HUFFMAN CODING

cs2420 | Introduction to Algorithms and Data Structures | Spring 2015

administrivia…

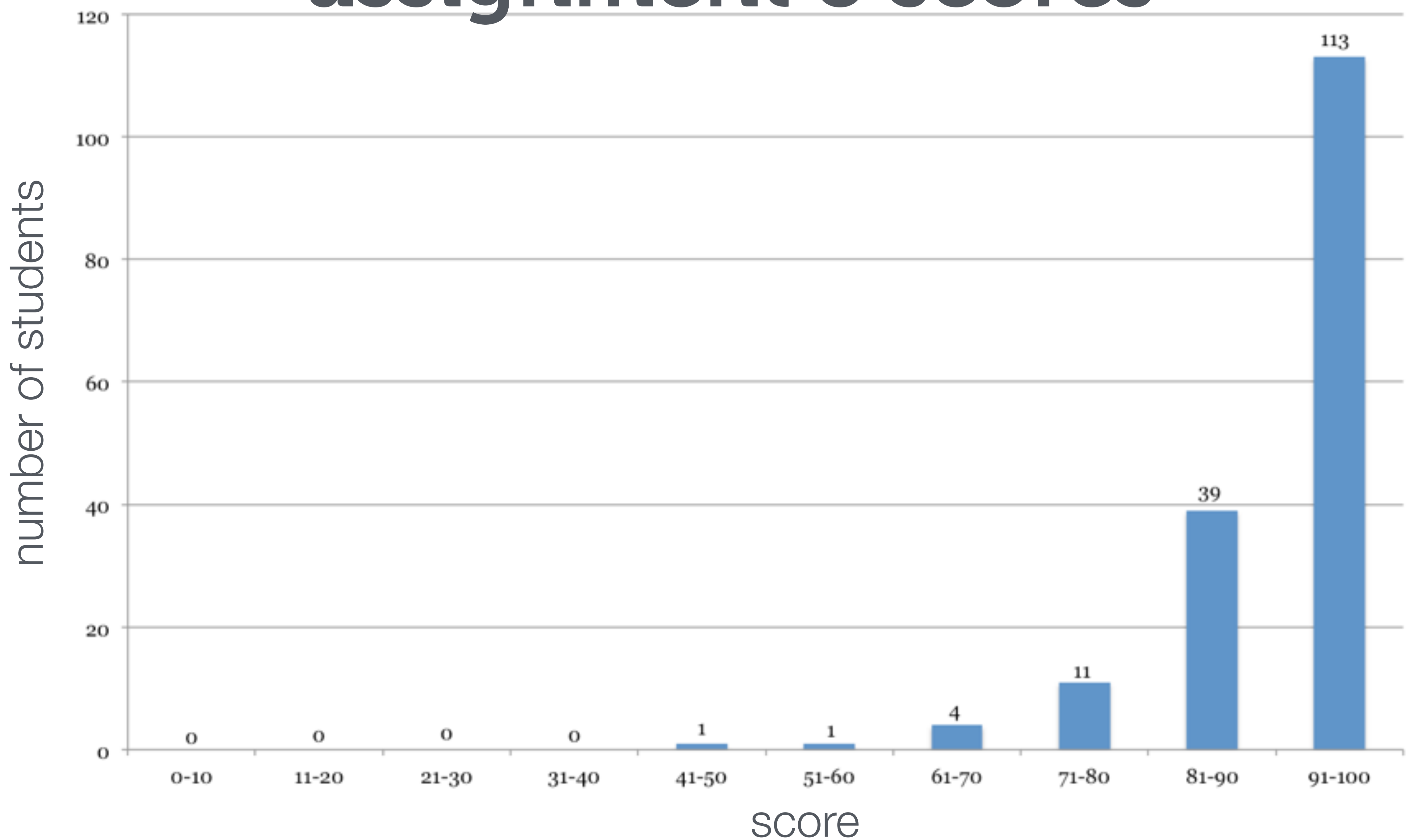
-assignment 11 is due tomorrow night

 -extra day to complete

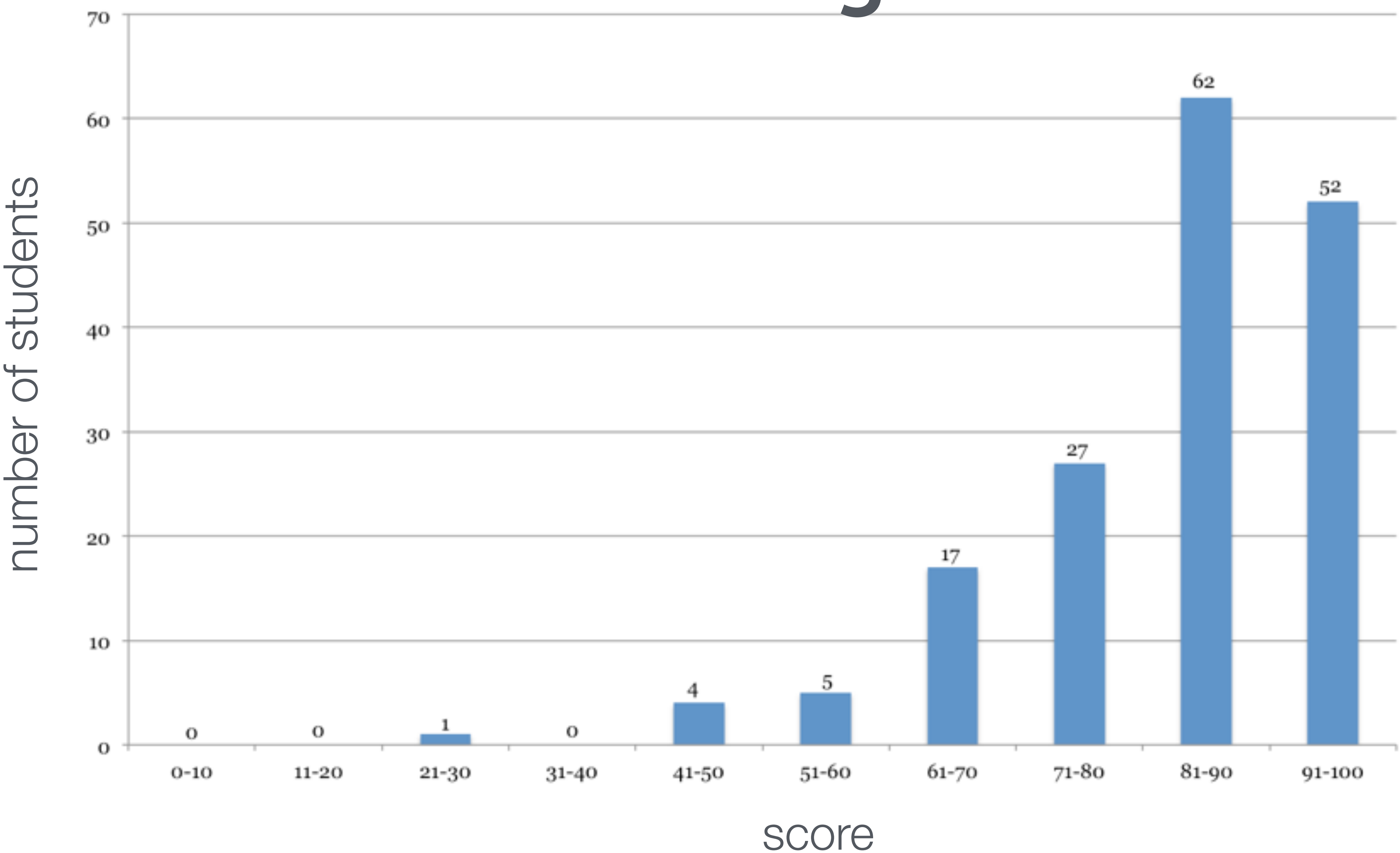
-assignment 12 up later today, due Tuesday, April 21st

 -we will finish-up necessary lecture material next Tuesday

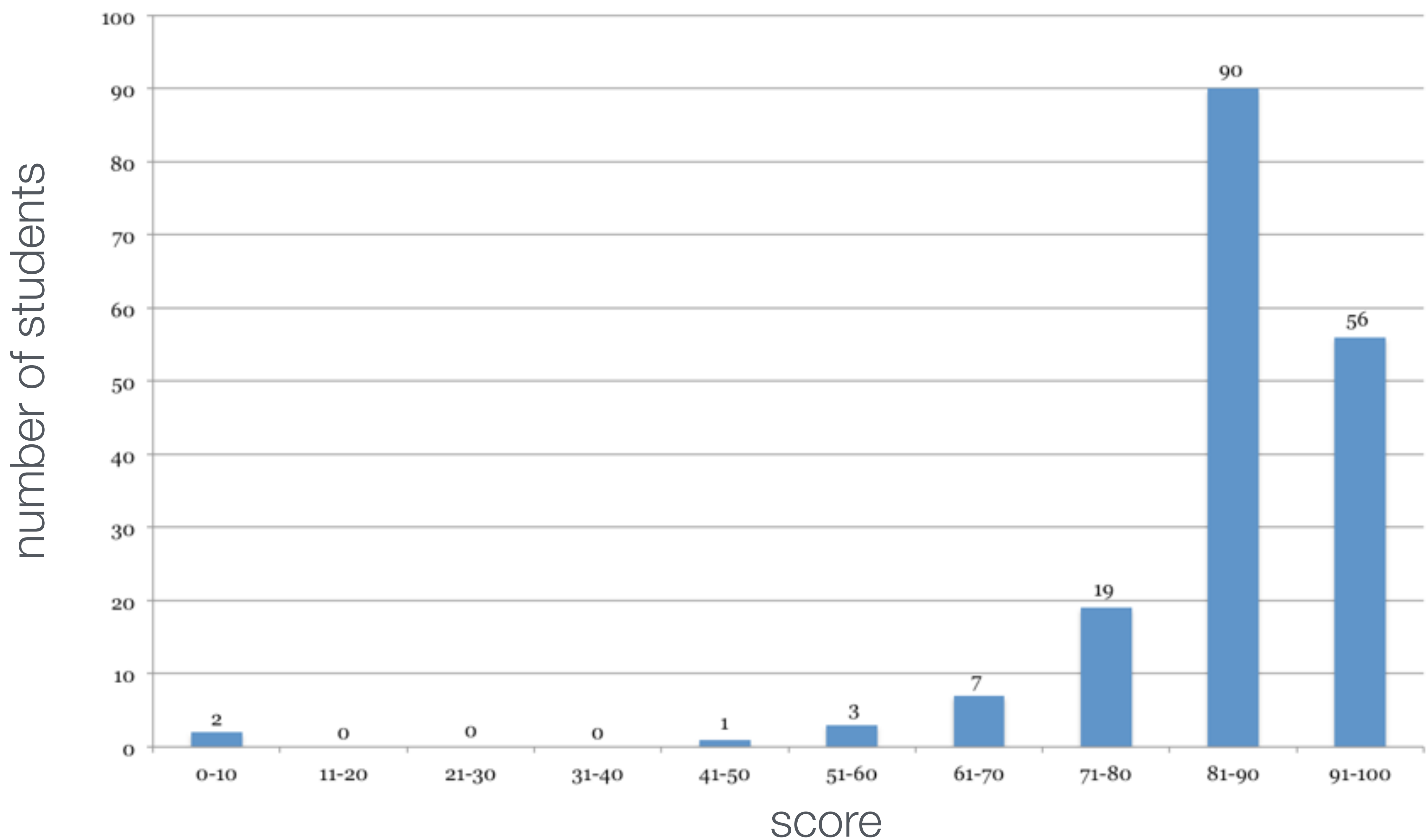
assignment 8 scores



midterm 2 grades



final scores (so far)



jason…

-St Thomas Moore Church, 3015 Creek Rd, Cottonwood Heights, UT 84093

- -vigil tomorrow night 6-8pm
- -funeral Saturday at 2pm

-Multifaith Memory Service at the U

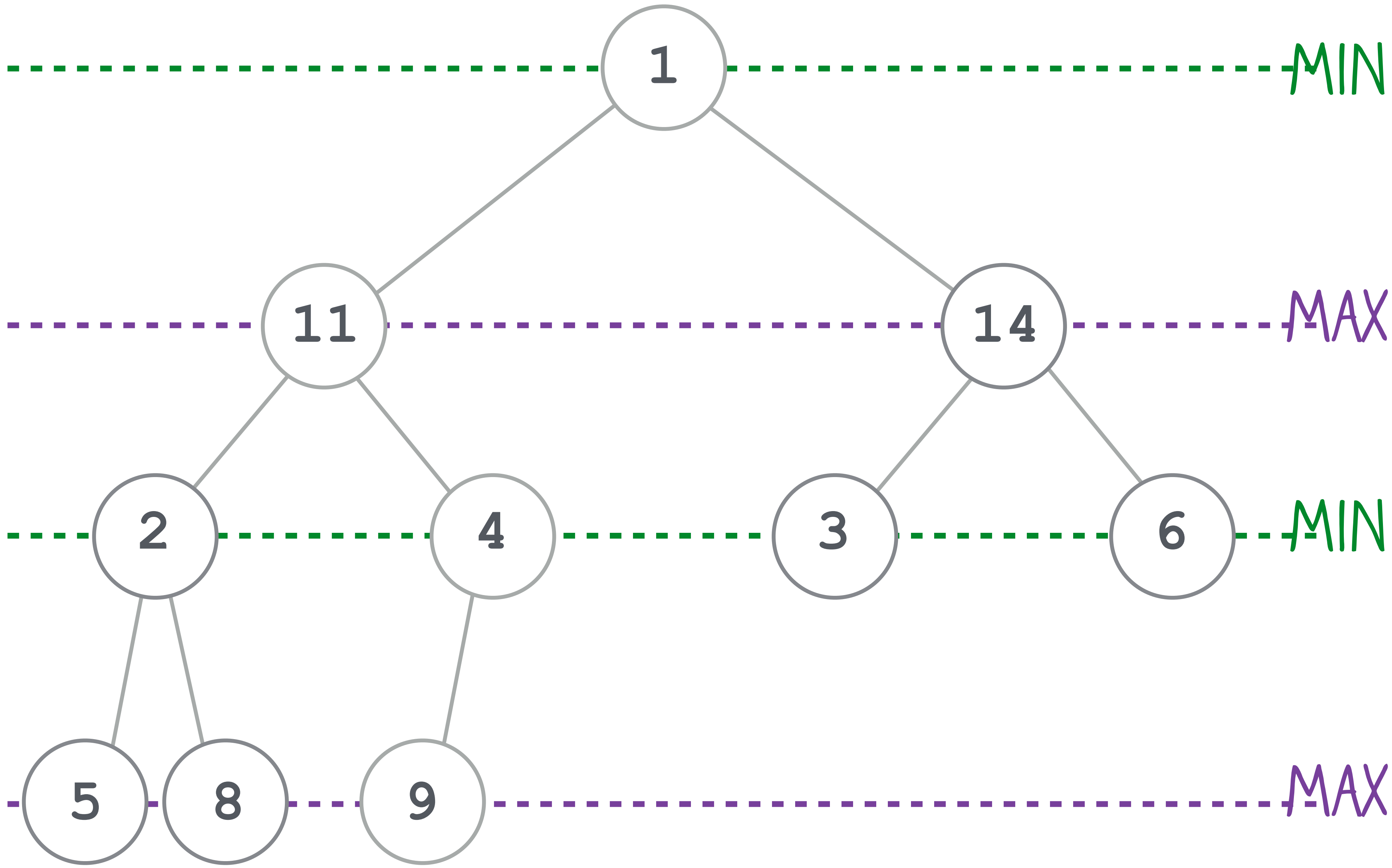
- -Friday, April 17th at 2pm, Saltair Room of the Union

last time…

-a **min-max heap** further extends the heap order property

- -for any node E at *even* depth, E is the minimum element in its subtree
- -for any node O at *odd* depth, O is the maximum element in its subtree

- -the root is considered to be at even depth (zero)


1
MIN
11
14
MAX
2
4
3
6
MIN
5
8
9
MAX

add

- AGAIN, we must ensure the heap property structure
  - must be a complete tree
  - add an item to the next open leaf node

- THEN, restore order with its parent
  - does it belong on a min level or a max level?
  - swap if necessary
  - the new location determines if it is a min or max node

- percolate up the appropriate levels
  - if new item is a max node, percolate up max levels
  - else, percolate up min levels

delete

delete max (min is analogous)

1. locate node X (node containing max item)

2. replace X with last node in tree (last index in array!)

3. determine if new X is violating order property with direct children
   - if so, swap contents of X with the largest child

4. percolate new item X down max levels

5. if lowest max level reached, restore order with lowest min level (if applicable)

today…

-number encodings

-file compression

-data compression

-data decompression

-Huffman’s algorithm

-building the compression tree

number encodings

binary

-each bit represents power of 2

1	0	0	1	0	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
on	off	off	on	off	off	on	on

-sum up all the bits that are on

-128 + 16 + 2 + 1 = 147

-how can we convert the other way?

WHAT IS THE BINARY REPRESENTATION OF THE NUMBER 39?

A) **1 0 1 0 0 1**
B) **1 0 0 1 1 1**
C) **0 1 0 1 1 1**
D) **1 1 0 0 0 1**

There are 10 types of people in the world:
those who understand binary, and those who don't.

-all data in a computer is stored in binary

-a **bit** is a **b**inary dig**it**

-a **byte** is an 8-bit value

-common until of data in a computer

-bytes

10100110

00000000

HOW MANY DIFFERENT VALUES CAN 4 BITS HOLD?

A) **7**
B) **8**
C) **15**
D) **16**
E) **31**
F) **32**

ASCII

-each character corresponds to one byte

-remember, a byte is just an 8-bit number! (0-255)

-for example:

00100000 = 32 = ‘ ‘ (blank space)

00111011 = 59 = ;

01000001 = 65 = A

01000010 = 66 = B

-a simple file containing the text “Hello” is stored on the disk as:

0100100001100101011011000110110001101111

-the text editor know to treat each byte as an ASCII value

-in reality, they are just bytes

-how much disk space do you need for a 1000 character text file?

hexadecimal

-hexadecimal is the base-16 number system

-we only have 10 digits (0-9), so to use a number base greater than 10 we need more symbols

-in hex, we use the letters A through F
- -A represents the value ten
- -F represents the value fifteen

counting

-in binary, we reset and add a digit at 1

-in decimal, we reset and add a digit at 9

-in hex, we reset and add a digit at F (ie. 15)

…
8
9
A
B
C
D
E
F
10 (= sixteen)
11 (= seventeen)

-each decimal place represents a power of 16

1AF

$= (1 * 16^2) + (A * 16^1) + (F * 16^0)$

$= 256 + 160 + 15$

$= 431$

-hex is useful because each hex digit corresponds to one half-byte (ie. 4 bits)

-reading raw byte data is almost always done in hex

- -two hex digits makes up a single byte

-bytes in hex:

1A

00

13

FF

D5

hex to binary

-each hex digit is a specific 4-bit sequence

    0 = 0000
    1 = 0001
    …
    E = 1110
    F = 1111

-converting from hex to binary is as simple as representing each digit with its bit-sequence

    12 EF = 0001 0010 1110 1111

-a single byte is two hex digits

- -the bytes in the above are 12 and EF

WHAT IS THE HEX VALUE OF THESE 8 BITS? 1010 0010

A) **B2**
B) **A2**
C) **12**
D) **10**

file compression

-why do we care about file compression?

-reducing traffic on networks and the internet

-reducing disk space requirements

-various media formats
- -MP3
- -MPEG4
- -JPEG
- -…

-YouTube, Netflix, GoogleMaps, etc. would not be possible without compression

exercise

-suppose we the following string stored in a text file:

ddddddddddddabc

-how many bytes of disk space does it take to store these 15 characters using ASCII?

-is there any way to represent this file in fewer bytes?

-hint: take advantage of repeated characters…

- two phases for data compression:
  - compression (encode data in fewer bits)
  - decompression (decode back to original data)

data compression

-allow number of bits for each character to vary, instead of using the full 8 for every character

-represent common characters with fewer bits, represent rare characters like ‘q’ and ‘z’ with more bits

-with ASCII we now each char is 8 bits

- -how do we reproduce (decompress) original file if characters are of varying lengths?

-for example:

- -‘e’ may be the two-bit value 11
- -‘q’ may be the four-bit value 1100
- -‘z’ may be the four-bit value 0011

-what characters does the following file contain?

- -001100
- -is there any way to know?

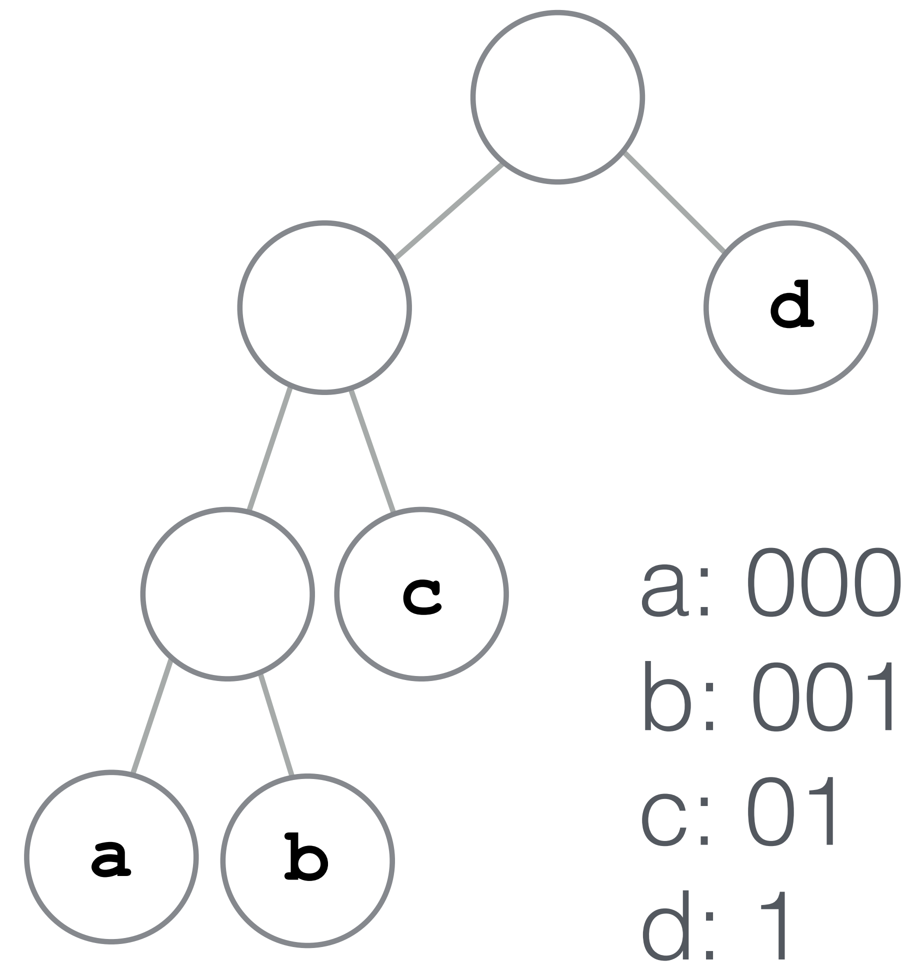
-we must include the secret decoder ring with the compressed file

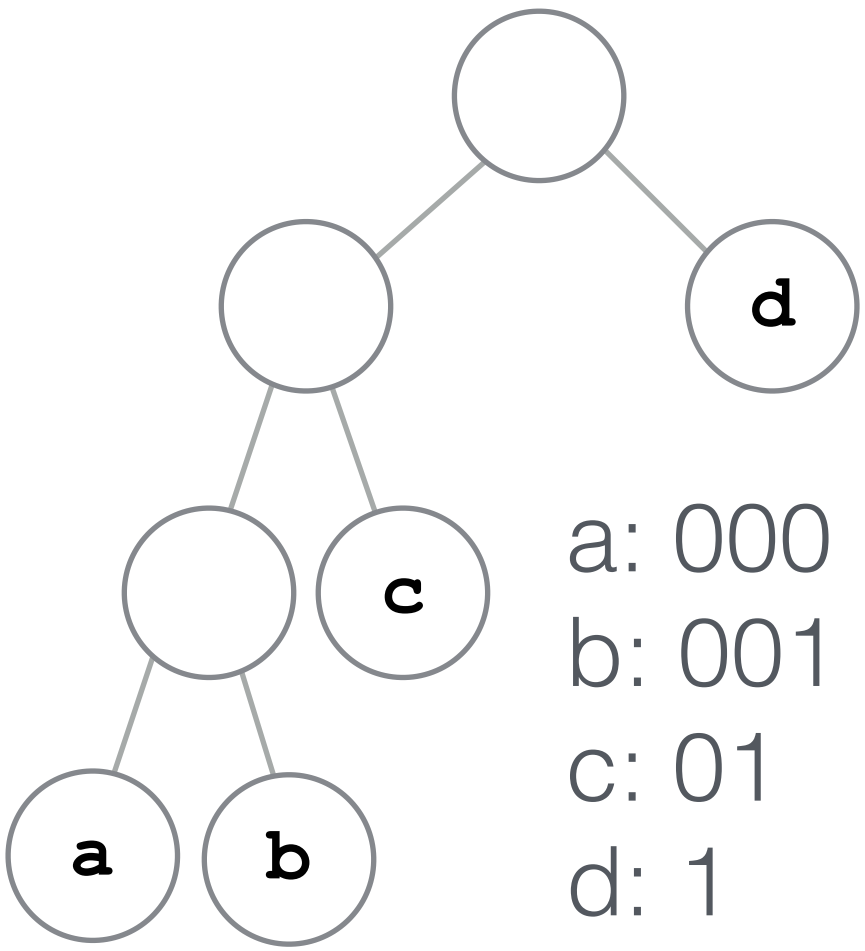
binary trie

-no, I did not misspell it!

-a **binary trie** is a binary tree in which a left branch represents a 0 and a right branch represents a 1

-the path to a node represents its encoding





a: 000
b: 001
c: 01
d: 1

“ddddddddddddabc” takes 15 bytes (120 bits) in ASCII

11111111111100000101 is less than 3 bytes (20 bits)

WHY IS THE D NEAR THE TOP OF THE TREE?

data decompression

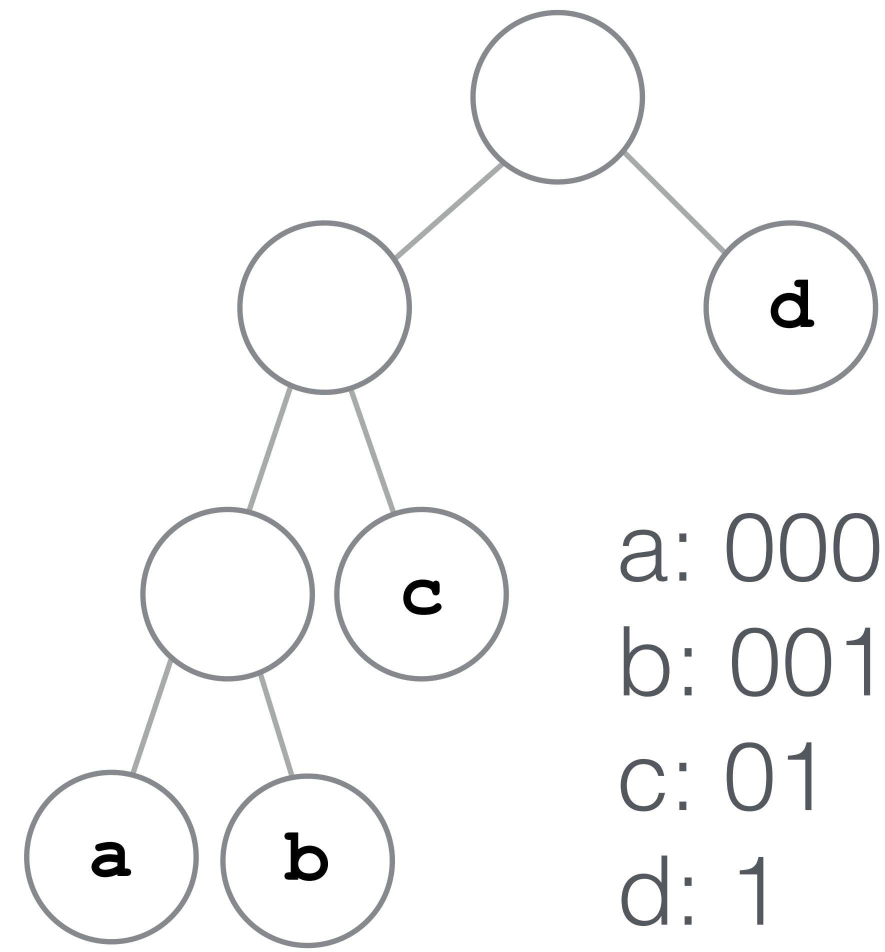
-read the bits of the compressed file one at a time

1111111111111100000101

-on 0 go left, on 1 go right

-when a leaf node is reached, print char contained in node

- -start back at the root for the next bit

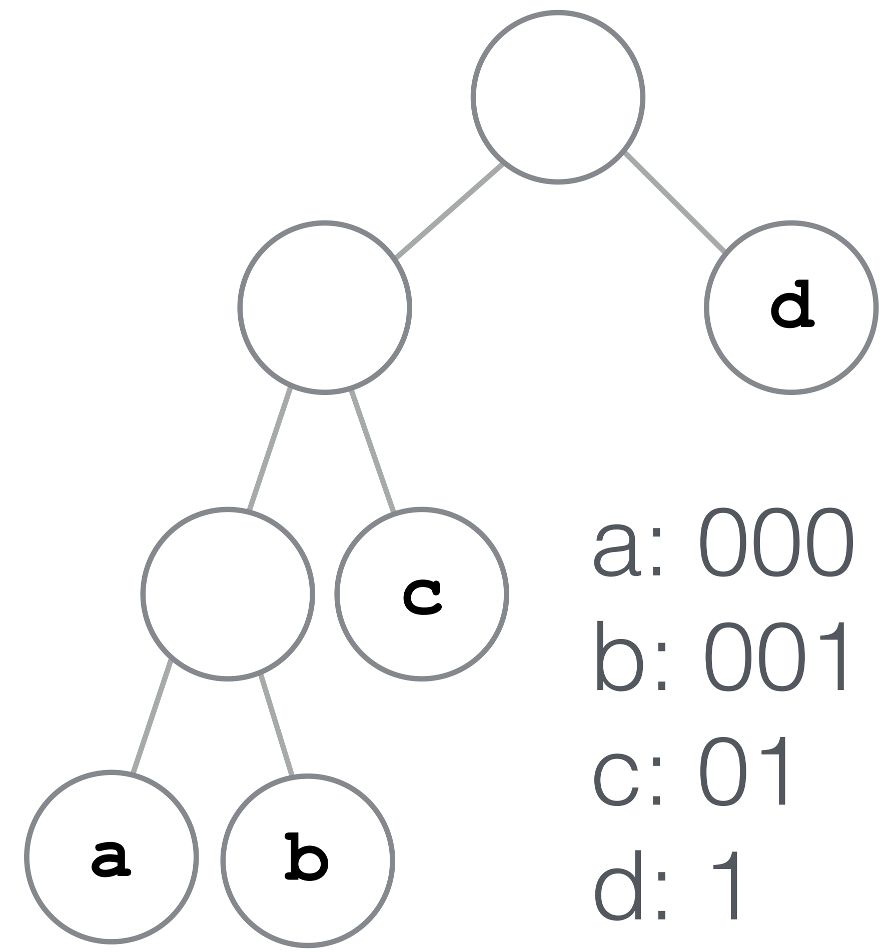


a: 000
b: 001
c: 01
d: 1

-what does the following bit-string encode?

0 1 0 0 0 0 0 1

-how many bytes is this encoded in? what if it was in ASCII?

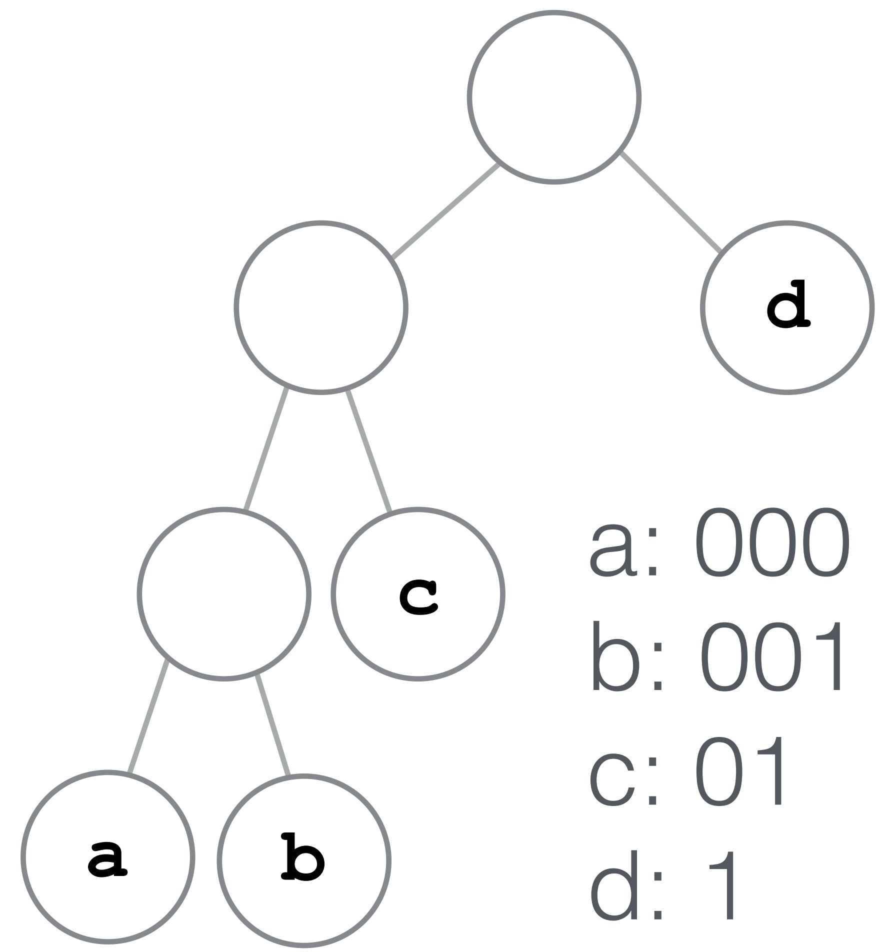


a: 000
b: 001
c: 01
d: 1

-read the bits of the compressed file one at a time

-a **binary trie** is a binary tree in which a left branch represents a 0 and a right branch represents a 1

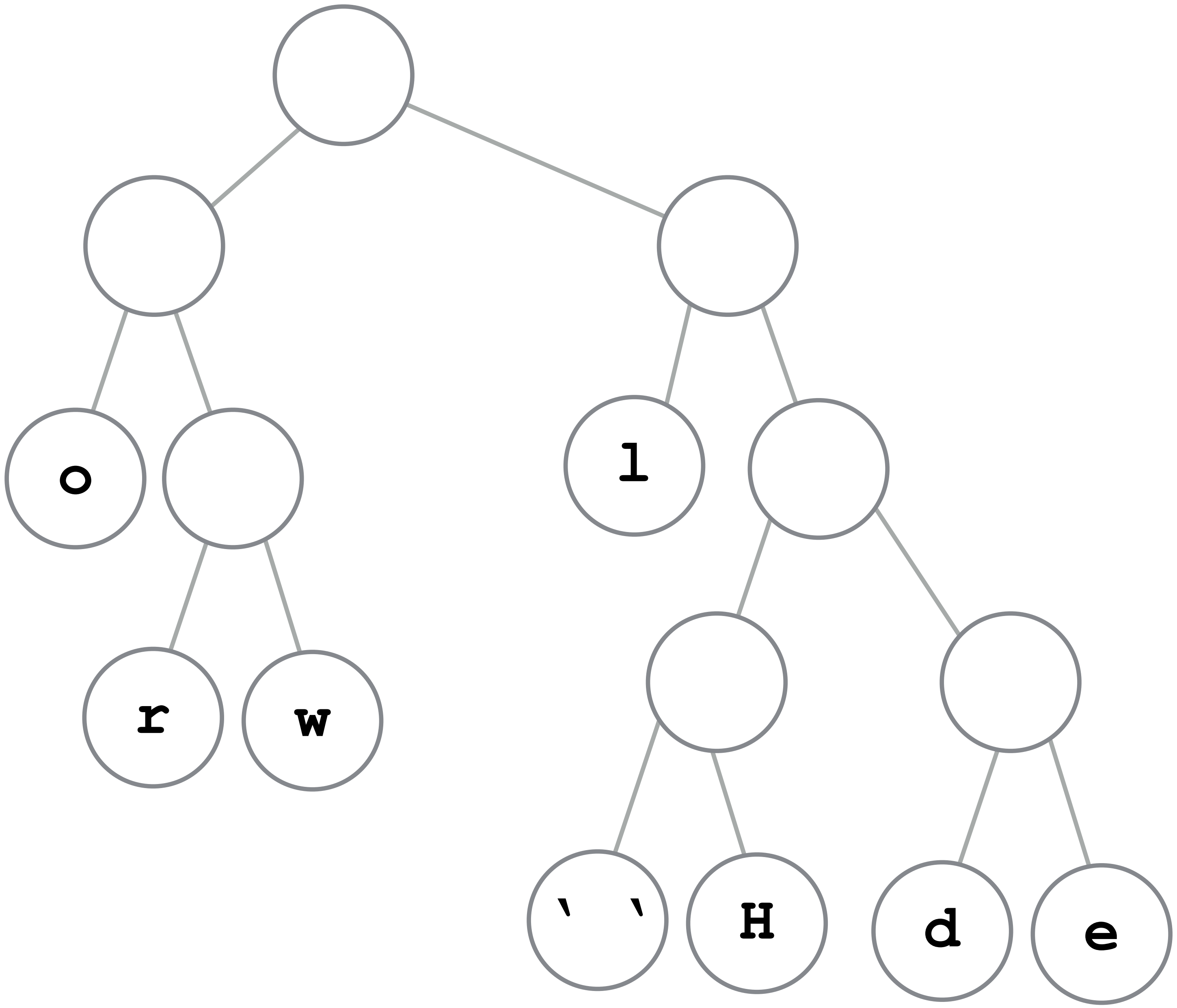
-the path to a node represents its encoding



a: 000
b: 001
c: 01
d: 1

WHAT STRING DO THESE BITS ENCODE?
0 1 1 0 0 0 0 1 0

A) **low**
B) **wow**
C) **wool**
D) **were**



Huffman’s algorithm

-idea is to encode most frequent characters with fewest bits

-common characters will be near the top of the tree

-uncommon near the bottom

-every file has a different frequency of characters, and therefore a different compression tree

Huffman's algo

-count the frequency of each different character

- -"hello world"
  - h = 1
  - e = 1
  - l = 3
  - o = 2
  - ' ' = 1
  - w = 1
  - r = 1
  - d = 1

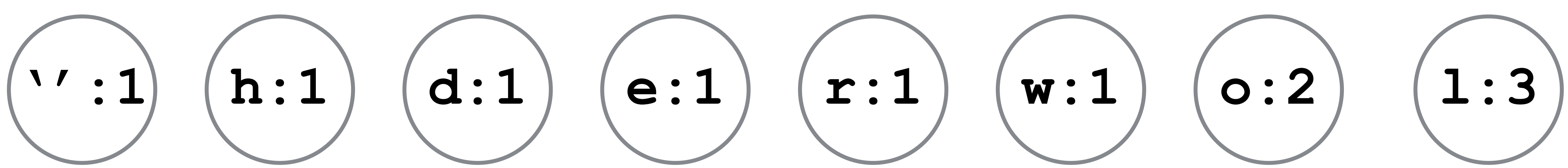
-construct a binary trie with highest frequency characters near the top, lowest near the bottom

-include this tree representation with the compressed file

building the compression tree

START WITH A SEPARATE TREE FOR EACH CHARACTER

EACH TREE IS A SINGLE ROOT NODE, HOLDING THE CHARACTER AND ITS FREQUENCY

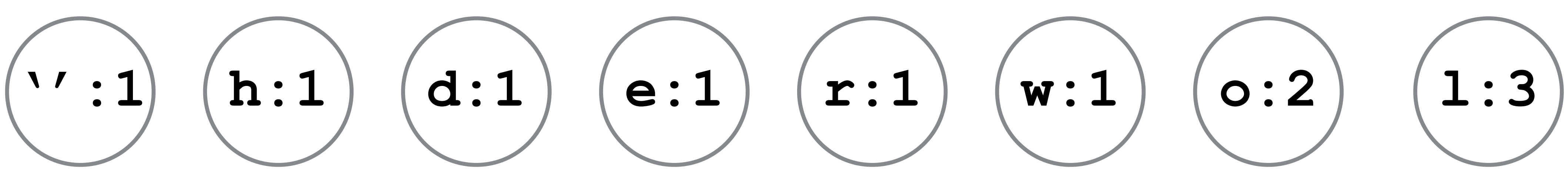


hello world

MERGE THE TWO LOWEST WEIGHT TREES TOGETHER INTO ONE NEW TREE

MAKE A NEW PARENT NODE WITH THEIR COMBINED WEIGHT; SMALLER NODE ON THE LEFT, LARGER ON THE RIGHT

LOTS OF TIES HERE! H, E, W, R, D, AND ‘ ‘
NEED A TIE-BREAKER . . . MORE ON THIS LATER . . .

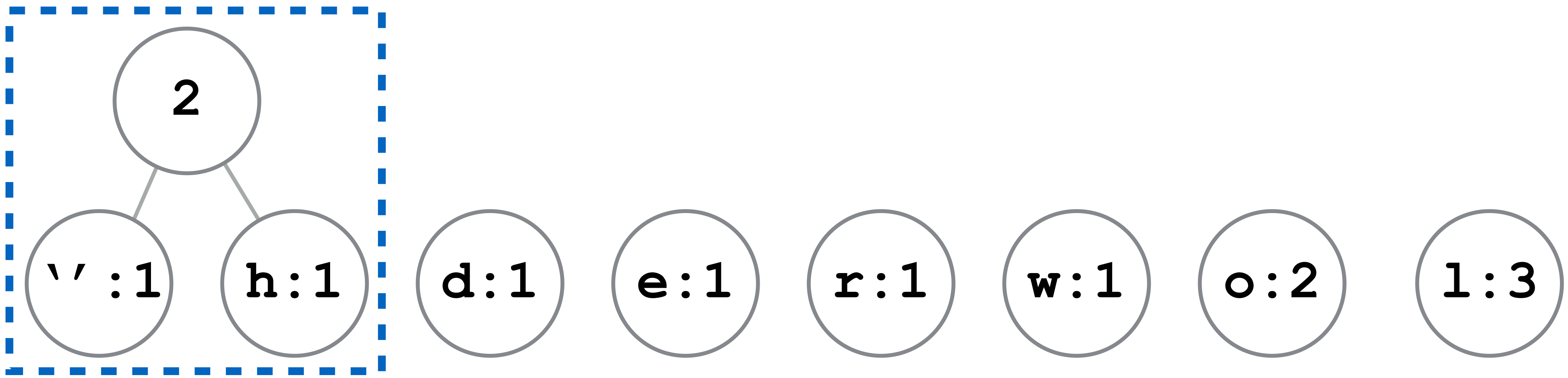


hello world

REMOVE ' ' AND H TREES FROM THE LIST

ADD NEW MERGED TREE TO THE LIST

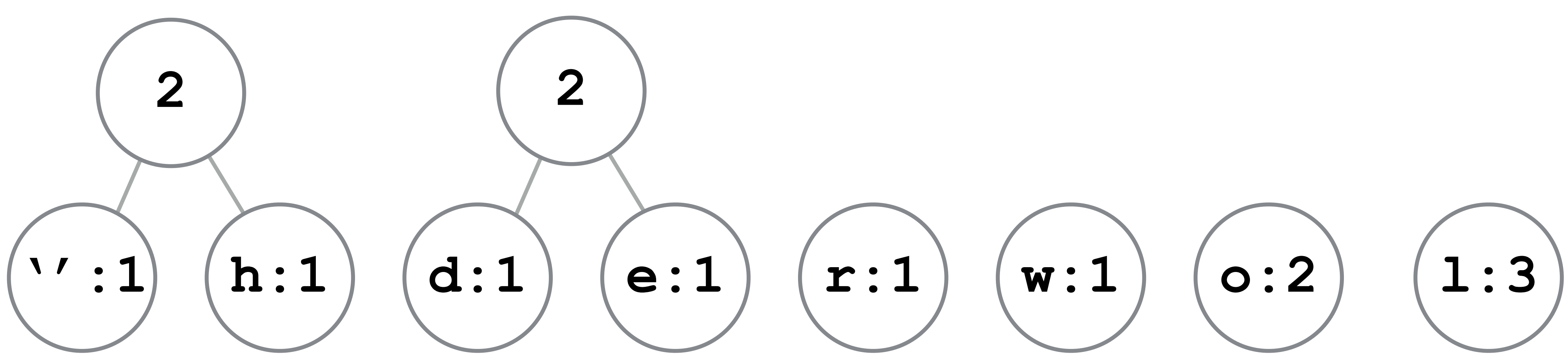
NOTE: NON-LEAF NODES DON'T HAVE A CHARACTER, ONLY A WEIGHT



hello world

CONTINUE PROCESS OF MERGING TWO SMALLEST TREES

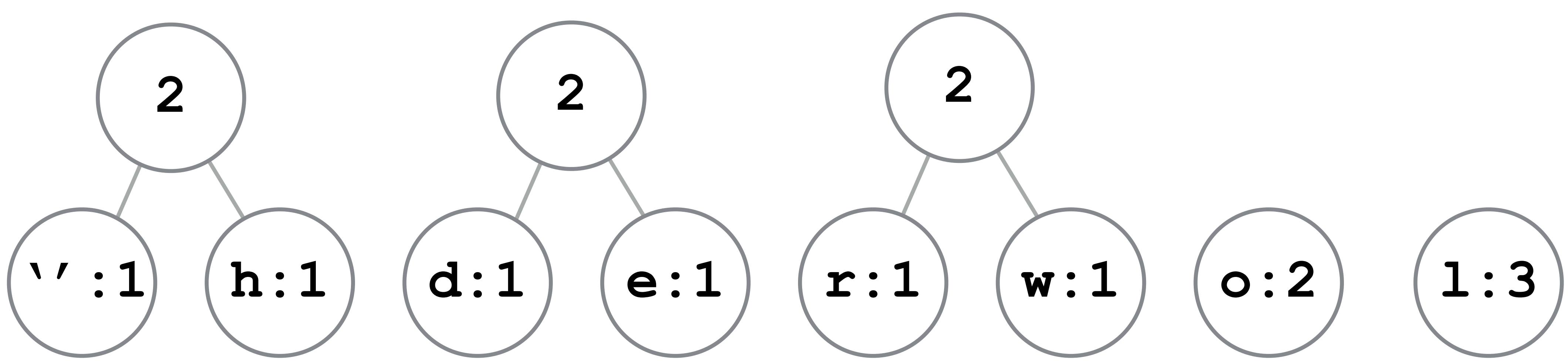
WHICH IS NEXT?



hello world

CONTINUE PROCESS OF MERGING TWO SMALLEST TREES

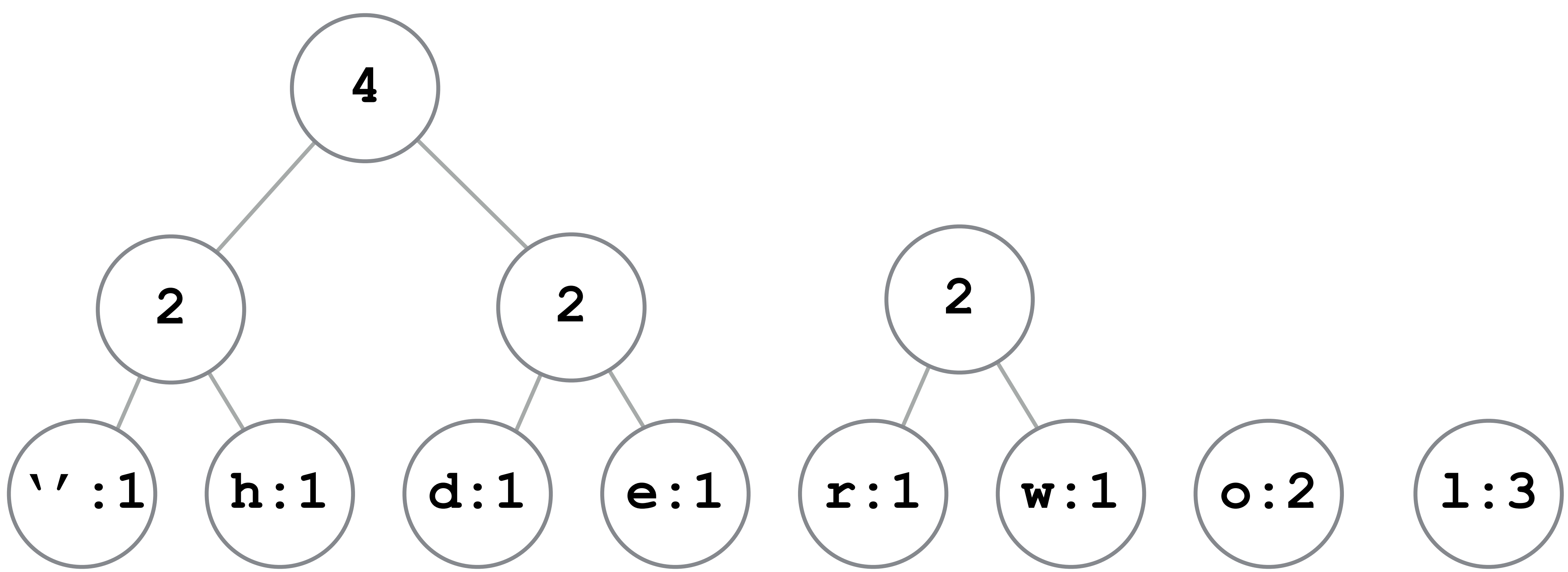
WHICH IS NEXT?



hello world

CONTINUE PROCESS OF MERGING TWO SMALLEST TREES

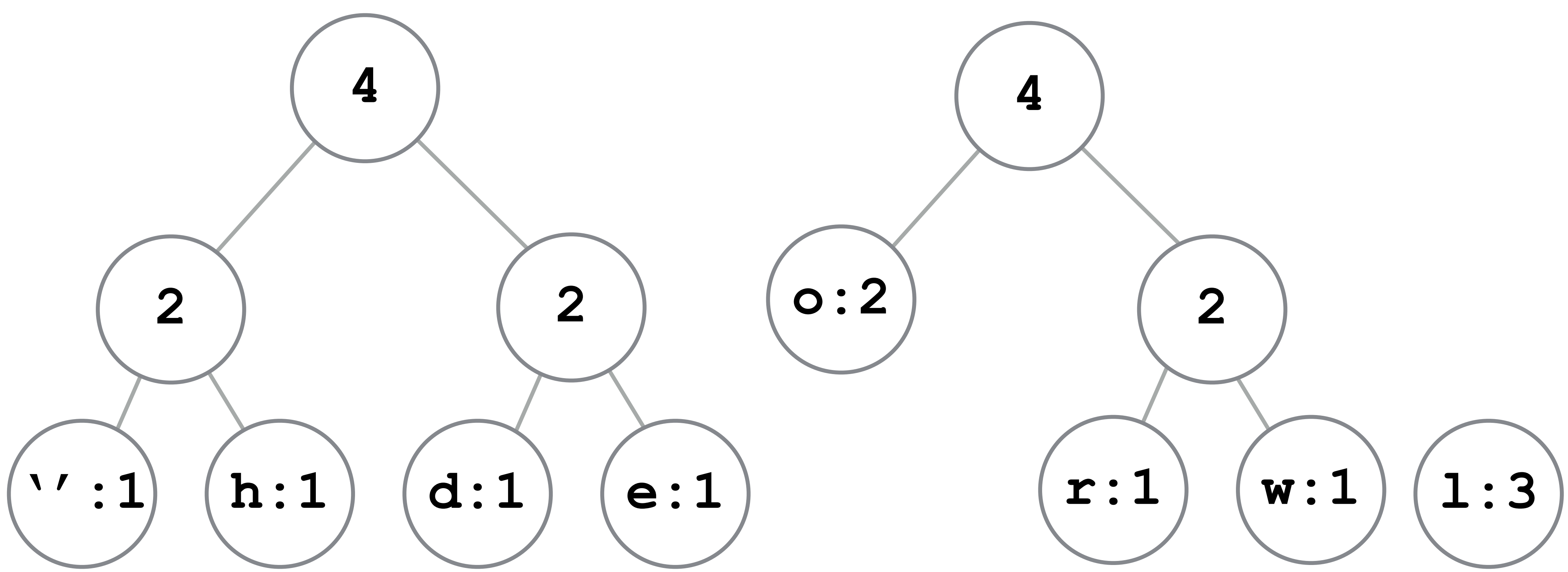
WHICH IS NEXT?



hello world

CONTINUE PROCESS OF MERGING TWO SMALLEST TREES

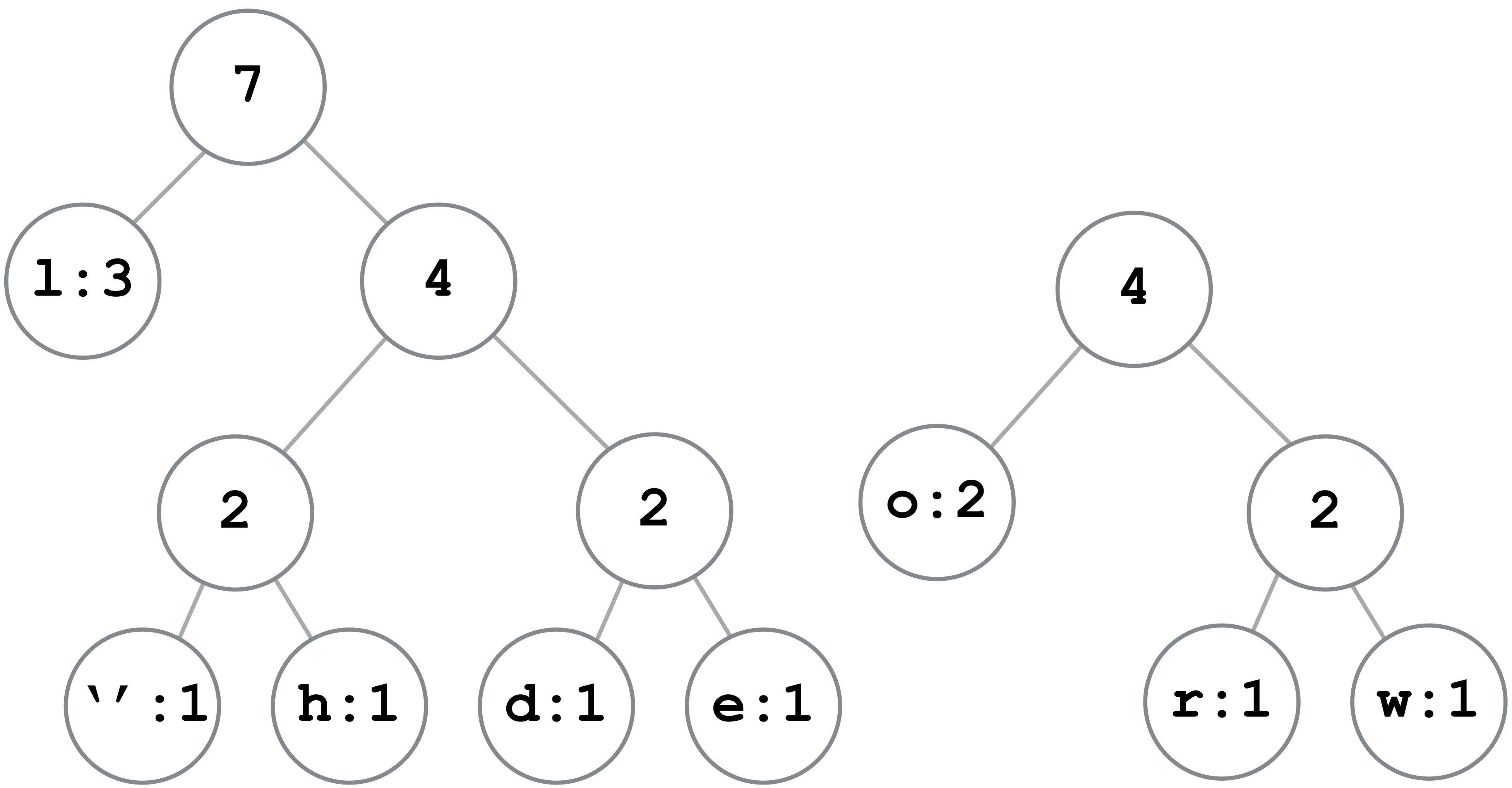
WHICH IS NEXT?



hello world

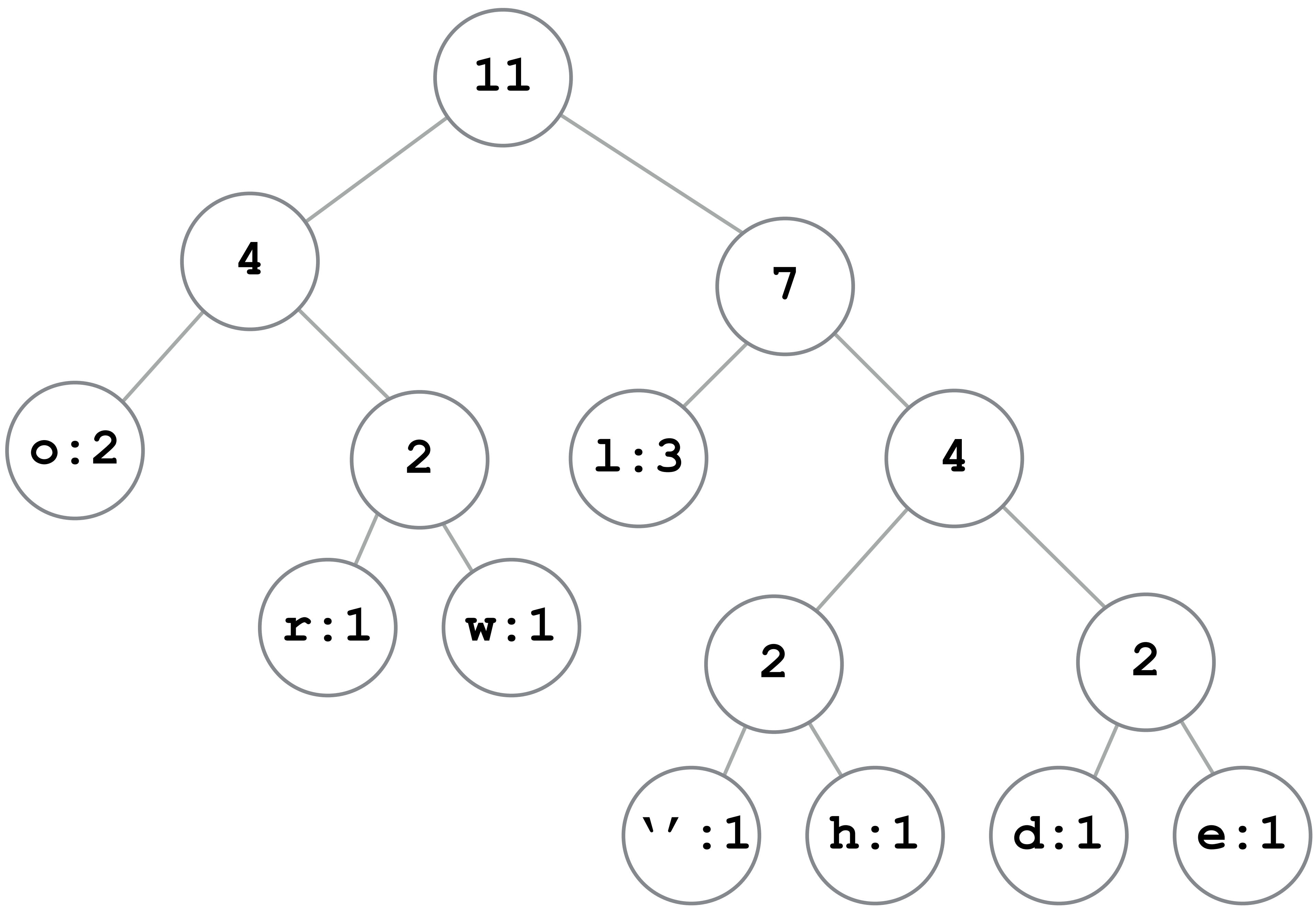
CONTINUE PROCESS OF MERGING TWO SMALLEST TREES

WHICH IS NEXT?



hello world

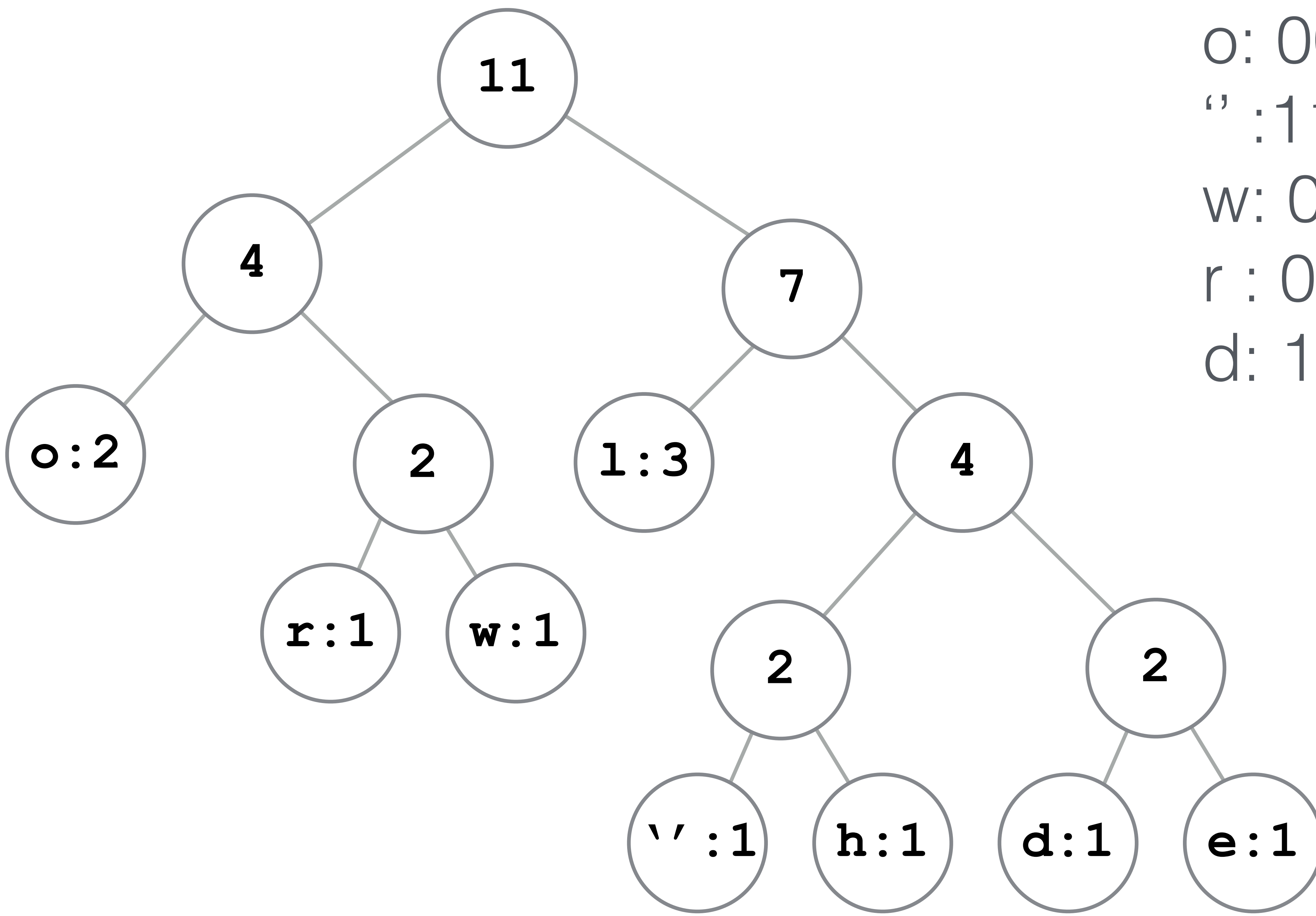
DONE!



hello world

NEW CHARACTER ENCODING

h: 1101
e: 1111
l : 10
o: 00
‘’ :1100
w: 011
r : 010
d: 1110



hello world 110111111101000110001100010101110

(re)building the compression tree

-the compressed file now contains non-ASCII, varying length bytes

-decompression needs the same tree in order to decode

-must add *header* information that describes the tree

- -this header should not be compressed!

-next time…

next time…

- reading
  - chapter 12 in book

- homework
  - assignment 11 due tomorrow night
  - assignment 12 is out, due a week from Tuesday