

CS 100

Tuesday

29 September 2015

Today's Agenda

0. Announcements

1. Calendar Review and Homework 3 Overview

2. Internship Day

3. *The Magic of Computer Science*

A. Numbers

- Representing information digitally: Digital/Analog; Binary/Decimal
- Powers of 2 (hierarchy / divide and conquer)

B. Algorithms and Structure

C. Massive Parallelism (map reduce)

0. Announcements

- Quiz 1:
 - 20 October
 - Begin to organize your notes now!
- Office hours
 - Can check with Amy or Diane in CS department to see if a meeting and/or trip has curtailed office hours

1. Homework 3 Overview

- Numbers and Encodings
 - Learning about the ASCII, Unicode standards
 - I give you several pieces of data. You have to figure out what “information” they contain.
 - This will require some “detective work”
 - Due next week (Tues 6 Oct by midnight)

2. Internship Day!

- Roxanne Coburn
- Michael Royal
- Michael Carlton
- Melissa Shankle
- Zack Anderson
- Stephen Parsons (via skype)

The Magic of Computer Science

Any sufficiently advanced technology is indistinguishable from magic

- Arthur C. Clarke

There is no magic.

- Ken Calvert

Numbers

- Computers manipulate and store numbers
- These numbers are actually stored as bits. (1s and 0s)
- All information must be encoded as bits.




Terms

- bit
- byte, octet
- nibble
- word
- file

Big Ideas in CS: Digital Information

Challenge: how to represent information as (binary) symbols?

– Examples:

- Text message (letters, numbers, symbols)
- Sound
- Speed your car is traveling
- Water pressure in a pipe
- Image of a flower 
- Number of votes cast in an election

History:

Digital Communication

When was digital communication invented?

History:

Digital Communication

When was digital communication invented?

- Ancient times: bonfires, smoke signals
- Late 1700's: Semaphore lines
 - Line-of-sight arrangement of towers with movable arms
- 1833: Electromagnetic telegraph (Gauss & Weber)
- 1834: Morse Code
 - Transatlantic cable: 1866

While it might seem otherwise...

There is no magic.

Computer Science Humor

There are 10 kinds of people in the world: those who know how to count in binary, and those who don't.

The Number 73

<http://www.youtube.com/watch?v=TIYMmbHik08>

CS Magic: Binary Numbers

- Computers use binary values for all calculations

0/1 on/off open/closed

- 1 **binary digit** = 1 **bit**
- 8 bits = 1 **byte** (4 bits is a nibble)
- Computers represent numbers in **binary**

0 = 0 3 = 11 6 = 110 9 = 1001

1 = 1 4 = 100 7 = 111 10 = 1010

2 = 10 5 = 101 8 = 1000

Powers of 2

- Just as powers of 10 are important in the decimal number system, the powers of 2 are important in the binary number system
- How many distinct **patterns** of 8 bits are there?

00000000

00000001

00000010

00000011

00000100

00000101

00000110

00000111

00001000

00001001

00001010

00001011 ...

Usefulness continued

- Given a fixed number of bits, there are a limited number of distinct binary values that can be represented with that many bits
 - Consider: 7 decimal digits can represent at most 10 million phone numbers
- **Question: How many distinct patterns of 8 bits are there?**

00000000	00000001	00000010
00000011	00000100	00000101
00000110	00000111	00001000
00001001	00001010	00001011 ...

Usefulness, continued

- There are $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ different 8-bit patterns of 1's and 0's
 - That's 2^8
- Similarly, there are $2^5 (= 32)$ different 5-bit patterns, and $2^3 (= 8)$ different 3-bit patterns
- In general there are 2^k different k-bit patterns
- It's useful to know 2^n for the smaller values of n
 - $2^1 = 2$ $2^2 = 4$ $2^3 = 8$ $2^4 = 16$
 - $2^5 = 32$ $2^6 = 64$ $2^7 = 128$ $2^8 = 256$

Usefulness, continued

$$2^9 = 512 \quad 2^{10} = 1024 \quad 2^{11} = 2048 \quad 2^{12} = 4096$$

- Extremely useful:

$$2^{10} = 1,024 \approx 1,000 = 10^3$$

$$2^{20} = 1,048,576 \approx 1,000,000 = 10^6$$

$$2^{30} = 1,073,741,824 \approx 1,000,000,000 = 10^9$$

Digital Encodings

- Morse (1837)
- Baudot (1870)
- ASCII (1963-1986)
- Unicode (1991-today)

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A ● —
B — ● ● ●
C — ● — ●
D — ● ●
E ●
F ● ● — ●
G — — ●
H ● ● ● ●
I ● ●
J ● — — —
K — ● —
L ● — ● ●
M — —
N — ●
O — — —
P ● — — ●
Q — — ● —
R ● — ●
S ● ● ●
T —

U ● ● —
V ● ● ● —
W ● — —
X — ● ● —
Y — ● — —
Z — — ● ●

1 ● — — —
2 ● ● — —
3 ● ● ● — —
4 ● ● ● ● —
5 ● ● ● ● ●
6 — ● ● ● ●
7 — — ● ● ●
8 — — — ● ●
9 — — — — ●
0 — — — — —

Baudot Code

(circa 1870)

V	IV		I	II	III	V	IV		I	II	III
		A /	●			●	●	P %	●	●	●
	●	B 8			●	●	●	Q /	●		●
	●	C 9	●		●	●	●	R -			●
	●	D 0	●	●	●	●		S ;			●
		E 2		●		●		T !	●		●
		E &	●	●				U 4	●		●
	●	F ̄		●	●	●		V '	●	●	●
	●	G 7		●		●		W ?		●	●
	●	H ̄	●	●		●		X ,		●	
		I ̄		●	●			Y 3			●
	●	J 6	●			●		Z :	●	●	
●	●	K (●			●		̄ .	●		
●	●	L =	●	●		●	●	* * Erasure			
●	●	M)		●		●		Figure Blank			
●	●	N N°		●	●	●		Letter Blank			
		O 5	●	●	●						

Letters	Figures						Letters	Figures					
		V	IV	I	II	III			V	IV	I	II	III
A	1			●			-	.	●		●		
E	2				●		X	9/1	●			●	
Y	3				●		S	7/	●				●
/	4			●	●		Z	:	●		●	●	
1	3/			●	●		W	?	●			●	●
U	4			●		●	T	2	●		●		●
O	5			●	●	●	V	1	●		●	●	●
							Letter Blank		●				
J	6		●	●			K	(●	●	●		
G	7		●		●		M)	●	●		●	
B	8		●			●	R	-	●	●			●
H	1		●	●	●		L	=	●	●	●	●	
F	5/		●		●	●	N	£	●	●		●	●
C	9		●	●		●	Q	/	●	●	●	●	●
D	0		●	●	●	●	P	+	●	●	●	●	●
figure blank		●					* *	* *	●	●			

Fig 1. The Baudot code

US-ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[59	3B	;	91	5B	[123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-`	63	3F	?	95	5F	`	127	7F	DEL

Encoding Characters

- A Character Encoding maps a **set of characters** to a **set of numbers** (called “code points”)
 - If you have no more than 256 characters, you are done: **Each character can be represented with one (8-bit) byte.** (See ASCII.)
 - If (like much of the world’s population) your language uses more than 256 characters, **life gets more complicated**

Unicode

*“Unicode provides a unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.”*

--- www.unicode.org, “What is Unicode?”

Unicode

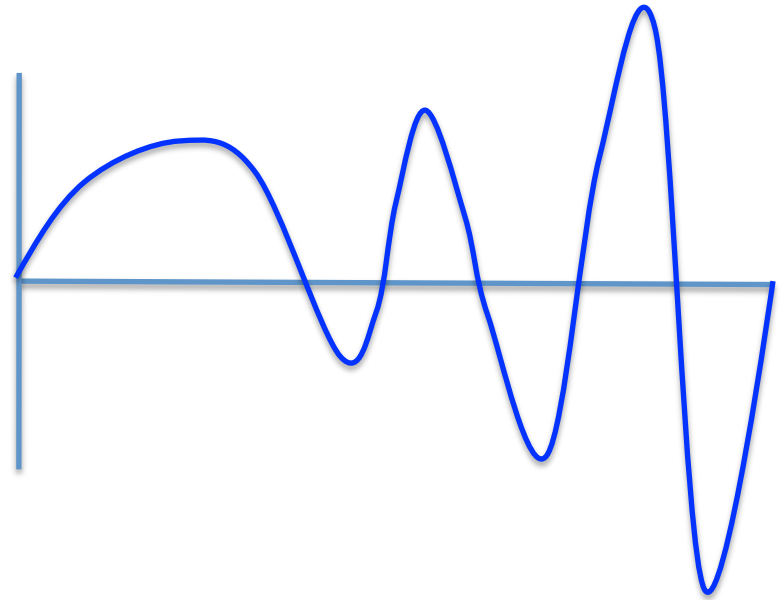
- Provides a single consistent encoding for **all the world's languages (!)**
- Characters (note: *not* “glyphs”) are mapped to numbers in the range 0-1,114,112
- On most platforms: characters represented by 16-bit numbers
 - Note: character encoding defines **integers**, *not bit representations* (endianness has to be specified)

Analog and Digital Phenomena

- **Analog**: **continuous** in time and value
 - there is **always** a value
 - no instantaneous “jumps” in value
 - values are ~**real numbers**
 - infinite precision
 - **uncountable** set of possible values
- **Digital**: **discrete** in time and value
 - values are only meaningful at **certain times**
 - there can be “jumps” between values
 - values are ~**integers**
 - or rationals
 - **countable** set of possible values (often finite)

Analog and Digital

- Analog signals
 - AM and FM Radio
 - NTSC Television signals
- Digital signals
 - Cell Phone signals
 - HDTV
 - JPEG Images
 - MP3 Audio
 - Text

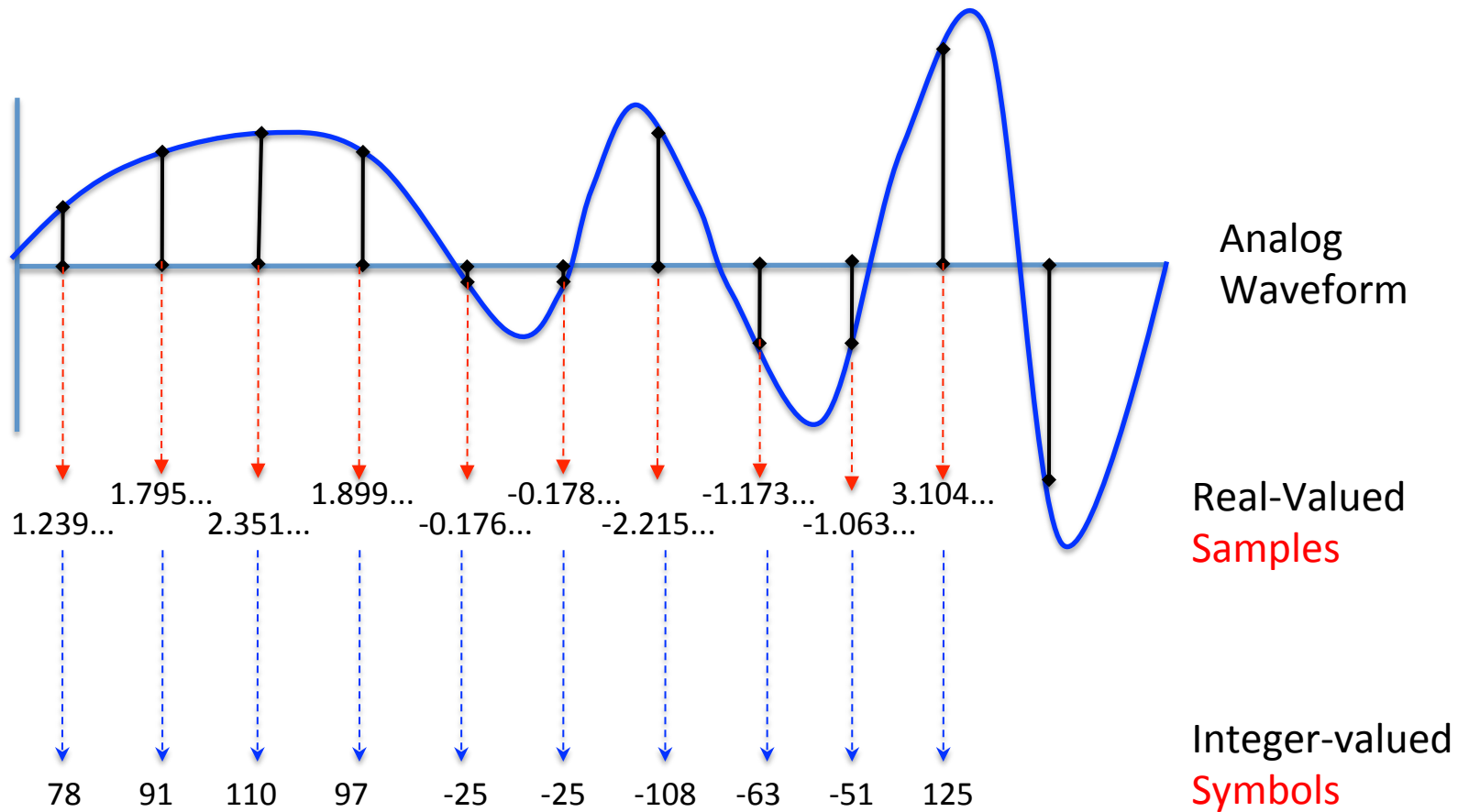


```
1000101010010101010010  
0100101010010101011010  
10011111101011010101000  
11111101011100101010010  
1000101010010101010010  
100000000000000010010
```

Analog-to-Digital Conversion

- Representing analog signals (e.g., music) digitally requires that the signal be converted
- Process steps:
 - **Sampling**: record the value of the signal (waveform) at **periodic, discrete** instants of **time**
 - Waveform represented as a series of **real numbers**
 - **To get an accurate reproduction, must sample at a sufficiently high rate** (Nyquist Theorem)
 - **Quantizing**: convert real-valued **samples** to integer-valued **symbols**

A-to-D Conversion



D-to-A is more or less the reverse: play out the samples in sequence...

Binary Symbols

- We can represent any number of distinct values with different **groups of bits**
- The Important Principle:
 - By adding **one more bit**, we **double** the number of distinct values that can be represented
 - **Each additional bit doubles the precision**
 - Precision **grows exponentially** with the number of bits

Review: Powers of 2

$2^0 = 1$	$2^4 = 16$	$2^8 = 256$	$2^{12} = 4096$
$2^1 = 2$	$2^5 = 32$	$2^9 = 512$	$2^{13} = 8192$
$2^2 = 4$	$2^6 = 64$	$2^{10} = 1024$	$2^{14} = 16384$
$2^3 = 8$	$2^7 = 128$	$2^{11} = 2048$	$2^{15} = 32768$

Recap

- The power of computing comes from the ability to manipulate digital information
- Analog-to-Digital conversion translates natural phenomena (e.g., waveforms) into digital representation
 - Music (MP3), voice telephony
 - Images (JPEG, HDTV)
- Once information is encoded digitally, it can be:
 - Stored on any digital medium
 - Manipulated by any computer
 - **Duplicated perfectly at very low cost**

Bits and Bytes

- A **byte** is a group of **8 bits**
 - The bits in a byte are **ordered**, like the digits in a number
 - What's the biggest integer that can be stored in one byte?
It depends.
 - If we only want to encode **unsigned** values (i.e., no negative numbers): **255** ($=2^8 - 1$)
 - If we need to encode **signed** (both negative and positive) values: **127** ($=2^7 - 1$)
 - There are 256 ($=2^8$) different bytes
- Computers generally represent values using **fixed-size** groups of bytes – 2, 4, or 8 bytes
 - i.e., 16, 32, or 64 bits

Binary Representation of Values

- Obviously only finitely many distinct values can be represented with a fixed number of bits
 - 8 bits: $2^8 = 256$ values
 - 16 bits: $2^{16} = 65,536$ values
 - 32 bits: $2^{32} = 4,294,967,296$ values
 - 64 bits: $2^{64} = 18,446,744,073,709,551,616$ values
- **Most software tools (e.g., Excel) cannot represent arbitrary numbers**

Note: some tools can do arbitrary-precision calculations
...but they are relatively slow

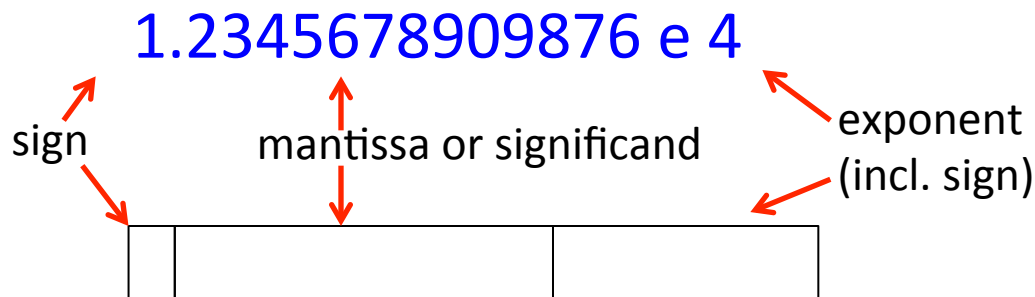
Binary Representation of Values

(cont.)

exponent

mantissa

- **What about “real numbers”?**
 - E.g., 3.141592653589793..., 12345.678909876
- Represented as fixed-size **floating-point** binary numbers
 - “single-precision:” 32 bits
 - “double-precision:” 64 bits
- Similar to (decimal) scientific notation:



Floating-Point Values

- Single-precision (4 bytes)
 - 1-bit sign
 - 8-bit exponent (power of **ten**)
 - 23-bit significand (**binary** fraction)
 - Effect is 7+ decimal digits of precision
- **Double-precision** (8 bytes)
 - 1-bit sign
 - 11-bit exponent (power of ten)
 - 52-bit significand (**binary** fraction)
 - Effect is just under **16 decimal digits** of precision

3. Decimal-to-Binary Conversion (for whole numbers)

- Write down powers of 2 until you get one **larger** than the number to be converted
(To get powers of 2: start with 1 and keep doubling)
- Subtract the **next-to-last** power of 2 **from the given number** and write down a 1 as the first digit of the binary number
 - Call the difference the **remaining number**
- While the **remaining number** is not zero:
 - Go to the **next lower power of 2**:
 - **If it is larger than the current number**, write a **0** (to the right of the previous binary digits)
 - **Otherwise**, write a **1** (to the right of the previous binary digits) and **subtract** it from the remaining number to get the new **remaining #**

3. Binary-to-Decimal Conversion

- Count digits (from the right) to the **leftmost “1”**
 - Note well: **the rightmost digit is the “0th”**
- Write down the corresponding power of 2
 - Leftmost 1 is kth from right => write down 2^k
 - Example: leftmost 1 is digit 4 => write 16
- Now move right
 - for each 1, write down the corresponding power of 2
- Add up the powers of 2 you wrote down
- Example:
1001010 = 2⁶ + 2³ + 2¹ = 64 + 8 + 2 = 74

Some Further Exploration

<https://www.khanacademy.org/science/computer-science/v/binary-numbers>

http://en.wikipedia.org/wiki/Binary_number

Algorithms

Exploit structure of a problem / ordering

Very common problem:

Order a set of information

Index Cards

- Last Name, First Name
- Age
- County in Kentucky where you graduated from High School
 - Write “Kentucky: Adair” for example
 - (if from outside Kentucky, just put your home state + name of school country)
- Number of AP classes you took in High School

Ordering is Crucial for Queries

- Queries (or “data mining”)
 - Find information in the set of data to answer questions
 - Information can be composite:
 - How many people are from Jessamine county?
 - Or individual
 - What is Jamie Lewis’ age?
- Queries are *hard* without structure / ordering

Algorithm for Ordering

- Multiple stages
 - “Row sort”
 - Row reduce via merge

Ordering a.k.a. Sorting

- Row Sort
 - Each row do an insertion sort
 - Insert your name at the right place alphabetically when the stack comes to you
 - At the aisle boundary, you should have a stack of alphabetized cards

Row Reduce

- Two adjacent aisles negotiate and designate one of you the “merger”
- Merger task: merge the two stacks together, keeping them in alphabetical order
- Now find another merger and negotiate again (one of you drops out, the other continues to the next round as a merger)
- Double down!

While the Mergers Merge....

- How many merger rounds must take place to reduce to a single stack?
- Any relationship to binary numbers?
- There is no magic.

Ordered Data: Now Answer Queries

- Ordered / structured data very powerful
- But there is just one of *me* to rifle through the data source to answer questions....
- What about information like this?
 - What are the top Kentucky counties (by count) represented in this class?

Parallelism

- Structure of many problems allows the possibility to solve them in pieces, where the pieces happen simultaneously

(There is no magic)

MapReduce

- Programming model for solving problems
 - In parallel
 - On a cluster
- Back to the reading for the day...
 - Low level abstractions (binary numbers)
 - Structure of information (algorithms)
 - Massively parallel / distributed solutions driven by data centers available on a global scale

MapReduce

- Input reader
- Mapper / Partition
- Reduce (comparison / calculation)
- Output reader

Remember: MapReduce was one of three ingredients in the Data Center Secret Sauce (+ the GFS and BigTable)

Example

- Runners (will perform mapping / partition function)
 - Take a card from stack (input)
 - Put it on work table by county (mapping)
- Reducers (will perform “counting” function)
 - Pick a single county stack
 - Count the number of cards in the stack (reduce)
 - Report the result (output)

Results

- Collect and show results

TakeAways

- Numbers are an abstraction
 - At the lowest level, they are binary sequences
- Media is an abstraction
 - At the lowest level, digital media is a binary sequence
- Algorithms are abstractions
 - At the lowest level, they are sequences of operations on binary sequences
- Parallelism is an abstraction
 - At the lowest level, it is simultaneous, structured activity on binary sequences