A Bit of Information

Representing Data Using Bits

Charlie Chiccarine

PaTTAN Computer Science Praxis Prep

Quick Recap

What is a bit?

- The smallest form of data representation
- Either a 0 or a 1
- 0/1, off/on, false/true, etc.
- Bits are used in representing binary numbers

What is a byte?

- Eight bits
- 00000000 - 11111111
- Can represent binary numbers 0-255

There's More Than One

$$\begin{aligned}
\text{byte} &: 8 \text{ bits} \\
\text{kilo} &: 2^{10} = 1024 \approx 1000 \\
\text{mega} &: 2^{20} \approx 1,000,000 \\
\text{giga} &: 2^{30} \approx 10^{9} \\
\text{tera} &: 2^{40} \approx 10^{12} \\
\text{peta} &: 2^{50} \approx 10^{15}
\end{aligned}$$

1 gigabyte is 2^{30} bytes which is 2^{33} bits

Normally, kilo would mean 1000, however, since we're working with bits and binary, we're using base two. While 1000 (10^3) is not a power of two, 1024 (2^{10}) is. We use 1024 since it's close enough to 1000.

Representing Text

How can we represent the letter 'A' using 0s and 1s?

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- Commonly known as ASCII
- A table of 128 symbols and their 7 bit representation
- A link to the full table
- Keep in mind that this is for the visual representation of symbols
  - ASCII 1 is different from the numeric value 1

Binary	Decimal	Symbol
010 0100	36	$
011 0001	49	1
011 0010	50	2
011 0011	51	3
100 0001	65	A
100 0010	66	B
100 0011	66	C
110 0001	97	a
110 0010	98	b
110 0011	99	c

Representing Symbols

How can we represent more than 128 symbols using 0s and 1s?

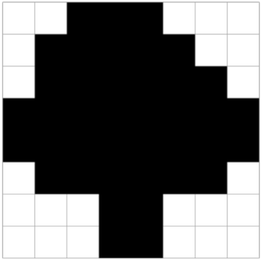
- Unicode : Universal Coded Character Set
- A set of over 1 million code-points that represent a wide range of characters
  - Characters for non-Latin alphabets
  - Mathematical characters
  - Emojis
  - etc etc etc
- Each code-point is identifiable by a 4 digit hexadecimal number
  - Since each hex digit is 4 bits, each code-point is 16 bits
  - Recently the table was extended to 32 bits, but for our purposes, we're going to use the nonextended 16 bit version
- A link to the full table

Representing Images

- Each pixel in the image has to be represented to get all the data across
- Each pixel is a data point that holds important information about the color
- The next slides go through how many bits are needed to represent a pixel in different situations
- Notice how we get more information about the image as we increase the data
- The image stays 8x8 pixels, however, the size of the file increases

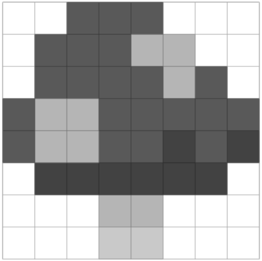
Representing Images - Black and White

- Only two values for each pixel - black and white
- Since 0 correlates to off, black will be 0 while white will be 1
- Each pixel can be represented with 1 bit
- 1 bit * 64 pixels = 64 bits
- The top row would be 11000111



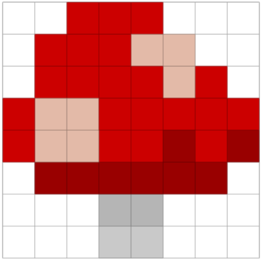
Representing Images - Greyscale

- Gray values range from 0-255 with 0 being black and 255 being white
- Each pixel can be represented with 8 bits
- 8 bits * 64 pixels = 512 bits
- The top row would be
  11111111 11111111
  01011001 01011001
  01011001 11111111
  11111111 11111111



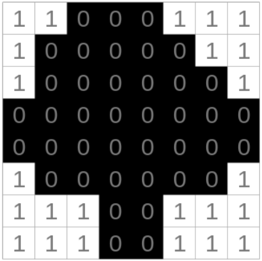
Representing Images - Color

- Each pixel has red, green, and blue values ranging 0-255 each
- Each pixel can be represented with 3 sets of 8 bits (24 bits)
- 24 bits * 64 pixels = 1536 bits
- The start of the top row would be 111111111111111111111111 111111111111111111111111 110011000000000000000000



Representing Images - Encoding Images

- We can encode this image, in order for it to take less space
- The first row originally is 11000111
- This image alternates 0s and 1s
- It starts with zero 0s, then two 1s, then three 0s, then three 1s.
- The encoding becomes 0231 - 00 10 11 01



Calculating Storage Space

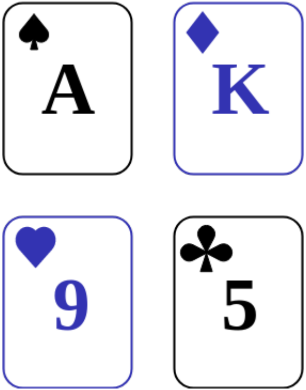
- Prioritize important data
- Choose best method of representation
- Convert to bits
- Make sure each unique data point has a unique representation

Deck of Cards Example - Set Up

Goal: Represent a standard deck of 52 cards using the smallest amount of bits possible

Relevant Card Information:

- Value
  - Ace
  - 2, 3, 4, 5, 6, 7, 8, 9, 10
  - Jack, Queen, King
- Suit
  - Clubs
  - Diamonds
  - Hearts
  - Spades



Deck of Cards - Value

How can we represent the value of the cards?

- Needs to be representable using bits
- ASCII - 7 bits
  - Use the ASCII value for each value A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
  - A=100001, 2=0110010, 3=0110011, ... , K=1001011
  - The largest value we use is Q at 1010001
- Binary - 4 bits
  - Use the binary value for each value: 1(A), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11(J), 12(Q), 13(K)
  - 1=0001, 2=0010, 3=0011, ... , 13=1101
  - The largest value we use is 13 at 1101

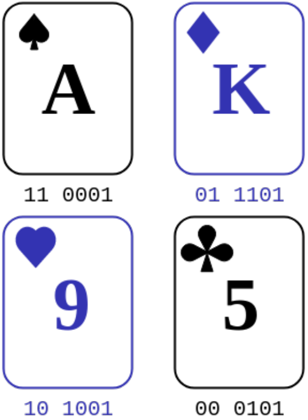
Deck of Cards - Suit

How can we represent the suit of the cards?

- Unicode - 16 bits
  - Unicode uses 4 base-16 numbers to create symbols
  - Use the Unicode symbol for each suit: ♣, ♦, ♥, ♠
  - ♣=0010011001100011, ♦=0010011001100110, ...
- ASCII - 7 bits
  - Use the ASCII value for each suit: **C**lub, **D**iamond, **H**eart, **S**pade
  - C=100011, D=1000100, H=1001000, S=1010011
  - The largest value we use is S at 1010011
- Binary - 2 bits
  - Give each a binary value from 0-3 since there are only 4 data points
  - Club=00, Diamond=01, Heart=10, Spade=11
  - The largest value we use is Spade at 11

Deck of Cards - The Verdict

- By using binary to keep track of the value and suit, we only need 6 bits per card
- 6 bits * 52 cards = 312 bits
- 312 bits = 39 bytes
- The only work we have to do is remember which binary value each suit corresponds to



Deck of Cards - The Verdict

Can we represent each card using fewer than 6 bits?

- No!
- 6 bits provides us with 64 unique code-points
  - With 52 cards each needing a unique code-point, this is good
- If we would try to use 5 bit instead, we would only have 32 unique code-points
  - Due to the pidgeon-hole principle, 32 code-points is not enough for 52 cards

Recap

- Bits are the smallest unit of data
- We use bits to represent all forms of data
  - Numbers
  - Letters
  - Symbols
  - Images
- We can encode data using bits
- More information needs more bits