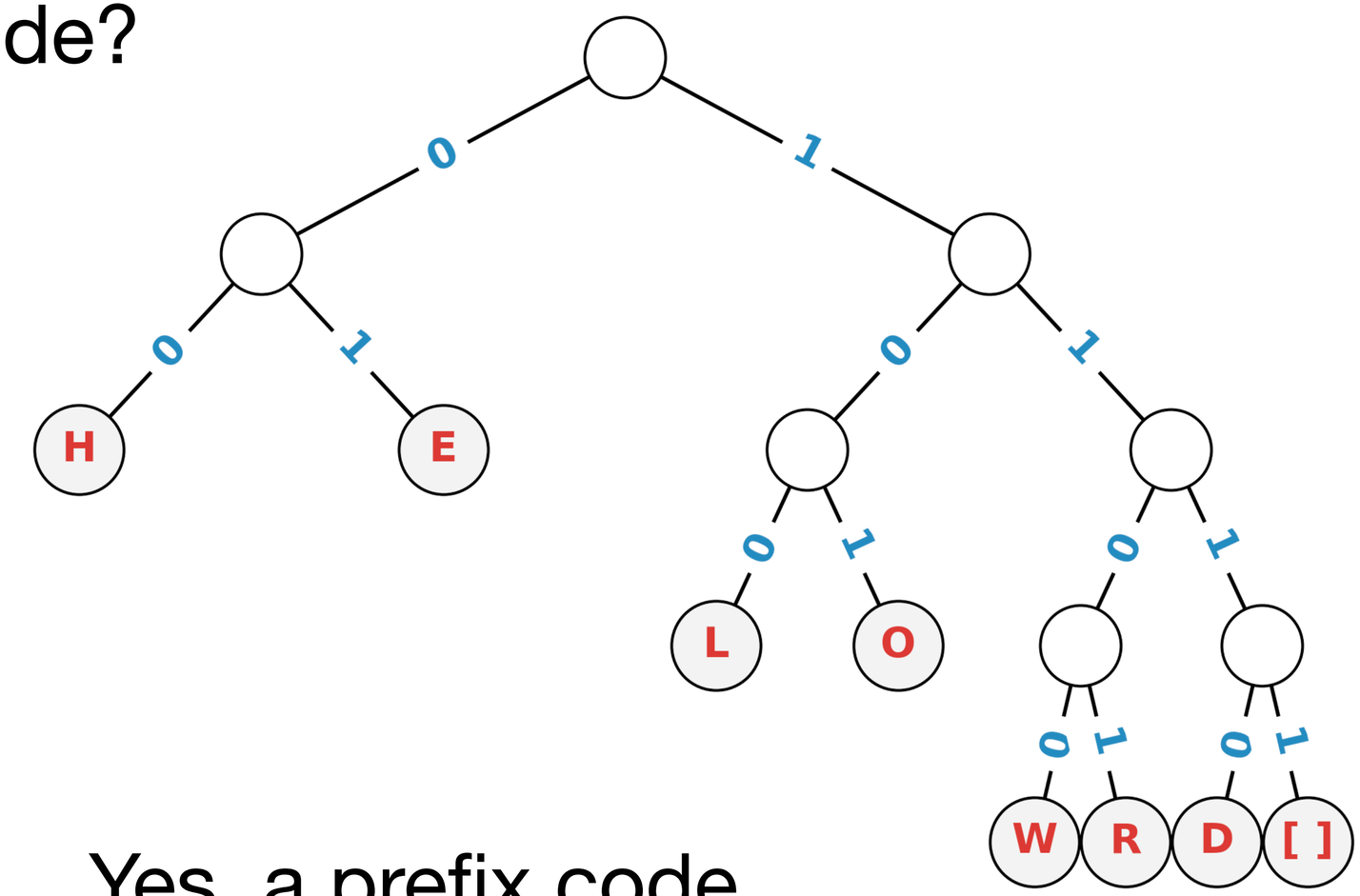
H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Codes and Binary Trees

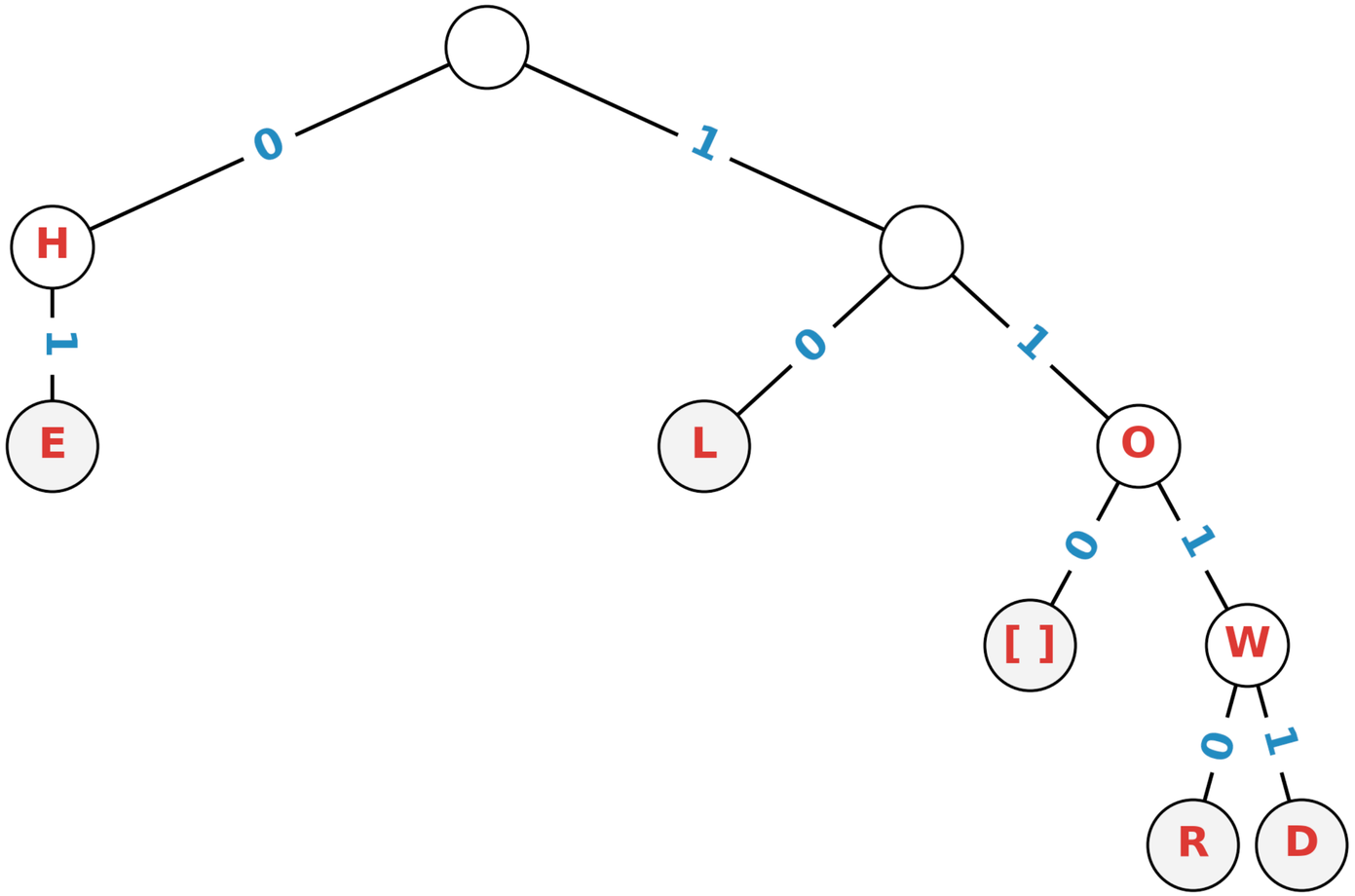
Is this a prefix code?

H → 00
E → 01
L → 100
O → 101
W → 1100
R → 1101
D → 1110
(space) → 1111



Yes, a prefix code

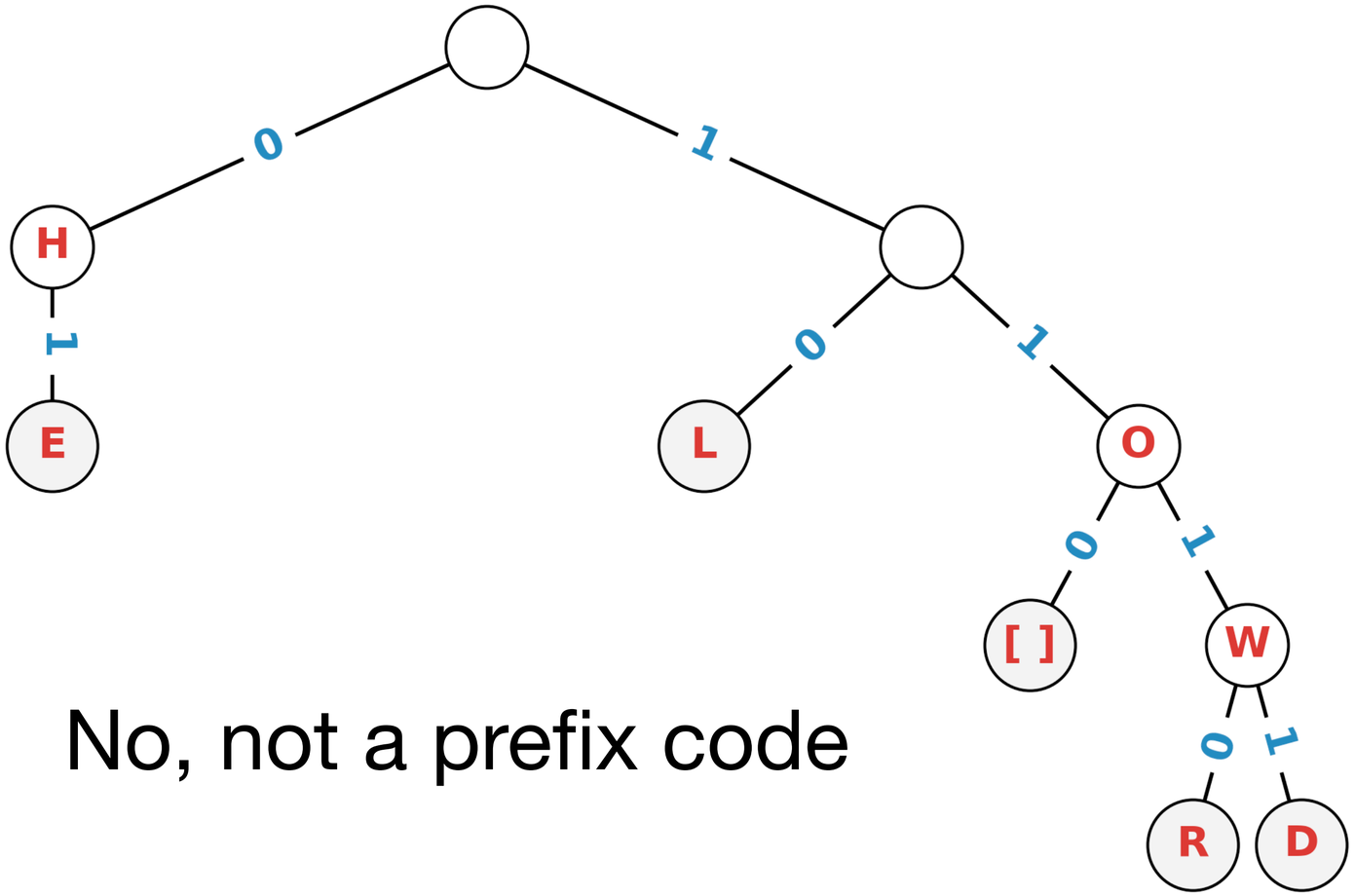
Codes and Binary Trees



Is this a prefix code?

H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

Codes and Binary Trees

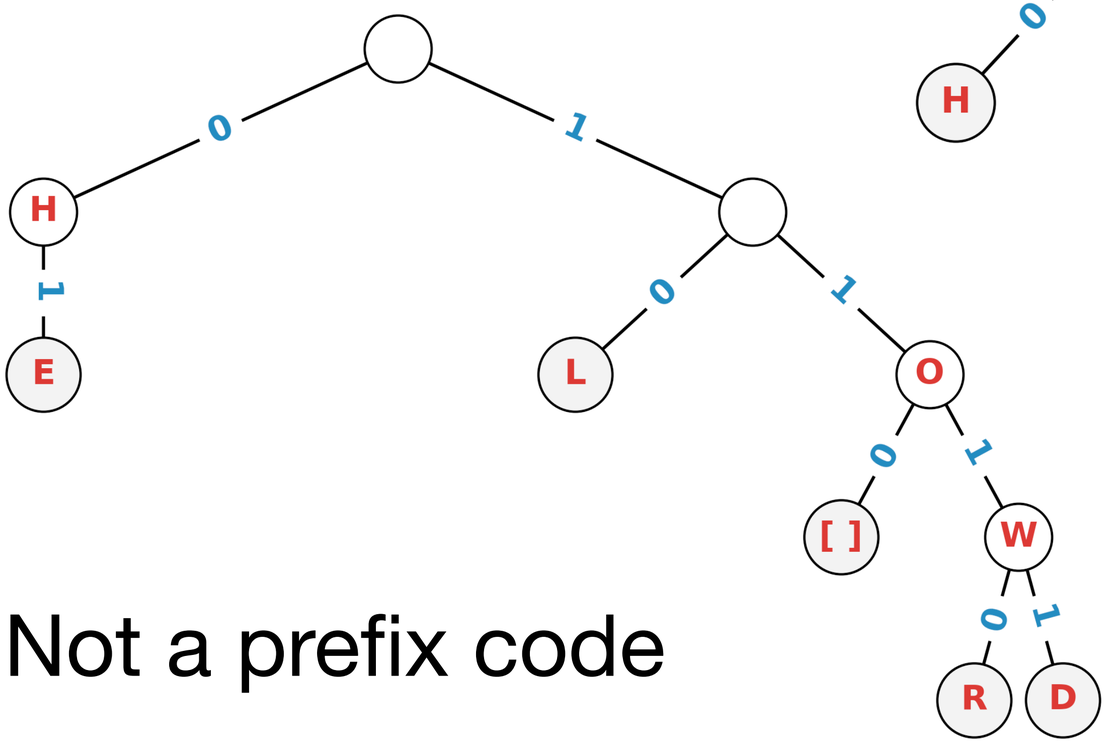


Is this a prefix code?

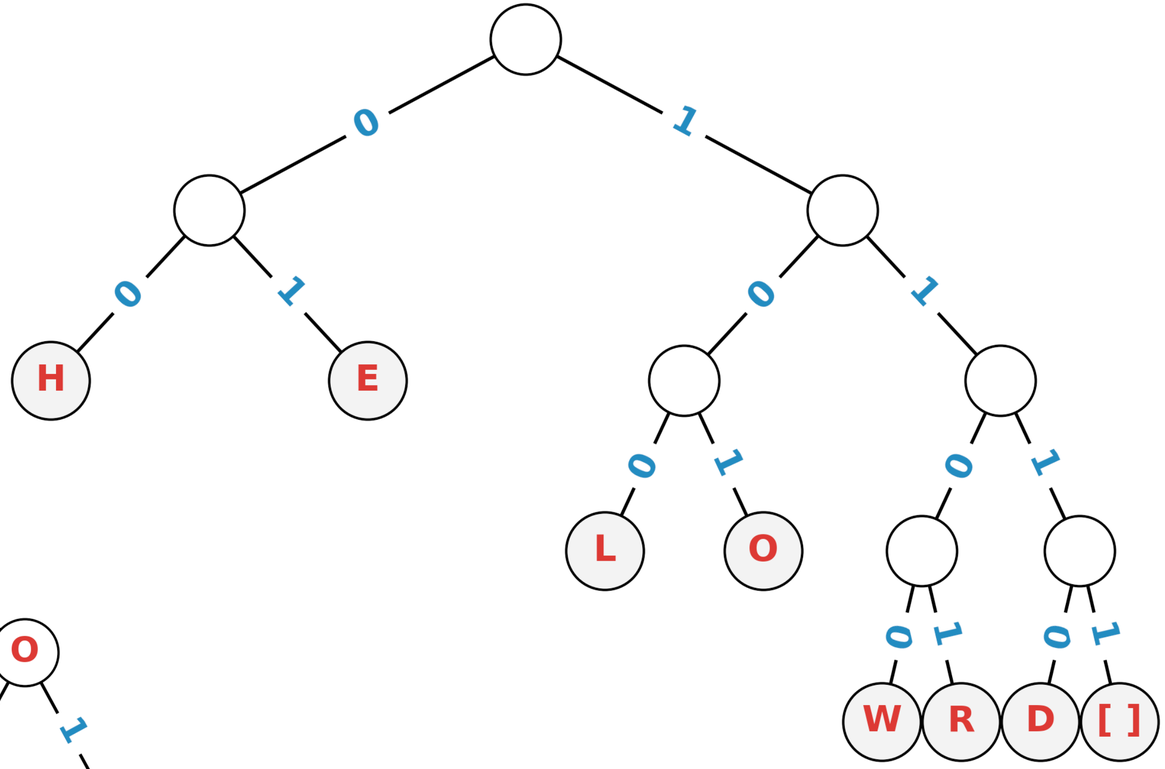
H → 0
E → 01
L → 10
O → 11
(space) → 110
W → 111
R → 1110
D → 1111

No, not a prefix code

Codes and Binary Trees



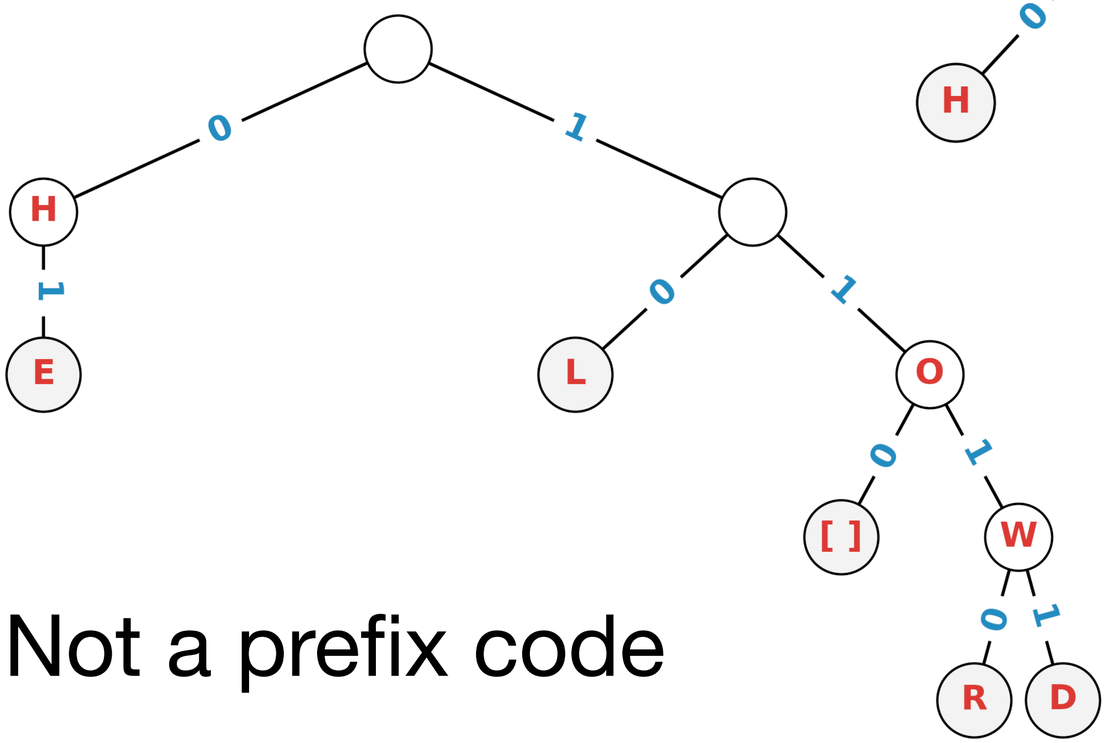
Not a prefix code



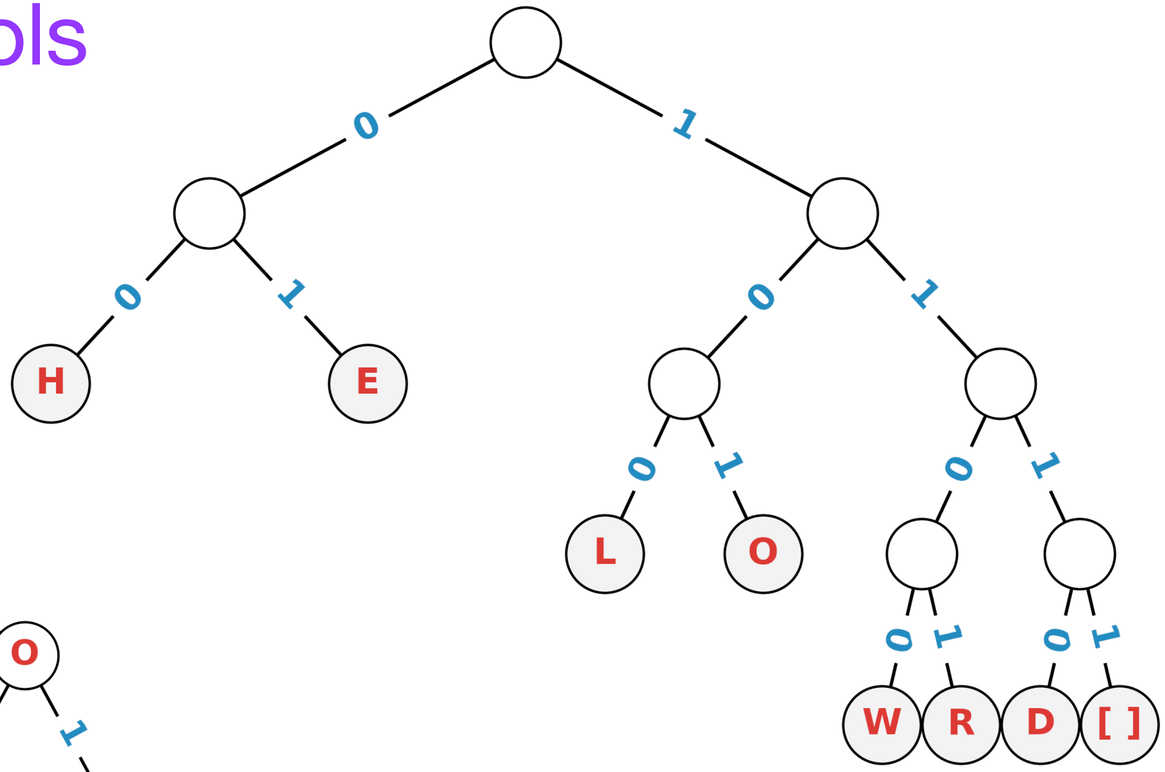
Prefix code

Codes and Binary Trees

Prefix codes have symbols only at the leaves



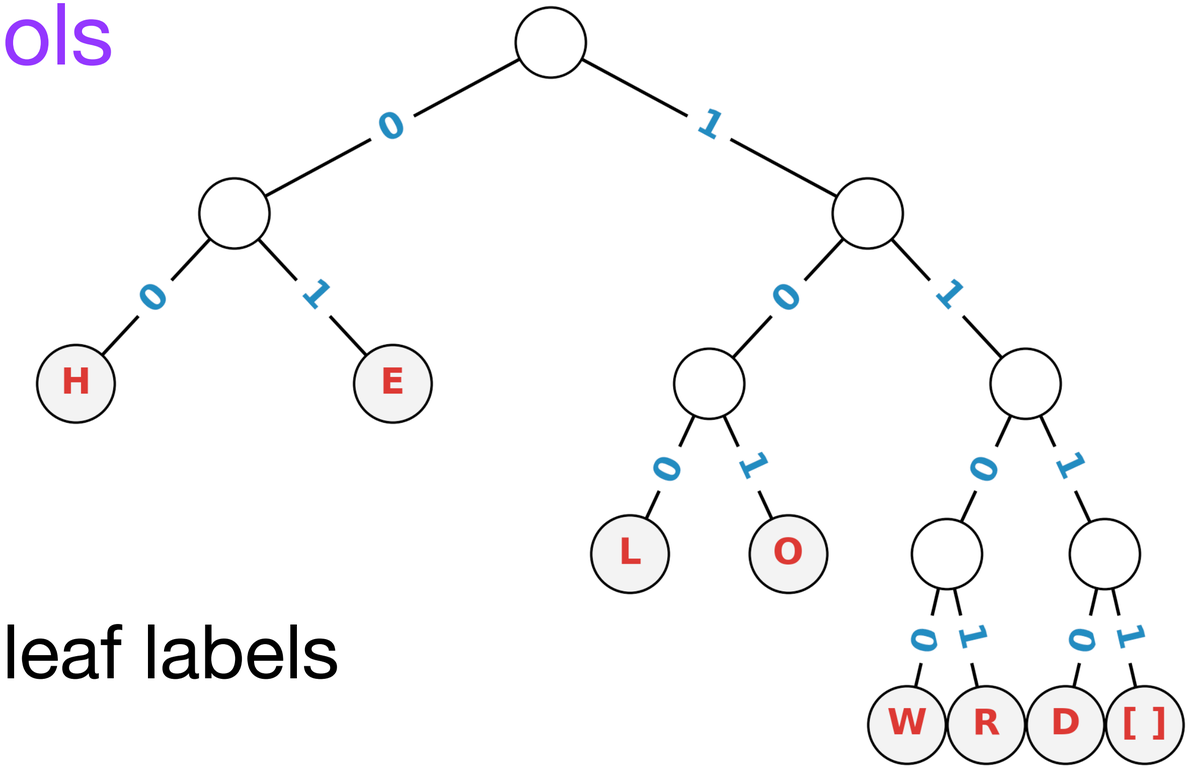
Not a prefix code



Prefix code

Codes and Binary Trees

Prefix codes have symbols only at the leaves

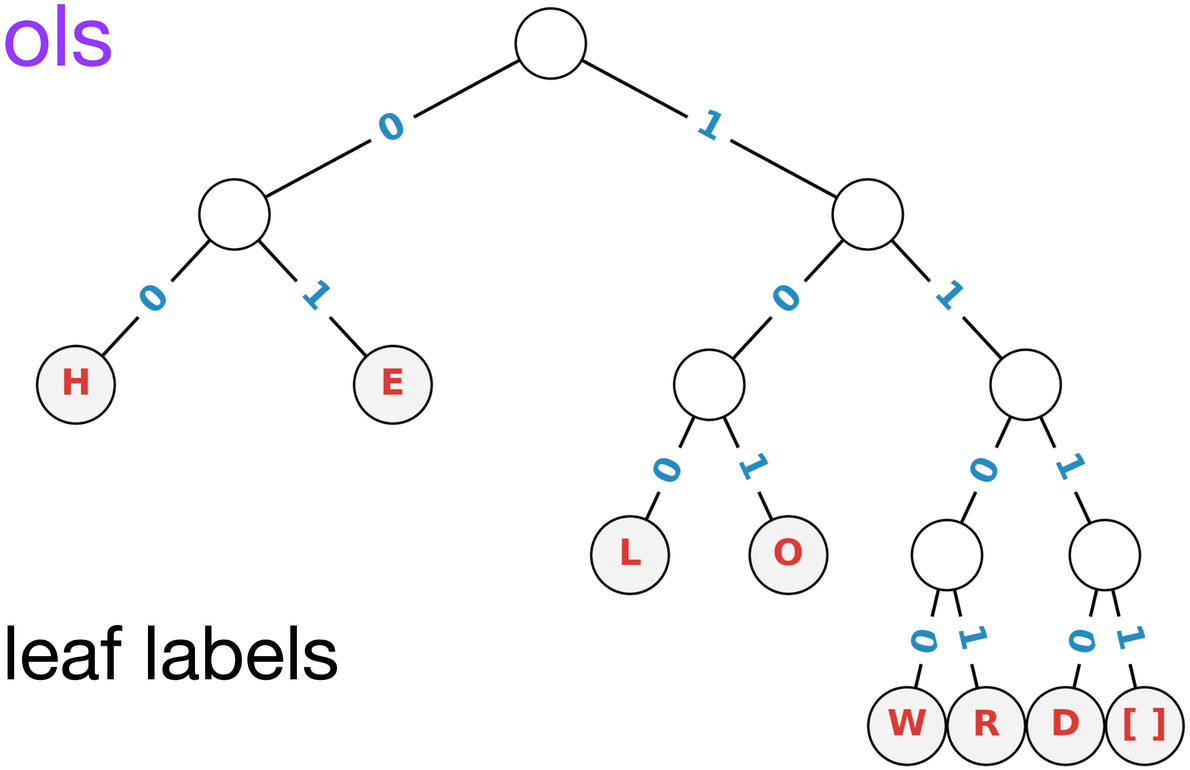


Prefix code → Binary tree with leaf labels

Prefix code

Codes and Binary Trees

Prefix codes have symbols only at the leaves

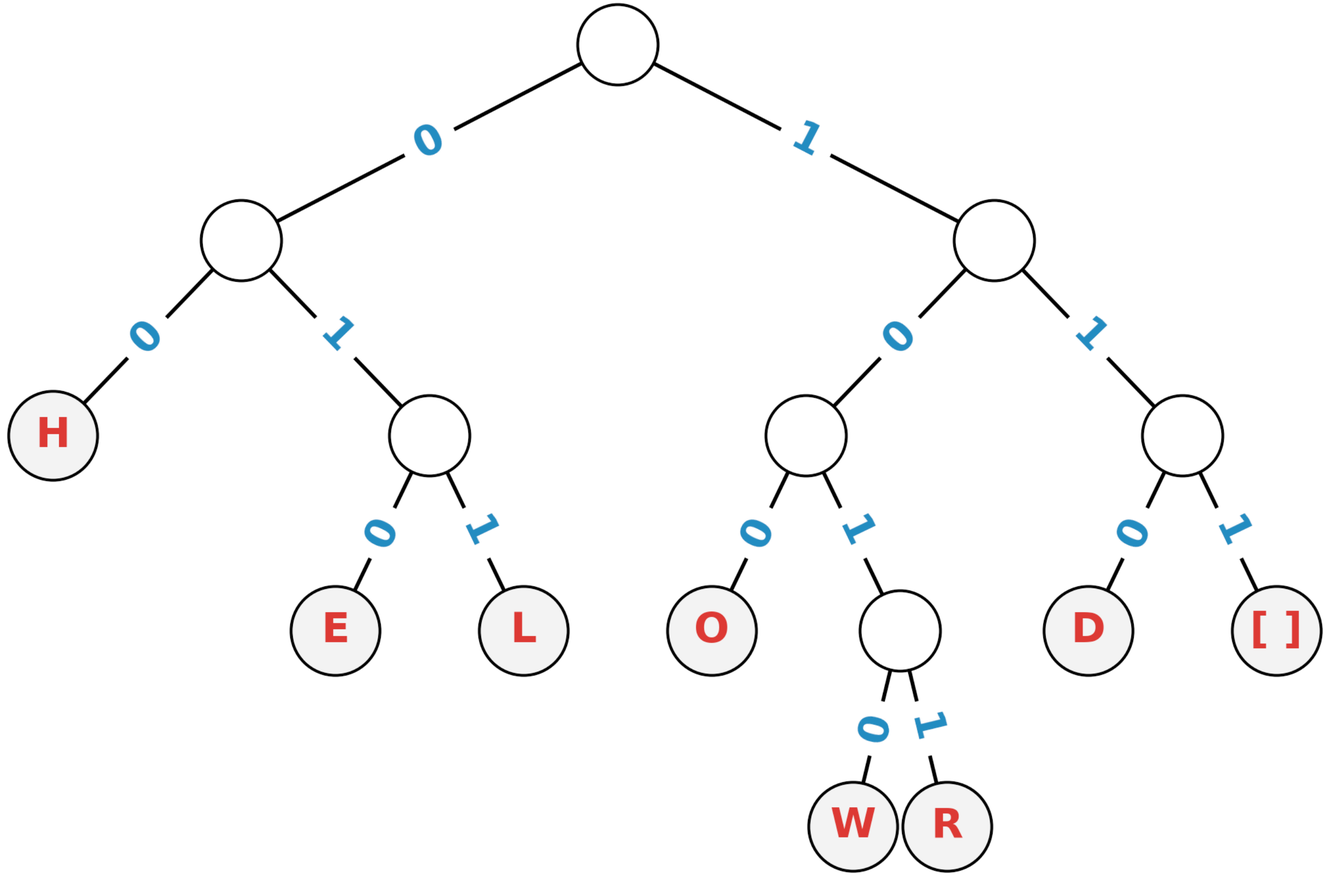


Prefix code → Binary tree with leaf labels

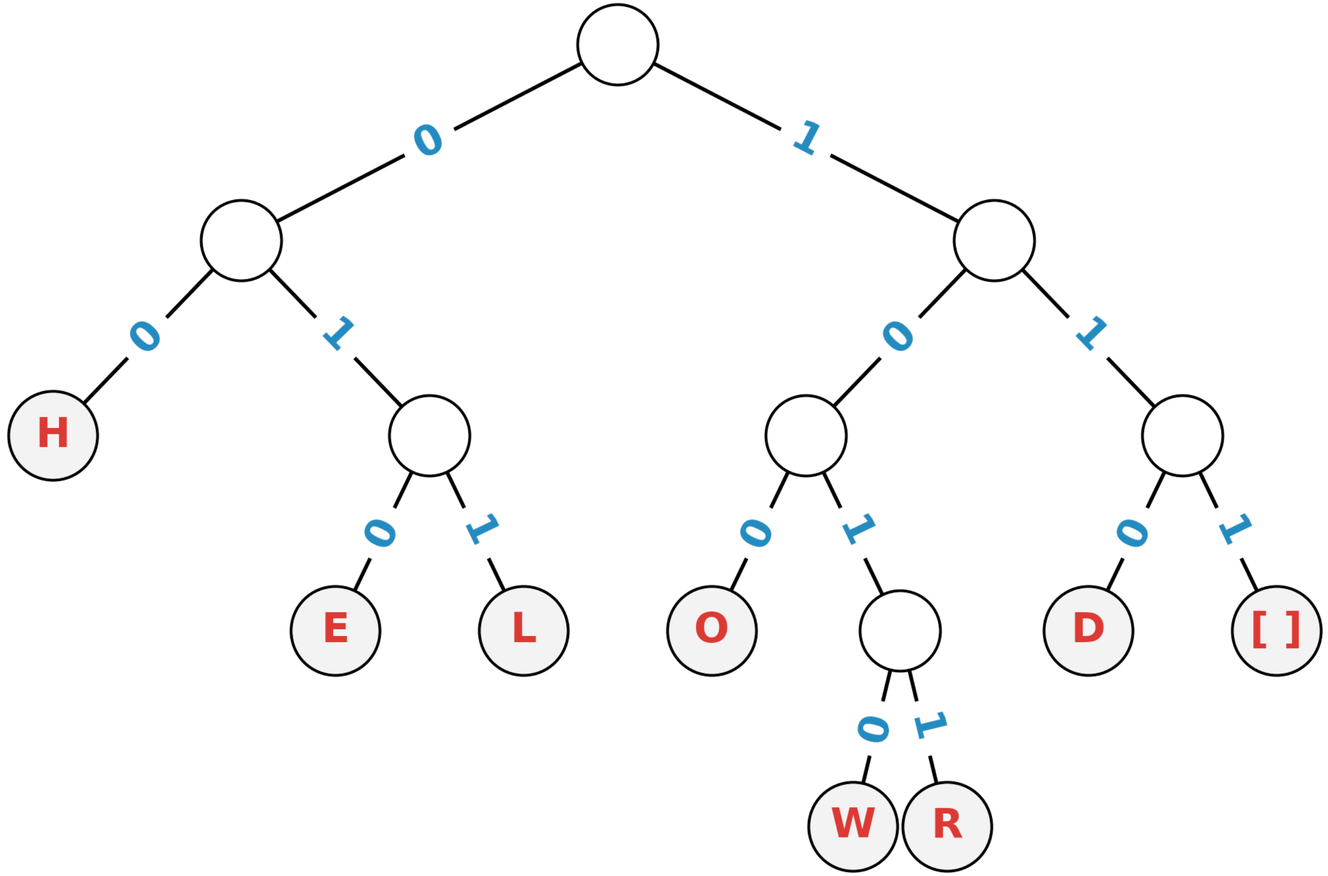
Binary tree with leaf labels → Prefix Code

Prefix code

Binary tree with leaf labels → Prefix Code

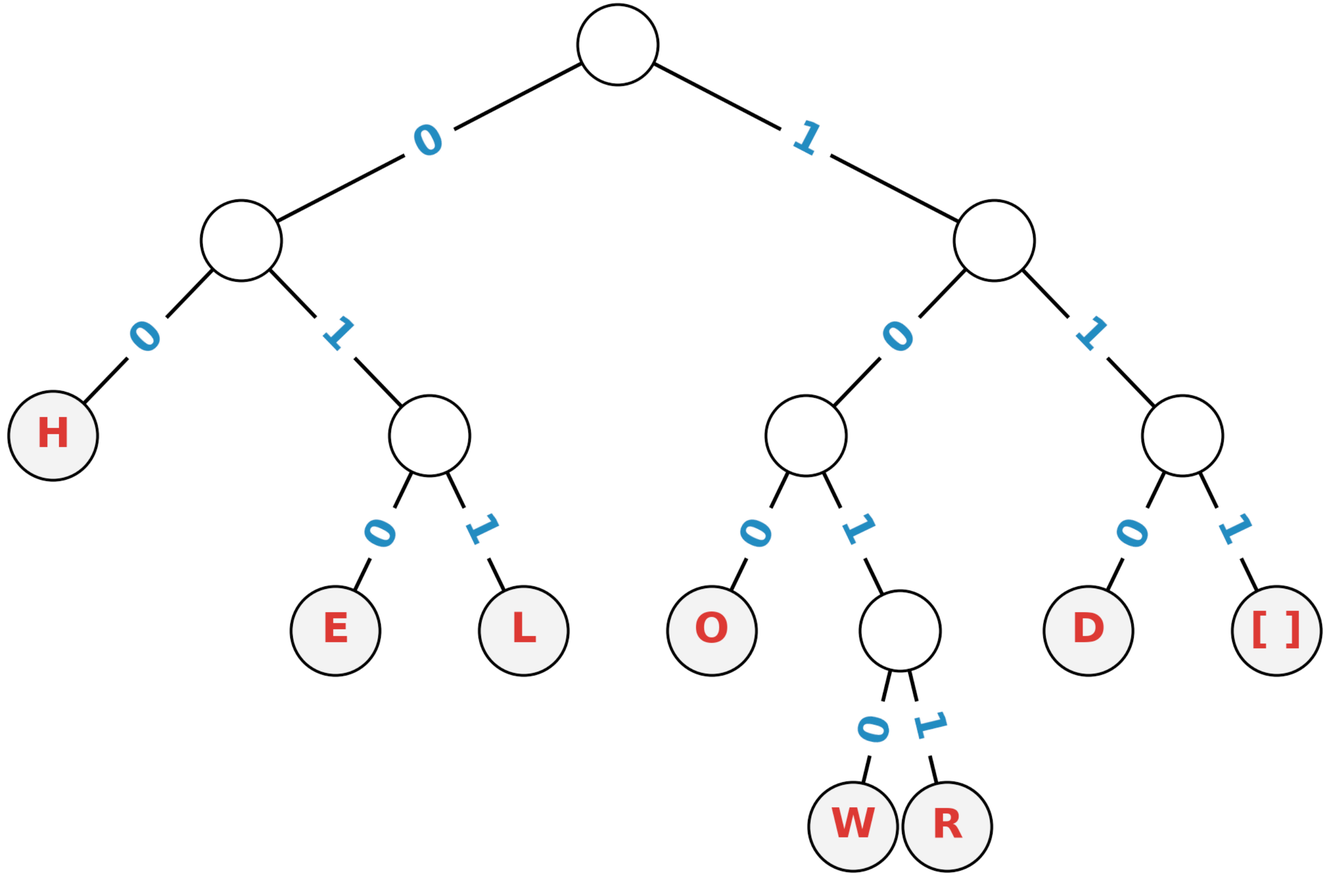


Binary tree with leaf labels → Prefix Code



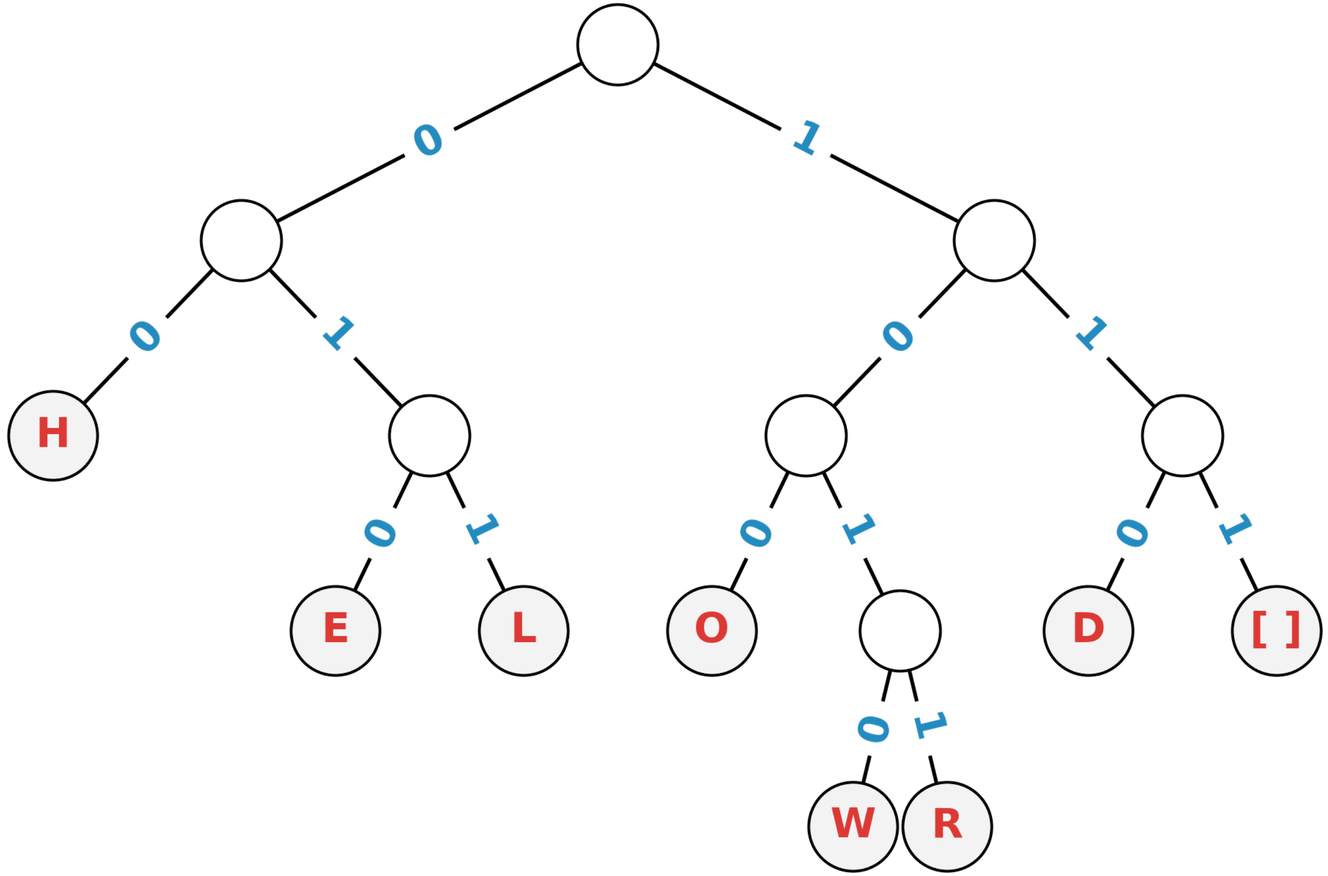
H → 00

Binary tree with leaf labels → Prefix Code



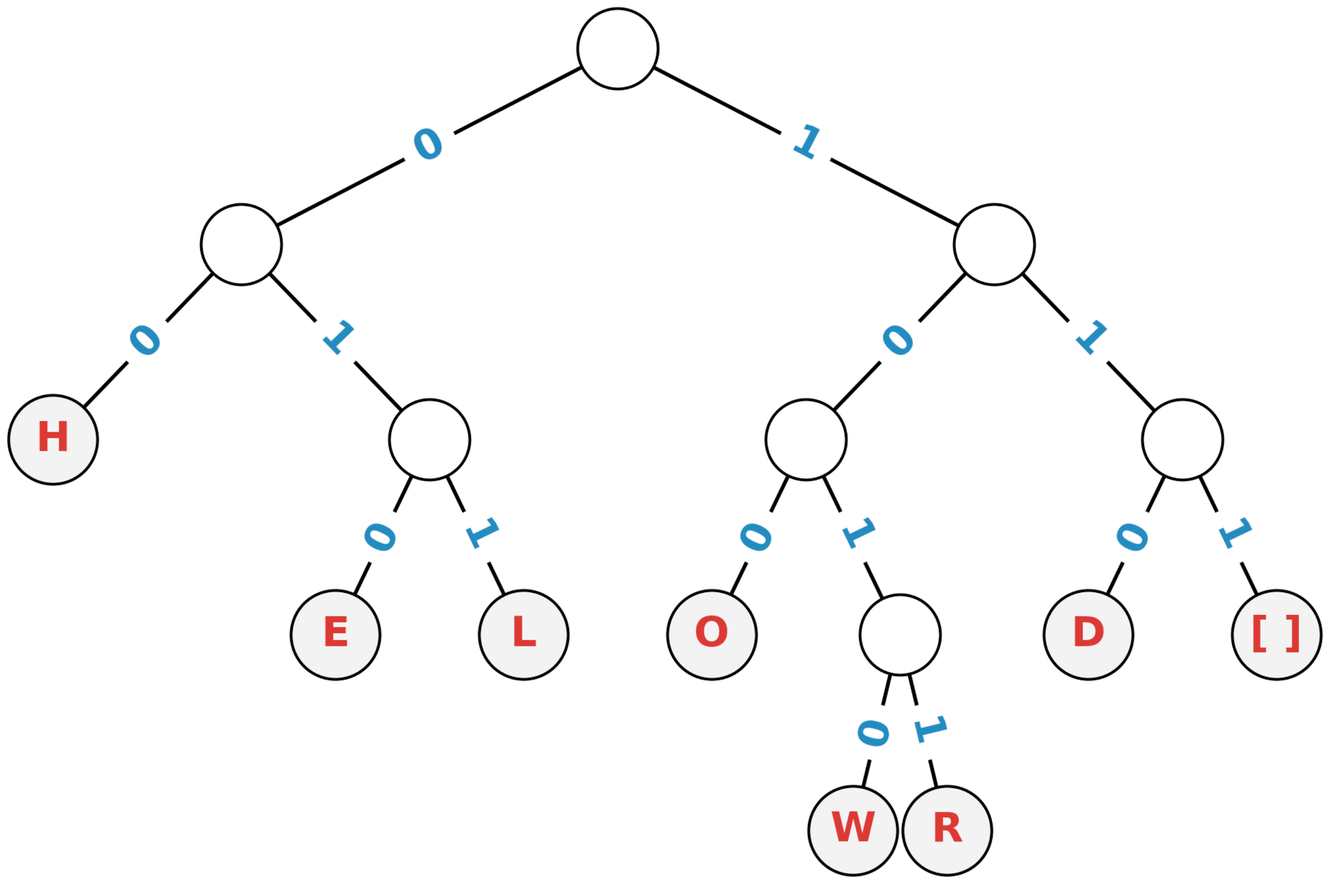
H → 00
E → 010

Binary tree with leaf labels → Prefix Code



H → 00
E → 010
L → 011

Binary tree with leaf labels → Prefix Code



H → 00
E → 010
L → 011
O → 100
W → 1010
R → 1011
D → 110
[ ] → 111

Codes and Binary Trees

Prefix code → Binary tree with leaf labels

Binary tree with leaf labels → Prefix Code

Codes and Binary Trees

Prefix code → Binary tree with leaf labels ✓

Binary tree with leaf labels → Prefix Code

Codes and Binary Trees

Prefix code → Binary tree with leaf labels ✓

Binary tree with leaf labels → Prefix Code ✓

Codes and Binary Trees

Prefix code → Binary tree with leaf labels ✓

Binary tree with leaf labels → Prefix Code ✓

Can think about how to build trees
instead of how to construct codes

An Iterative Approach

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

An Iterative Approach

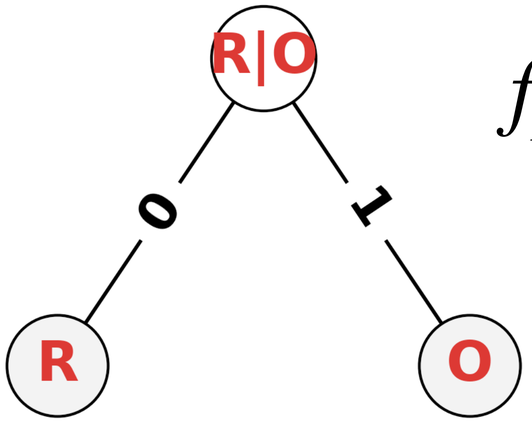
Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them

An Iterative Approach

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them



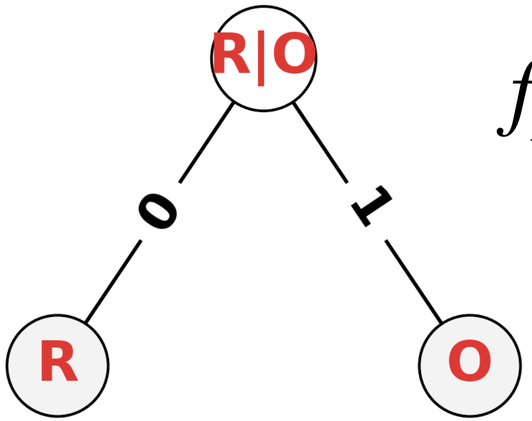
Think of R|O a new char with freq

$$f_R + f_O = 10.31 + 13.23 = 23.54$$

An Iterative Approach

Char	%
[ ]	27.49
E	20.62
H	10.48
R|O	**23.54**
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them



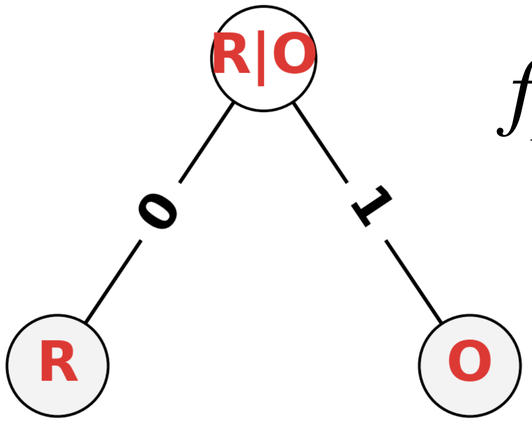
Think of R|O a new char with freq

$$f_R + f_O = 10.31 + 13.23 = 23.54$$

An Iterative Approach

Char	%
[ ]	27.49
E	20.62
H	10.48
R|O	**23.54**
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them



Think of R|O a new char with freq

$$f_R + f_O = 10.31 + 13.23 = 23.54$$

Intuitively as we iterate, we’ll build a tree UP from R and O

How do we use frequencies?

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

How do we use frequencies?

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Intuitively we know we want to assign shorter codewords to common characters

And longer codewords to less common ones

How do we use frequencies?

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Intuitively we know we want to assign shorter codewords to common characters

And longer codewords to less common ones

In the binary tree, this means common chars are high up and rare chars are low down

Huffman Coding

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them

Huffman Coding

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them

At each step, greedily choose the lowest two frequencies and merge them

Huffman Coding

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

At each step, choose two chars, make them siblings, and “merge” them

At each step, greedily choose the lowest two frequencies and merge them

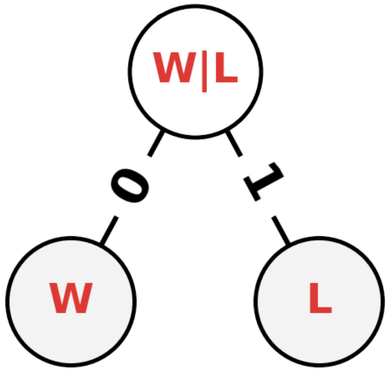
Intuition: we’re building up, so they’ll end up lower

Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

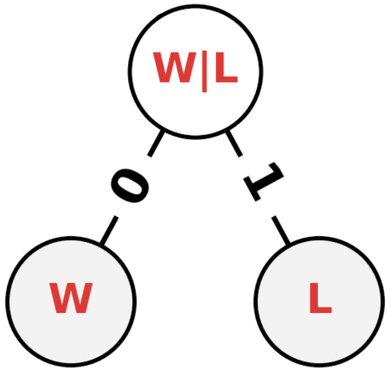
Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61



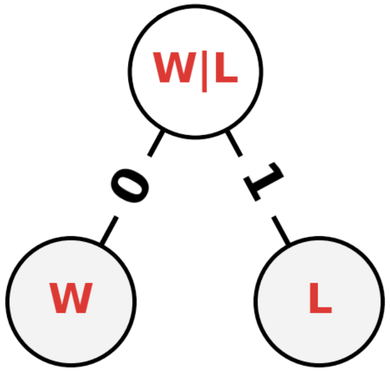
Huffman Coding Example

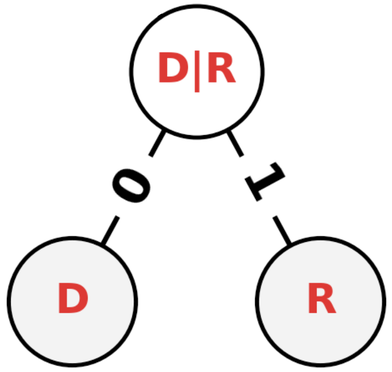
Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
W|L	10.48



Huffman Coding Example

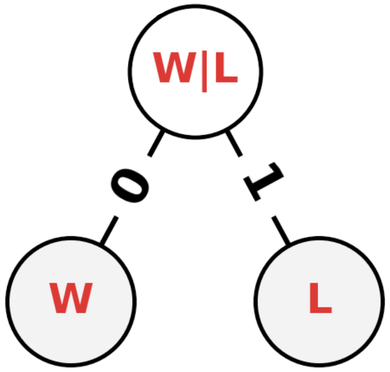
Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
W|L	10.48

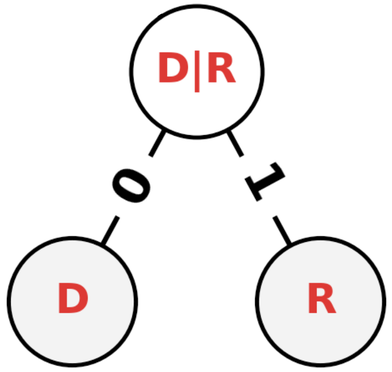




Huffman Coding Example

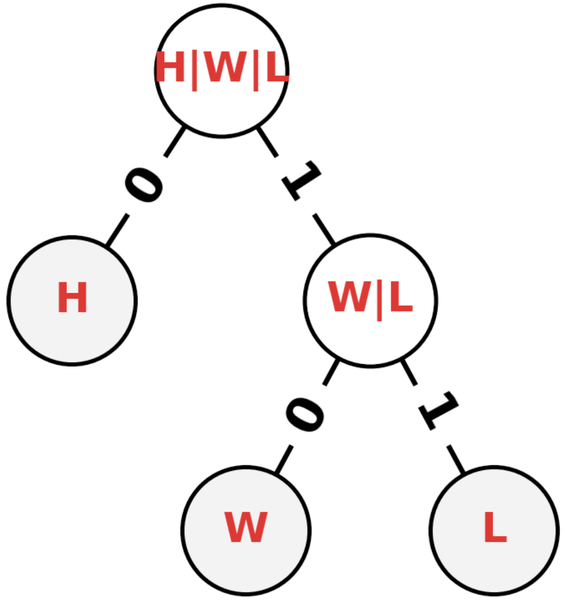
Char	%
[ ]	27.49
E	20.62
H	10.48
D|R	17.70
O	13.23
W|L	10.48

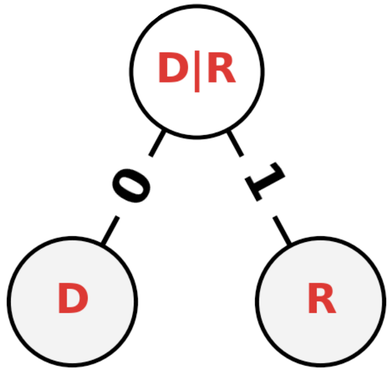




Huffman Coding Example

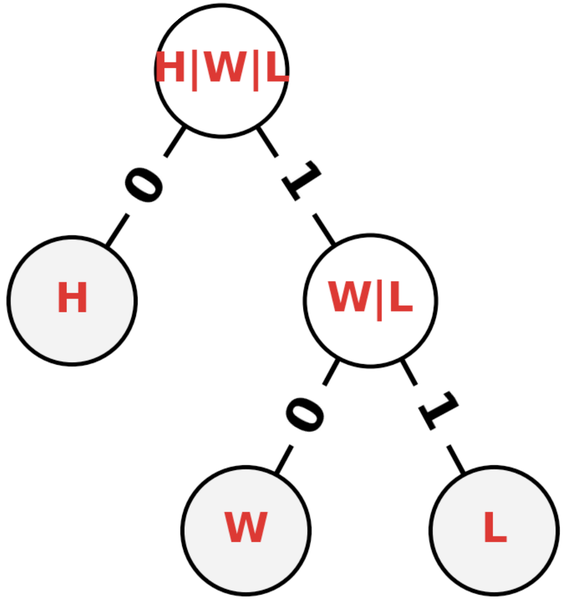
Char	%
[ ]	27.49
E	20.62
H	10.48
D|R	17.70
O	13.23
W|L	10.48

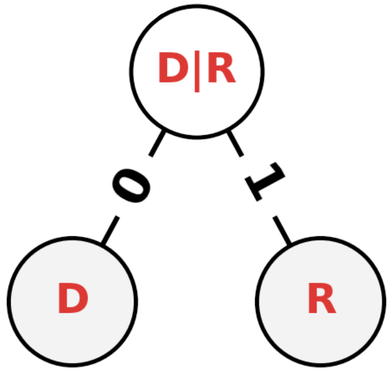




Huffman Coding Example

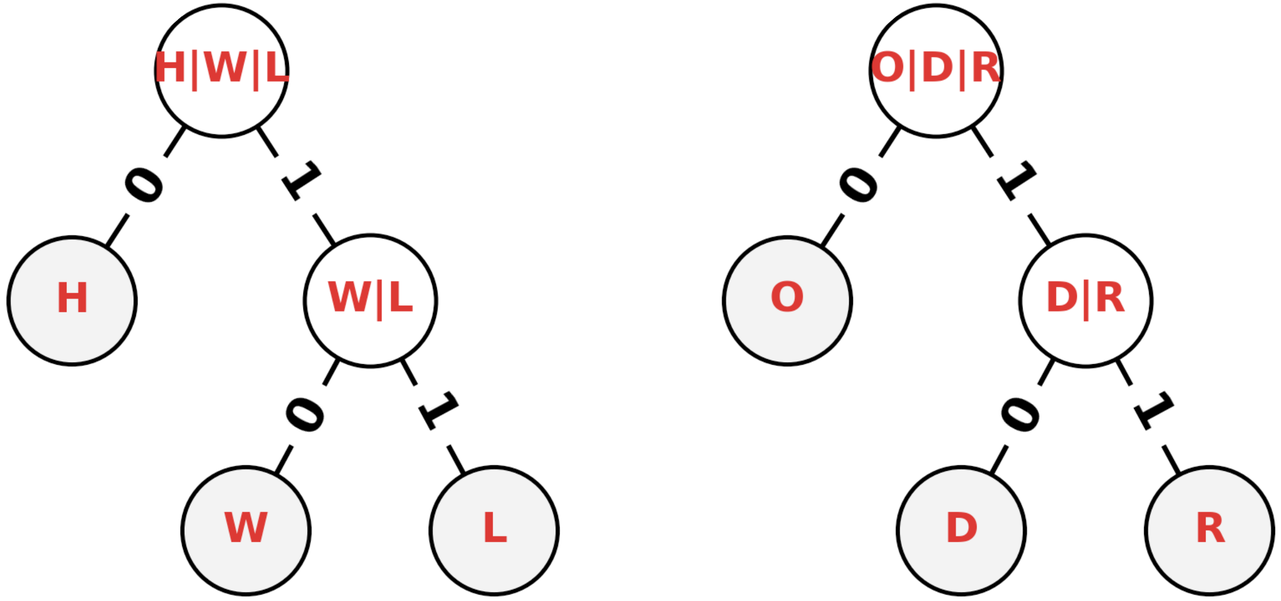
Char	%
[ ]	27.49
E	20.62
H|W|L	20.96
D|R	17.70
O	13.23





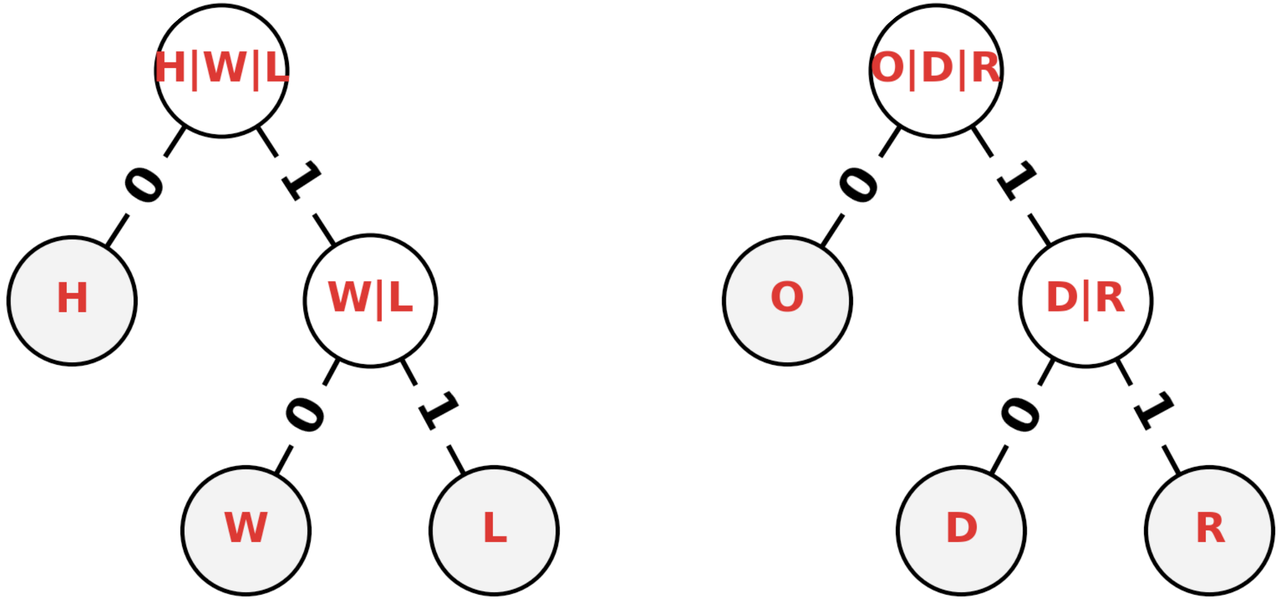
Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H|W|L	20.96
D|R	17.70
O	13.23



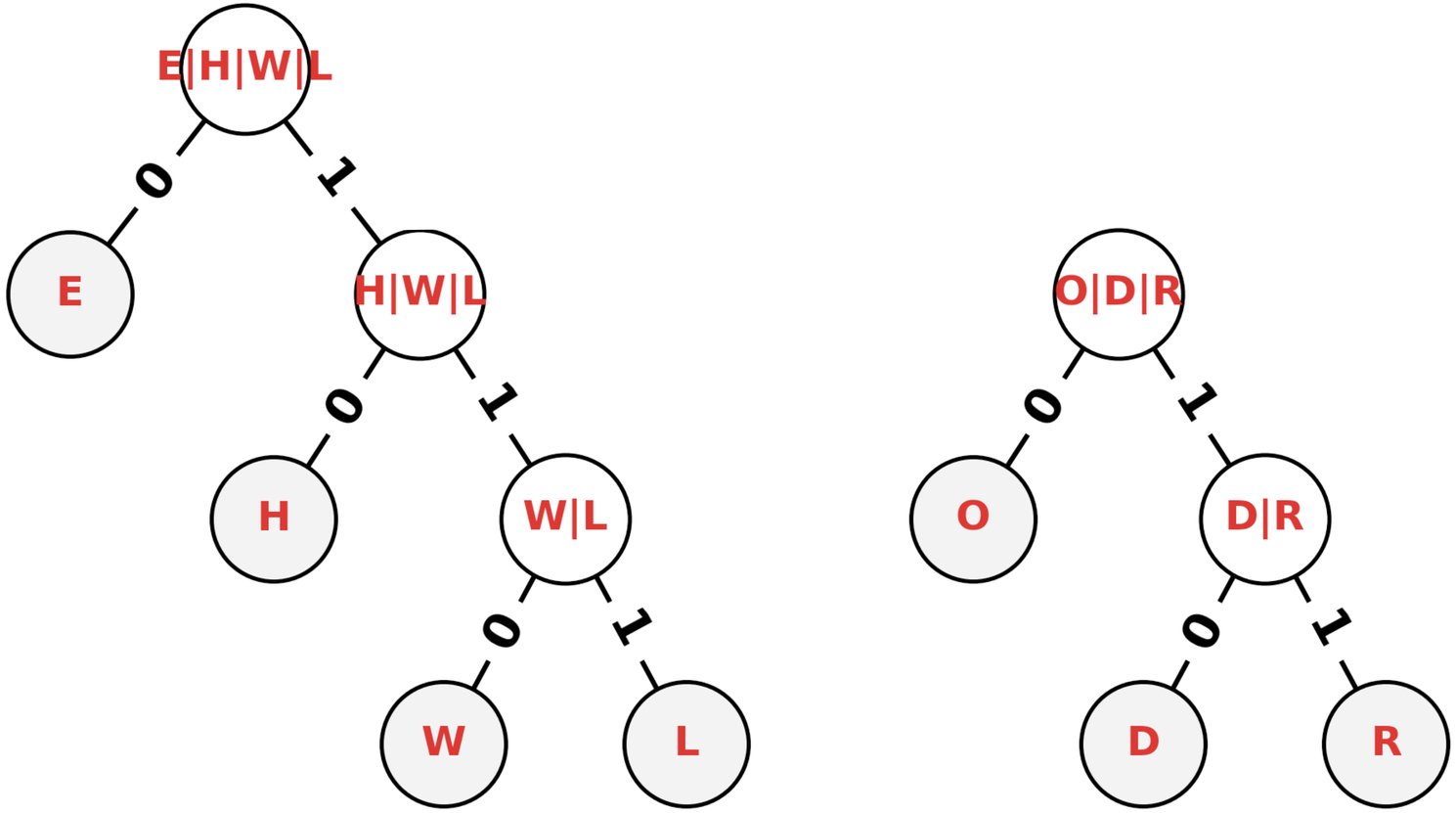
Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H|W|L	20.96
O|D|R	30.93



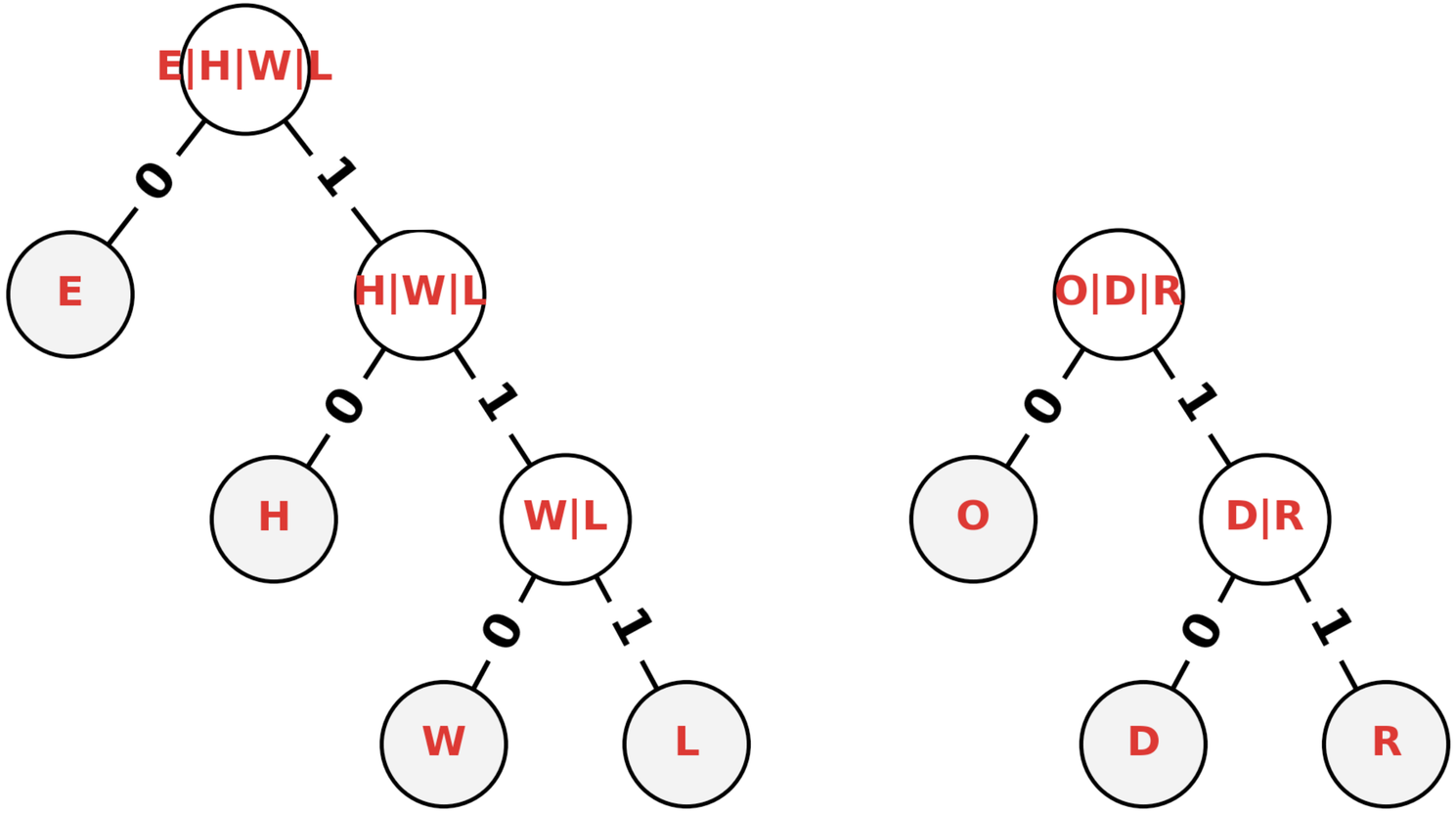
Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H|W|L	20.96
O|D|R	30.93



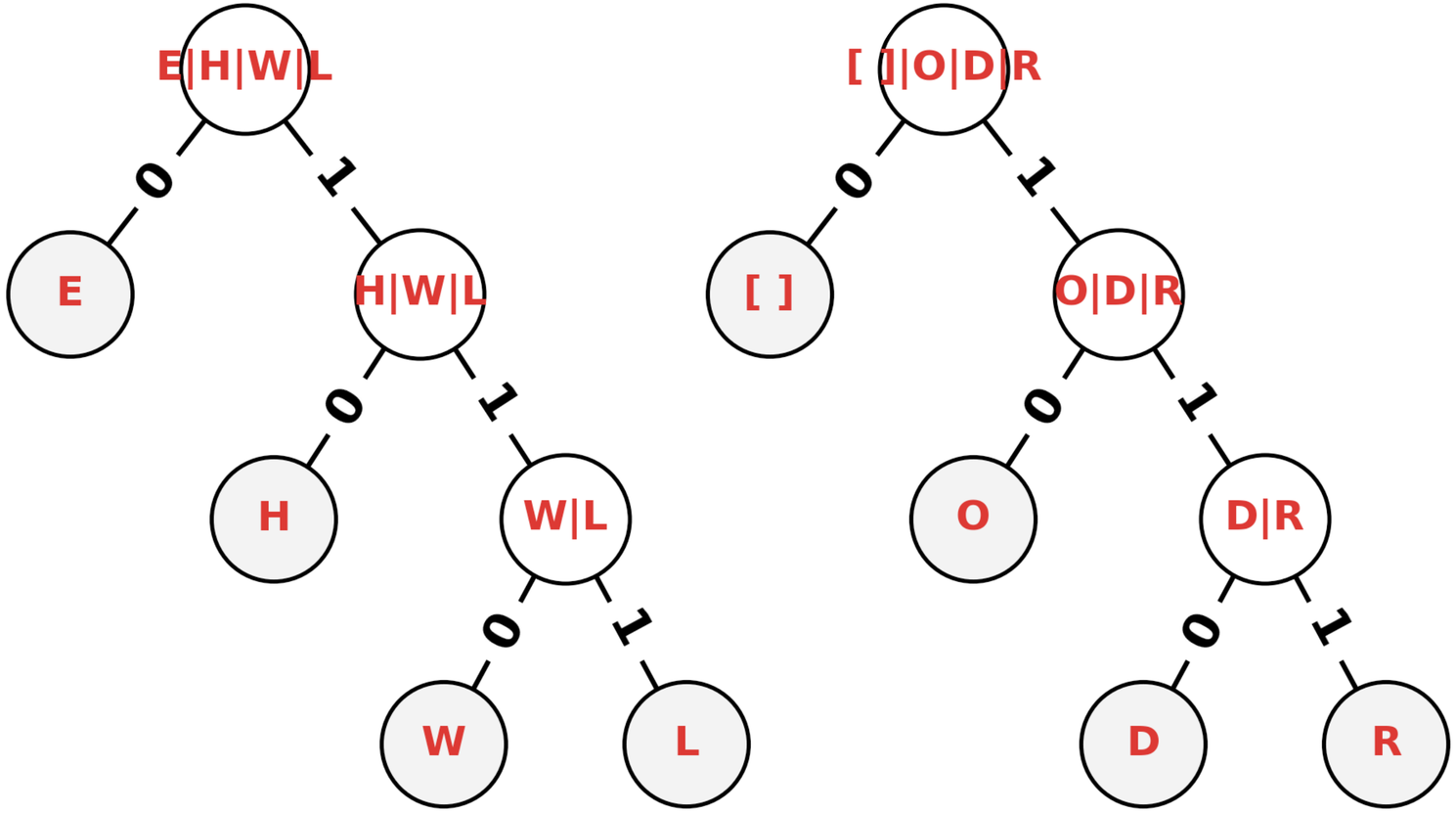
Huffman Coding Example

Char	%
[ ]	27.49
E|H|W|L	41.58
O|D|R	30.93



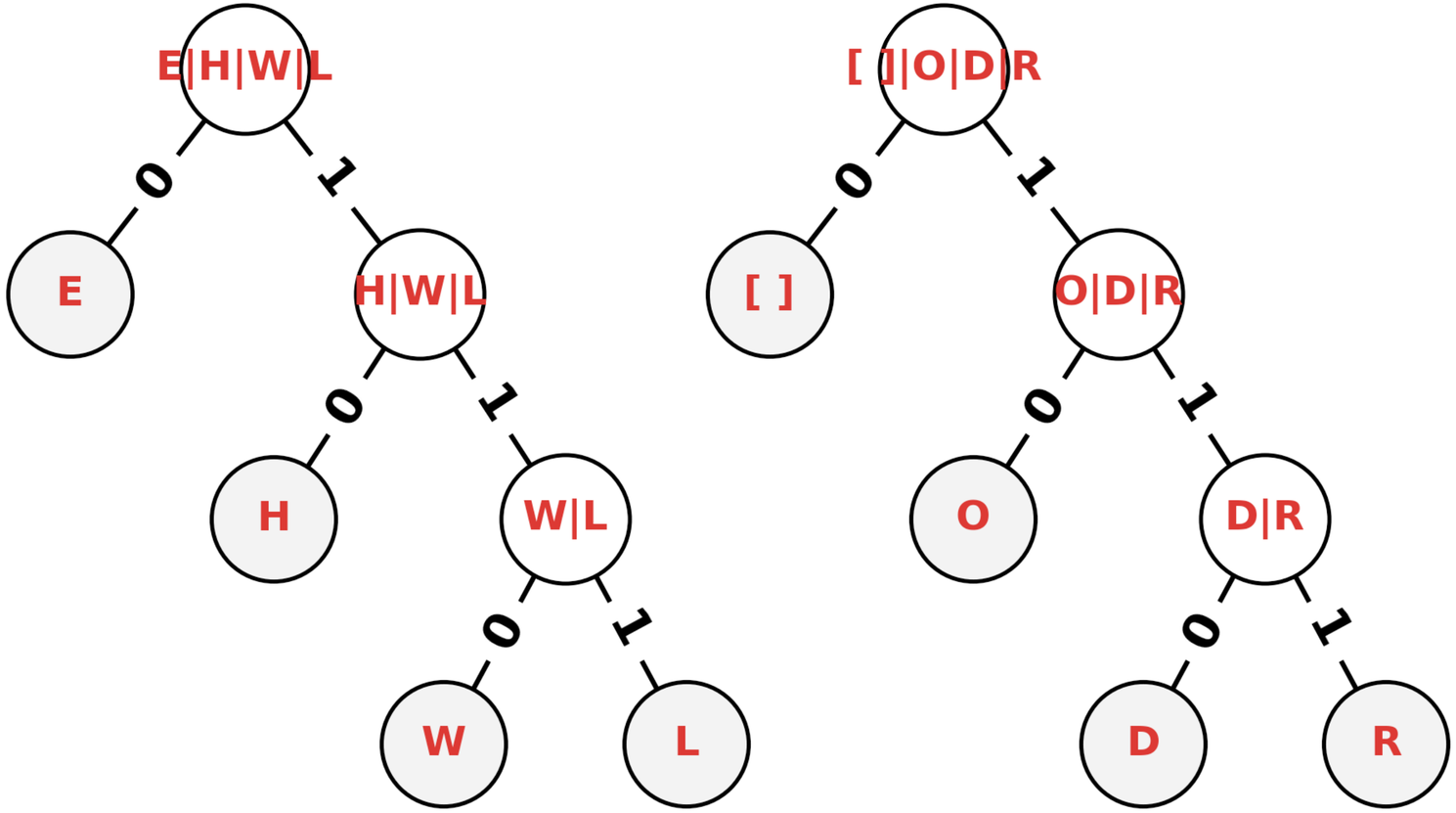
Huffman Coding Example

Char	%
[ ]	27.49
E|H|W|L	41.58
O|D|R	30.93



Huffman Coding Example

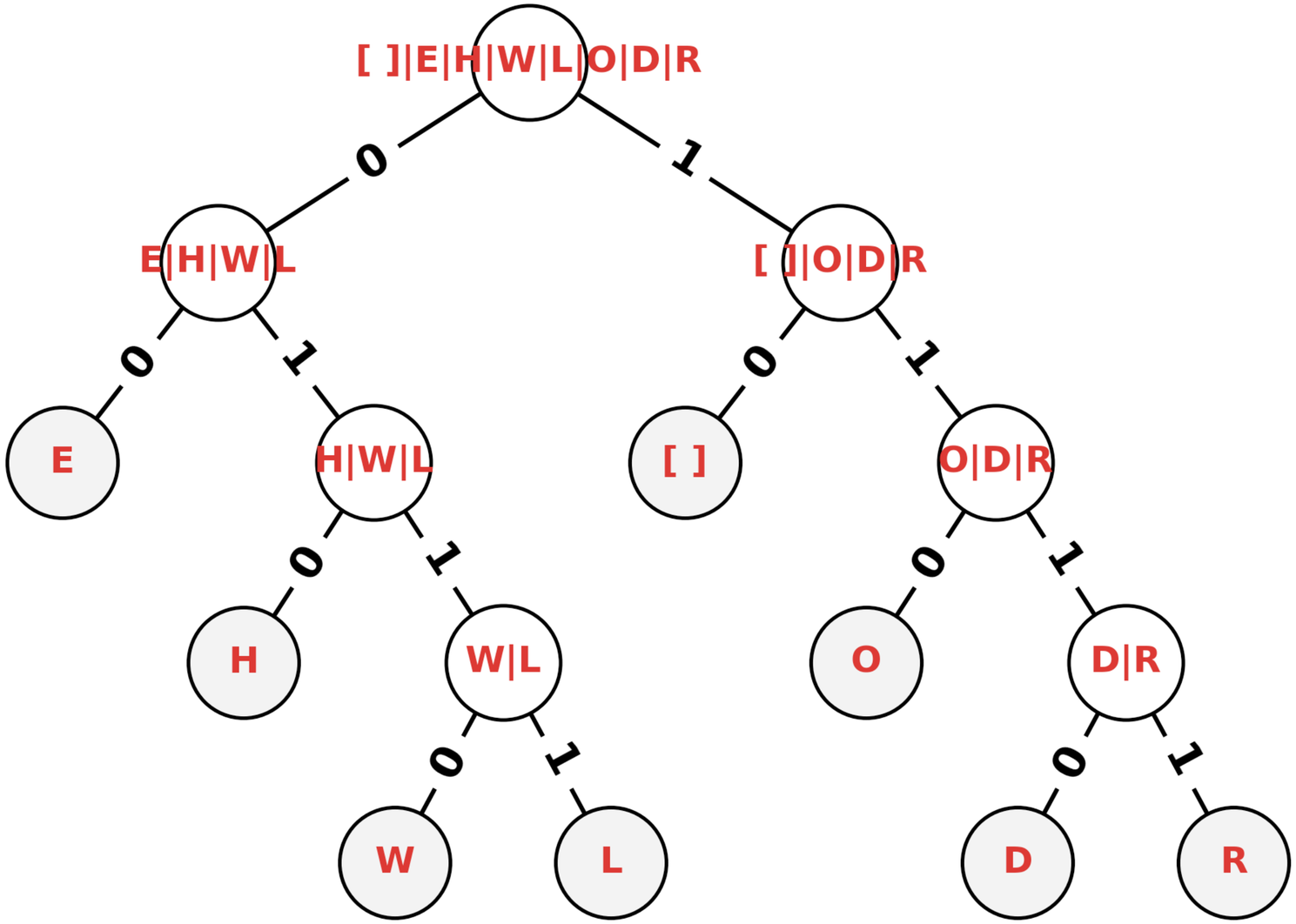
Char	%
[ ]| O|D|R	58.42
E|H|W|L	41.58



Huffman Coding Example

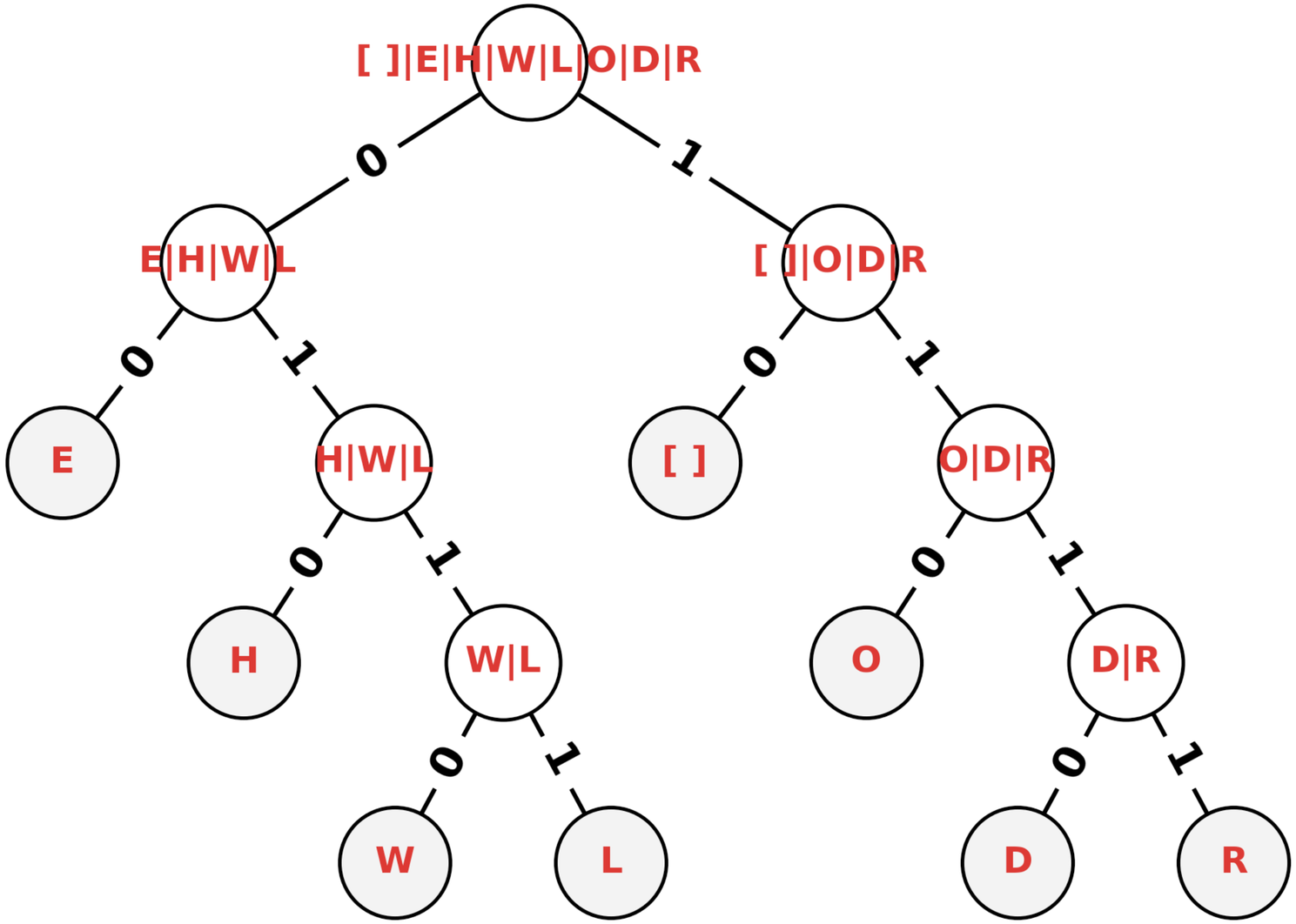
Char	%
[ ]| O|D|R	58.42
E|H|W|L	41.58

This is the complete Huffman tree



Huffman Coding Example

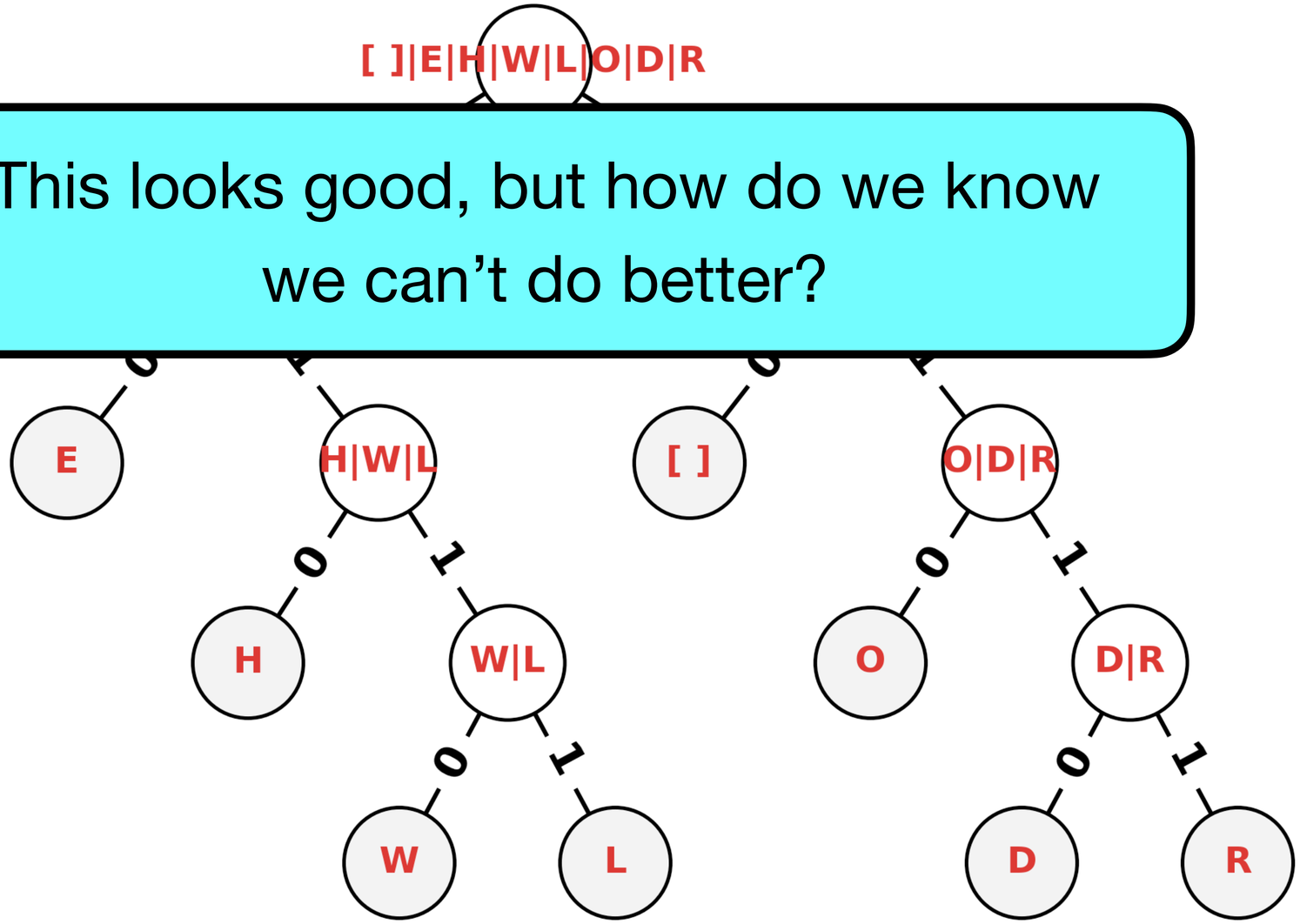
Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61



Huffman Coding Example

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

This looks good, but how do we know we can't do better?



Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Goal:

Minimize the average message length

Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Goal:

Minimize the average message length

Equivalently:

Minimize the bits/char in the average message

Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Goal:

Minimize the average message length

Equivalently:

Minimize the bits/char in the average message

When I send a character, it's D with $f_{\mathrm{D}} = 7.39\,\%$ probability

Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Goal:

Minimize the average message length

When I send a character, it's D with $f_{\mathrm{D}} = 7.39\,\%$ probability

Equivalently:

Minimize the bits/char in the average message

In general it's χ with f_χ probability

Remember the Goal

Char	%
[ ]	27.49
E	20.62
H	10.48
R	10.31
O	13.23
D	7.39
L	6.87
W	3.61

Goal:

Minimize the average message length

Equivalently:

Minimize the bits/char in the average message

When I send a character, it's D with $f_{\mathrm{D}} = 7.39\,\%$ probability

In general it's χ with f_χ probability

Bits/char is the weighted average:

$$\sum_{\chi} f_\chi \, |c_\chi|$$

c_χ is the codeword for χ