

Linked Structures

Songs, Games, Movies

Part IV

Fall 2013

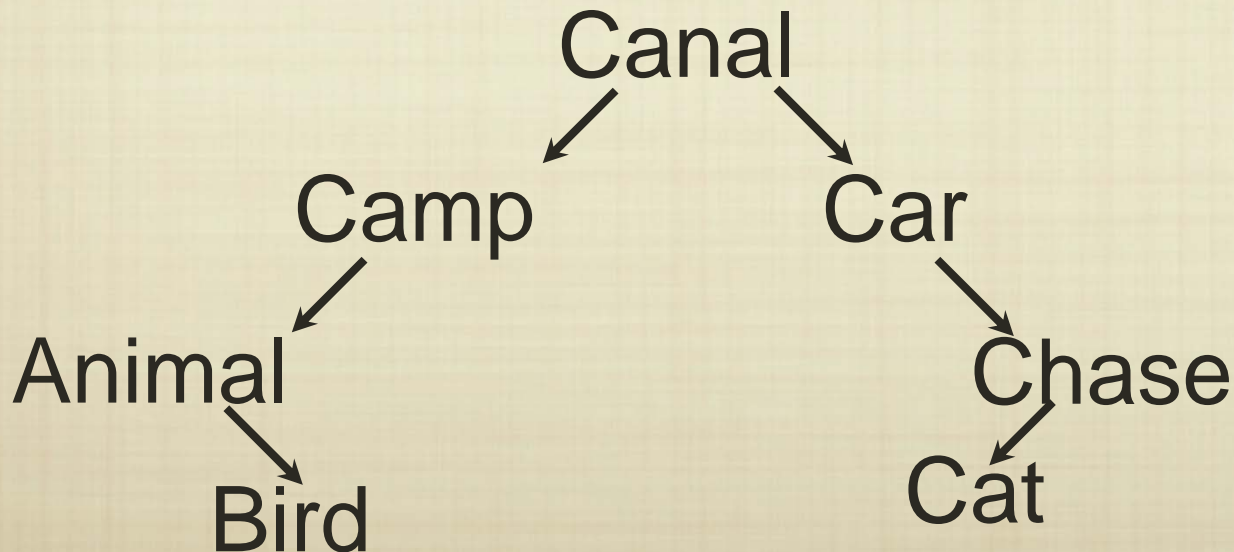
Carola Wenk

Storing Text

- We've been focusing on numbers. What about text?

“Animal”, “Bird”, “Cat”, “Car”, “Chase”, “Camp”,
“Canal”

We can compare the lexicographic ordering of strings, and then construct a binary search tree:

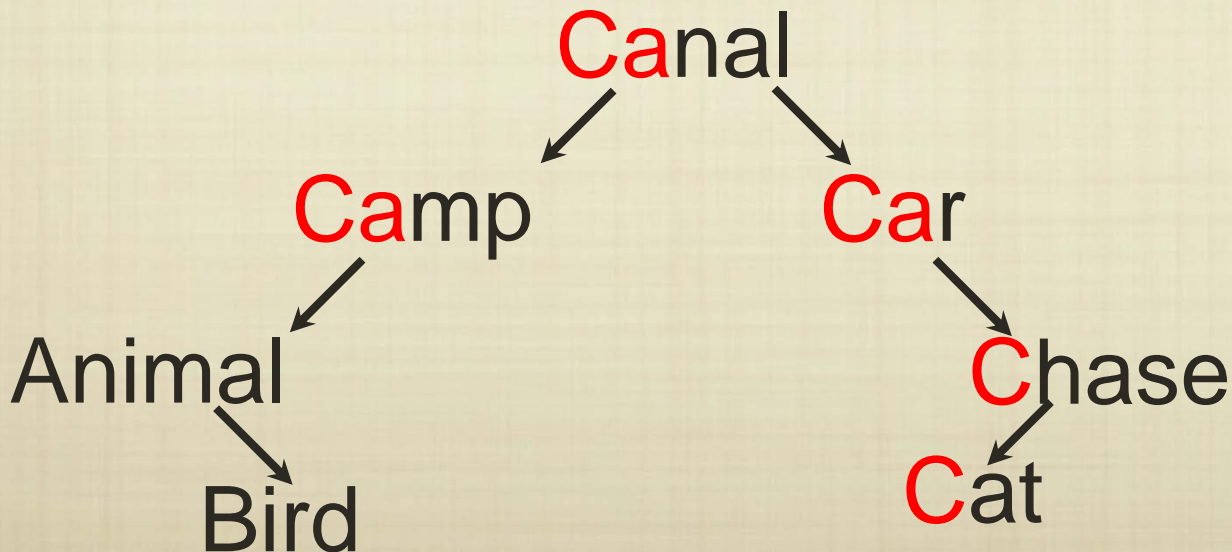


Storing Text

- We've been focusing on numbers. What about text?

“Animal”, “Bird”, “Cat”, “Car”, “Chase”, “Camp”,
“Canal”

In many cases, it would be beneficial to eliminate redundancy:

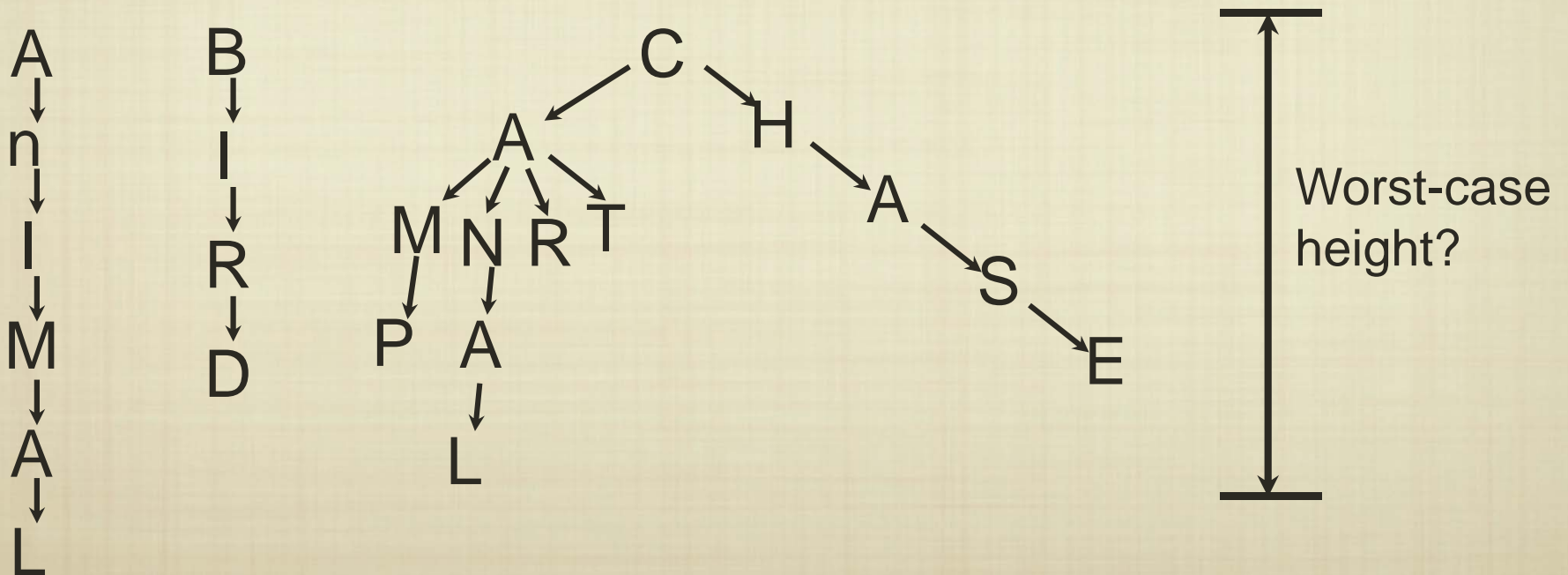


Storing Text

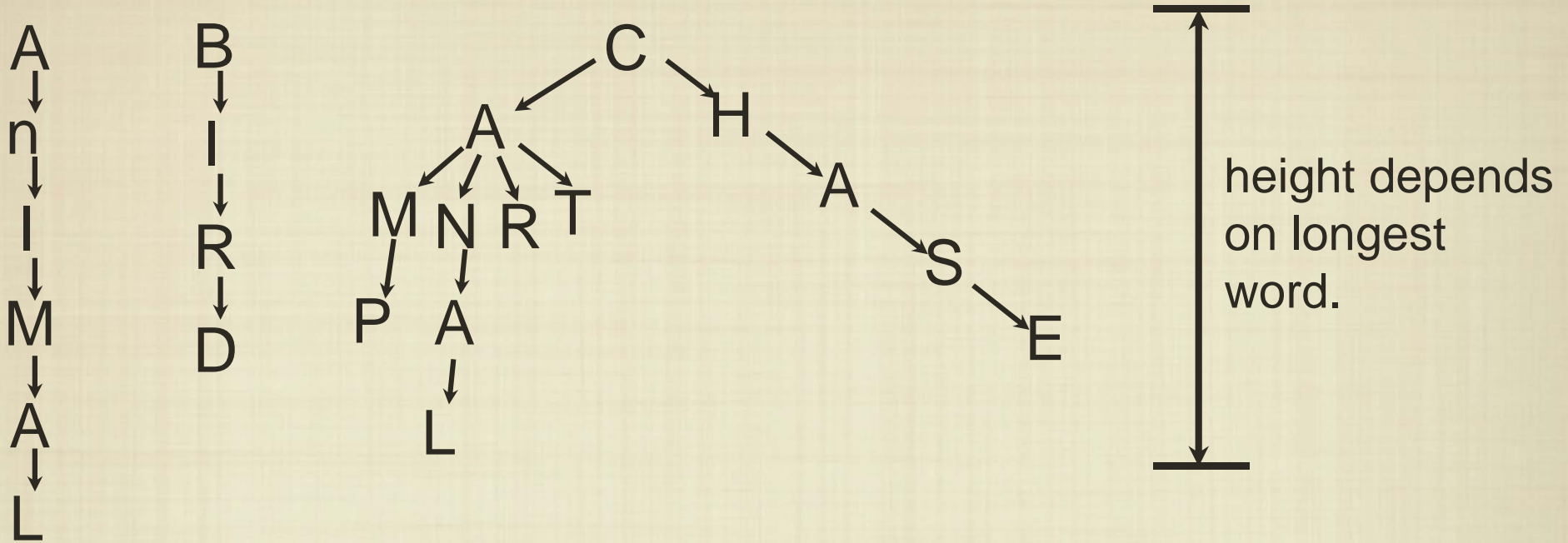
- We've been focusing on numbers. What about text?

“Animal”, “Bird”, “Cat”, “Car”, “Chase”, “Camp”, “Canal”

A prefix tree (or trie) has characters as nodes, and stores each string as a path in the tree.



Prefix Trees



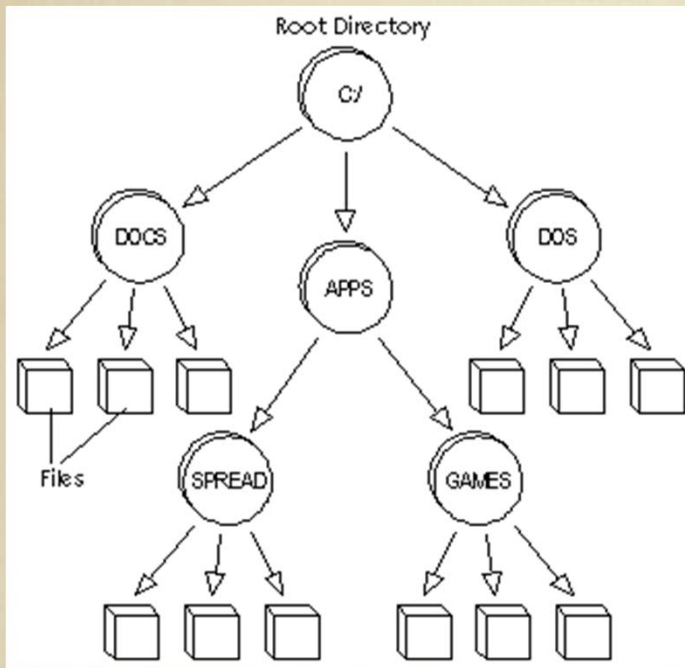
The advantage of a prefix tree is that finding any element requires height proportional to the associated string (the average English word is about 5 letters).

This representation allows much faster performance than the best-case scenario for a binary search tree (e.g. the Oxford English Dictionary has about 175K words).

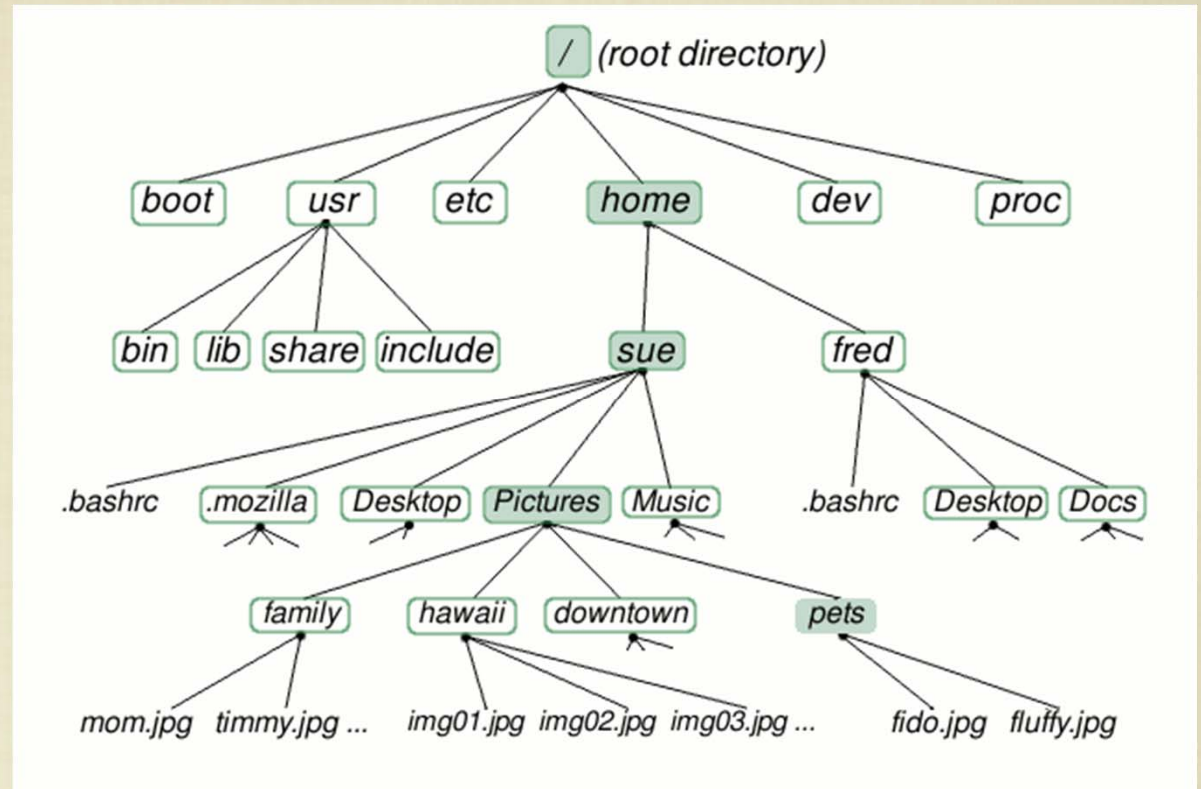
Linked Structures in Software

- Nearly every modern file system uses some type of hierarchical layout, as implemented by a tree structure.
- In the most general sense, structuring information as a tree uses particular attributes (e.g. values, spelling) to form subtrees.
- We can also think of our data structure as making decisions as we go traverse downward.
- Decision trees are a basic abstraction that are used for a large variety of tasks.

File Systems



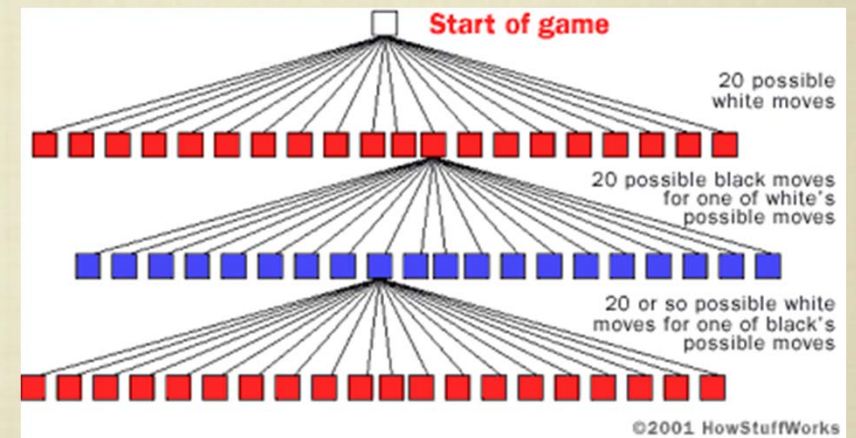
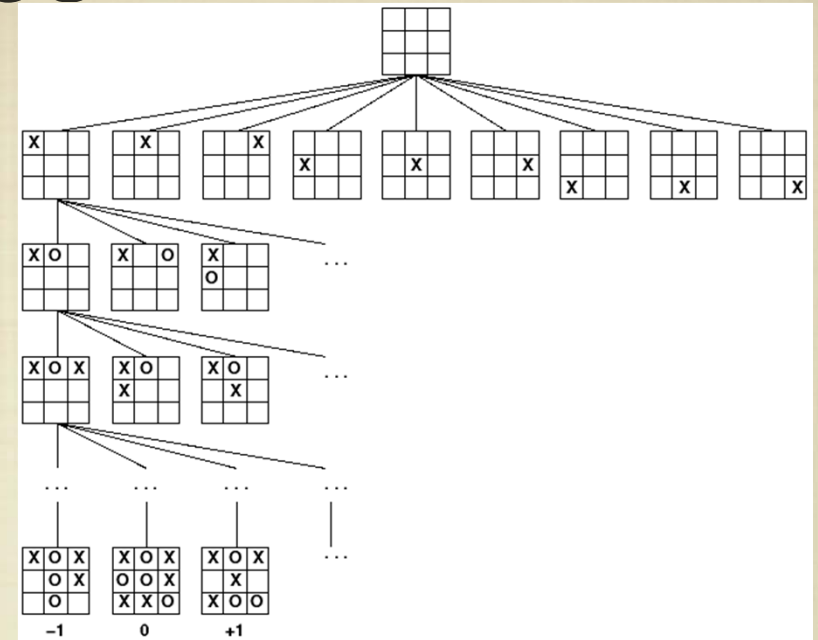
MS-DOS



Linux

Files in every operating system are organized in a tree structure. Moreover, files are laid out on a disk in a tree-structured manner for efficient access.

Game (Decision) Trees



In adventure and strategy games, player decisions are used to decide how the game will progress. This decision tree is used by the computer opponent to decide the most “advantageous” move.

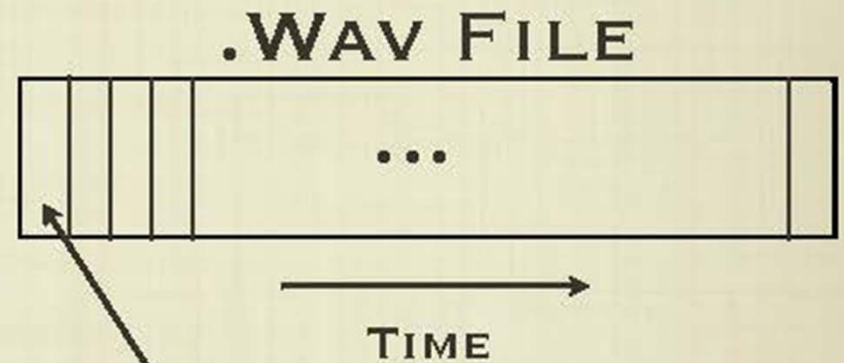
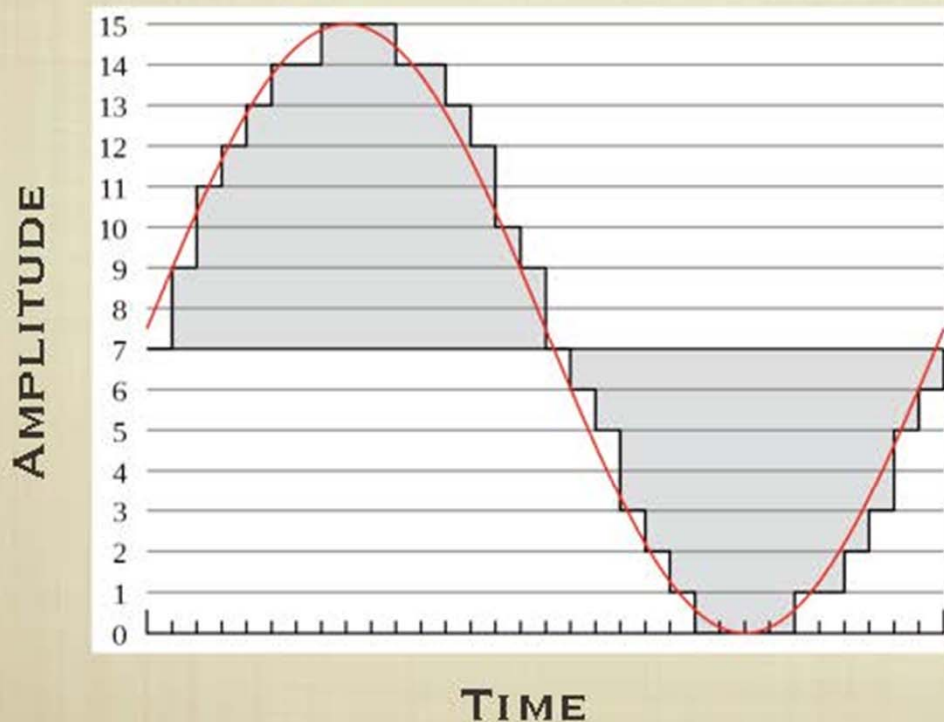
Recap: Linked/Hierarchical Structures

- What is the “standard” representation of lists in Python?
- What is the main advantage of array-based lists?
- What is the primary limitation of array-based lists?
- What is the “layout” of a linked structure? How do we construct and access a linked structure?
- In a linked structure with one neighbor relationship per item, how quickly can we add/remove items?
- How do we add, remove and find elements in a binary search tree?
- What is the high-level organization of any tree structure?

Data Compression

How are sounds, images and movies represented in a computer?

Sounds and images are continuous signals that can be “digitized”.

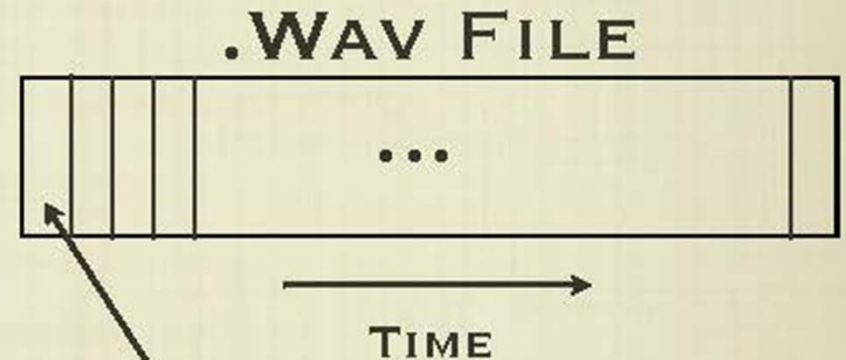
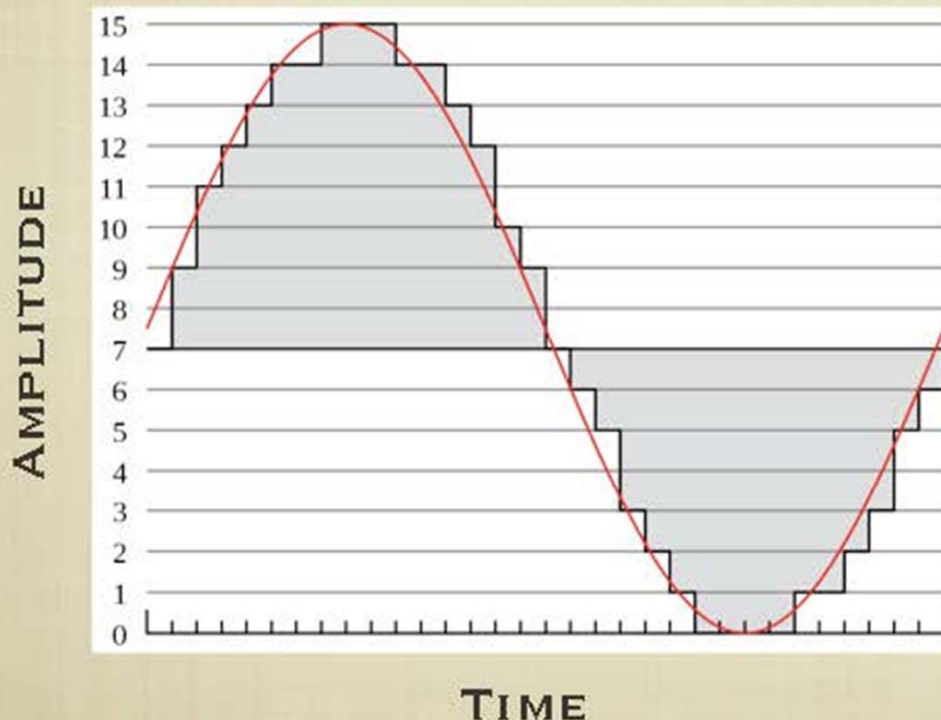


“Samples” (numbers) that capture the amplitude of the signal at each time point.

Data Compression

We can store the amplitude (as a number) of a sound signal at chosen time intervals; this is the sampling rate. The higher the rate, the more “accurate” the sound, and more space we need to store the signal.

A WAV file requires about 100MB per minute of audio - can we do better?

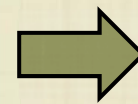
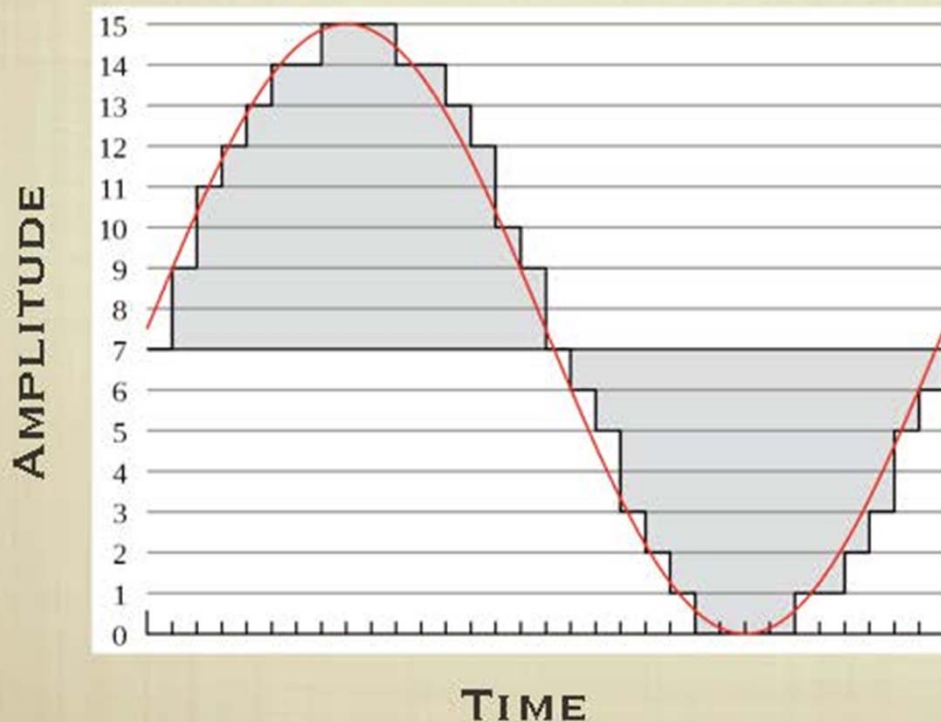


“Samples” (numbers) that capture the amplitude of the signal at each time point.

Data Compression

We can store the amplitude (as a number) of a sound signal at chosen time intervals; this is the sampling rate. The higher the rate, the more “accurate” the sound, and more space we need to store the signal.

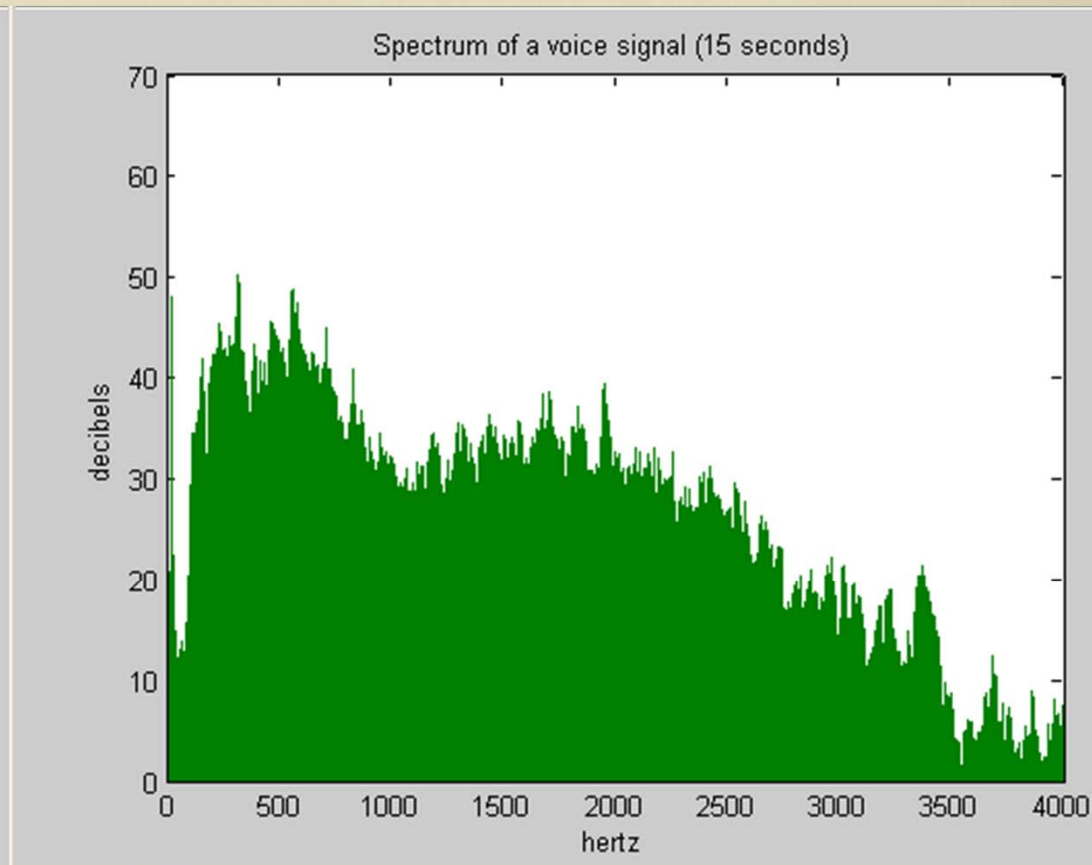
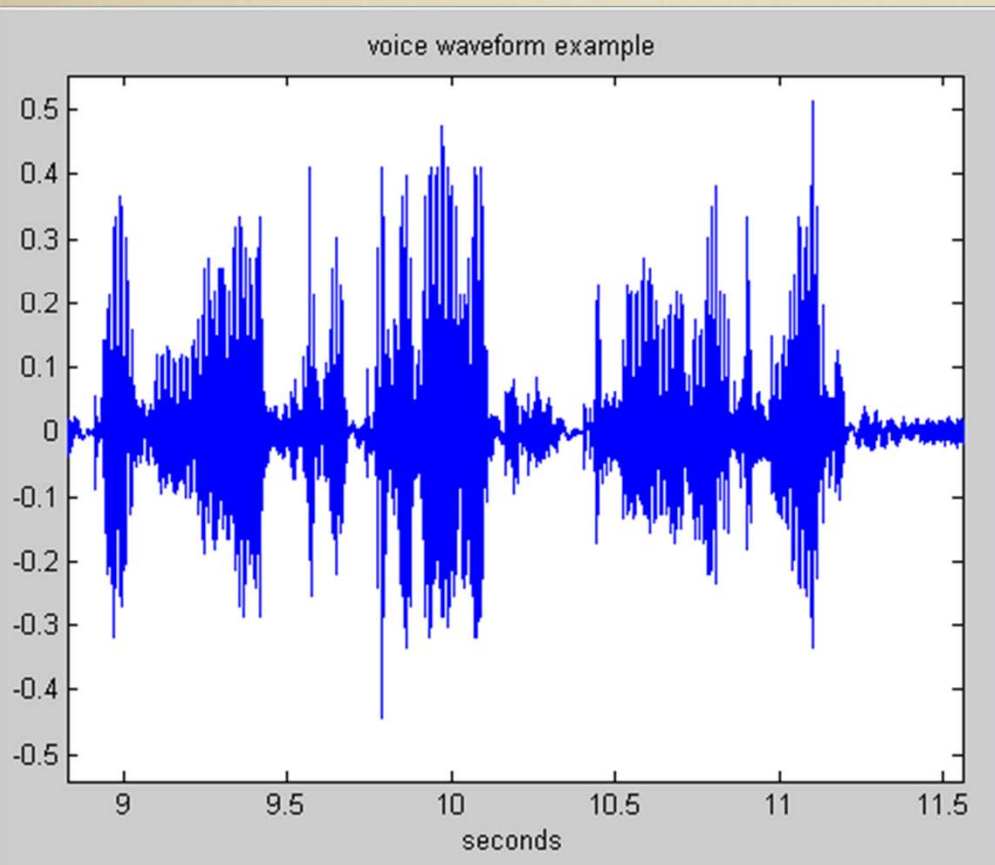
A WAV file requires about 100MB per minute of audio - can we do better?



MP3

“Moving Pictures Expert Group
Audio Layer III”

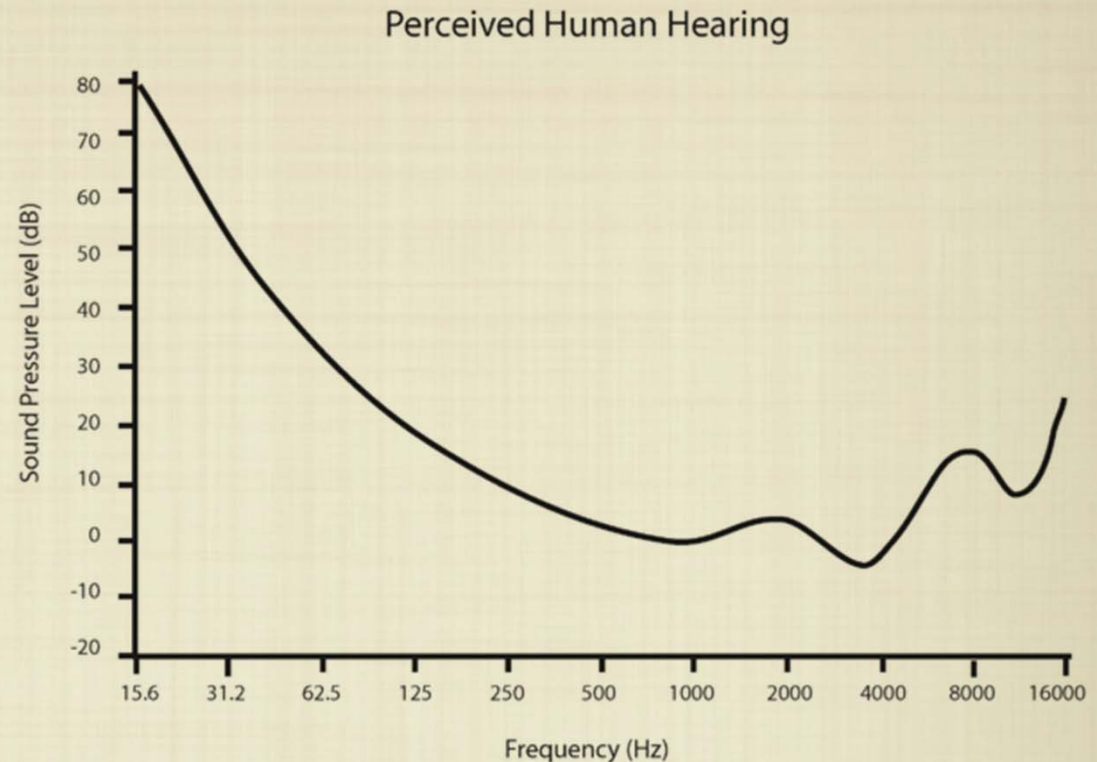
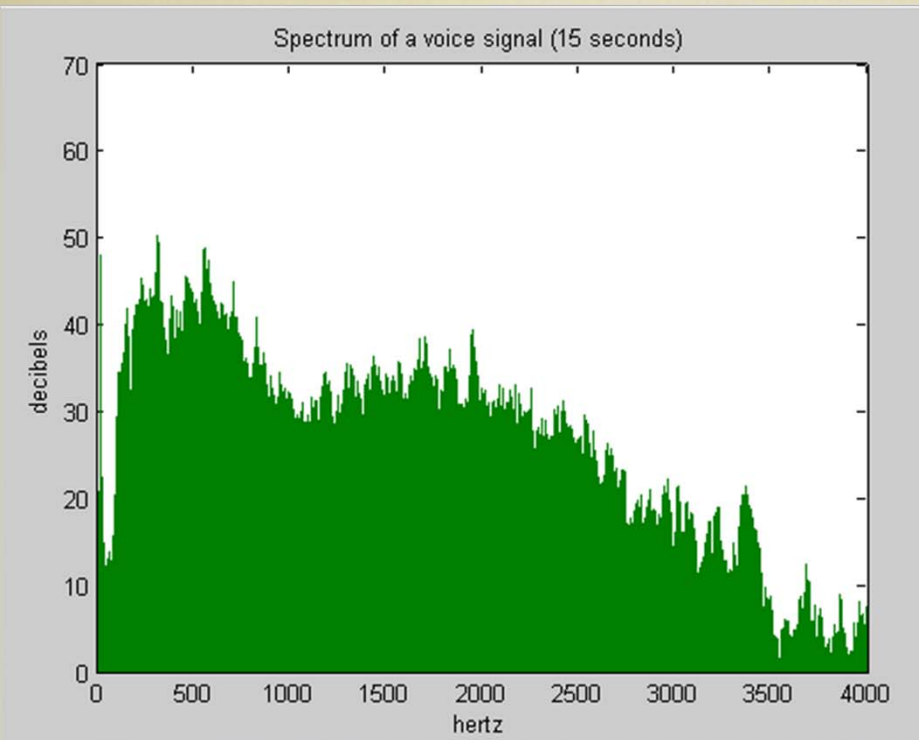
Time and Frequency Domains



We can also represent a sound wave as a collection of frequencies and the intensity with which they appear.

A decibel is a logarithmic quantity, so one intensity may need more bits than another.

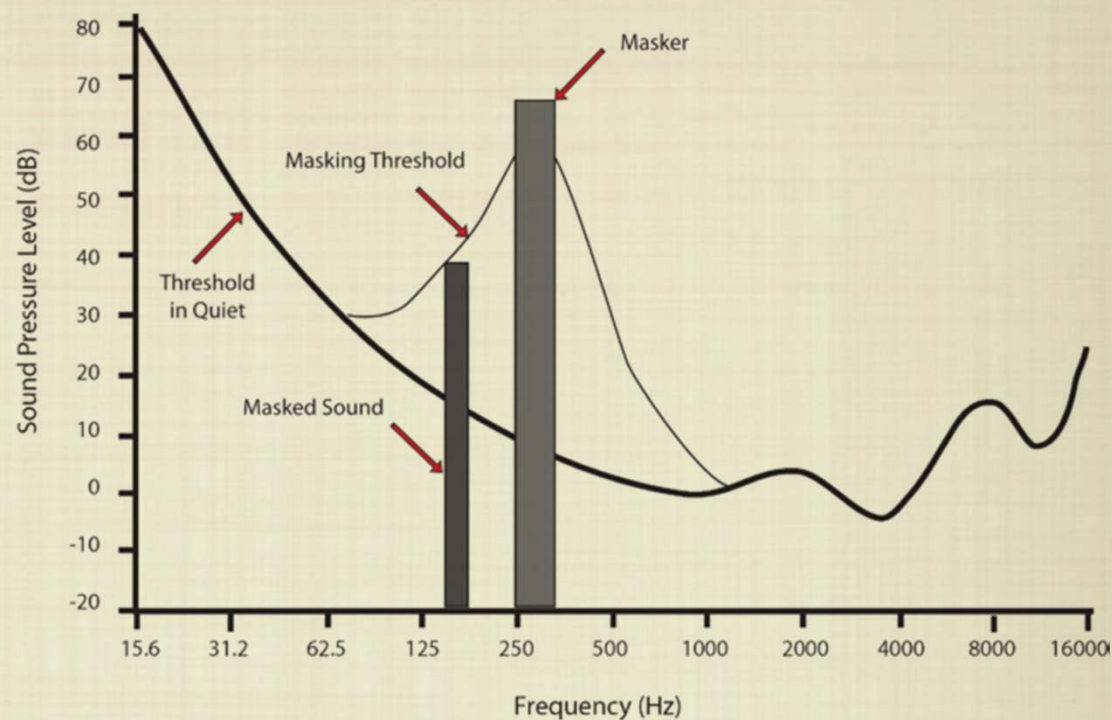
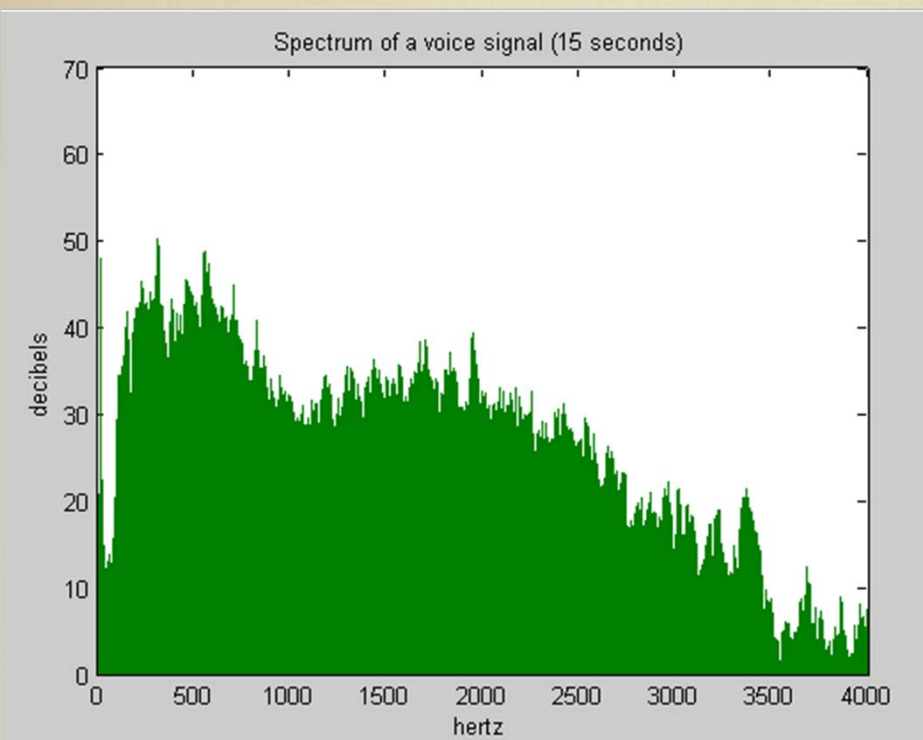
Psychoacoustic Filtering



The MP3 encoding algorithm consists of two high-level steps:

1. Apply psychoacoustic filters to remove information not “perceivable” by the human ear/brain.
2. Take the remaining signal and compress it to eliminate redundancy.

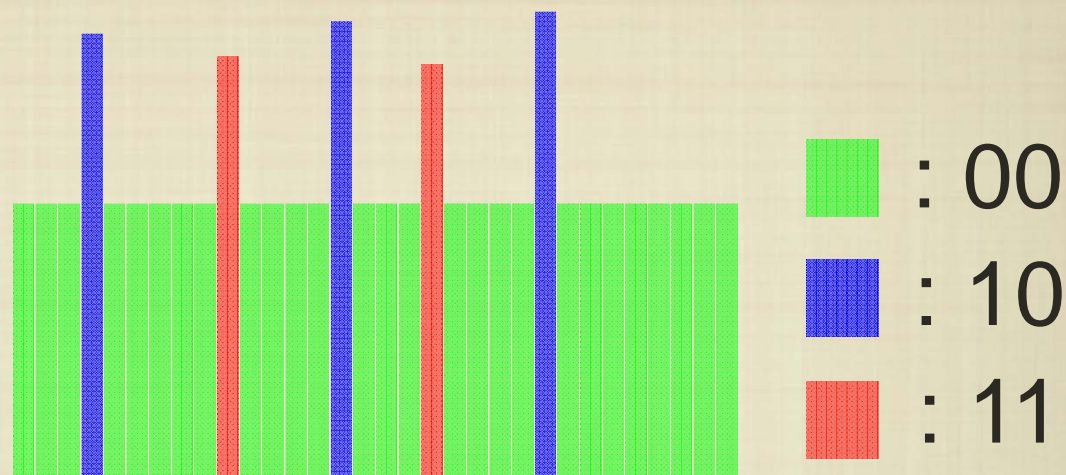
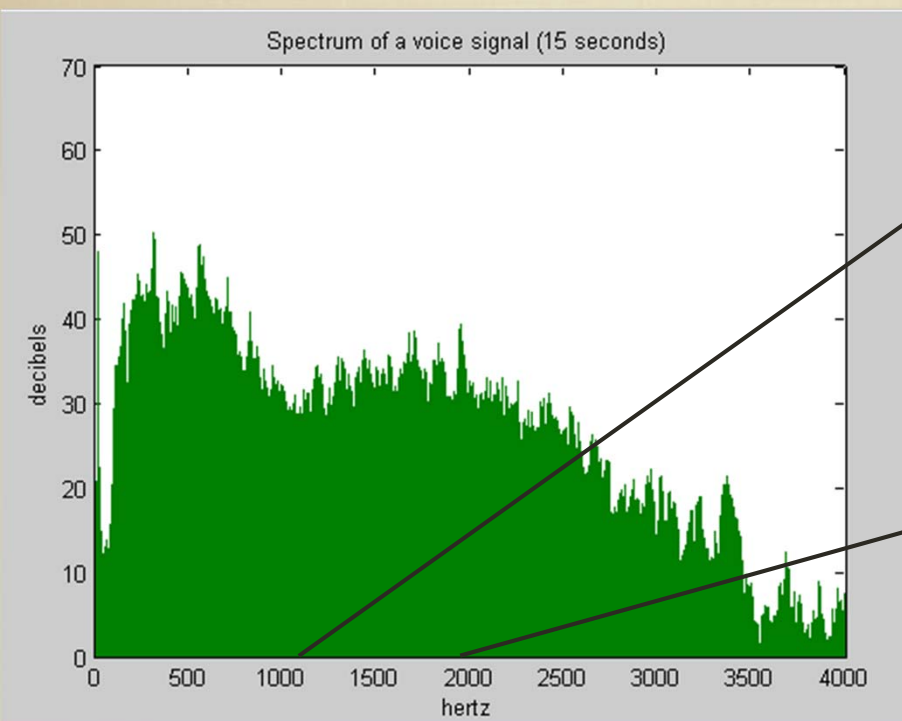
Psychoacoustic Filtering



The MP3 encoding algorithm consists of two steps:

1. Apply psychoacoustic filters to remove information not “perceivable” by the human ear/brain.
2. Take the remaining signal and compress it to eliminate redundancy.

Eliminating Redundancy

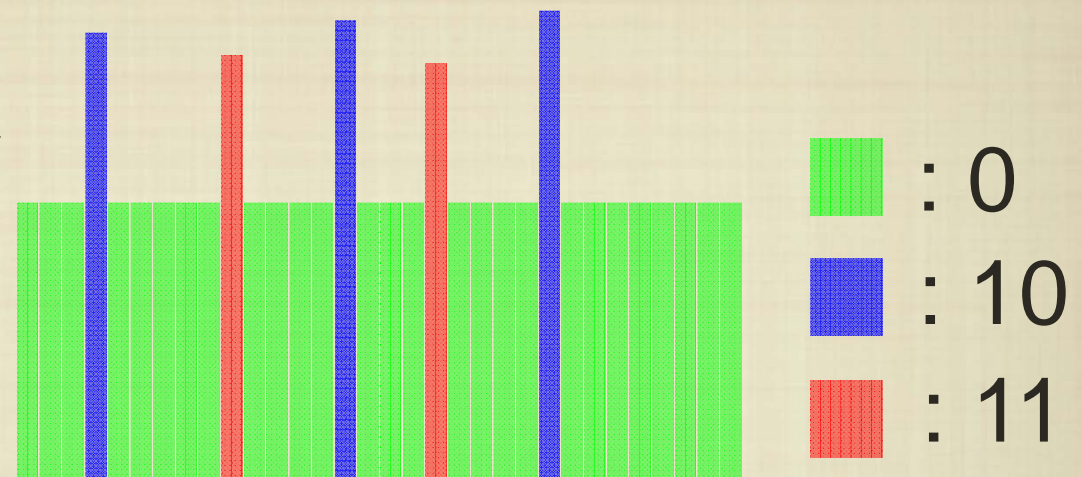
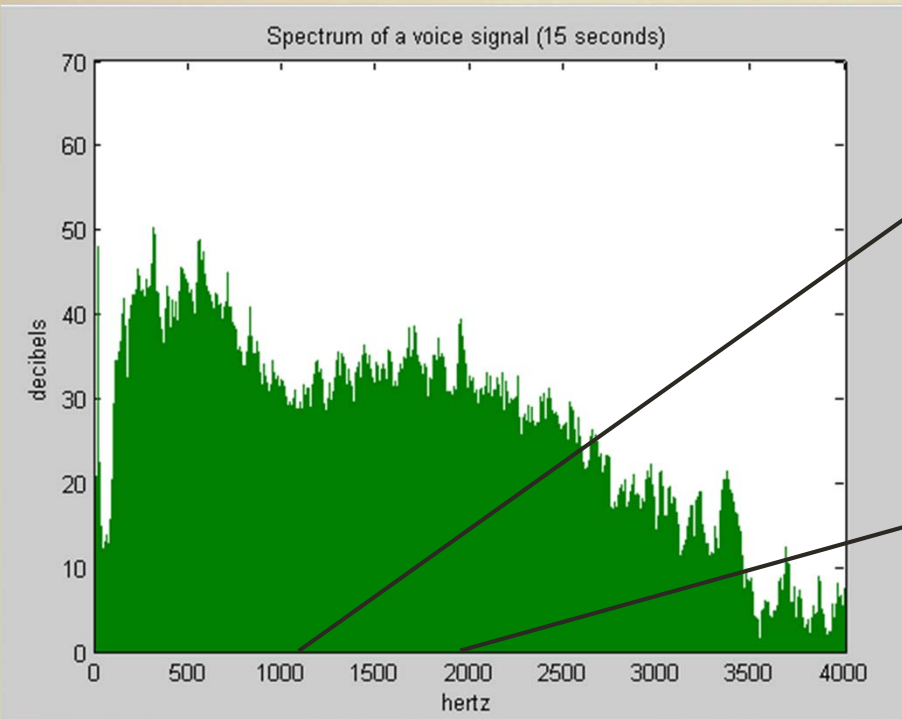


Once we have eliminated sounds that a human is unlikely to be able to hear, can we further compress the signal?

What if we have the same (or nearly the same) intensities at a large number of frequencies?

We can construct a “code” which takes advantage of this redundancy.

Eliminating Redundancy



Once we have eliminated sounds that a human is unlikely to be able to hear, can we further compress the signal?

What if we have the same (or nearly the same) intensities at a large number of frequencies?

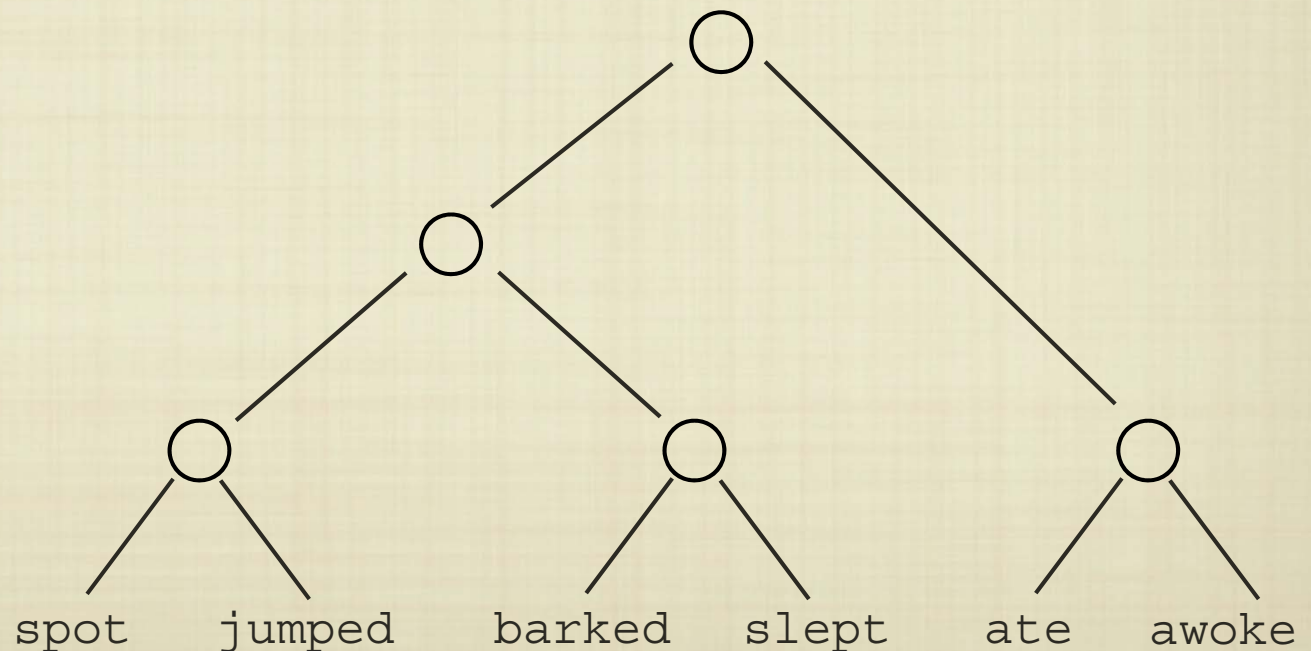
We can construct a “code” which takes advantage of this redundancy.

Encoding Symbols with Trees

- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- A binary tree can serve as a means to encode any set of symbols:

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```

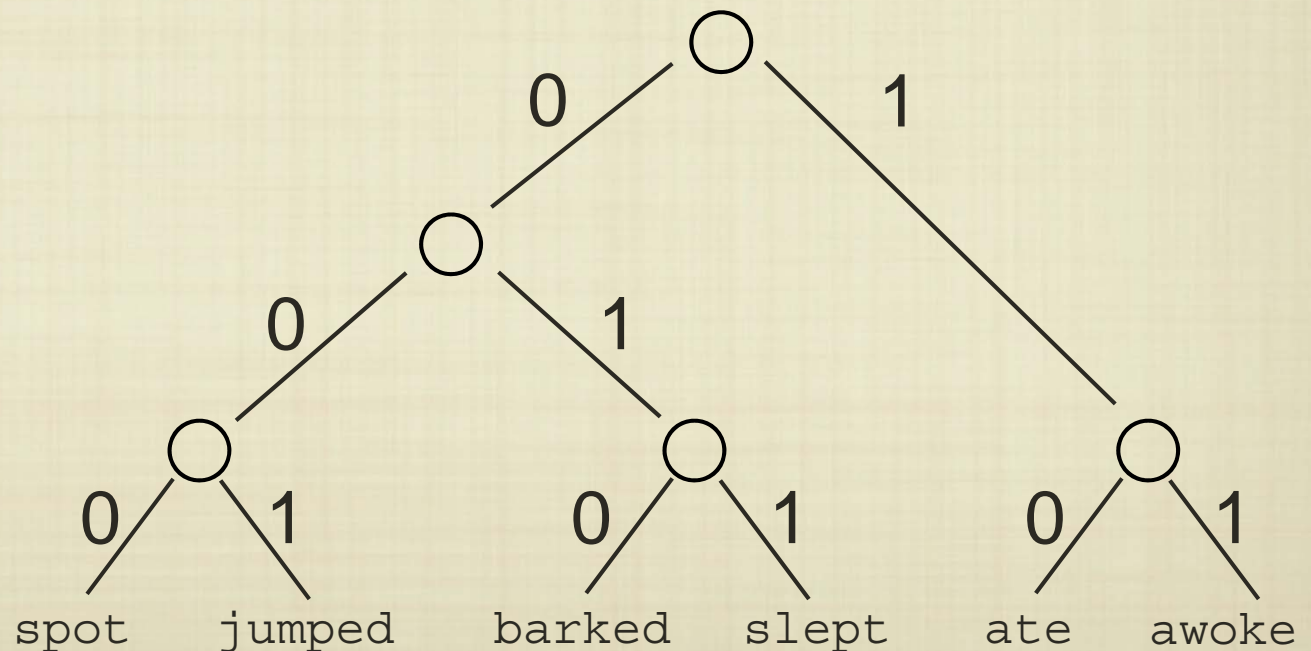


Encoding Symbols with Trees

- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- A binary tree can serve as a means to encode any set of symbols:

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```



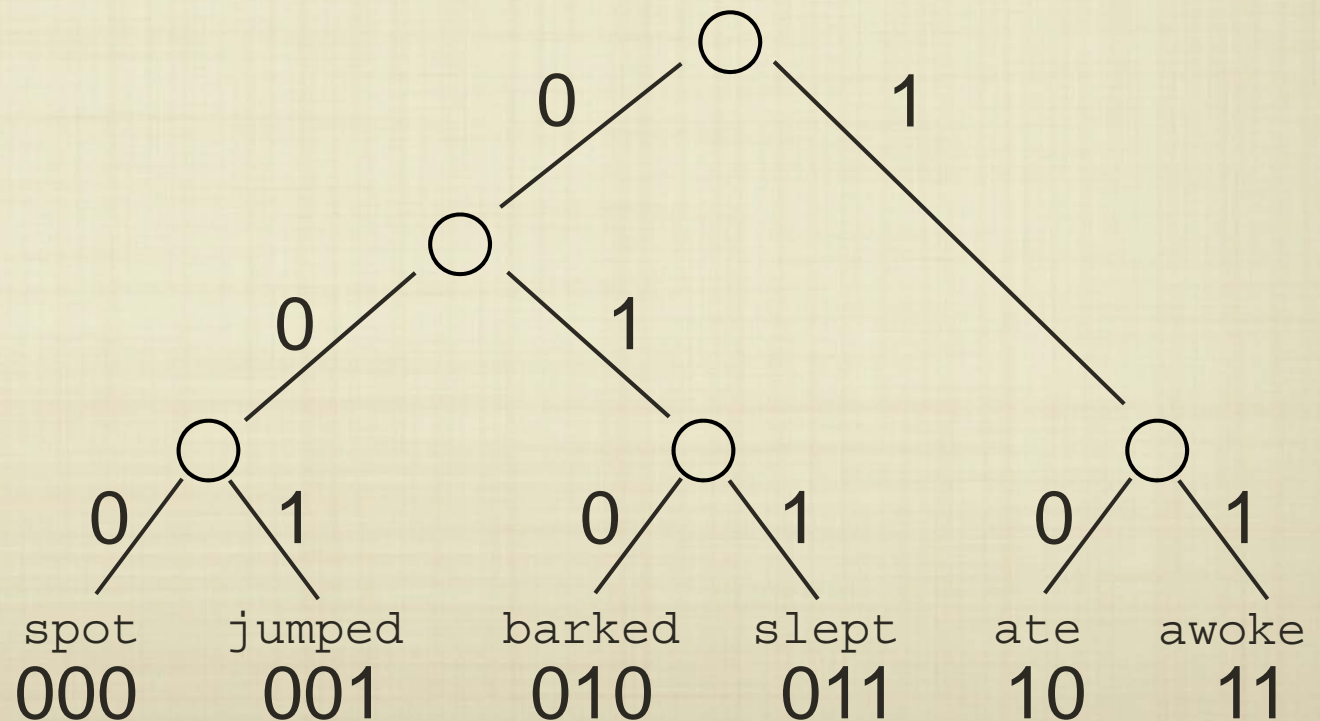
Encoding Symbols with Trees

- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- A binary tree can serve as a means to encode any set of symbols:

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```

Space Used:
 $5 \cdot 3 + 3 + 3 + 3 + 2 + 2 =$
28 bits



Encoding Symbols with Trees

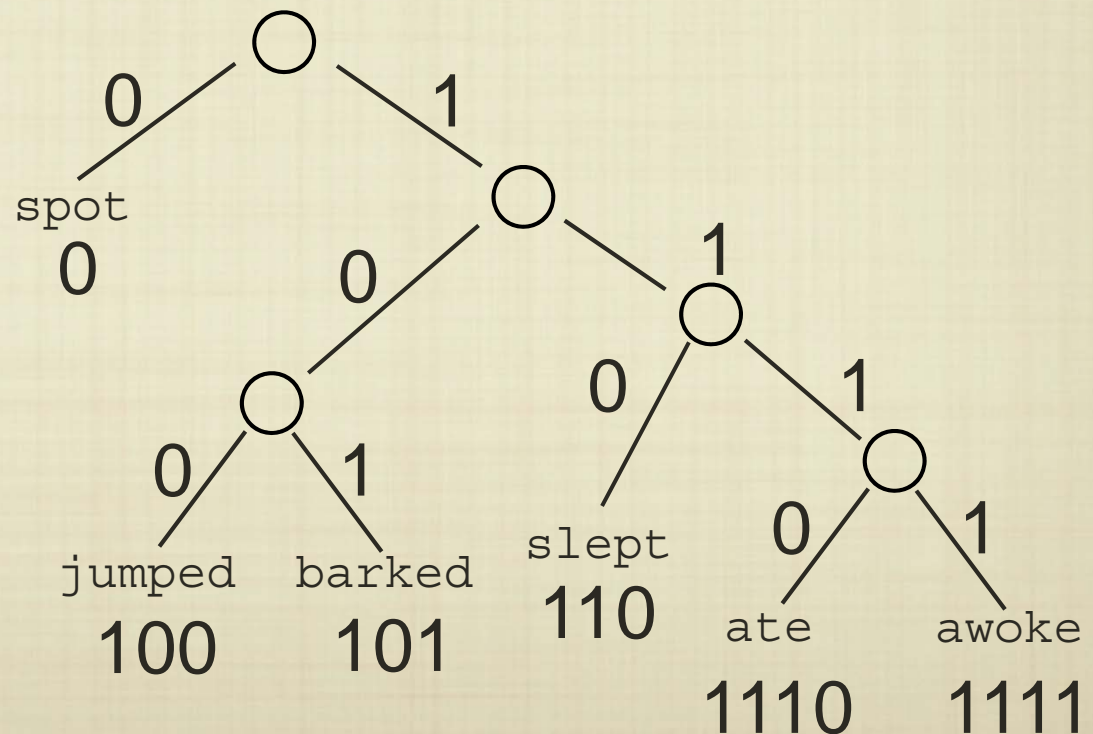
- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- We can construct any binary tree we want - the goal is to minimize the total space used to encode the source symbols.

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```

Space Used:

$$5 * 1 + 3 + 3 + 3 + 4 + 4 = 22 \text{ bits}$$



Encoding Symbols with Trees

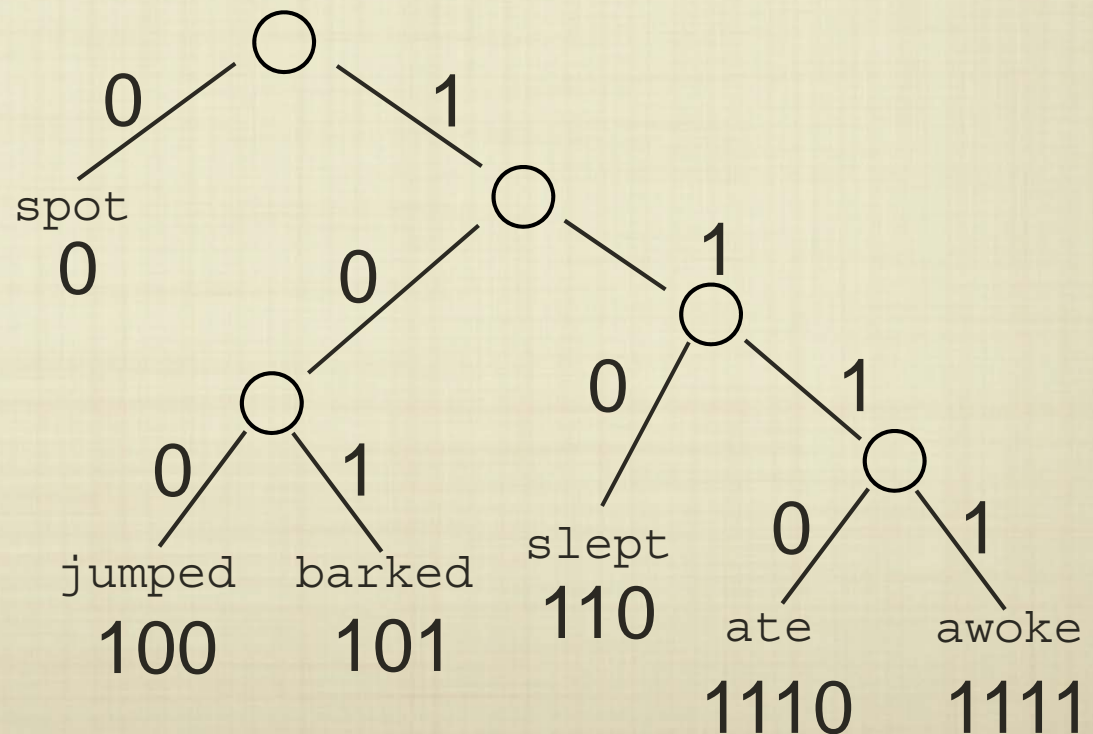
- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- Can we use the frequencies of symbols? Intuitively, we can save space by using shorter encodings for frequent symbols.

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```

Space Used:

$$5 \cdot 1 + 3 + 3 + 3 + 4 + 4 = 22 \text{ bits}$$



Encoding Symbols with Trees

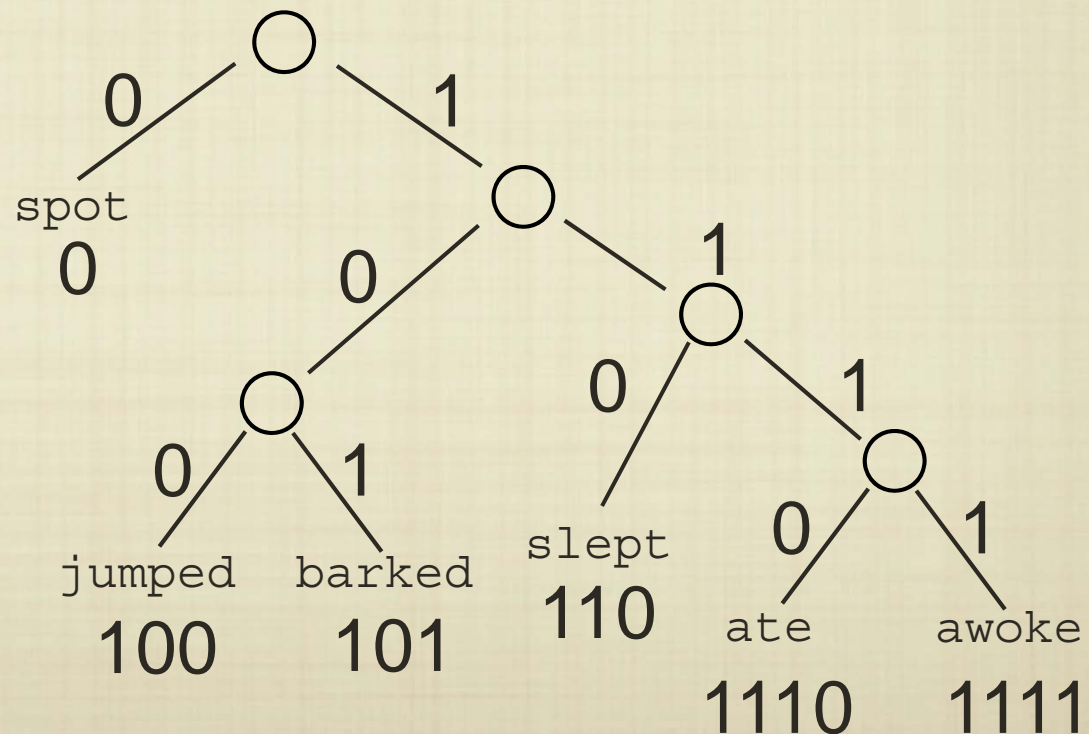
- Given a set of symbols and the frequency with which they appear, how can we encode the symbols using as few bits as possible?
- How do we find the “optimal” encoding? Is this always possible to do quickly?

Text File

```
spot jumped,  
spot barked,  
spot ate,  
spot slept,  
spot awoke
```

Space Used:

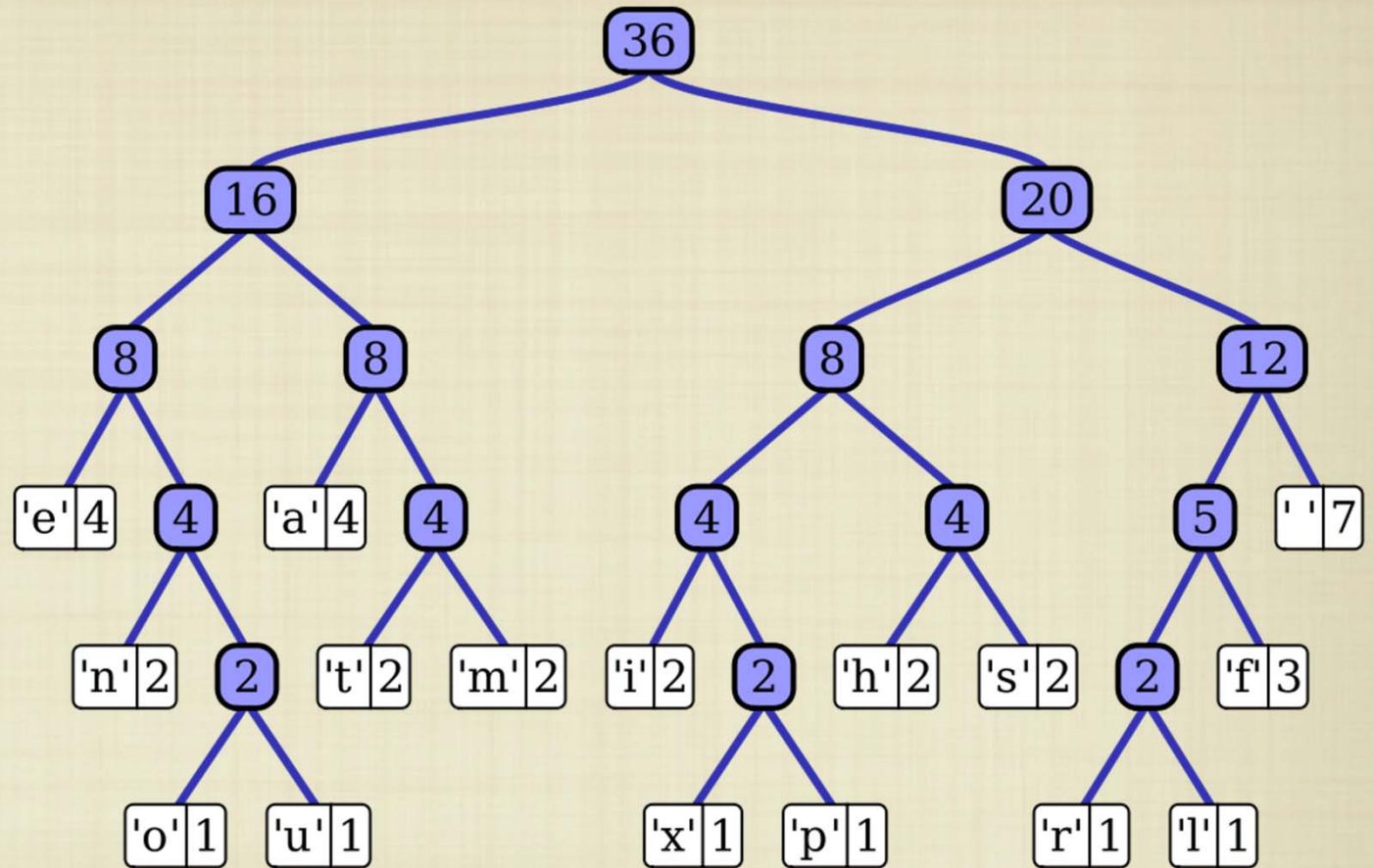
$$5 \cdot 1 + 3 + 3 + 3 + 4 + 4 = 22 \text{ bits}$$



Huffman Coding

Symbols/Frequencies:

'o': 1
'u': 1
'x': 1
'p': 1
'r': 1
'l': 1
'n': 2
't': 2
'm': 2
'i': 2
'h': 2
's': 2
'f': 3
'e': 4
'a': 4
' ': 7



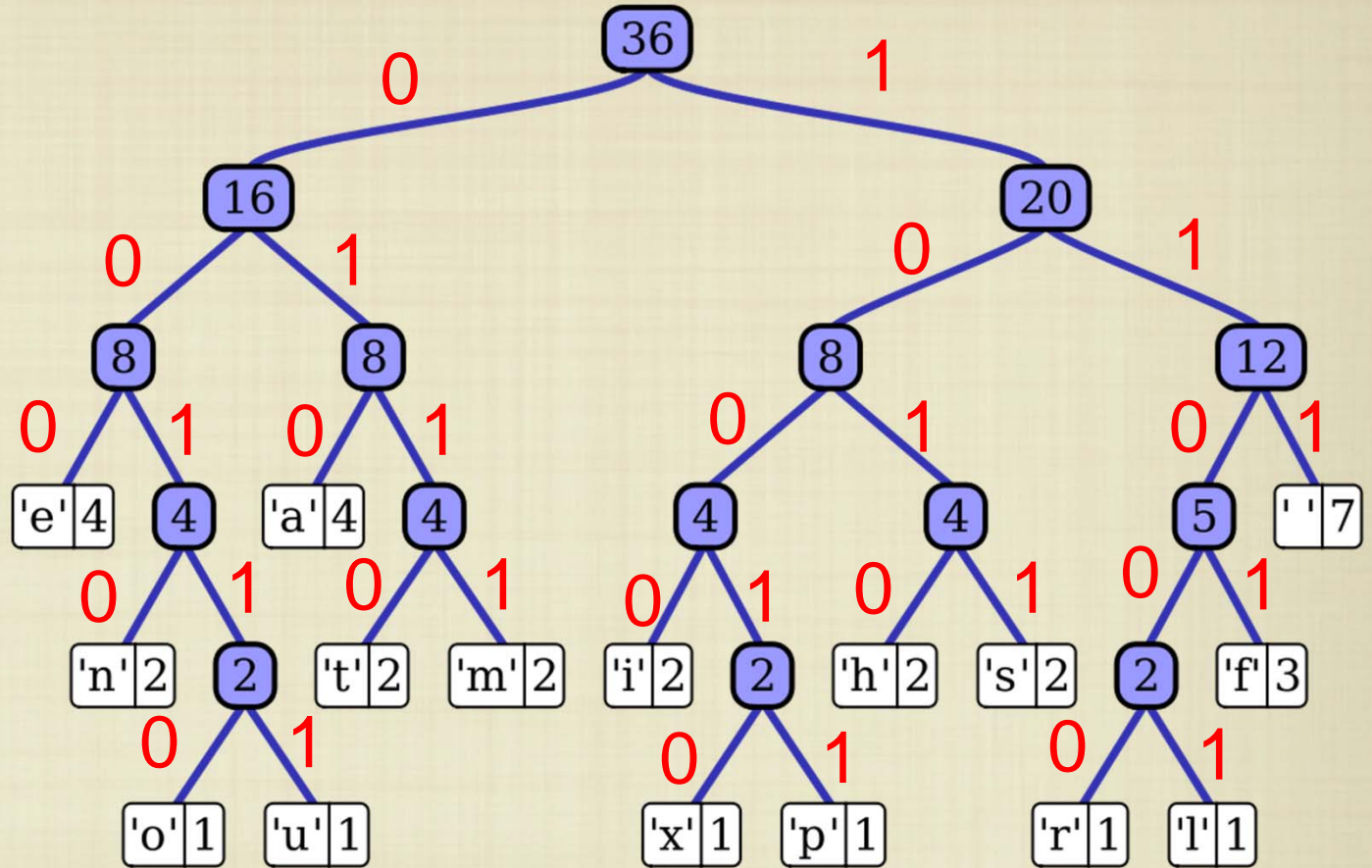
Algorithm

1. Take the two least frequent symbols, make them two 'sibling' leaves.
2. Replace these two symbols with a 'pseudo-symbol' whose frequency is the sum of the two smallest frequencies.
3. Repeat until only a single symbol remains.

Huffman Coding

Symbols/Frequencies:

00110	'o': 1
00111	'u': 1
10010	'x': 1
10011	'p': 1
11000	'r': 1
11001	'l': 1
0010	'n': 2
0110	't': 2
0111	'm': 2
1000	'i': 2
1010	'h': 2
1011	's': 2
1101	'f': 3
000	'e': 4
010	'a': 4
111	' ': 7

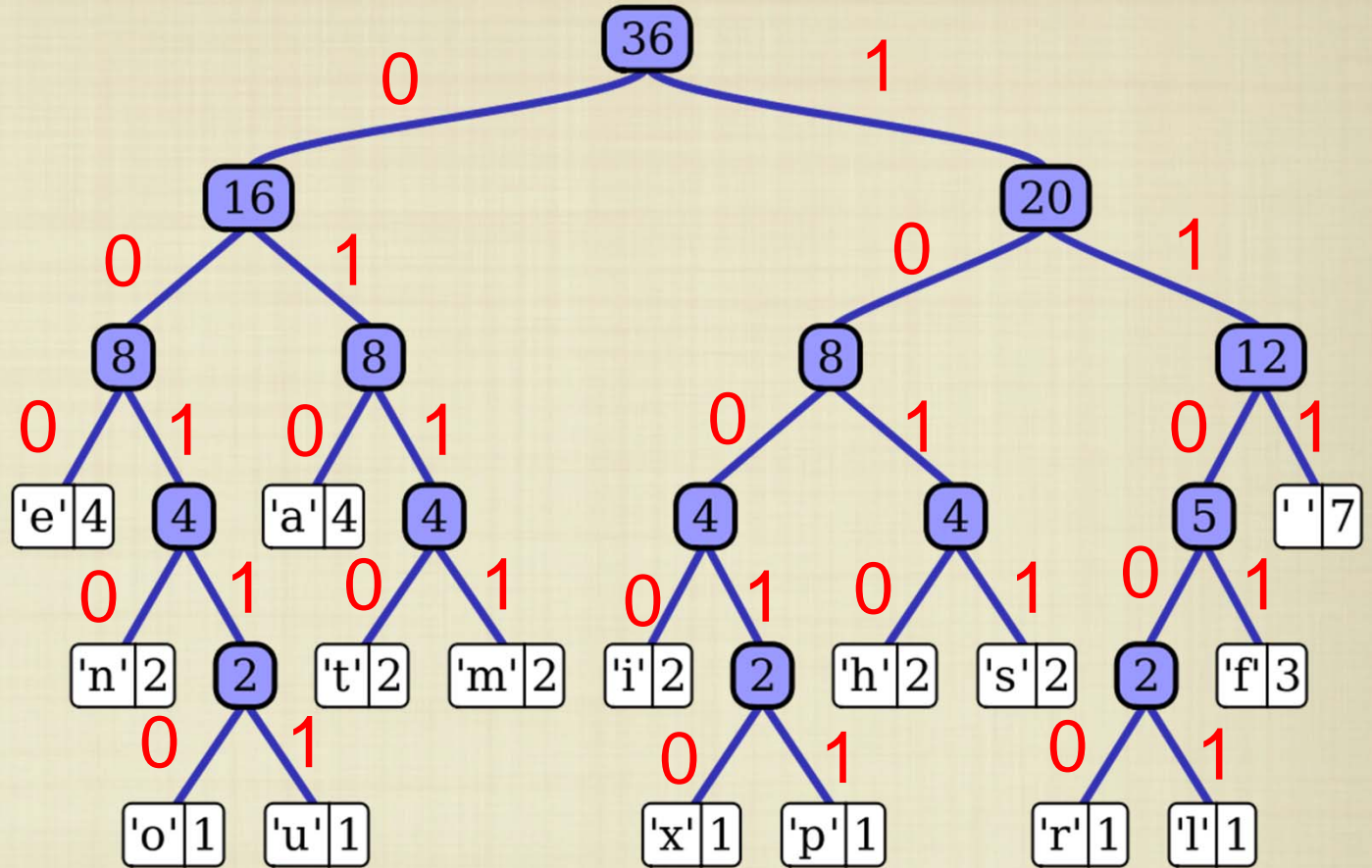


Intuitively, this algorithm places the lowest frequency symbols at the bottom of the tree. But does it always produce the best encoding? David Huffman came up with this approach in 1954 (as a graduate student) and proved that it is optimal.

Huffman Encoding and Decoding

Symbols/Frequencies:

00110	'o': 1
00111	'u': 1
10010	'x': 1
10011	'p': 1
11000	'r': 1
11001	'l': 1
0010	'n': 2
0110	't': 2
0111	'm': 2
1000	'i': 2
1010	'h': 2
1011	's': 2
1101	'f': 3
000	'e': 4
010	'a': 4
111	': 7



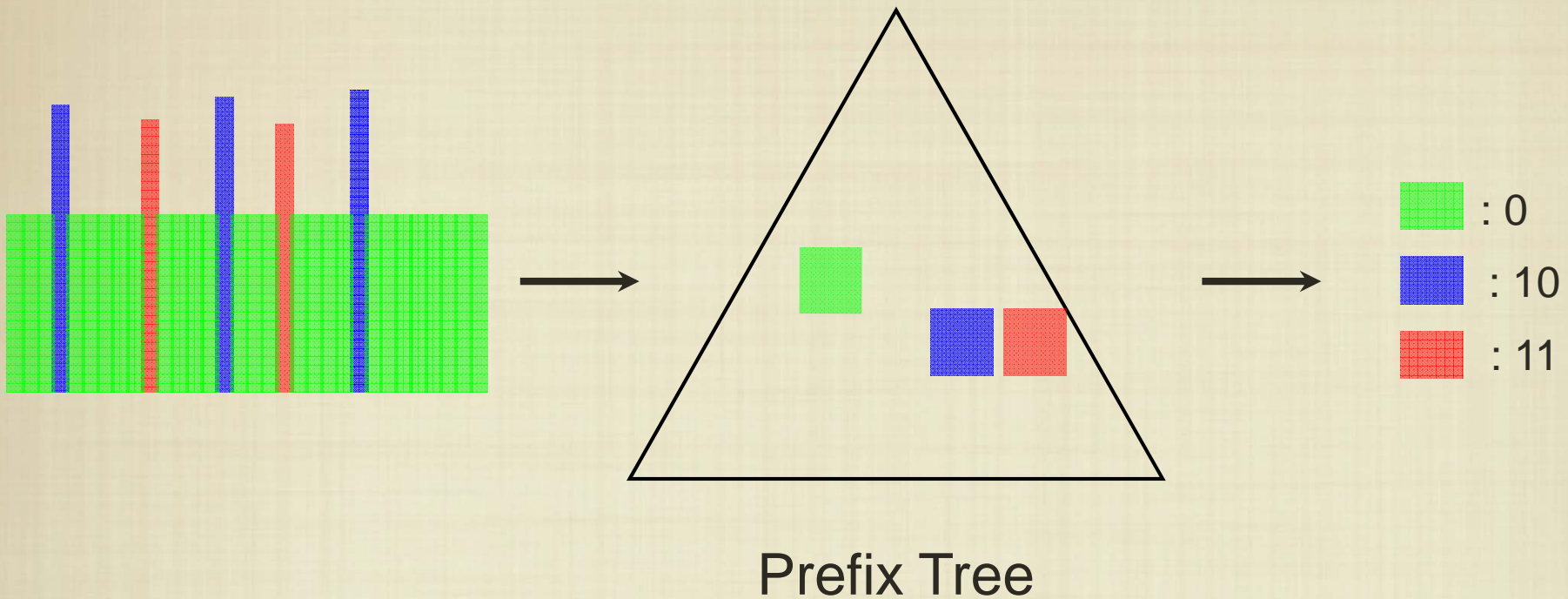
Encoding: Convert sequence of symbols into sequence of bits:

hello → 1010 000 11001 11001 00110

Decoding: Scan encoded file from left to right and simultaneously follow path in tree

1101100010111010 → fish

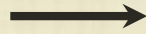
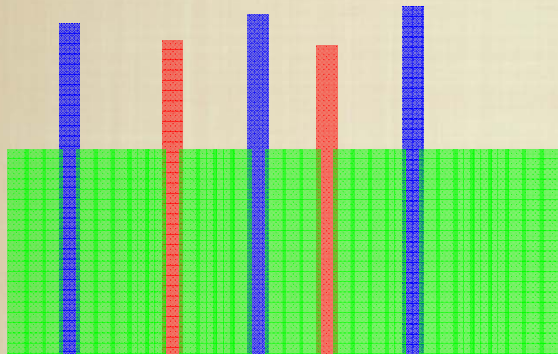
Huffman Coding






The last phase of MP3 encoding compresses the filtered signal using Huffman coding. Intensities are the symbols, and the frequencies are how often they appear in the spectrum.

This algorithm is also widely used for compressing any type of file that may have redundancy (e.g., ZIP, JPEG, MPEG).

Huffman Coding



MP3 Format

	: 0	...
	: 10	0001000000110000
	: 11	1000011000010000
		00000
		...

The last phase of MP3 encoding compresses the filtered signal using Huffman coding. Intensities are the symbols, and the frequencies are how often they appear in the spectrum.

This algorithm is also widely use for compressing any type of file that may have redundancy (e.g., ZIP, JPEG, MPEG).