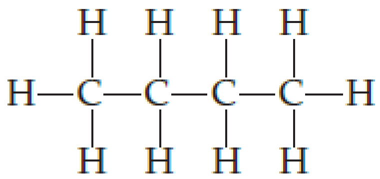
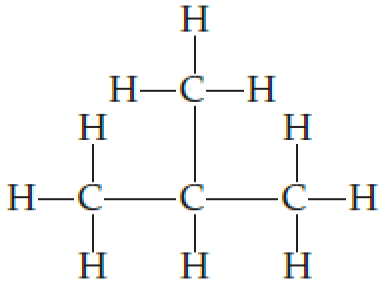
Trees

Background

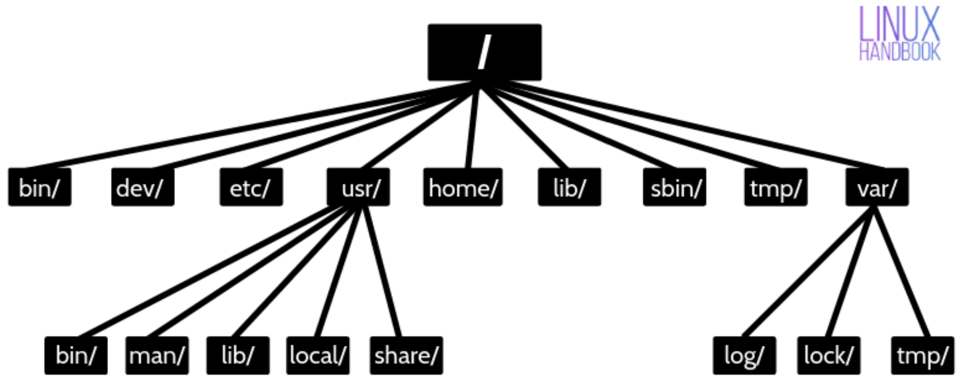
- Arthur Cayley, English mathematician, discovered trees in 1857 when he needed a way to count chemical compounds
- Fast Searching
- Data compression
- Modeling games/procedures/decisions/relationships
- Filesystem hierarch
- Evaluating mathematical expressions

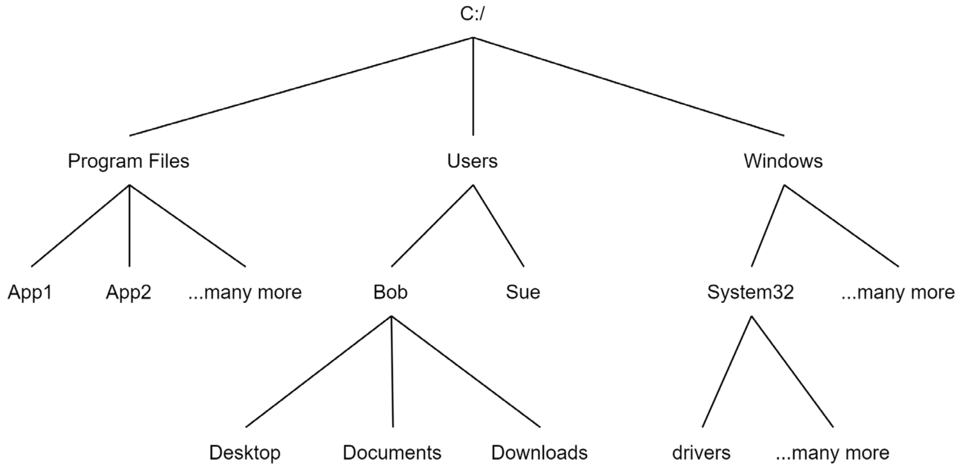


n-butane



isobutane

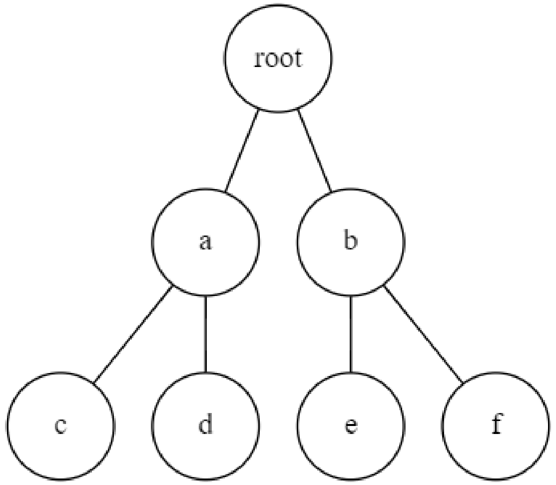


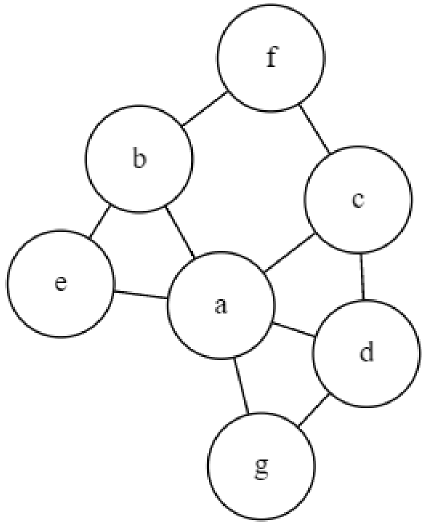


Neff, Rick, and Gopalakrishnan, Ganesh. *First Three Odds Double Halve Divide Reciprocate*. CRC Press, 2021
Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Trees vs Graphs

- Both are composed of vertices and edges
- Tree: Only one path from one vertex to any other vertex
- Graph: More than one path between vertices





Graphic from: https://medium.com/swlh/making-data-trees-in-python-3a3ceb050cfd

Tree Terminology

Free Tree

Rooted Tree

Root

Level

Height

Parent

Child

Ancestor

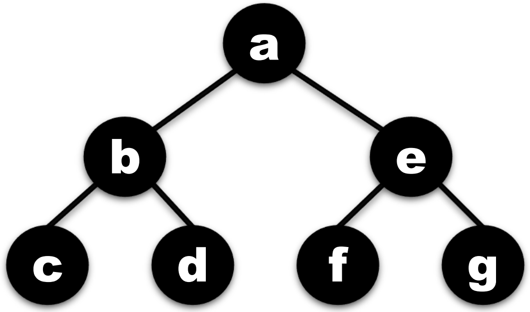
Descendant

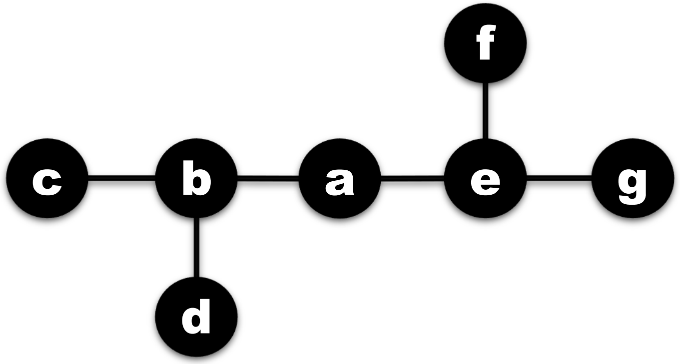
Leaf

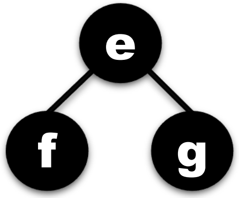
Internal vertex

Sibling

Subtree







Review:

What does it mean if two graphs are isomorphic?

With a partner, complete Additional Exercises:

14.1.2

A Few Tree Applications

Binary Search Trees

- Fast searching. Optimize **time**.

Huffman Trees

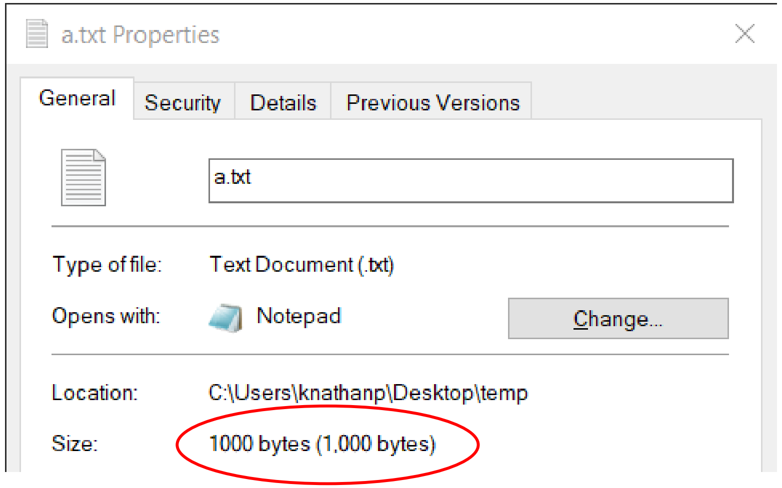
- Data compression. Optimize **space**.
- Binary tree, but not a binary **search** tree.
- Used to create an optimal prefix code

Game trees

Mathematical expressions

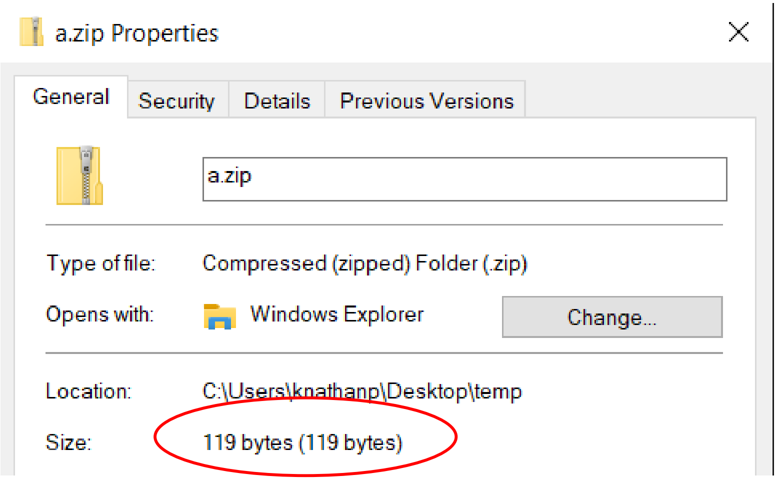
Consider a text file that consists of 1000 a's.

aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
...



How much space would it take to store this file if we use standard ASCII or UTF-8, which each use 1-byte (8-bits) to store the letter "a"?

ZIP compression



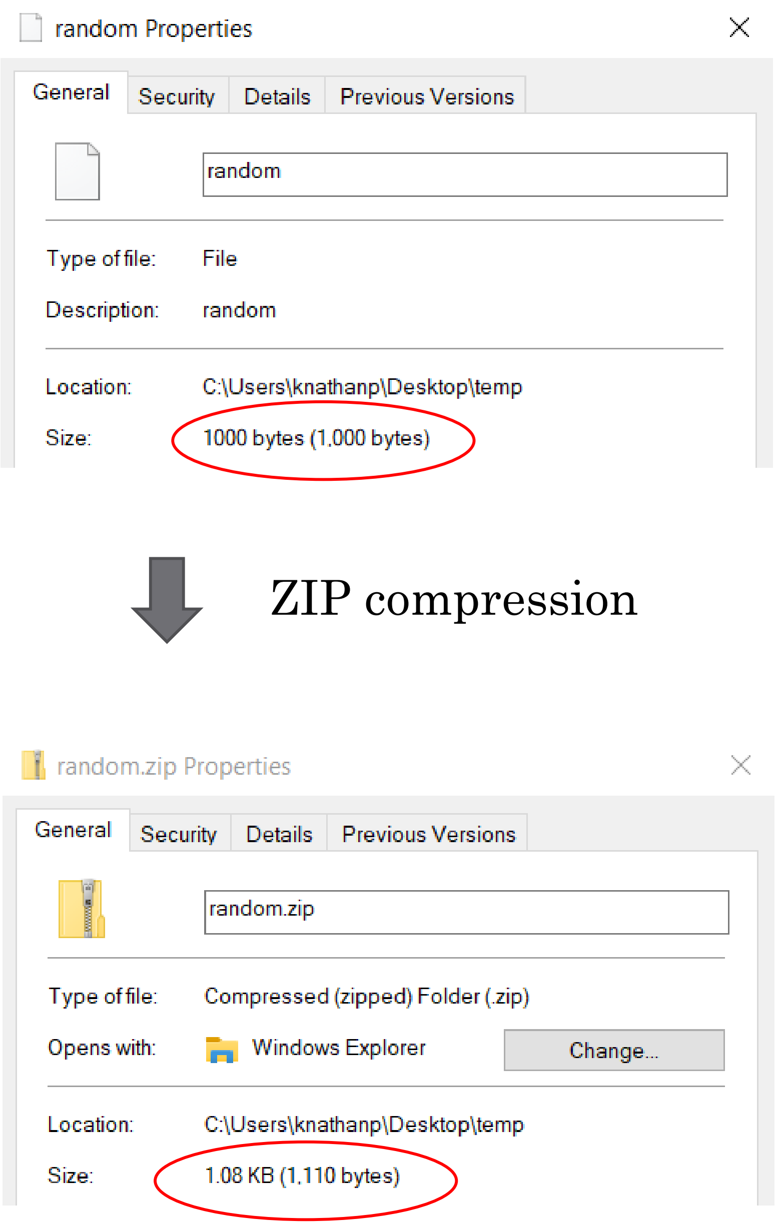
Can you think of a way to store this file in a way that would save space?

What is the compression ratio?

$$\frac{\text{orignal} - \text{compressed}}{\text{original}} \qquad \frac{1000 - 119}{1000} = 0.88 = \mathbf{88\%}$$

What about a file that consists of 1000 completely random bytes?

q2KVZRbqRiflgI56Es4plUeWoWBk
w2b892Ss5JWSJAk5SPaIksHucUo
GFMszZ2pqBB
...



Can you think of a way to store this file in a way that would save space?

Why don't we save any space by compressing it?

Fixed-length Encoding

Encode means "represent"

- How many bits would it take to encode the letters of the English alphabet? (the minimum number of bits)
  - There are 26 letters
  - 4 bits (2^4) will only give 16 possible encodings
  - Need at least 5 bits
  - But that gives $2^5 = 32$ different possibilities. Waste of bits
  - There are six combinations of bits that are unused

- This is a **fixed-length** encoding. Each symbol/letter uses five bits.
- Wastes **space** and **time**.
- Can we find a coding scheme that uses fewer bits?

A	0000	00000	0	0
B	0001	00001	1	1
C	0010	00010	2	2
D	0011	00011	3	3
...	...	...		
O / Z	1110	11001	14	25
P / Unused	1111	11010		26
Q	?	11011		27
R	?	11100		28
S	?	11101		29
T	?	11110		30
...	?	11111		31

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Variable-length Encoding

- Instead of using a fixed number of bits to represent each letter/symbol, we can use bit strings of different lengths to encode letters.
- For efficiency, letters that occur more frequently should be represented using shorter bit strings. Letters that are rarely used should be represented using longer bit strings.
- Example:

  The letters **e**, **a**, **t** are more common than x and z.

  What if we represent the letter **e** with 0, **a** with 1, and **t** with 01?

  Then the bit string 0101 could represent **eat**, **tea**, **eaea**, or **tt**.

e	0
a	1
t	01

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Prefix Codes

- One way to make sure no bit string corresponds to more than one sequence of letters is to encode letters in such a way that the bit string for a letter **never occurs as the first part of a bit string for another letter**.
- This is called a Prefix Code.
- Example

  To represent the letters **e**, **a**, **t** using a prefix code, we could choose 0 for **e**, 10 for **a**, and 11 for **t.**

  The bitstring 10110 represents the word **ate**. There is no other possible interpretation.

  Now we can decode or recover a word from a bitstring without ambiguity.

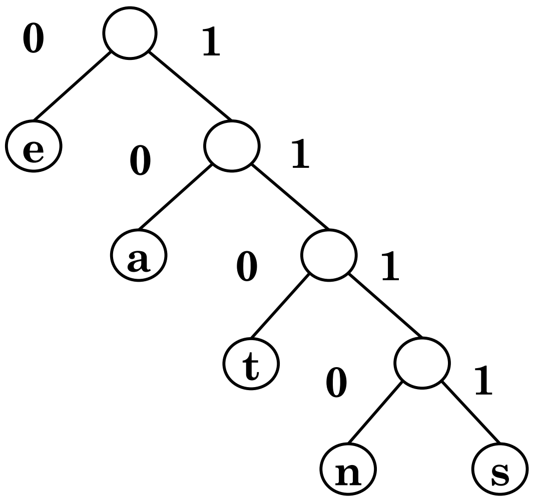
e	0
a	10
t	11

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Prefix Codes

- A prefix code can be represented using a binary tree.
- Leaves represent the characters/symbols we need to represent
- Edges represent the encoding bits. A left edge is a 0 and a right edge is a 1.
- The bit string used to encode a symbol is the sequence of edges to reach the symbol.
- Because this is a tree, the path to get to any leaf is **unique.**
- The sequence of bits formed by the path is a **prefix code**.

e	0
a	10
t	110
n	1110
s	1111




```
11111011100
S   A N   E
```

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Prefix Code Examples

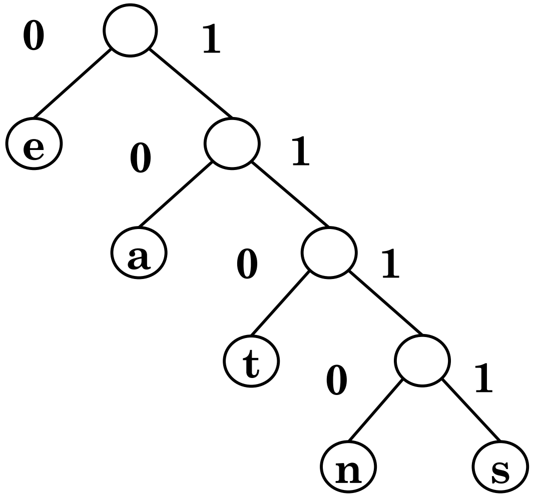
- Huffman codes
- Country calling codes
- ISBN country and publisher assignments
- Machine language instruction sets

Where is Huffman Coding used?

- JPEG
- MP3
- DEFLATE algorithm (Huffman Coding is combined with LZ77 compression)
  - zip, gzip, 7-Zip, C zlib library, PuTTY, ...
  - PNG files

Huffman Trees

- Used to produce a **Huffman Coding**, which is a prefix code.
- It is not only a prefix code, but it is a prefix code using the fewest number of bits possible.
- Fundamental algorithm in data compression.
- Uses a binary tree.
- Store more frequently seen symbols closer to root.
- Label the links rather than the nodes. Left link is 0, right link is 1.
- Symbols we want to represent are stored at the leaves.
- Each symbol is encoded into a unique bitstring by following the structure of the tree.



Compression Ratio

- How do we know how much space/time we are saving by using a variable-length encoding instead of a fixed-length encoding?
- Calculate the **compression ratio!**

Compression Ratio

f: bits per symbol for fixed length encoding

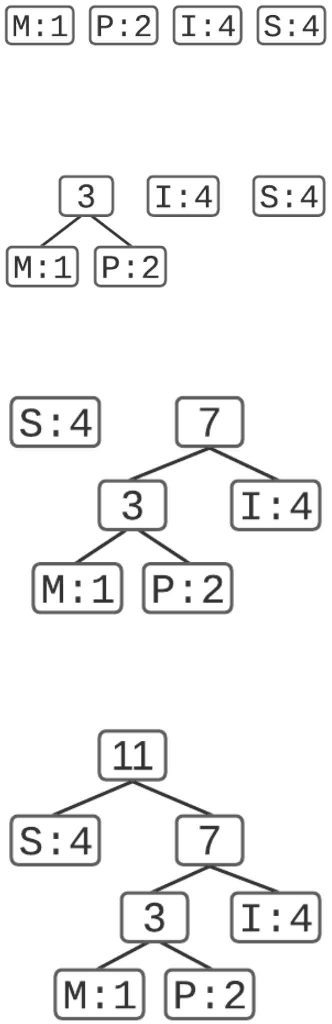
v: average bits per symbol with variable length encoding

$$\frac{f-v}{f} \cdot 100$$

Example: Create a Huffman Tree for the string "MISSISSIPPI"

1. List out the counts
2. Go through the algorithm:
    1. List symbols in a priority queue sorted by count (smallest count first)
    2. Combine first two symbols (M and P) into a tree and add to the proper place in the queue
    3. Repeat again, combining first two items in queue into a tree (M, P, and I) and adding back to the queue.
    4. Continue to repeat until only one item is on the queue. This is the completed tree.
3. Extract the encoding for each letter:

```
100110011001110110111    21 bits
M  I SSI SSI P  P  I
```



Letter	Count
M	1
P	2
I	4
S	4

Letter	encoding
M	100
P	101
I	11
S	0

Example: Create a Huffman Tree for the string "MISSISSIPPI"

Calculate the compression ratio.

How many bits per letter would it take for a fixed-length encoding?

2 bits per letter

Letter	encoding
M	00
P	01
I	10
S	11

`0010111110111110010110` 22 bits

`M I S S I S S I P P I`

How many bits on average with this Huffman encoding?

M: 3 bits * 1 letter = 3

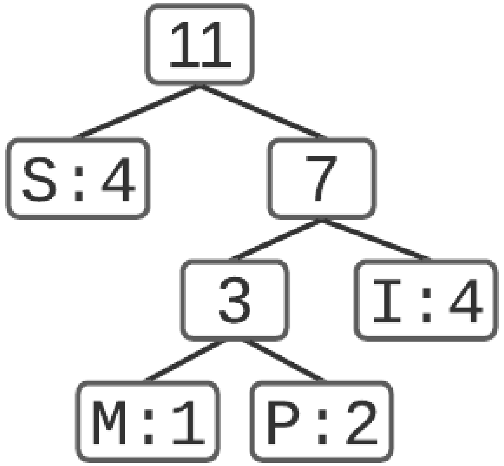
P: 3 bits * 2 letters = 6

I: 2 bits * 4 letters = 8

S: 1 bits * 4 letters = 4

Total: 3+6+8+4 = 21

Average bits per letter: 21/11 = 1.91



Letter	Count
M	1
P	2
I	4
S	4
total	**11**

Letter	encoding
M	100
P	101
I	11
S	0

Compression ratio:

$$\frac{f-v}{f} \cdot 100$$

$f = 2$
$v = 1.91$

$$\frac{2-1.91}{2} \cdot 100 = \mathbf{4.5\%}$$

Example: Create a Huffman Tree for the string "MISSISSIPPI"

```
0010111110111110010110
M I S S I S S I P P I


100110011001110110111
M  I SSI SSI P  P  I
```

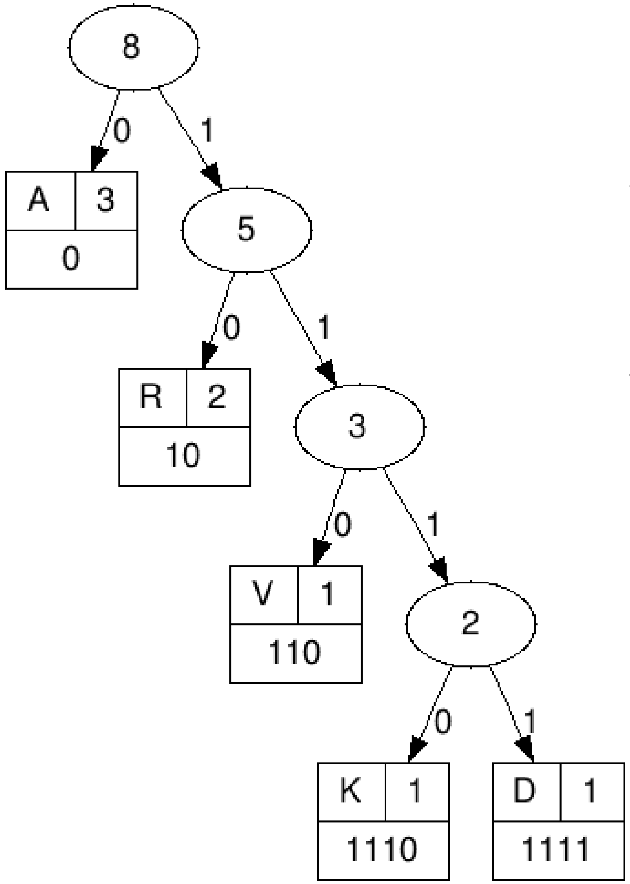
Fixed

Variable

Letter	fixed	variable
M	00	100
P	01	101
I	10	11
S	11	0

Try it:

Create a Huffman tree and Huffman encoding for "AARDVARK"



	count	encoding
D	1	1111
K	1	1110
V	1	110
R	2	10
A	3	0

```
0010111111001011 10
AAR D    V  AR K
```

How many bits for fixed length encoding?

How many bits for variable length encoding?

Compression ratio?

Suggested Additional Exercises

14.2.2

14.2.3

Huffman Practical Application

Python Example implementing Huffman Tree