Lossless Data Compression

"How does it work?"

First, a question

First, a question

Can you find "data compression" outside the computer?

Abbreviations

Association for Computing Machinery

->

ACM

Numbers

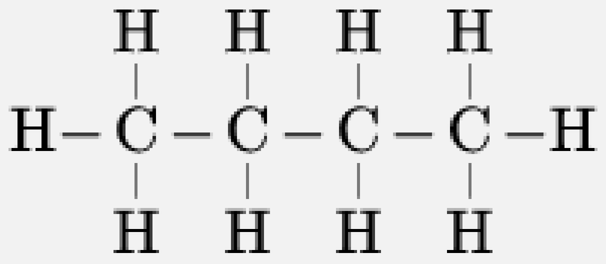
Dopey, Sneezy, Bashful, Doc, Happy, Grumpy, Sleepy *

->

The Seven Dwarfs

(* There are different versions)

Chemical Formulas

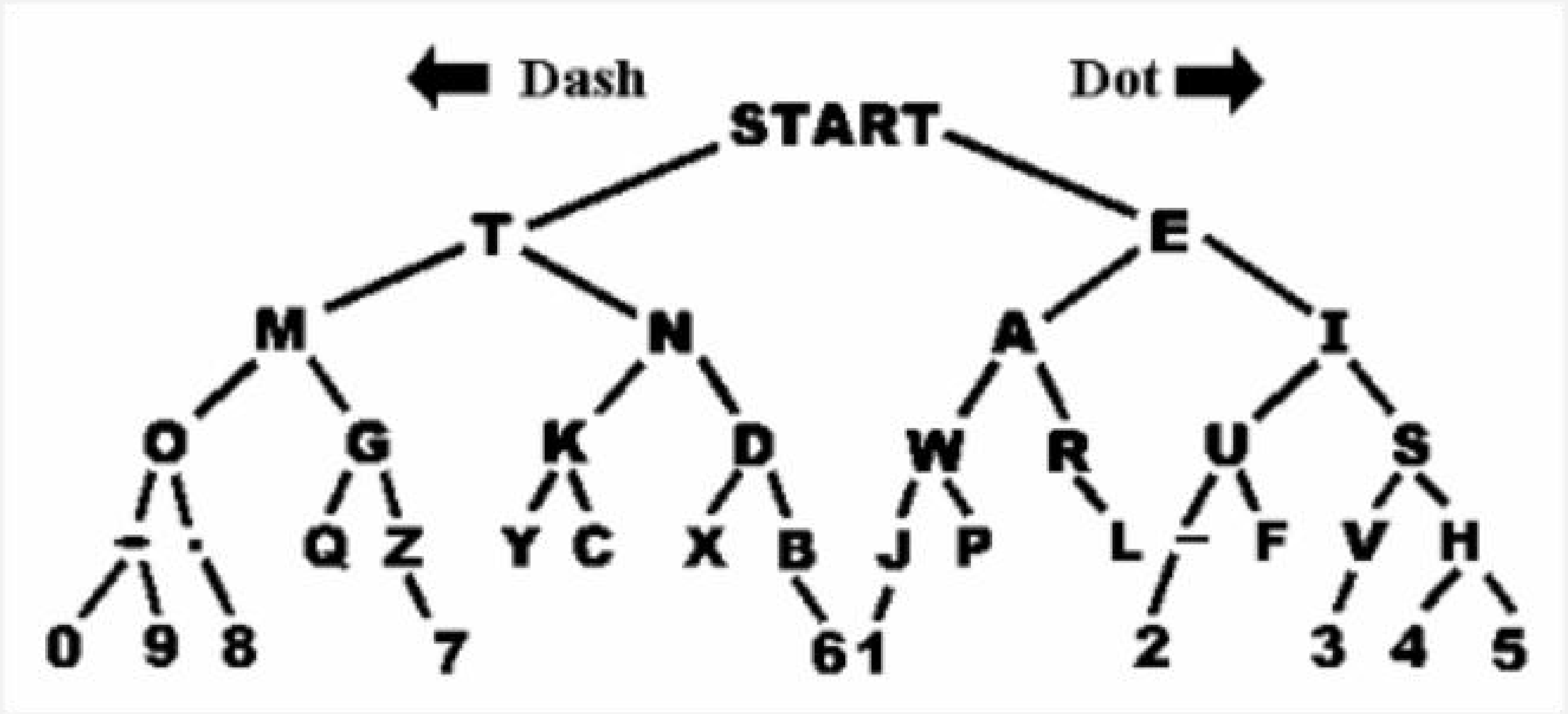


->

C_4H_{10}

and...

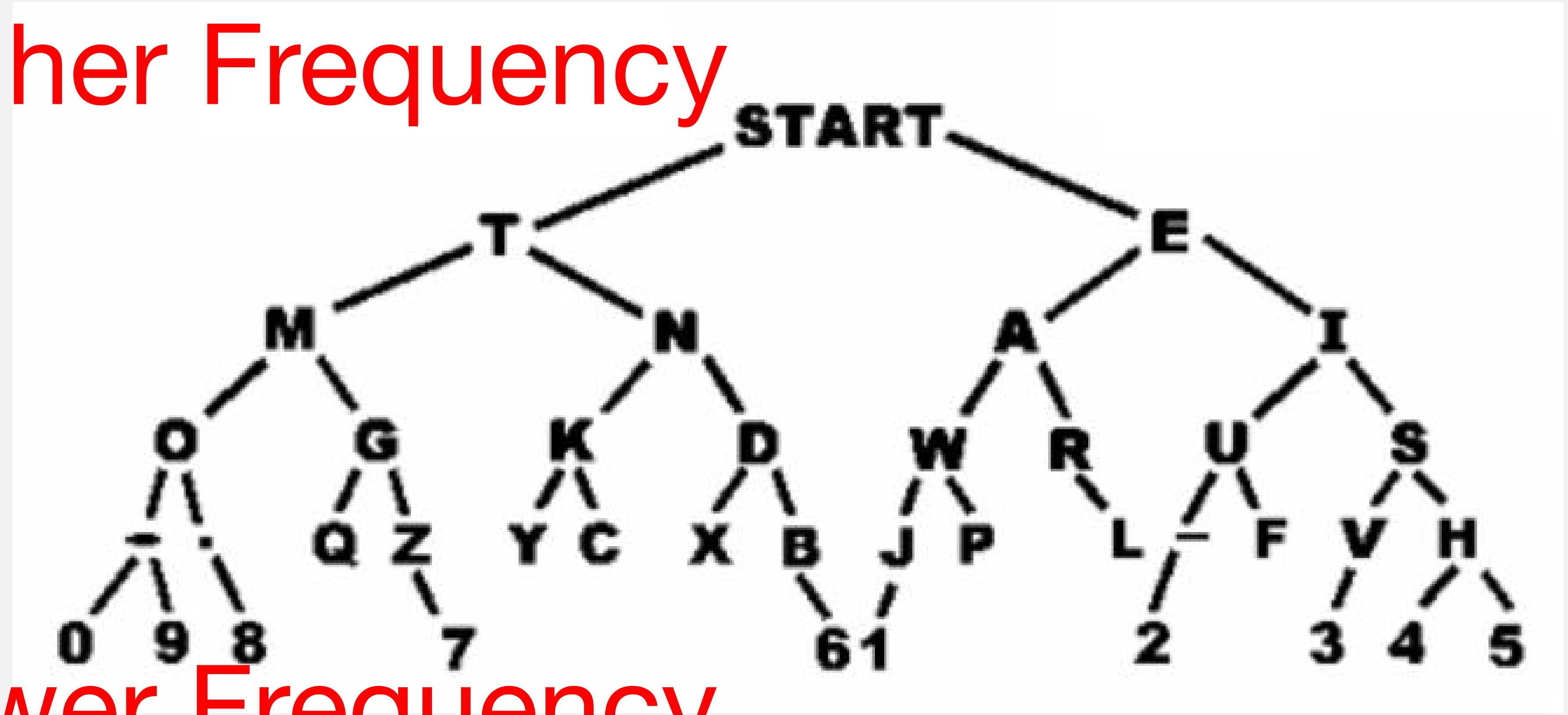
Morse Code



(Image source: UWashington)

Morse Code

Higher Frequency



Lower Frequency

Can we use Morse Code
to compress data on our computers?

Text

hello world

Morse Code

.... . .-.. .-.. --- ....-- --- .-. .-.. -..

Text

hello world

Binary

00000010 00100111 00000000
11111010 0100100

Text

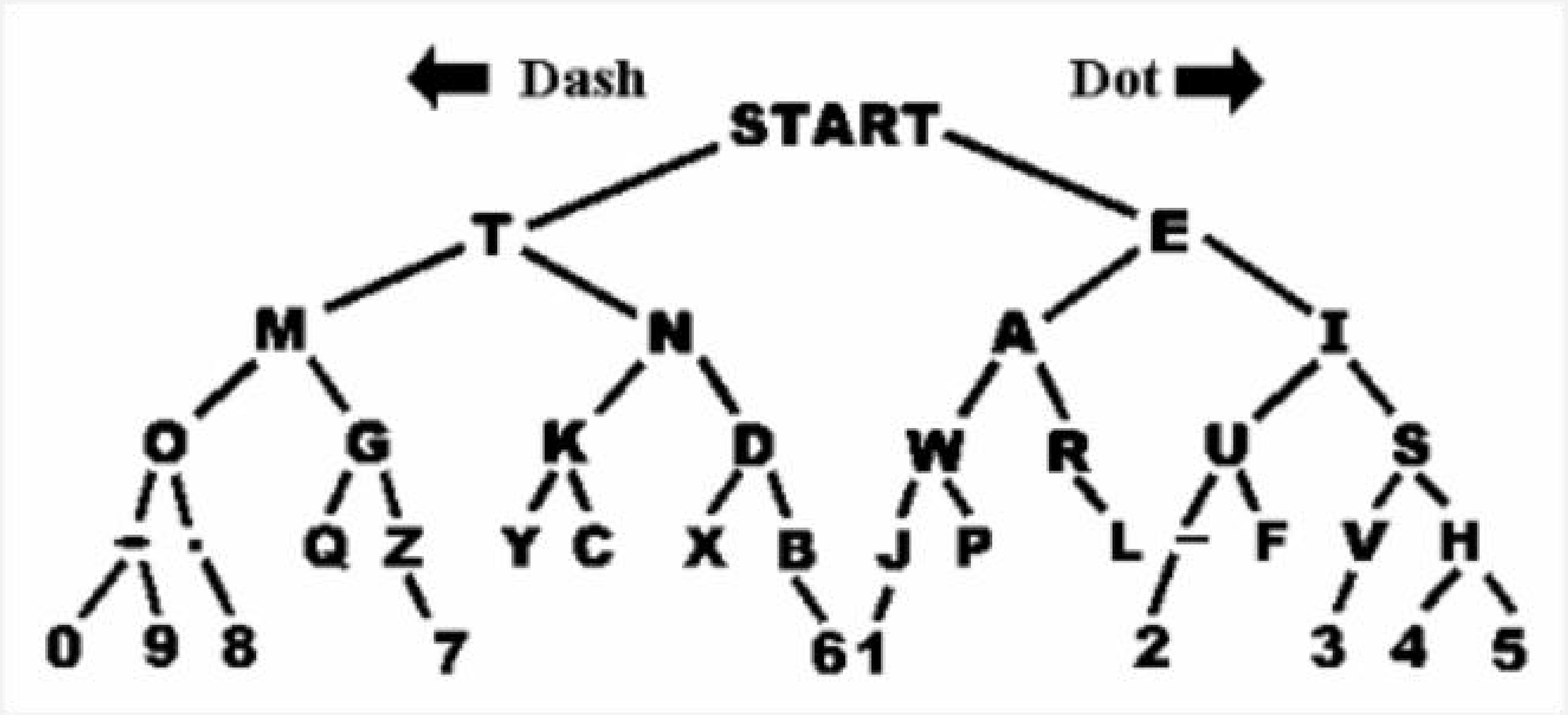
hello world

Binary

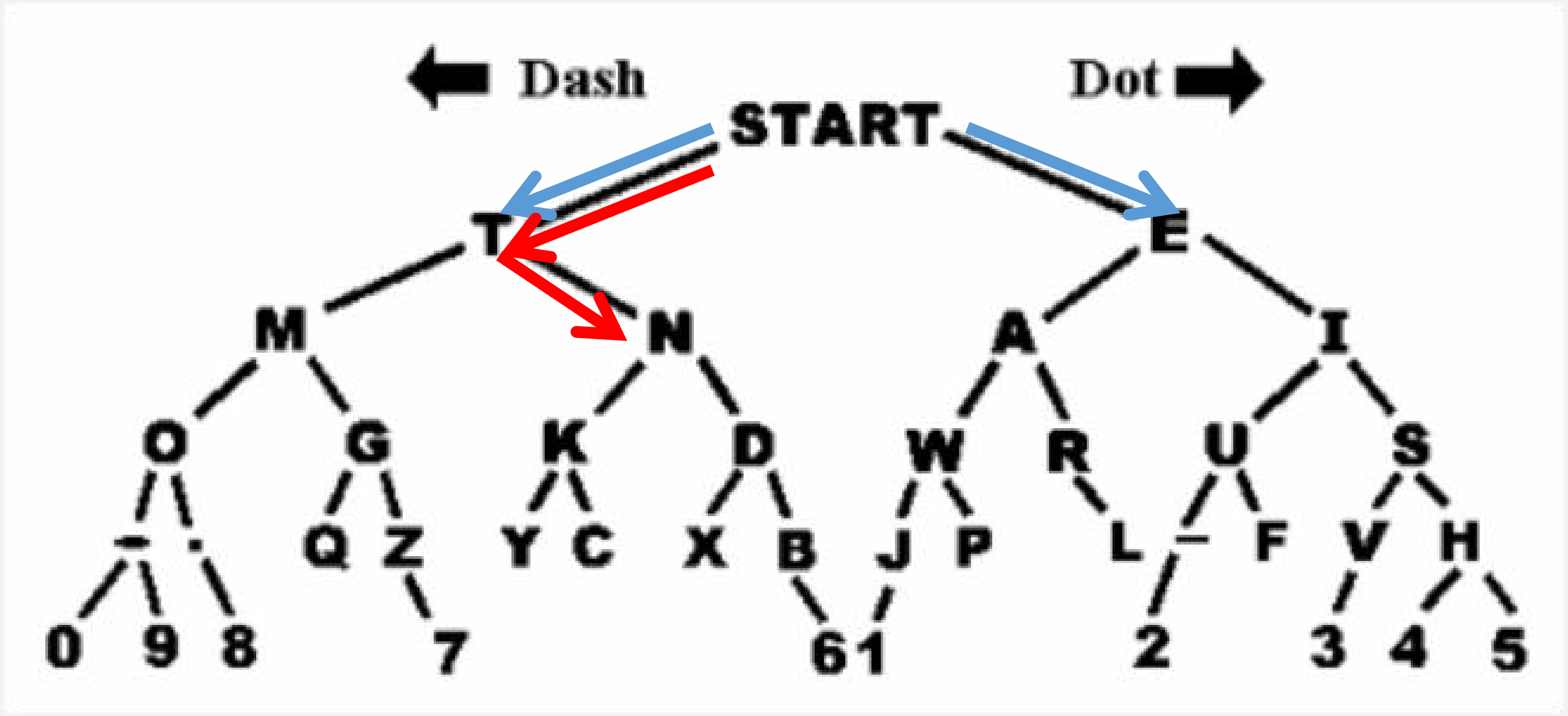
00000010 00100111 00000000
11111010 0100100

But, we do not have "separators" between letters

Look at the Tree Again

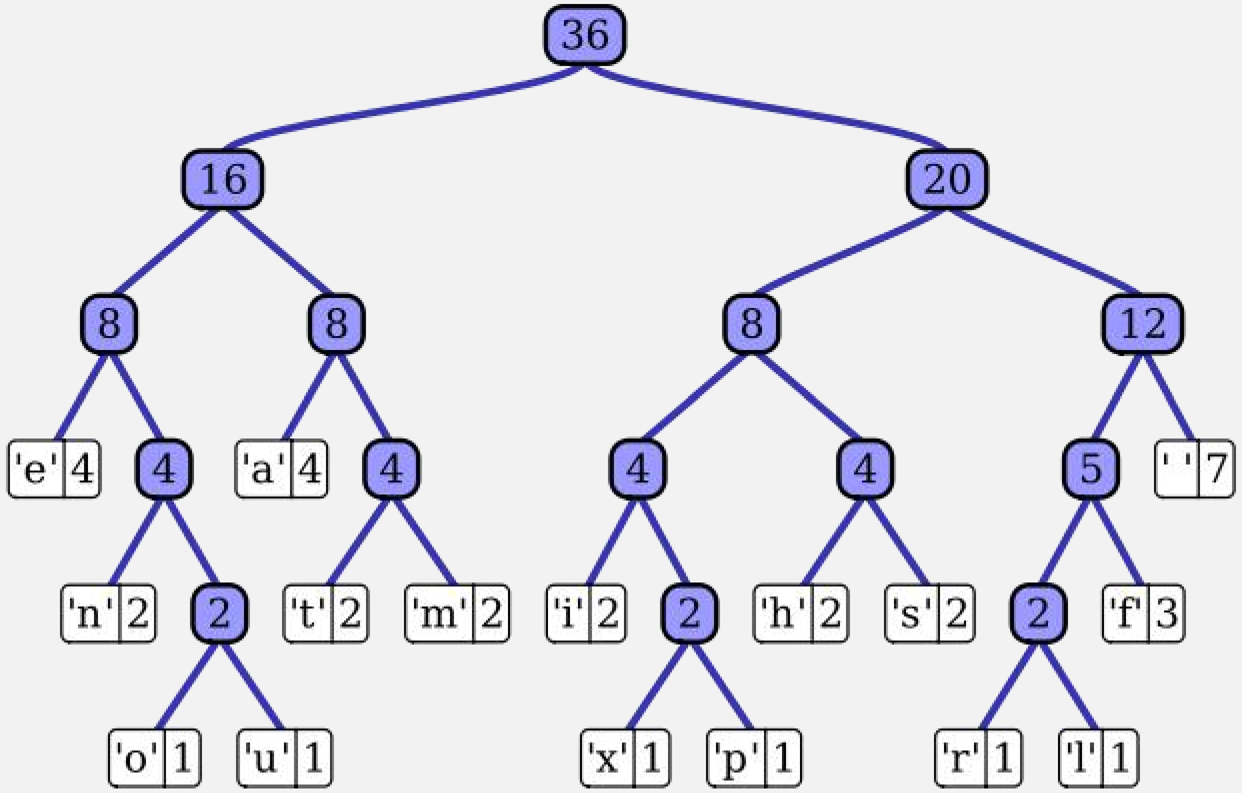


Look at the Tree Again



Ambiguity: N = TE

Huffman Tree



(Image source: Meteficha)

Huffman Tree

Put letters on leaf nodes only

Build the tree based on

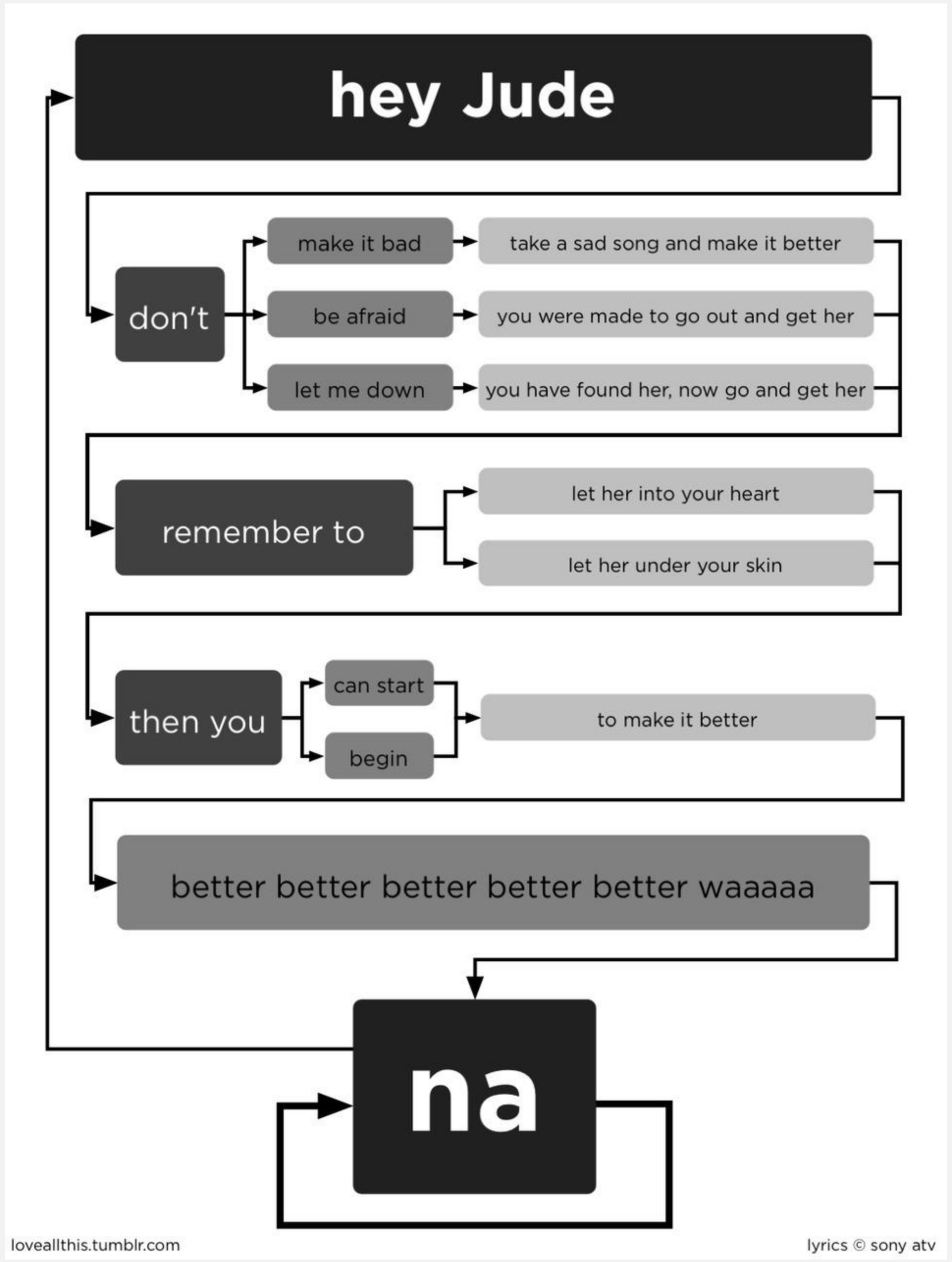
the occurrence rate of each letter

Minimize the sum of:

depth of node * occurrence of the letter

(= the size of compressed data)

Any other Approaches?



(Image source: Loveallthis)

Dictionary-Based Data Compression

- Find repeated patterns (strings)
- Replace them with references
- Lempel-Ziv Algorithms

Lempel-Ziv

hey Jude, don't make it bad
take a sad song and ...
then you can start to make it better
hey Jude, don't be afraid
you were made to ...
then you begin to make it better

Lempel-Ziv

hey Jude, don't make it bad
take a sad song and ...
then you can start to make it better
<1> be afraid
you were made to ...
<2> begin <3>

I have Huffman

I have Lempel-Ziv

DEFLATE

DEFLATE

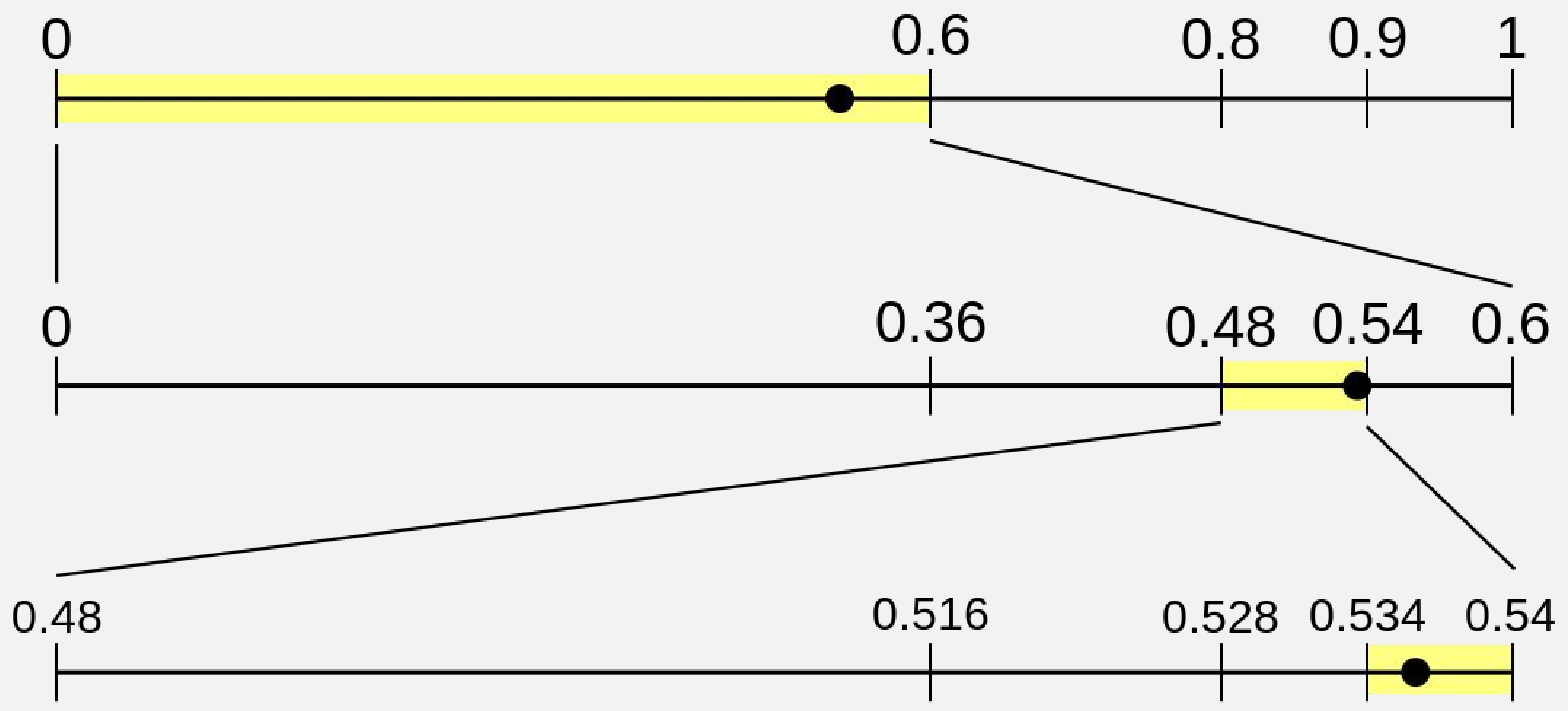
- Phil Katz, PKZip, 1989
- Use a Huffman tree to encode
- Dictionary references as tree leaves

Any other Approaches?

Probability-Based Data Compression

- Use a prediction model
- Arithmetic / range encoding
- Length ~ $\log_2(100\% / \text{probability})$

Arithmetic Encoding



(Image source: Dcoetzee)

Arithmetic Encoding

Final range:

0.534-0.540

Binary:

0.10001000-0.10001010

Encoding:

10001000

Can you imagine some possible prediction model?

Byte frequencies
(identical to Huffman)
Matching / partial matching
Neural network
...