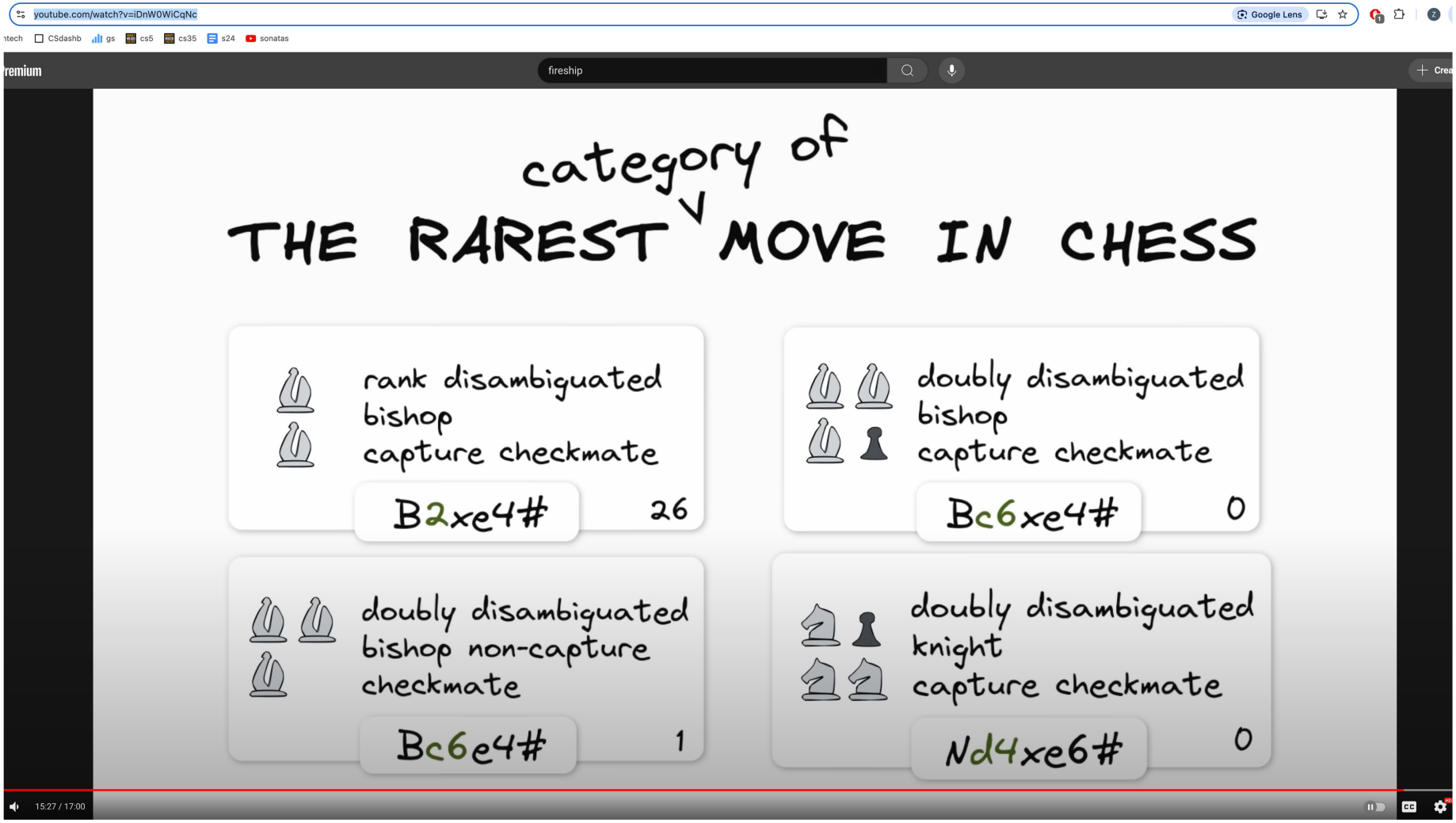
More ***bits*** of CS

Jotto Corner

CS5 guess ZD guess



https://www.youtube.com/watch?v=iDnW0WiCqNc

Welcome, CS5 black!

1/0 vs T/F vs P/F

In 2015, what was the *most-watched video* of all time?

More ***bits*** of CS

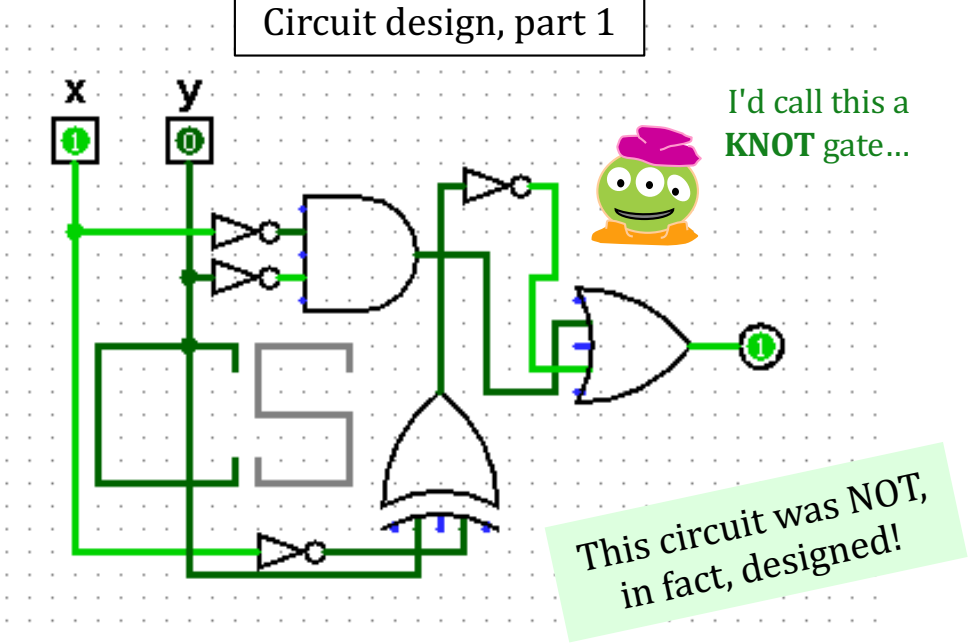
Too many bits? Compress!

Below binary? ***physical circuits***

Jotto Corner

CS5 guess	ZD guess
plane: 4	diner: 3
alien: 3	savvy: 1
?????: ?	ghost: ?

Circuit design, part 1



Hw #4 due Mon. **9/30**

pr0 *(reading)* A bug and a crash!

pr1 *(lab)* binary ~ decimal: *bases!*

pr2 conversions + compressions

extra image processing...

Lots of grutoring hrs - join in... !

Welcome, CS5 black!

1/0 vs T/F vs P/F

In 2015, what was the *most-watched video* of all time?

Last time…

Purpose: *Building CS … bit-by-bit …*

Tasks: *Seeing the bits … everywhere!*

(hw3pr2)

CS Colloquium

Upcoming Events

Presentation by Dr. Nicki Washington: Designing to Disrupt in Computing: Why We All Should "Color Outside the Lines"

Where: SHAN 1430

When: Friday, September 27th, 11:00 a.m.-12:15 p.m.

This talk discusses the importance of technology disruption from the lens of people, policies, and practices in computing environments, including why computing doesn't exist in a vacuum, how they impact technical innovation, and what we all can do to create and advocate for more disruption in spaces we navigate.

Dr. Nicki Washington

Later today…

well, *one year ago*

Summer Poster Session Today!

CS Colloquium

Hailey, Nina, and Sam '23

You're here! *This time, so are…*

Purpose: *Conserving* bits?!

(hw4pr2)

Tasks: *Binary Image Compression*

To do: Try two breakouts…

+ hand in *one* of them …

Sarah Kavassalis
Assistant Professor of Climate and Chemistry
Olin 1257A
skavassalis@g.hmc.edu

Prof. Kavassalis!

Welcome, CS5 black!

Prof. Medero!

You're here! *This time, so are…*

Purpose: *Conserving* bits?!

Tasks: *Binary*

To do: breakouts…

+ hand in *one* of them …

Next Tuesday's presenter: Prof. Kavassalis of the Hixon Center - and Chemistry

Welcome, CS5 black!

Sarah Kavassalis
Assistant Professor of Climate and Chemistry
Olin 1257A
skavassalis@g.hmc.edu

Prof. Kavassalis!

Prof. Medero!

Yo

HARVEY MUDD COLLEGE

Resources for Students Faculty and Staff Alumni Parents Mudd A–Z

Admission and Financial Aid Academics Clinic Research Student Life About Giving

/ Academics / Hixon Center for Climate and the Environment

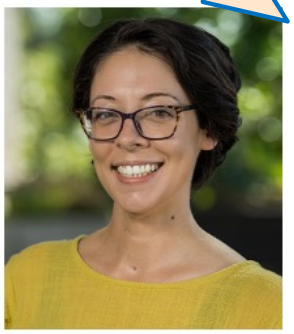
Hixon Center Menu

Hixon Center for Climate and the Environment

n

e

black!



Sarah Kavassalis

Assistant Professor of Climate and Chemistry

Olin 1257A

skavassalis@g.hmc.edu

Prof. Kavassalis!

Prof. Medero!

Lab + tutoring <u>happenings</u>!?

And mysteries…
HARVEY
MUDD
COLLEGE
TAYLOR SWIFT
THE ERAS TOUR

EXTRA PROTECTION!
Search
October 31, 2023
Edit
LIVE

Watch out…

C: > Users > ngupt > Desktop

```python
def numToTernary(N):
    """ Converts a decimal n
    The ternary representation
    it can go to is the 4th v
    left it skips the 9 colum
    a 1 there, and now it has
    or 27(2) + 9(0) + 3(1) + 1
    """
    if N == 0:
        return ''
    elif N%3 == 1:
        return numToBinary(N//3) + '1'
    elif N%3 == 2 :
        return numToBinary(N//3) + '2'
    else:
        return numToBinary(N//3) + '0'
```

CS5 lab story…

CS5'ing…

What's wrong here?

```
        return 3* ternaryToNum(S[0:-1]) + 2

# Function 8
def balancedTernaryToNum(S):
    """
    if S == '':
        return 0
    elif S[-1] == '+':
        newstring = 3* ternaryToNum(S[0:-1]) + 1
    elif S[-1] == '0':
        newstring =  3* ternaryToNum(S[0:-1]) + 0
    elif S[-1] == '-':
```

DEBUG CONSOLE TERMINAL PORTS Python

False

What's wrong here?

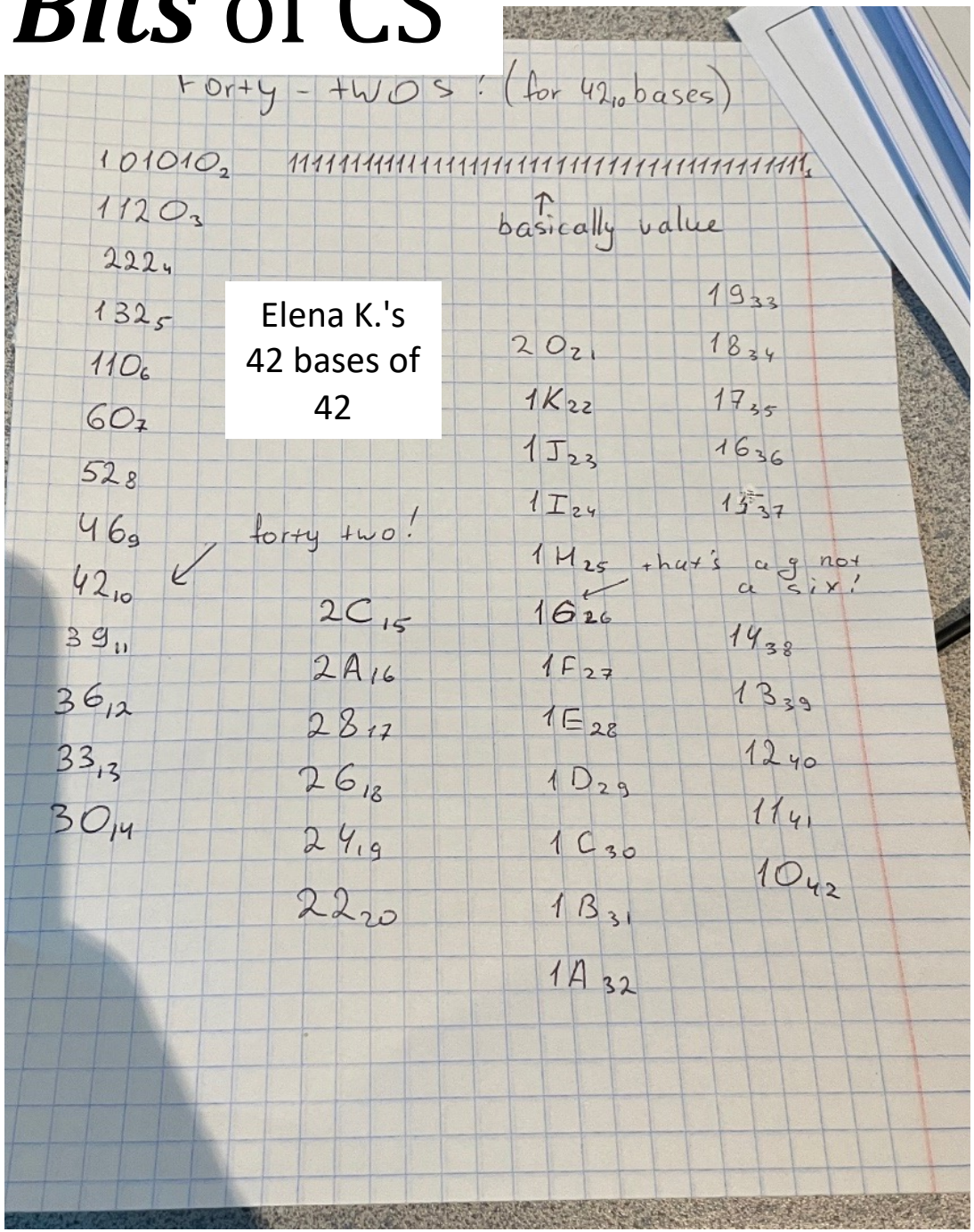
```
100        return 3* ternaryToNum(S[0:-1]) + 2
101
102
103    # Function 8
104    def balancedTernaryToNum(S):
105        """
106        if S == '':
107            return 0
108        elif S[-1] == '+':
109            newstring = 3* ternaryToNum(S[0:-1]) + 1
110        elif S[-1] == '0':
111            newstring =  3* ternaryToNum(S[0:-1]) + 0
112        elif S[-1] == '-':
```

using balanced ternary at all?!

DEBUG CONSOLE TERMINAL PORTS Python

False

Bits of CS



Elena K.'s 42 bases of 42

$CS\ 101_2$

$CS\ 5_{10}$

Why is Christmas the same as Halloween?!

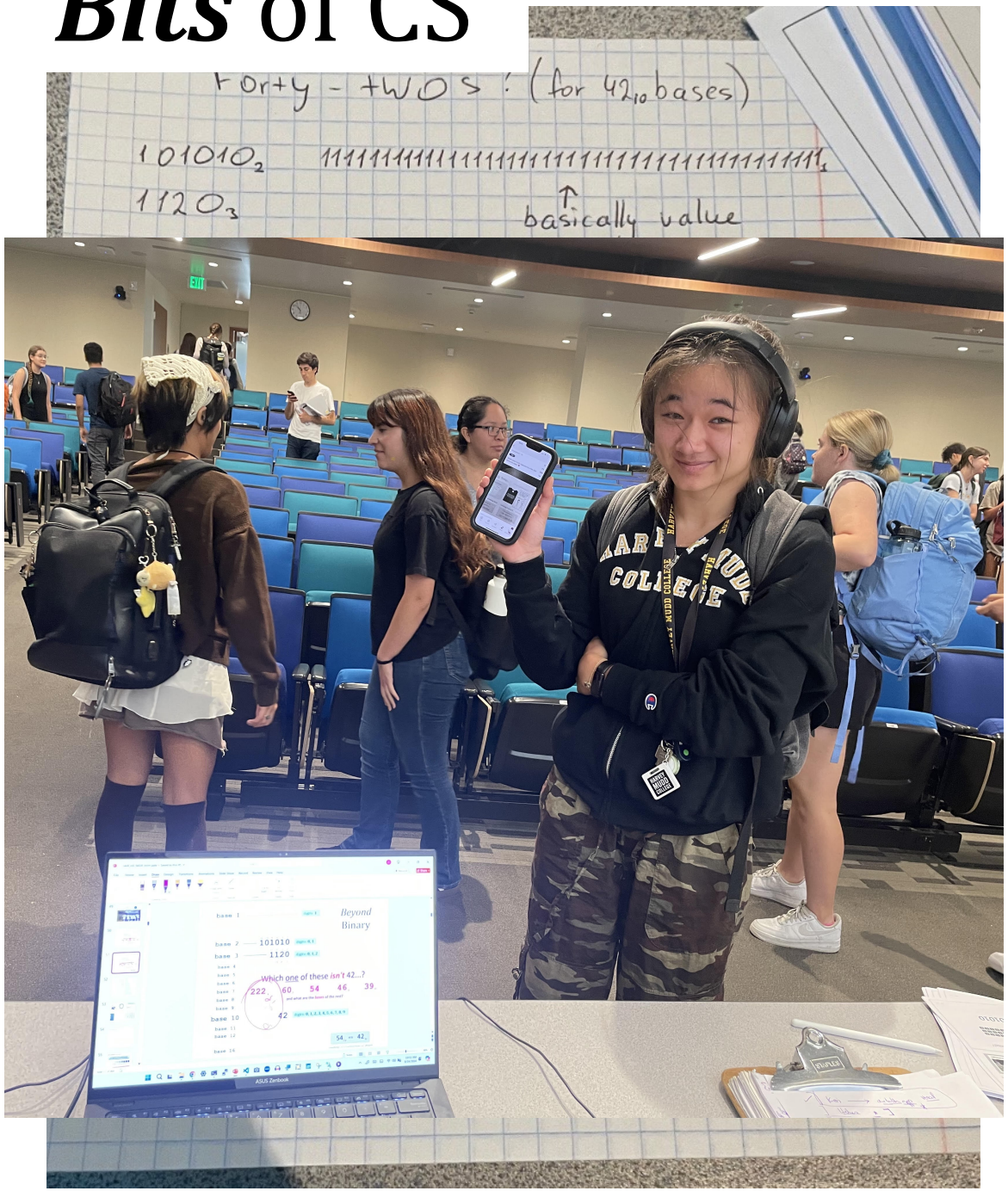
Dec 25 == Oct 31

Fractional bases are possible too!

$54_{7.6} == 42_{10}$

https://brilliant.org/wiki/fractional-number-base/

Bits of CS



$CS\ 101_2$

$CS\ 5_{10}$

Why is Christmas the same as Halloween?!

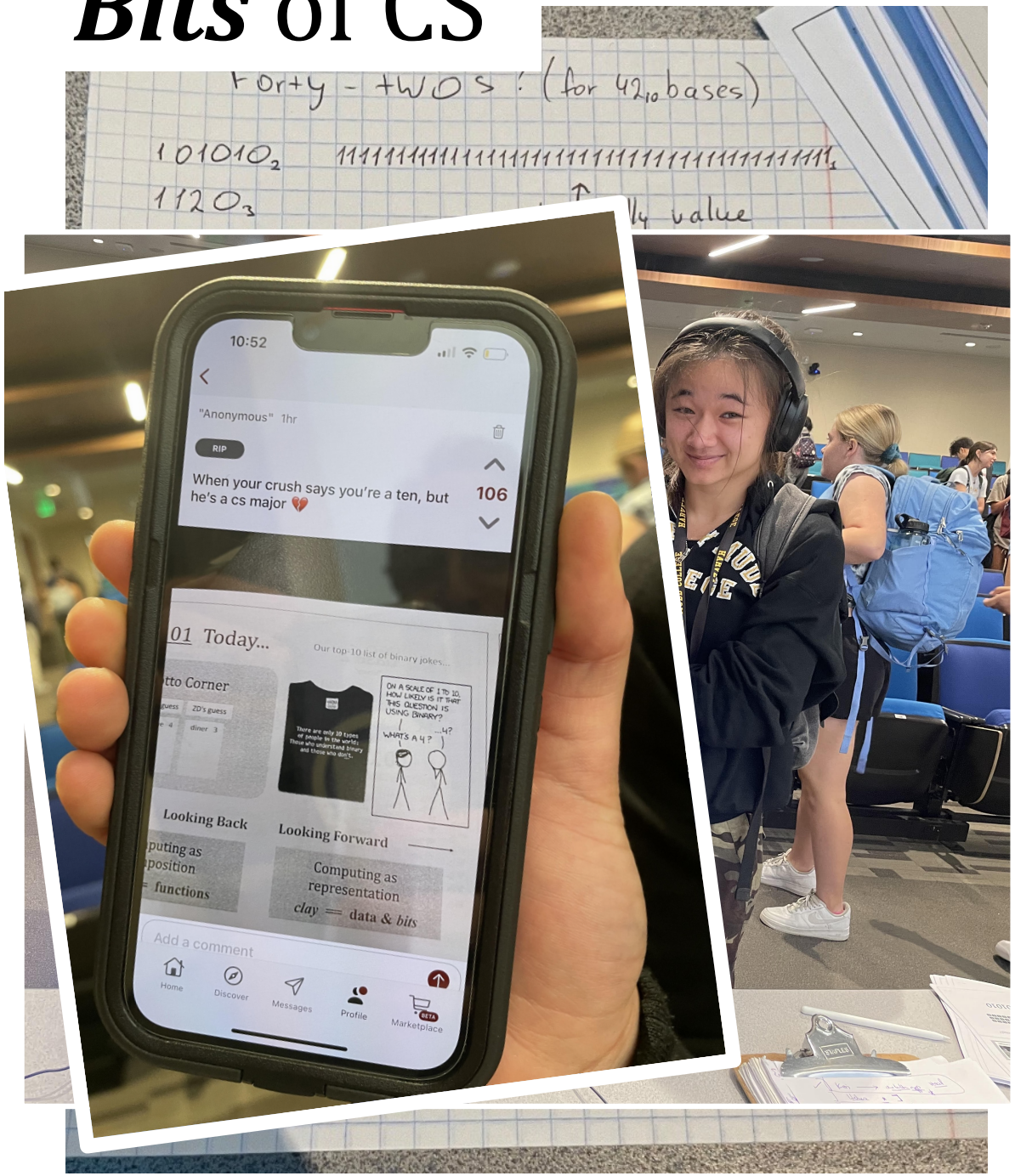
Dec 25 == Oct 31

Fractional bases are possible too!

$54_{7.6} == 42_{10}$

https://brilliant.org/wiki/fractional-number-base/

Bits of CS



$\text{CS } 101_2$

$\text{CS } 5_{10}$

Why is Christmas the same as Halloween?!

Dec 25 == Oct 31

Fractional bases are possible too!

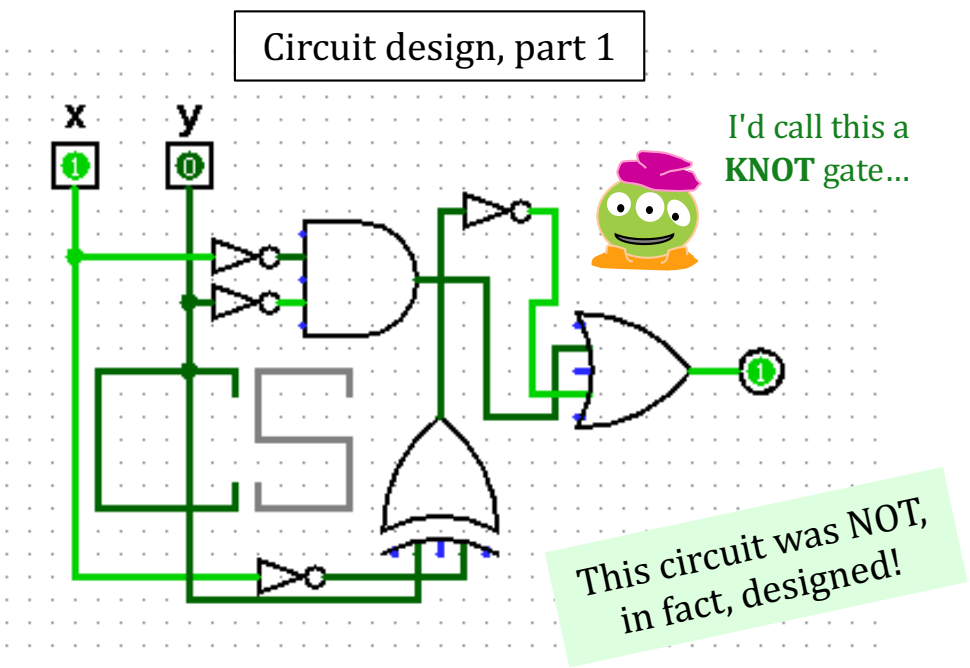
$54_{7.6} == 42_{10}$

https://brilliant.org/wiki/fractional-number-base/

More ***bits*** of CS

Too many bits? Compress!

Below binary? ***physical circuits***



1/0 vs T/F vs P/F

Jotto Corner

CS5 guess	ZD guess
plane: 4	diner: 3
alien: 3	savvy: 1
?????: ?	ghost: ?

Jotto guess?

Hw #4 due Mon. 9/30

pr0 *(reading)* A bug and a crash!

pr1 *(lab)* binary ~ decimal: *bases!*

pr2 conversions + compressions

extra image processing...

Lots of grutoring hrs - join in... !

Th+F+S+Su+M evenings

In 2015, what was the *most-watched video* of all time?

At its base ...

(case)

```
ntb( 42 )
ntb( 21 ) + '0'
ntb( 10 ) + '1'
ntb( 5 ) + '0'
ntb( 2 ) + '1'
ntb( 1 ) + '0'
ntb( 0 ) + '1'
''
```

in

42

out

'101010'

```
def numToBin( N      ):
    """ converts a decimal int to a binary string
    """
    if N==0:        return ''
    else:           return numToBin( N//2 ) + str(N%2)
```

use for hw4pr2

fleek-ified!

What if you wanted base-3 output?! ***base-B output?***

At its base ...

(case)

in

'101010'

```
btn('101010')
2*btn('10101') + 0
2*btn('1010') + 1
2*btn('101') + 0
2*btn('10') + 1
2*btn('1') + 0
2*btn('') + 1
0
```

2*1

2*2*2*1

2*2*2*2*2*1

out

42

```
def binToNum( S     ):
    """ converts a binary string to a decimal int
    """
    if S==''': return 0
    else:      return 2*binToNum(S[:-1]) + int( S[-1] )
```

use for hw4pr2

fleek-ified!

What if you wanted base-3 input?! ***base-B input?***

saves the need for another **`if`**

Bits & Base-2

patterns!?

"value"	base-2 syntax (64 32 16 8 4 2 1)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
42	101010

These are probably worth "knowing"

What's 12?

This, you *definitely* want to know!

What's 21?

Bits & Base-2

Name(s) _____

"value"	base-2 syntax (64 32 16 8 4 2 1)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
42	101010

probably worth "knowing"

definitely need-to-know

What's 42 in binary? 101010

What's 21 in binary?

What's 10 in binary?

What's 5 in binary?

What happens to the *value* if *you shift all bits **right** 1 place?*

What's 3 in binary?

What's 6 in binary?

What's 12 in binary?

What's 24 in binary?

What happens to the *value* if *you shift all bits **left** by 1 place?*

What's the max value 2 bits can hold?

3 bits' max value?

4 bits' max value?

7 bits' max value?

8 bits' max value?

N bits' max value?!

Extra: How could bits represent *negatives*? *fractions*?

Bits & Base-2

"value"	base-2 syntax (64 32 16 8 4 2 1)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
42	101010

probably worth "knowing"

definitely need-to-know

My answers...

What's 42 in binary? 101010
What's 21 in binary? 10101
What's 10 in binary? 1010
What's 5 in binary? 101

What happens to the *value* if *you shift all bits **right** 1 place?* //2

What's 3 in binary? 11
What's 6 in binary? 110
What's 12 in binary? 1100
What's 24 in binary? 11000

What happens to the *value* if *you shift all bits **left** by 1 place?* *2

What's the max value 2 bits can hold? 3 — 11

3 bits' max value? 7 — 111
4 bits' max value? 15 — 1111
7 bits' max value? 127 — 1111111
8 bits' max value? 255 — 11111111

N bits' max value?! ???

Extra: How could bits represent *negatives*? *fractions*?

Bits & Base-2

"value"	base-2 syntax
0	0
1	1
2	10
3	
4	
5	
6	
7	
8	1000
42	101010

64 32 16 8 4 2 1

definitely needed!

Hand these toward your favorite

One-Song Wednesdays

dorm...

My answers...

What's 42 in binary? 101010

What's 21 in binary? 10101

What's 10 in binary? 1010

What's 101

//2

11

10

0

*2

3 bits can hold? 3 11

3 bits' max value? 7 111

4 bits' max value? 15 1111

7 bits' max value? 127 1111111

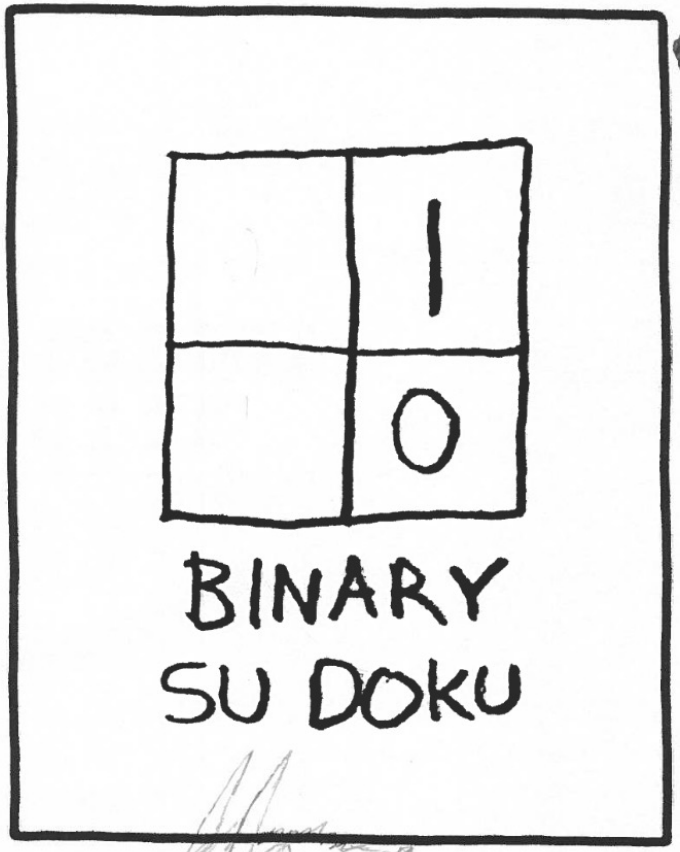
8 bits' max value? 255 11111111

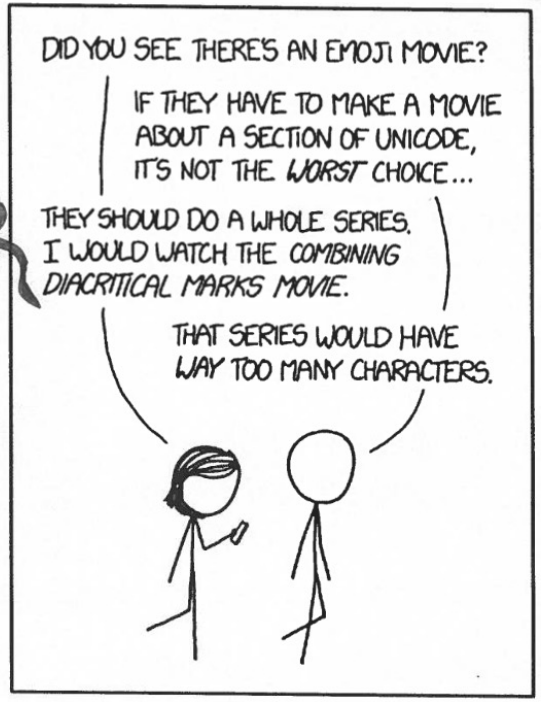
N *bits' max value?!* ???

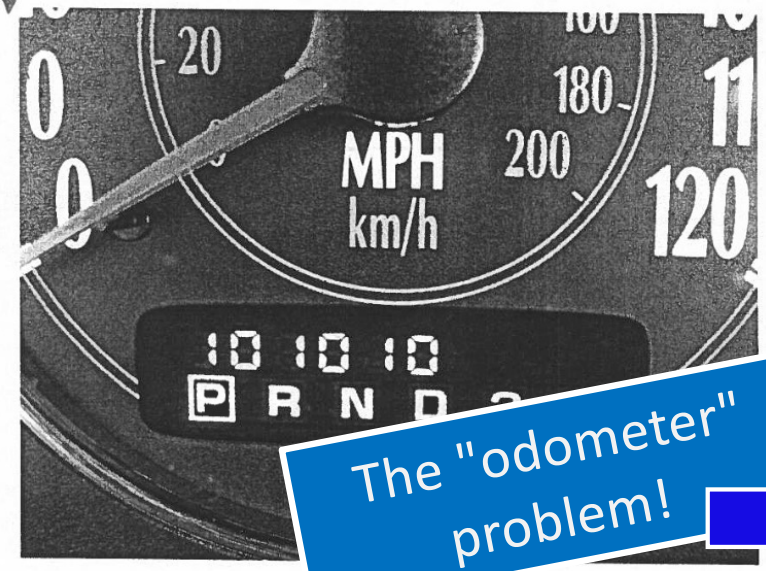
Extra: How could bits represent *negatives*? *fractions*?

Quiz Art!

The best thing about a Boolean is that even if you are wrong, you are only off by a bit.





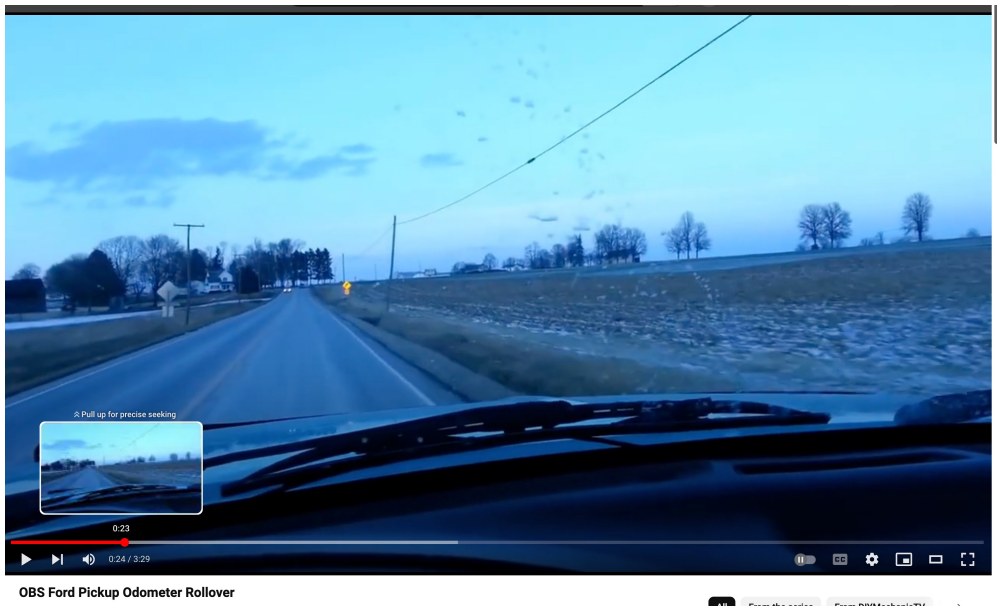


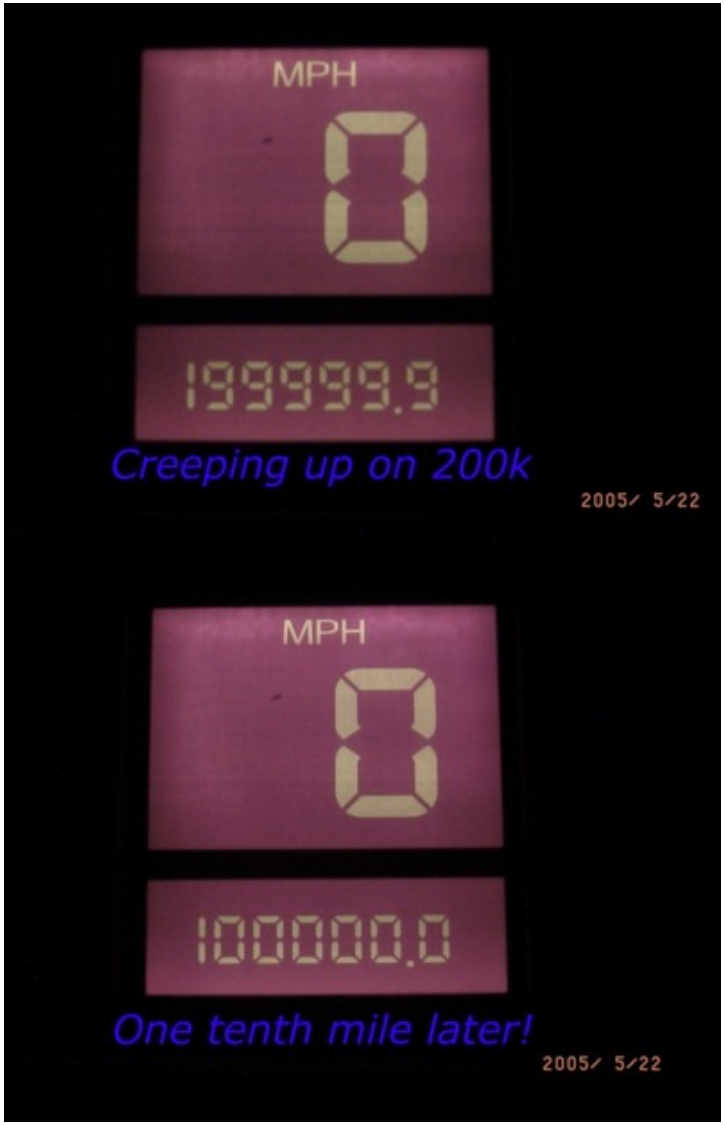
The "odometer" problem!

Odometer rollover: *mechanical*

m2-res_1080p.mp4

Odometer rollover: *electrical?!*





https://www.youtube.com/watch?v=RmF8gvtklT8

How *far* can we count...?

I see some patterns here –
even *with one eye closed!*

with **1 bit**	1	1
2 bits	11	3
3 bits	111	7
4 bits	1111	15
7 bits	1111111	127
8 bits	11111111	255
N bits	1111...1111	max count

Fractions?

Negative?

How *far* can we count...?

I see some patterns here –
even *with one eye closed!*

with **1 bit**	1	1
2 bits	11	3
3 bits	111	7
4 bits	1111	15
7 bits	1111111	127
8 bits	11111111	255
N bits	1111...1111	2^N-1 max

8-bit signed values
-128 to 127

8-bit unsigned values
0 to 255

15 bits

31 bits

Fractions?

Negative?

How *far* can we count...?

I see some patterns here –
even *with one eye closed!*

with **1 bit**	1	1
2 bits	11	3
3 bits	111	7
4 bits	1111	15
7 bits	1111111	127
8 bits	11111111	255
N bits	1111...1111	2^N-1 max

8-bit signed values
-128 to 127

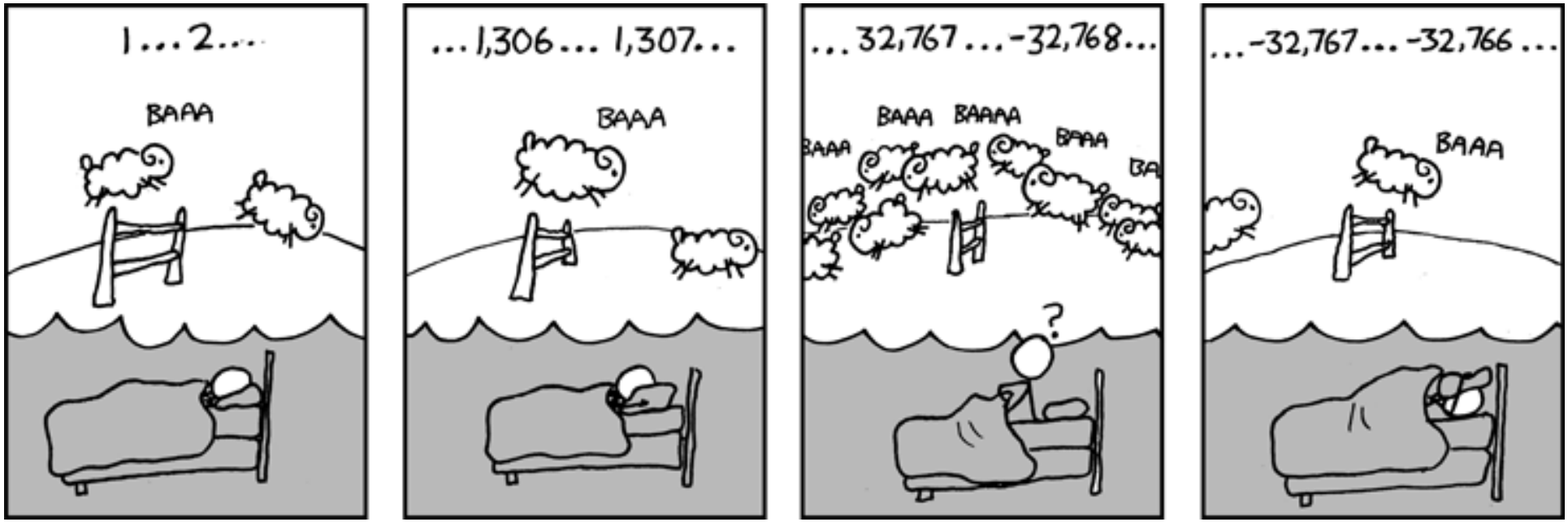
8-bit unsigned values
0 to 255

15 bits	***32,767***
31 bits	

Fractions?

Negative?

Counting sheep, xkcd *style...*



How many <u>bits</u> are being used here?

Ariane 5

This week's reading: ***bits can be vital***

`IndexError` `TypeError` `HumanError`

16 bits ← 64 bits

version 4 version 5

much less-critical errors...

How *far* can we count...?

I see some patterns here –
even *with one eye closed!*

with **1 bit**	1	1
2 bits	11	3
3 bits	111	7
4 bits	1111	15
7 bits	1111111	127
8 bits	11111111	255
N bits	1111...1111	2^N-1 max

8-bit signed values
-128 to 127

8-bit unsigned values
0 to 255

15 bits	***32,767***
31 bits	***2,147,483,647***

Fractions?

Negative?

How *high* can we count…?

in 2015?

List of most viewed YouTube videos

in 2024?

From Wikipedia, the free encyclopedia

How *high* can we count...?

in 2015?

List of most viewed YouTube videos

From Wikipedia, the free encyclopedia

This song?
The song that replaced it?
The song that replaced ***it***?

Top videos

indicates a video that is not a music video.

Rank	Video name[A]	Uploader / artist	Views (as of September 29, 2015)	Upload date	Notes
1.					
2.	"Baby"[4]	Justin Bieber featuring Ludacris	1,216,729,955	February 19, 2010	[C]
3.	"Blank Space"[5]	Taylor Swift	1,173,509,710	November 10, 2014	[D]

The two "Starbucks songs"

How *far* can we count…?

in 2015!

List of most viewed YouTube videos

From Wikipedia, the free encyclopedia

This **list of most viewed YouTube videos** consists of the 30 most viewed videos of all time as derived from YouTube charts.[1] Videos that YouTube suspects have had their view counts manipulated[2] are not included in this list. View counts are based on the YouTube website; many of the videos are music videos that play through YouTube's partner site, Vevo, and YouTube view counts will lag those of Vevo by a few days.[1]

As of September 2015, nine music videos have received over 1 billion views, with the top video, "Gangnam Style", exceeding 2 billion views.

Psy's "Gangnam Style" is the most watched video on YouTube as of September 2015, with over 2.4 billion views.

Top videos

indicates a video that is not a music video.

recognize this number?!

Rank	Video name[A]		Views (as of September 29, 2015)	Upload date	Notes
1.	"Gangnam Style"[3]	Psy	2,421,271,749	July 15, 2012	[B]
2.	"Baby"[4]	Justin Bieber featuring Ludacris	1,216,729,955	February 19, 2010	[C]
3.	"Blank Space"[5]	Taylor Swift	1,173,509,710	November 10, 2014	[D]

31 bits

How *far* can we count…?

in 2015!

List of most viewed YouTube videos

From Wikipedia, the free encyclopedia

This **list of most viewed YouTube videos** consists of the 30 most viewed videos of all time as derived from YouTube charts.[1] Videos that YouTube suspects have had their view counts manipulated[2] are not included in this list. View counts are based on the YouTube website; many of the videos are music videos that play through YouTube's partner site, Vevo, and YouTube view counts will lag those of Vevo by a few days.[1]

As of September 2015, nine music videos have received over 1 billion views, with the top video, "Gangnam Style", exceeding 2 billion views.

Psy's "Gangnam Style" is the most

15 bits	**32,767**
31 bits	**2,147,483,647**

Fractions?

Negative?

recognize this number?!

Top

indicates a video that is not a music video.

Rank	Video name[A]		Views (as of September 29, 2015)	Upload date	Notes
1.	"Gangnam Style"[3]	Psy	2,421,271,749	July 15, 2012	[B]
2.	"Baby"[4]	Justin Bieber featuring Ludacris	1,216,729,955	February 19, 2010	[C]
3.	"Blank Space"[5]	Taylor Swift	1,173,509,710	November 10, 2014	[D]

31 bits

Bits bite back!

THE WALL STREET JOURNAL. ≡ ARTS & ENTERTAINMENT

9:19 am ET
Dec 3, 2014 MUSIC

Psy's 'Gangnam Style' Has Forced YouTube to 'Upgrade' Systems

Gangnam Style Video Overflows YouTube Counter

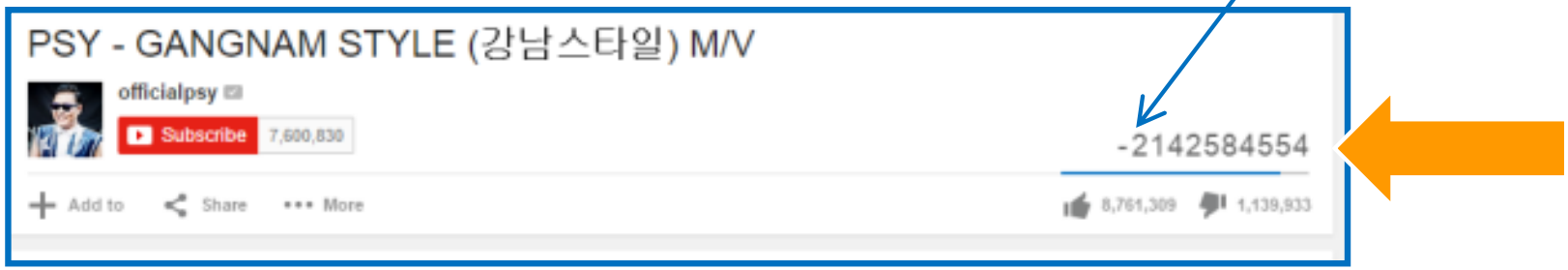
By Rick Regan (Published December 3rd, 2014)

On Monday, Psy's Gangnam Style video exceeded the limit of YouTube's view counter; this is what Google had to say

> *"We never thought a video would be watched in numbers greater than a 32-bit integer (=2,147,483,647 views)…"*

The "sign bit" has flipped to one.

Thus, the number has become *negative*… !



Bits back down...

When's our next wrap?

The billion-view mark was first passed by Gangnam Style in December 2012.

On 13 January 2022, Pinkfong's "Baby Shark" became the first video to hit 10,000,000,000 views.

	Video name	Uploader	Views (billions)	Date
1	"Baby Shark Dance"[7]	Pinkfong Baby Shark - Kids' Songs & Stories	15.09	June 17, 2016
2	"Despacito"[10]	Luis Fonsi	8.54	January 12, 2017
3	"Johny Johny Yes Papa"[18]	LooLoo Kids - Nursery Rhymes and Children's Songs	6.95	October 8, 2016
4	"Bath Song"[19]	Cocomelon - Nursery Rhymes	6.85	May 2, 2018
5	"Wheels on the Bus"[20]	Cocomelon - Nursery Rhymes	6.56	May 24, 2018
6	"See You Again"[21]	Wiz Khalifa	6.40	April 6, 2015
7	"Shape of You"[26]	Ed Sheeran	6.33	January 30, 2017
8	"Phonics Song with Two Words"[29]	ChuChu TV Nursery Rhymes & Kids Songs	6.00	March 6, 2014
9	"Uptown Funk"[30]	Mark Ronson	5.33	November 19, 2014
10	"Gangnam Style"[31]	Psy	5.29	July 15, 2012

one.

ome

Hw4: *images are just bits, too*!

hw4pr3 (*extra*)

<u>old</u> pixel at 42,42 has

red = 1 (out of 255)
green = 36 (out of 255)
blue = 117 (out of 255)

<u>new</u> pixel at 42,42 has

guesses as to this transformation?

how many <u>bits</u> are being used for each color channel?

Hw4: *images are just bits, too*!

hw4pr3 (*extra*)

<u>old</u> pixel at 42,42 has

red = 1 (out of 255)
green = 36 (out of 255)
blue = 117 (out of 255)

rgb-invert

<u>new</u> pixel at 42,42 has

red = 254 (out of 255)
green = 219 (out of 255)
blue = 138 (out of 255)

how many <u>bits</u> are being used for each color channel?

Hw4: *images are just bits, too*!

hw4pr3 (*extra*)

rgb-invert

<u>old</u> pixel at 42,42 has

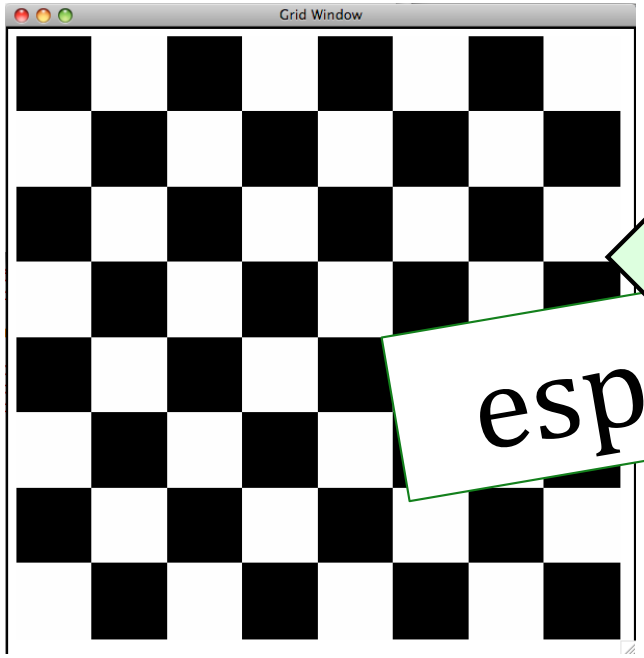
red = 1 (out of 255)
green = 36 (out of 255)
blue = 117 (out of 255)

<u>new</u> pixel at 42,42 has

red = 254 (out of 255)
green = 219 (out of 255)
blue = 138 (out of 255)

how many <u>bits</u> are being used for each color channel?

Hw4: *images are just bits, too*!




```
10101010
01010101
```

especially ***binary*** images

```
01010101
10101010
01010101
```

Binary Image

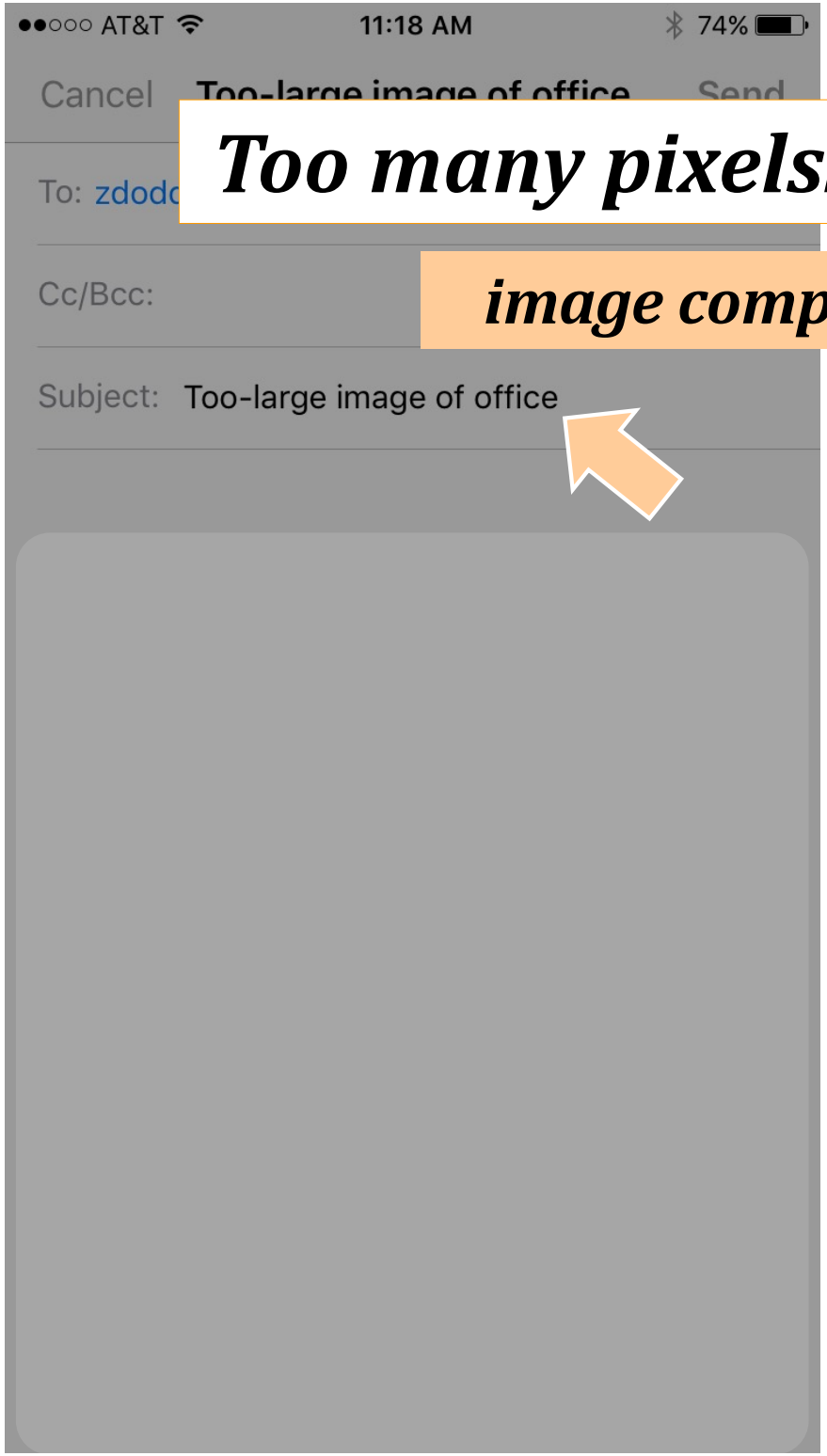
Encoding as raw bits

one big string of 64 characters

```
"1010101001010101101010100101010110101010010101011010101001010101"
```

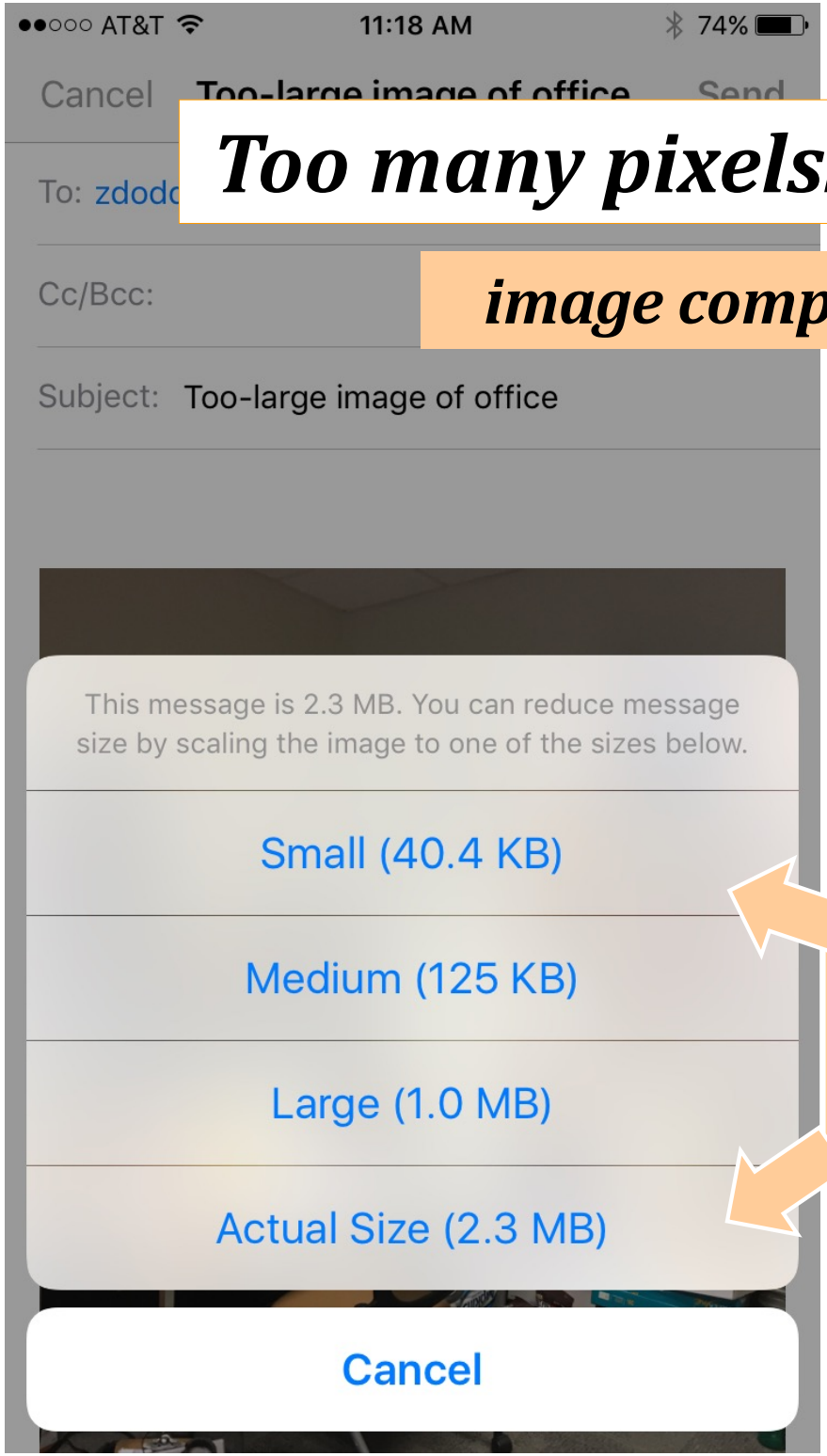
Too many pixels... too little time + space!

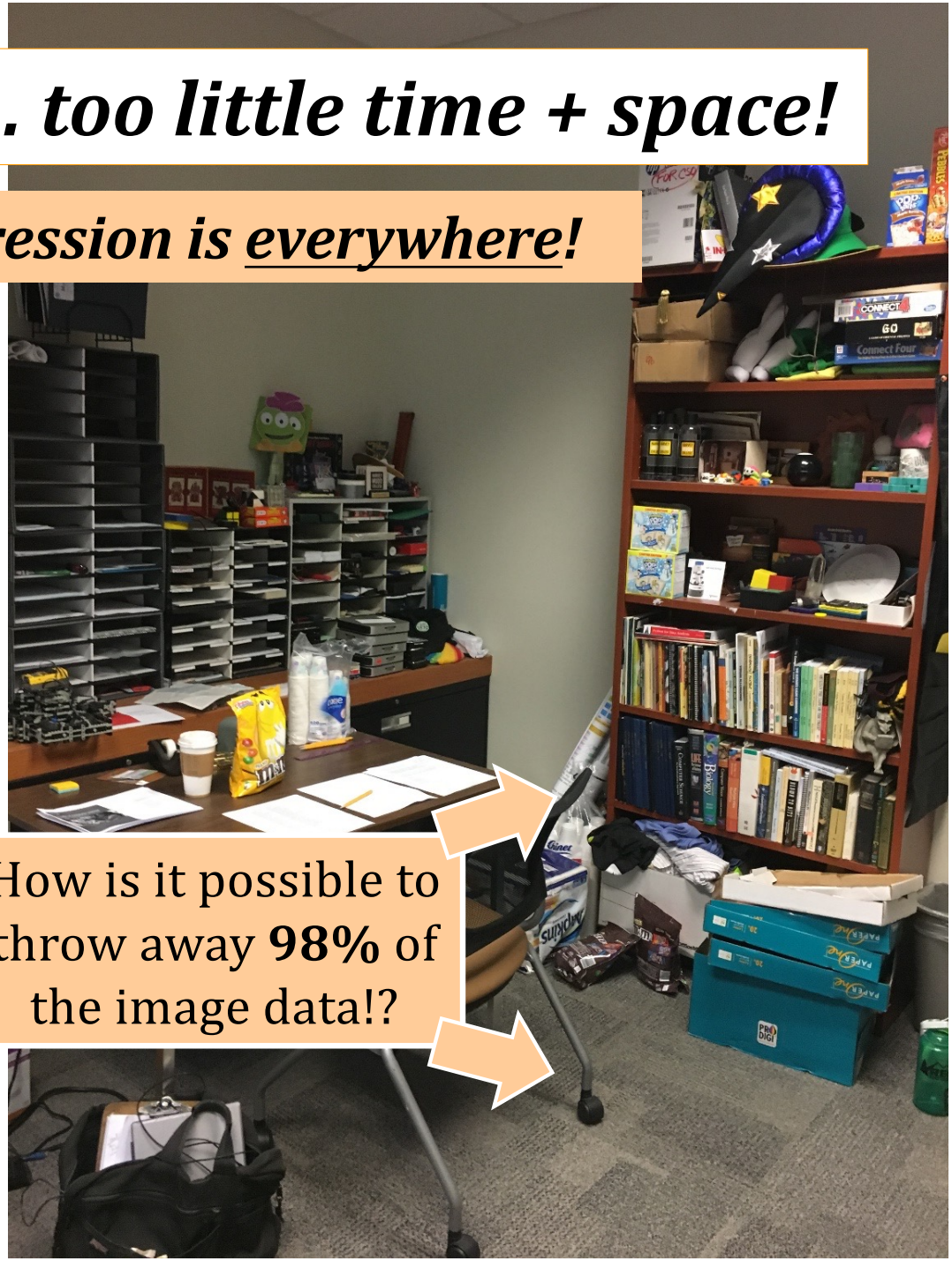
image compression is <u>everywhere</u>!



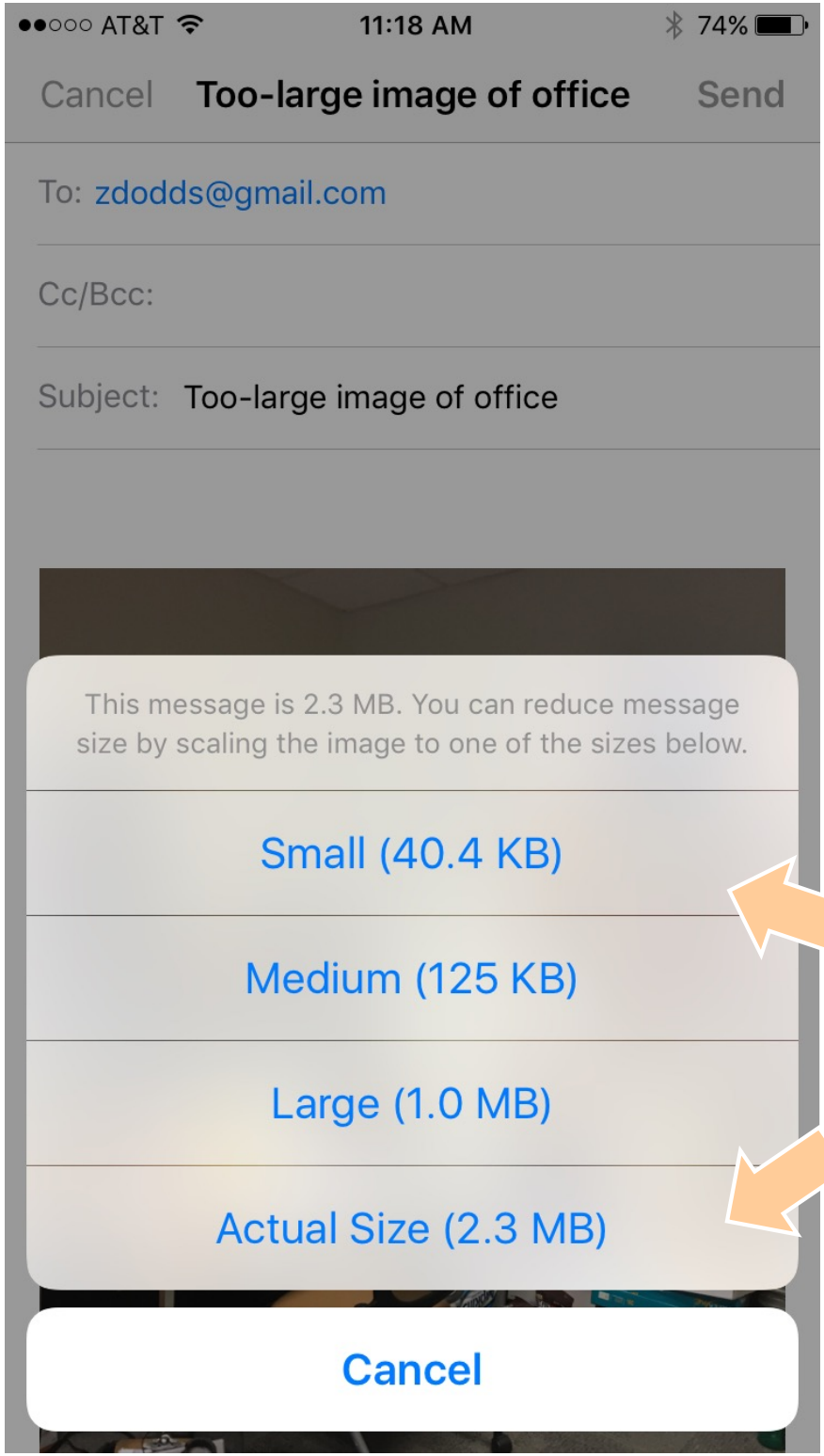
Too many pixels... too little time + space!

image compression is <u>everywhere</u>!





How is it possible to throw away **98%** of the image data!?



One possible solution!

We ***could*** throw away 98% of the image *area*!

Looks like the right 2% to keep!

How is it possible to throw away **98%** of the image data!?

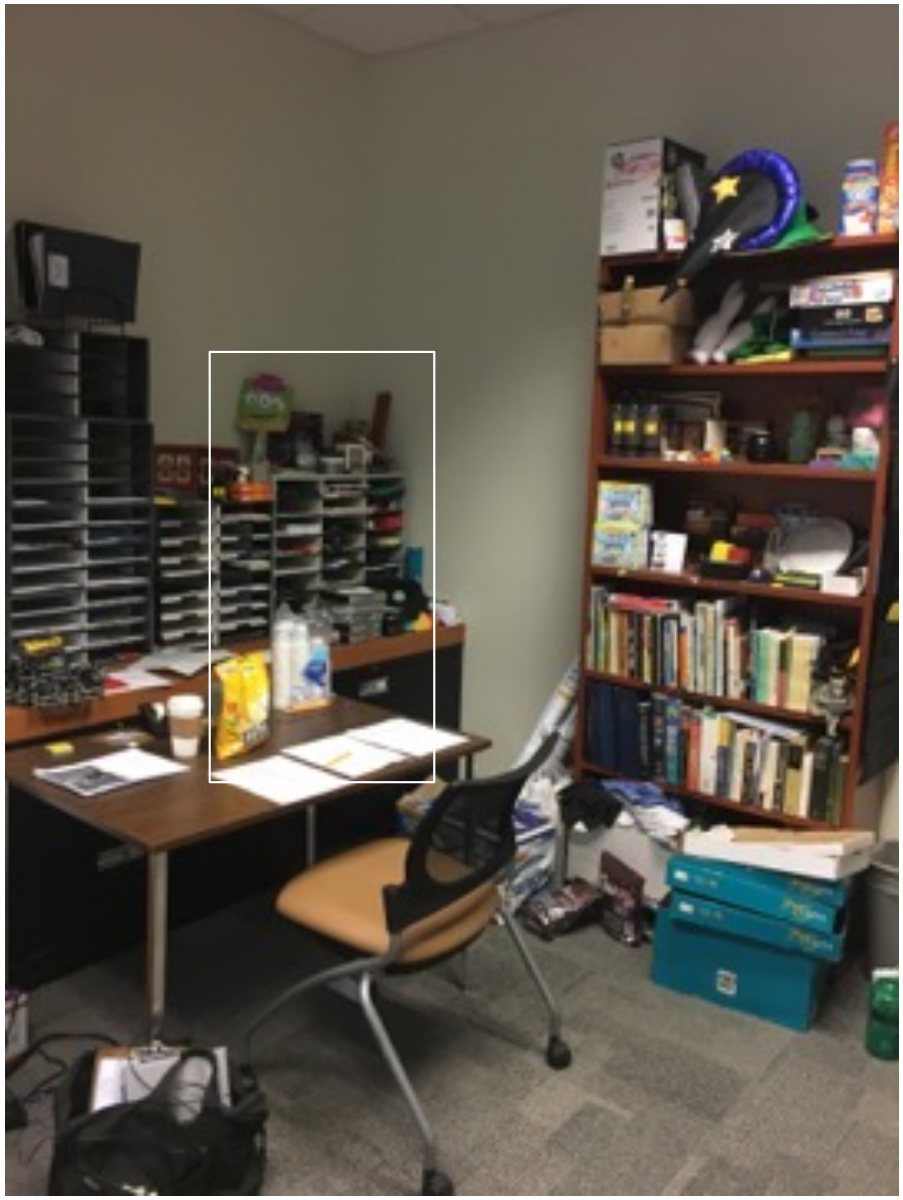
More often... what's done?

***compressed** to 40kb*

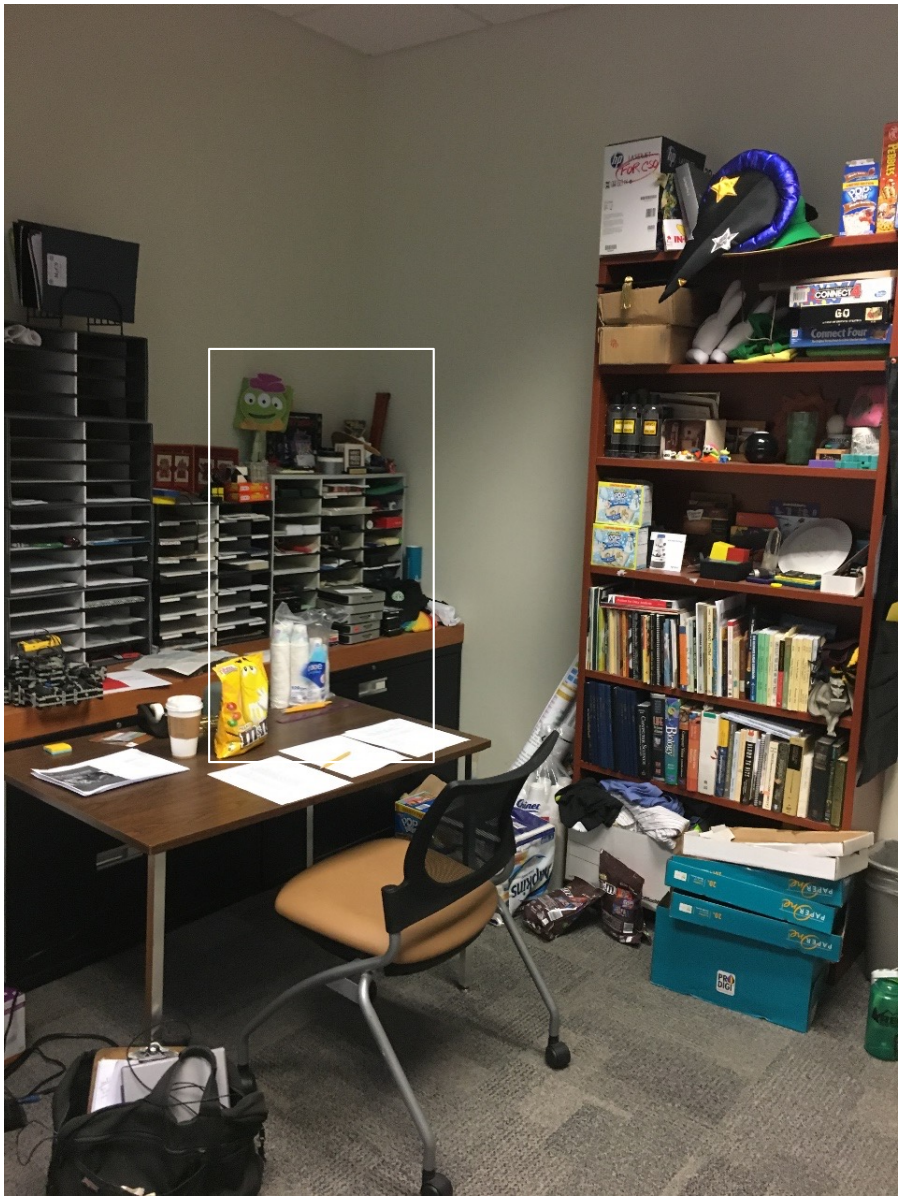
40,000 bytes

***original**: 2.3mb*

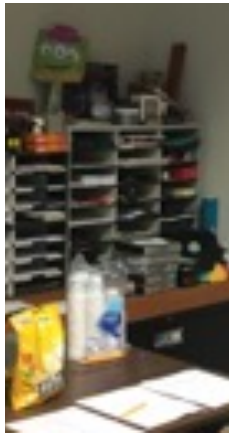
2,300,000 bytes

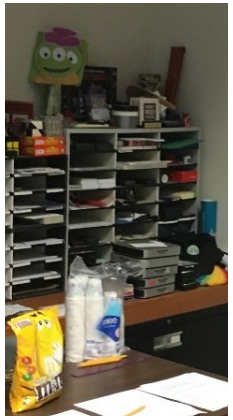


compressed *to 40kb*



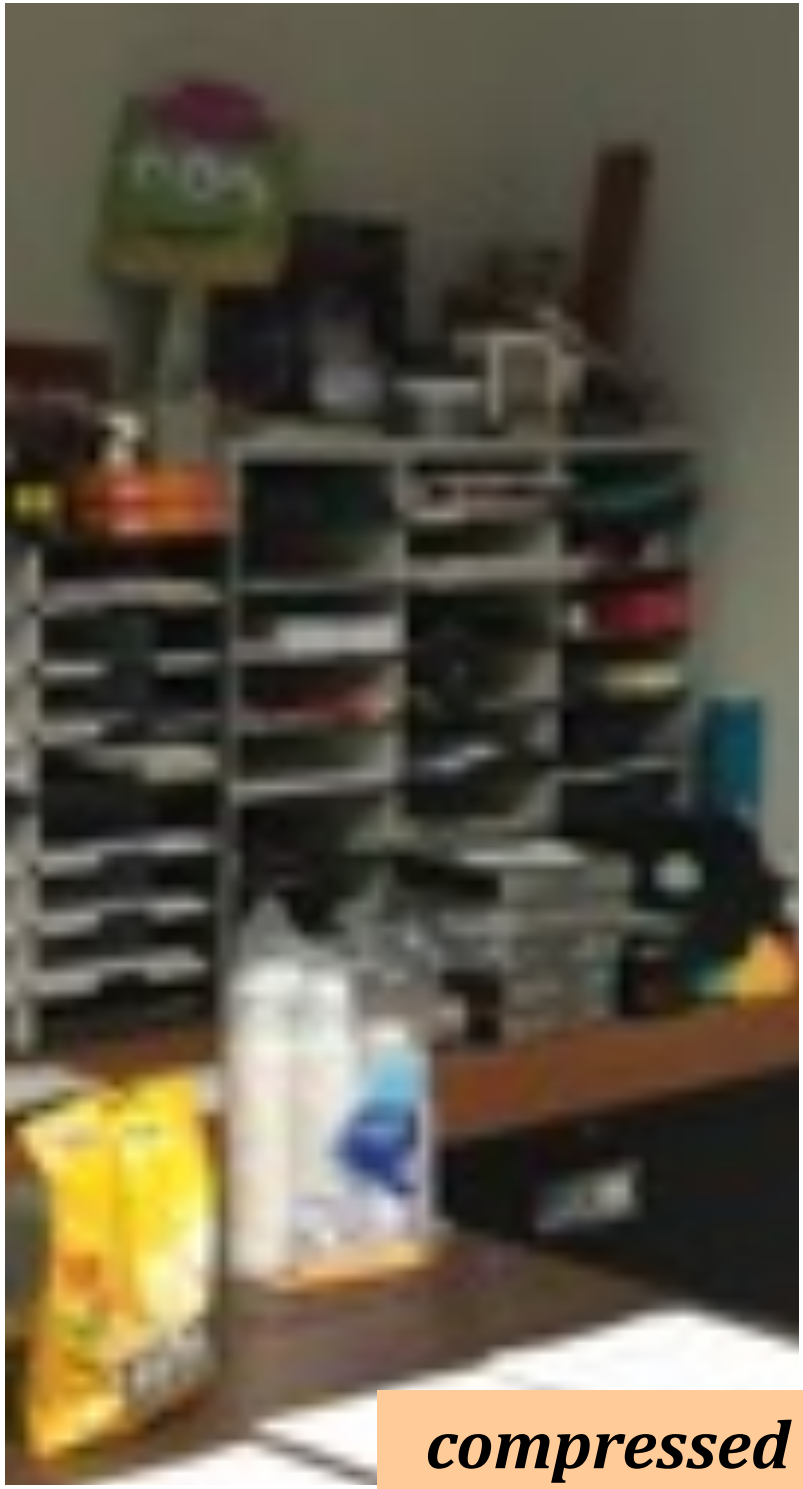
original*: 2.3mb*





compressed

original

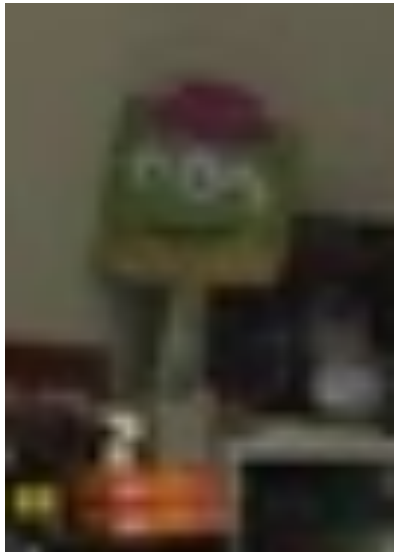


compressed

original

compressed

original



compressed

original

compressed

original

Lossy *compression vs.* ***Lossless***

Do we need both?

Think of three situations where ***Lossy*** *compression is better.*

Think of three situations where ***Lossless*** *compression is better.*

Lossy *compression vs.* ***Lossless***

We need both!

approximations	data
thought+uncertainty	science+math
age	money
... situations where ***Lossy*** *compression is better.*	*... situations where* ***Lossless*** *compression is better.*

Does this apply to numbers?

Yes!

56MB	56762440B
3.14	3.14159265358979323846264338327950288419716939937510
$1000	$1042.42

Let's use both!

Lossy compression

96 x 3670

Bitmap - Uncompressed
57.7 MB

JPEG - 90% Quality
9.2 MB

Lossless compression

2586

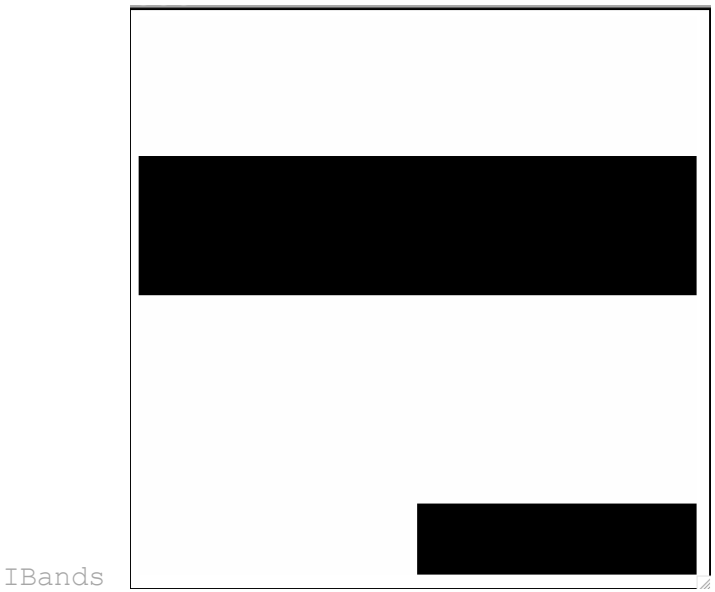
Bitmap – uncompressed
26.8 MB

PNG – compressed
14.5 MB

Starting ...

... with *BINARY* images

Hw4: *lossless* binary image compression



Binary Image

```
00000000
00000000
11111111
11111111
00000000
00000000
00000000
00001111
```

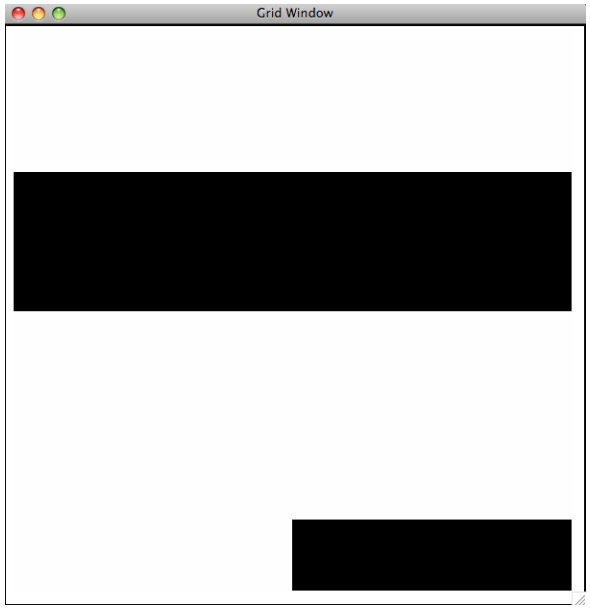
same-data streaks

Encoding as raw bits

one big string of 64 characters

a very <u>compressible</u> image...

Hw4: ***lossless*** binary image compression



IBands

```
00000000
00000000
11111111
11111111
00000000
00000000
00000000
00001111
```

same-data streaks

Binary Image

Encoding as raw bits

one big string of 64 characters

If our images tend to have <u>long streaks of unchanging data</u>, how might we represent it more efficiently, ***but still in binary***?

compress
uncompress

64 bits

"0000000000000000<u>1111111111111111</u>0000000000000000000000000000<u>1111</u>"

16 **0**'s 16 **1**'s 28 **0**'s 4 **1**'s

Hw4: lossless image compression

64 bits

16 0's 16 1's 28 0's 4 1's

"0000000000000000111111111111111100000000000000000000000000001111"

0 is the first digit

There are 16 of them.

1 is the next digit

Again, there are 16 of them.

0 is the next digit

There are 28

1 is the final digit

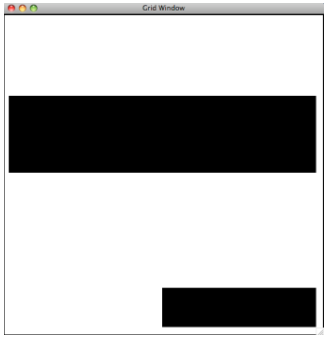
There are 4

0100001100000111001100

0100001100000111001100

22 bits !!!

00000000
00000000
11111111
11111111
00000000
00000000
00000000
00001111



One possible algorithm:

```
bit #repeats
```

Problems with this?

Hw4: lossless image compression

0 is the first digit

and there are 1,098,188 of them.

`0100001100000111001100`

It's ambiguous! this could just be a ***huge number*** of 0 pixels!

`0100001100000111001100`

our algorithm:

`bit #repeats`

could be misinterpreted!

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000000...
```

1,098,188 zeros!

huge empty image

fixed-width compression

64 bits

"0000000000000000111111111111111100000000000000000000000000001111"

0 is the first digit

There are 16 of them.

1 is the next digit

Again, there are 16 of them.

28 zeros

and so on...

4 ones

0001000010010000000111001 0000100

7 bits: # of repeats

7 bits: # of repeats

8-bit data block

8-bit data block

8-bit data block

8-bit data block

00000000
00000000
11111111
11111111
00000000
00000000
00000000
00001111

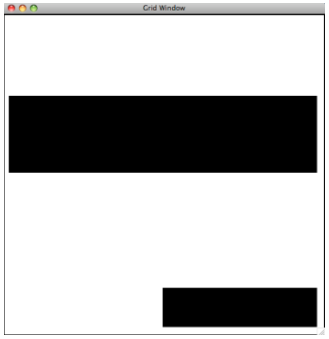
00010000100100000001110010000100

32 bits !!!

We need ***fixed-width*** blocks:

bit	#repeats
1 bit fill	7 bits for the # of repeats

← **8-bits in each "block"** →



fixed-width compression

64 bits

"0000000000000000111111111111111100000000000000000000000000001111"

0 is the first digit

There are 16 of them.

1 is the next digit

Again, there are 16 of them.

28 zeros

and so on...

4 ones

0001000010010000000111001000010 0

7 bits: # of repeats

7 bits: # of repeats

8-bit data block

8-bit data block

8-bit data block

8-bit data block

00010000100100000001110010000100

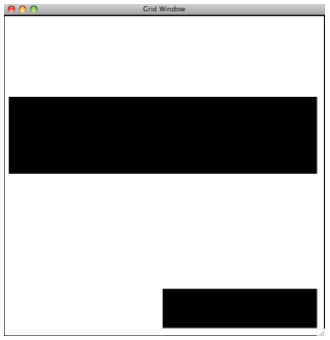
32 bits !!!

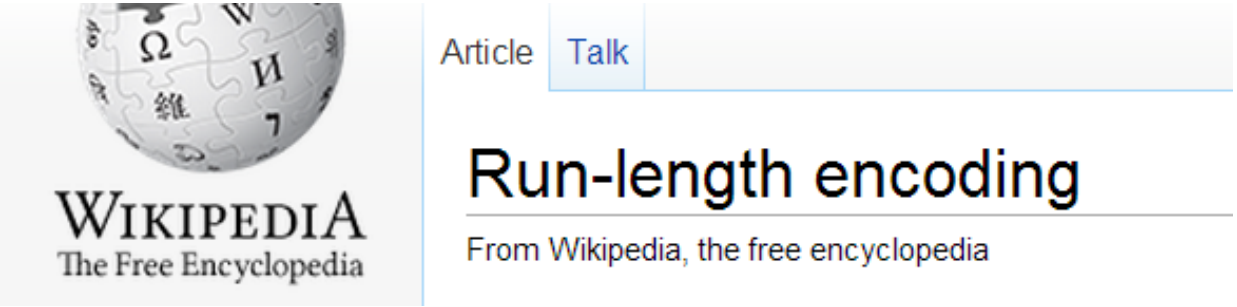
00000000
00000000
11111111
11111111
00000000
00000000
00000000
00001111

We need ***fixed-width*** blocks:

bit	#repeats
1 bit fill	7 bits for the # of repeats

← **8-bits in each "block"** →





If you use ***7 bits*** to hold the # of consecutive repeats, what is the largest number of bits that *one block can represent*?

00010000

1 bit: the initial pixel

7 bits: # of repeats

8-bit total data block

7 bits?

B bits?

*What if you need a **larger** # of repeats?*

hw4 pr2

```
def compress( I ):
    """ returns the RLE of the
        input binary image, I """
```

a binary image, **I**

"000000000000000000000000000000000000000000111111111111111111111111111111"

42 zeros

31 ones

"0010101010011111"

the bit	42, in binary # of repeats	the bit	31, in binary # of repeats

the "compressed" output returned by **compress(I)**

hw4 pr2

```
def compress( I ):
    """ returns the RLE of the
        input binary image, I """
```

a binary image, **I**

"000000000000000000000000000000000000000000111111111111111111111111111111"

42 zeros

31 ones

"0010101010011111"

the bit	42, in binary # of repeats	the bit	31, in binary # of repeats
0	101010100	1	0011111

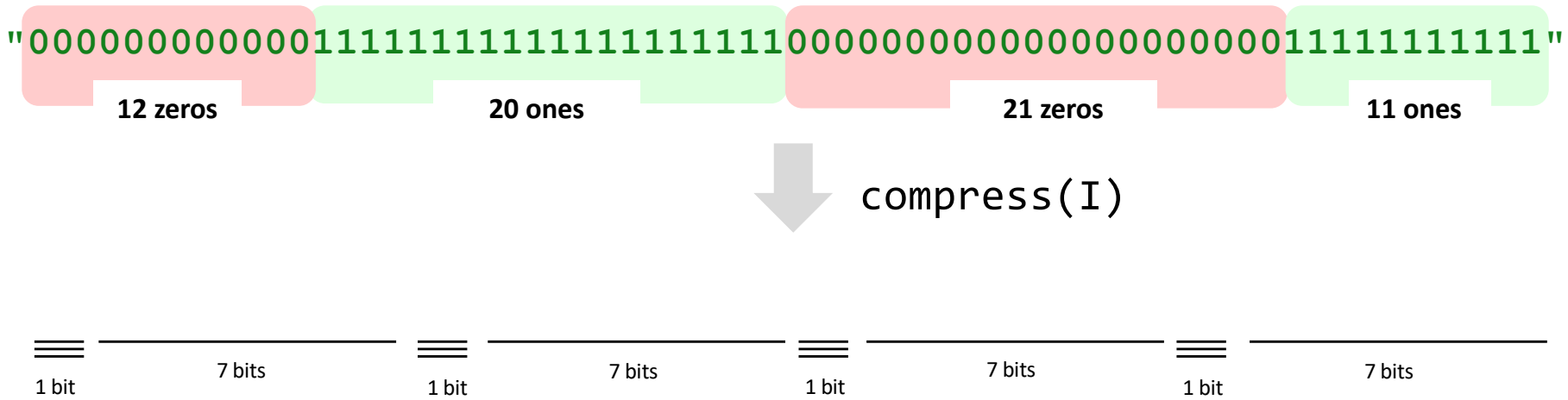
the "compressed" output returned by **compress(I)**

```
def compress( I ):
    """ returns the RLE of the
        input binary image, I """
```

Keep this for hw4pr2...

a 64-bit binary image, **I**

IQuiz



try creating the *compressed* output, as would be returned by **compress(I)**

Then, discuss

(1) What's an image **I** whose compressed output ***gets larger, not smaller***?

(2a) What is the BEST-compressible / the WORST-compressible 64-bit images?

(2b) How can we ***improve*** the algorithm so that it always compresses?!!

(3) What helper function might be useful here?

hw4 pr2

```
def compress( I ):
    """ returns the RLE of the
        input binary image, I """
```

most?
least?

a binary image, **IQuiz**

"000000000000111111111111111111110000000000000000000011111111111"

12 zeros | 20 ones | 21 zeros | 11 ones

pad | pad | pad | pad

"0000110010010100001010110001011"

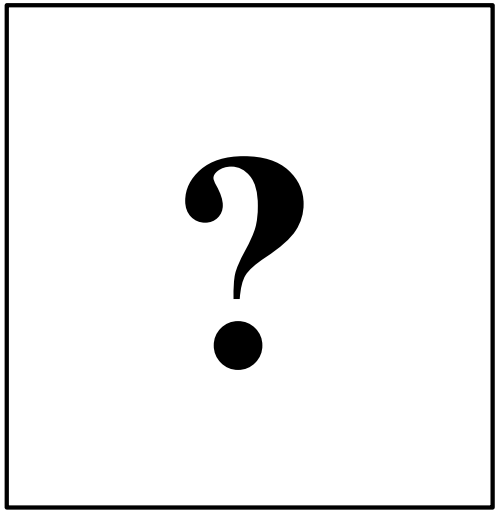
pixel | 12, in binary | pixel | 20, in binary | pixel | 21, in binary | pixel | 11, in binary

the "compressed" output, returned by compress(I)

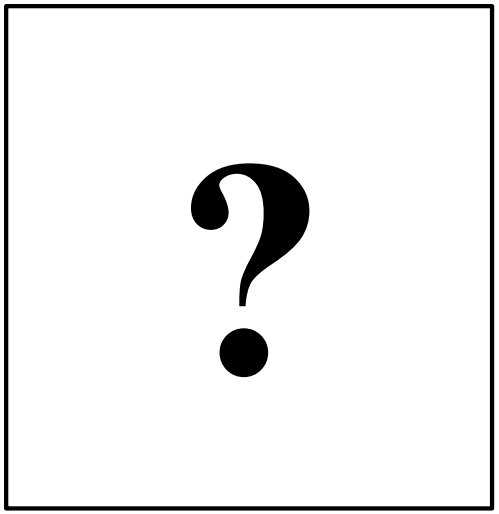
shortest compressed representation

longest compressed representation

What are the **BEST** and the **WORST** compression results you can get for an 8x8 image input (64 bits)?



BEST



WORST

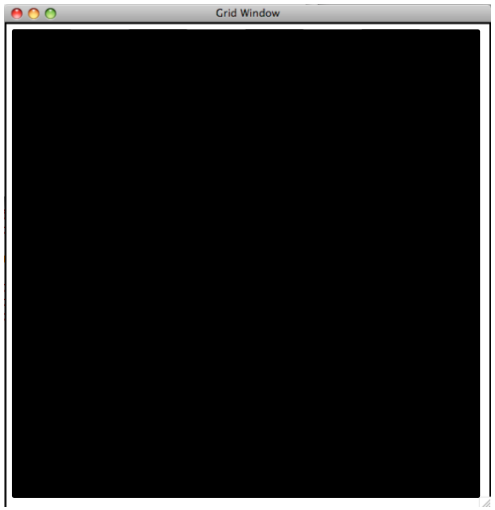
How could we improve this compression algorithm so that ***all images*** compress to smaller than the originals? That is, how can we make compression always <u>work</u> ?

?

shortest compressed representation

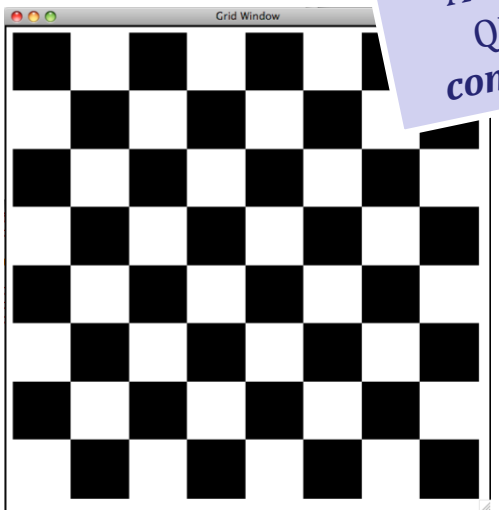
longest compressed representation

What are the **BEST** and the **WORST** compression results you can get for an 8x8 image input (64 bits)?



BEST

only 8 bits total!



Actually, this is NOT QUITE the ***worst-compressable*** image?

WORST

aargh!

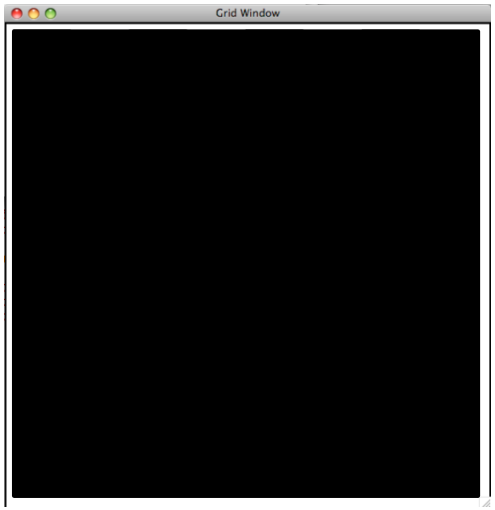
How could we improve this compression algorithm so that ***all images*** compress to smaller than the originals? That is, how can we make compression always <u>work</u> ?

?

shortest compressed representation

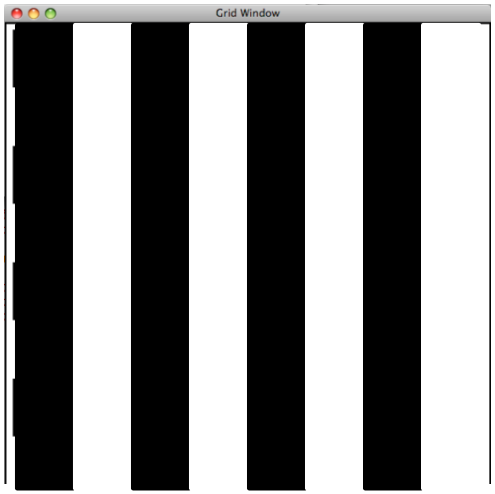
longest compressed representation

What are the **BEST** and the **WORST** compression results you can get for an 8x8 image input (64 bits)?



BEST

only 8 bits total!



WORST

aargh! 512 bits!

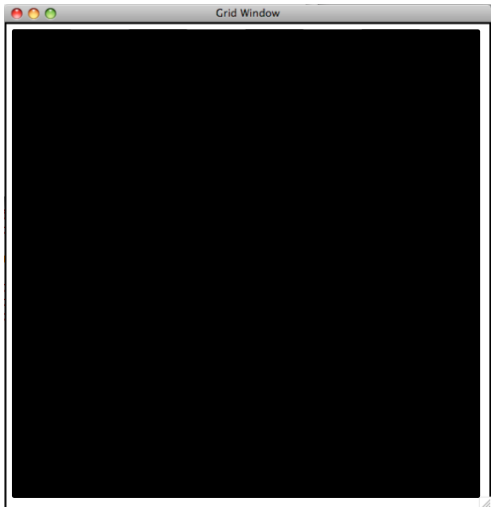
How could we improve this compression algorithm so that ***all images*** compress to smaller than the originals? That is, how can we make compression always <u>work</u> ?

?

shortest compressed representation

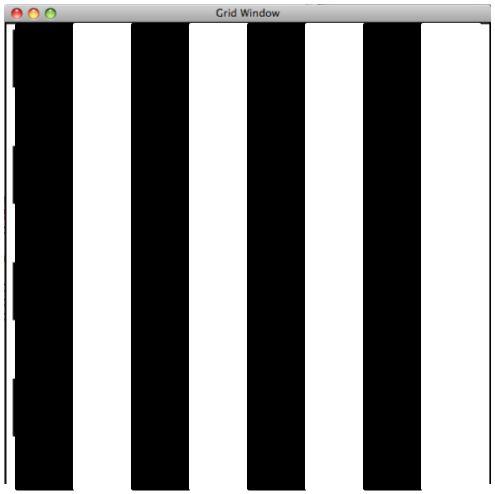
longest compressed representation

What are the **BEST** and the **WORST** compression results you can get for an 8x8 image input (64 bits)?



BEST

only 8 bits total!



WORST

aargh! 512 bits!

Impossible! *Provably so...!*

How could we improve this compression algorithm so that ***all images*** compress to smaller than the originals? That is, how can we make compression always <u>work</u> ?

!

Helper fun!

`frontNum(S)` returns the # of times the first element of the input **S** appears consecutively ***at the start*** of S:

```
                 S
frontNum('1111010')
4
                 S
frontNum('00110010')
2
                 S
frontNum('111010000')
3
```

```
def frontNum(S):

    if len(S) <= 1:
        return len(S)

    elif                    :
        return 1+frontNum(    )

    else:
        return
```

1 base case:

or 2 base cases:

```
len(S)== 0:
len(S)== 1:
```

or 3:

```
S== '' or
S== '1' or
S== '0'
```

If the first two bits **DO** match....

What should this be, if they ***do*** match?

If the first two bits ***DON'T*** match....

What should this be, if they ***don't*** match?

It's <u>all</u> bits!

images, text, sounds, data, ...

even the string `'Hi *'` is just a sequence of bits...

`'Hi *'`

4 ASCII characters
8 bits each

4*8 == 72 bits total

pad

`01001000 01101001 00100000 00101010`

`ord('H') == 72` `ord('i') == 105` `ord(' ') == 32` `ord('*') == 42`

All computation boils down to manipulating bits!

All computation is simply ***functions of bits***

binary inputs **S** and **T**		output, **S+T**
S	**T**	
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

This week: you'll build this in Python.

bitwise addition function

addB

Next week: you'll design this with *wires*!

```
if S[-1] == '0' and T[-1] == '0':
    return _____________________________ + '0'
```

binary inputs **S** and **T**

S	T	output, S+T
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

This week: you'll build this in Python.

bitwise addition function

addB

```
if S[-1] == '1' and T[-1] == '1' :
    return _____________________________ + '0'
```

binary inputs **S** and **T**

S	T	output, S+T
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

This week: you'll build this in Python.

bitwise addition function

addB

Adding strings?

is **circuit** *addition!*

is **syntactic** *addition!*

syntactic ~ *"meaning-free"*

Multiplying by machine:

is **circuit** *multiplying!*

is **syntactic** *multiplying!*

Doing anything by machine…

is **circuit** *interaction!*

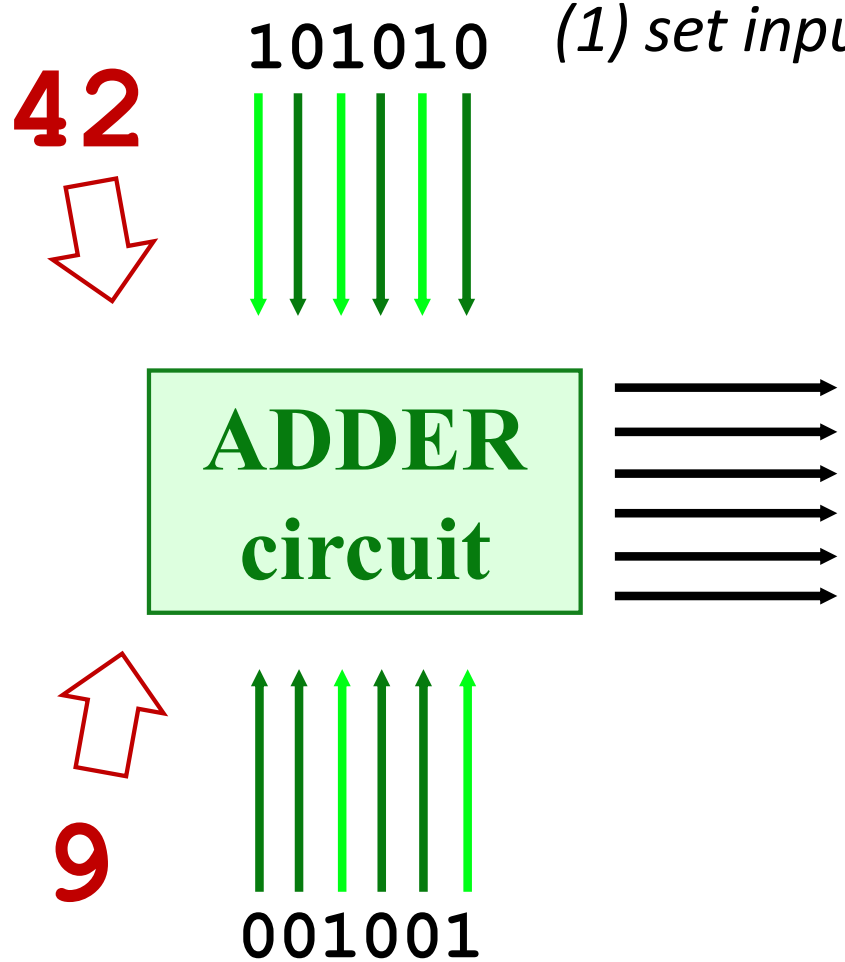
is **syntactic** *interaction!*

means it can be done purely via ***surface syntax****, which means it can be done* ***without thinking****…*

In a computer, each bit is represented as a <u>voltage</u> (**1** is +5v and **0** is 0v)

Computation is simply the **deliberate combination** of those voltages!

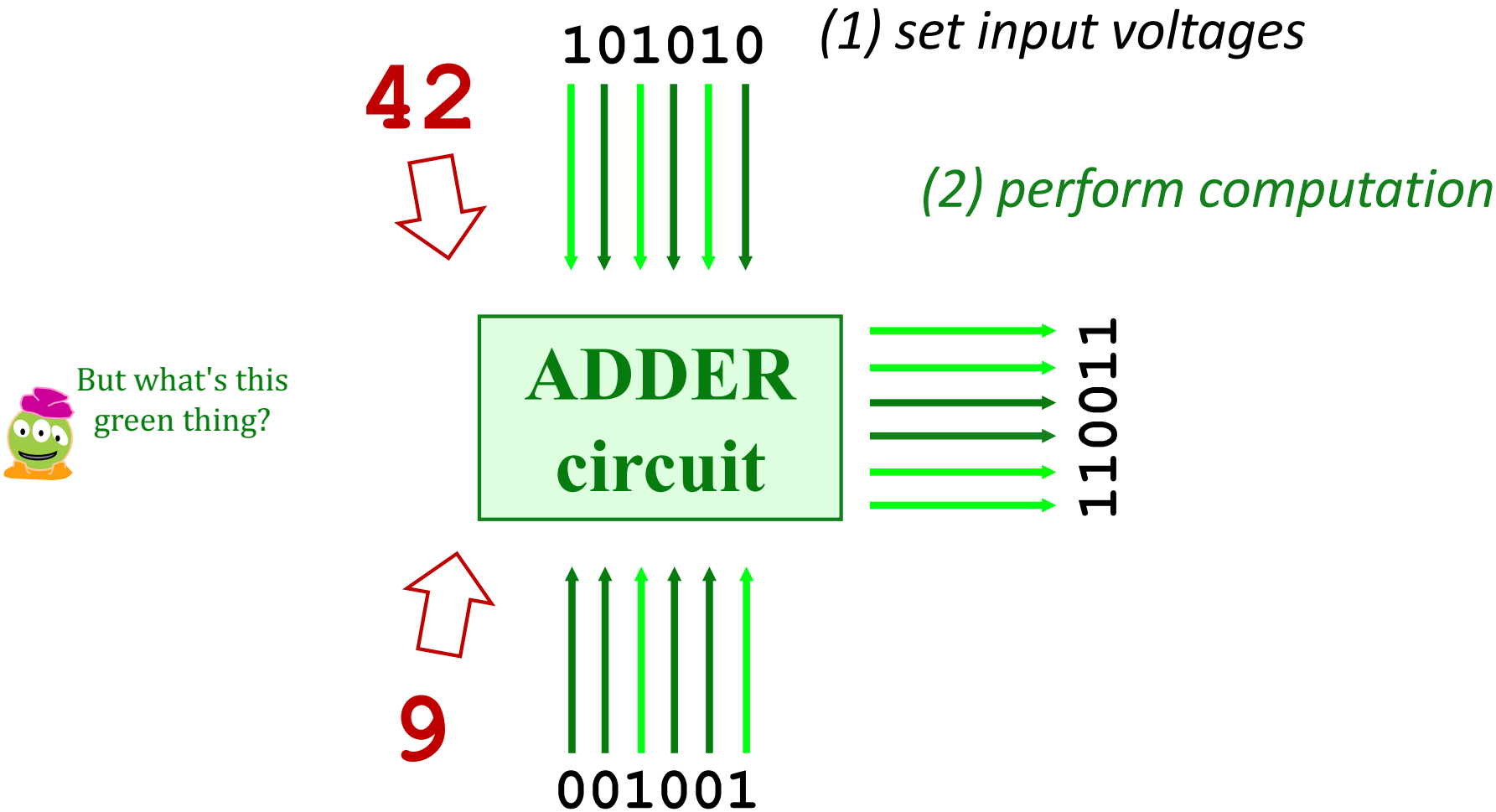
(1) set input voltages



But what's this green thing?

In a computer, each bit is represented as a <u>voltage</u> (**1** is +5v and **0** is 0v)

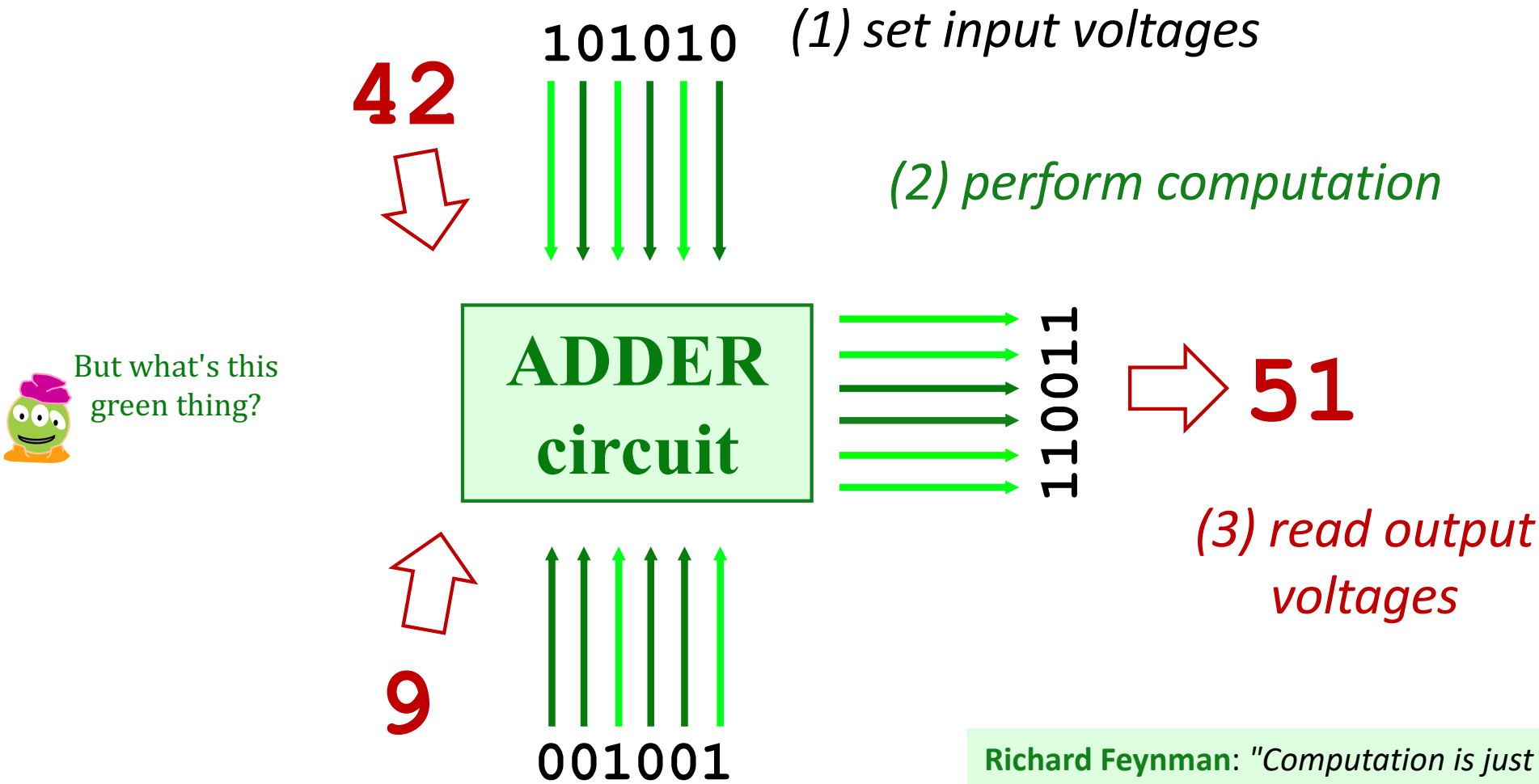
Computation is simply the **deliberate combination** of those voltages!



But what's this green thing?

In a computer, each bit is represented as a <u>voltage</u> (**1** is +5v and **0** is 0v)

Computation is simply the **deliberate combination** of those voltages!



But what's this green thing?

Richard Feynman: *"Computation is just a physics experiment that always works!"*

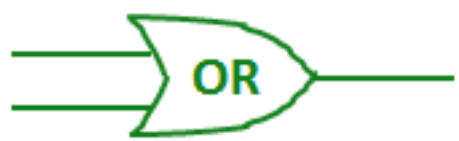
Our building blocks: ***logic gates***

AND outputs 1 only if **ALL** inputs are 1

AND

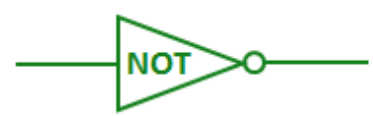
OR outputs 1 if **ANY** input is 1

OR



NOT reverses its input

NOT



These circuits are ***physical*** functions of bits…

… and *all* mathematical functions can be built from them!

Our building blocks: ***logic gates***

AND outputs 1 only if **ALL** inputs are 1

AND

OR outputs 1 if **ANY** input is 1

OR

NOT reverses its input

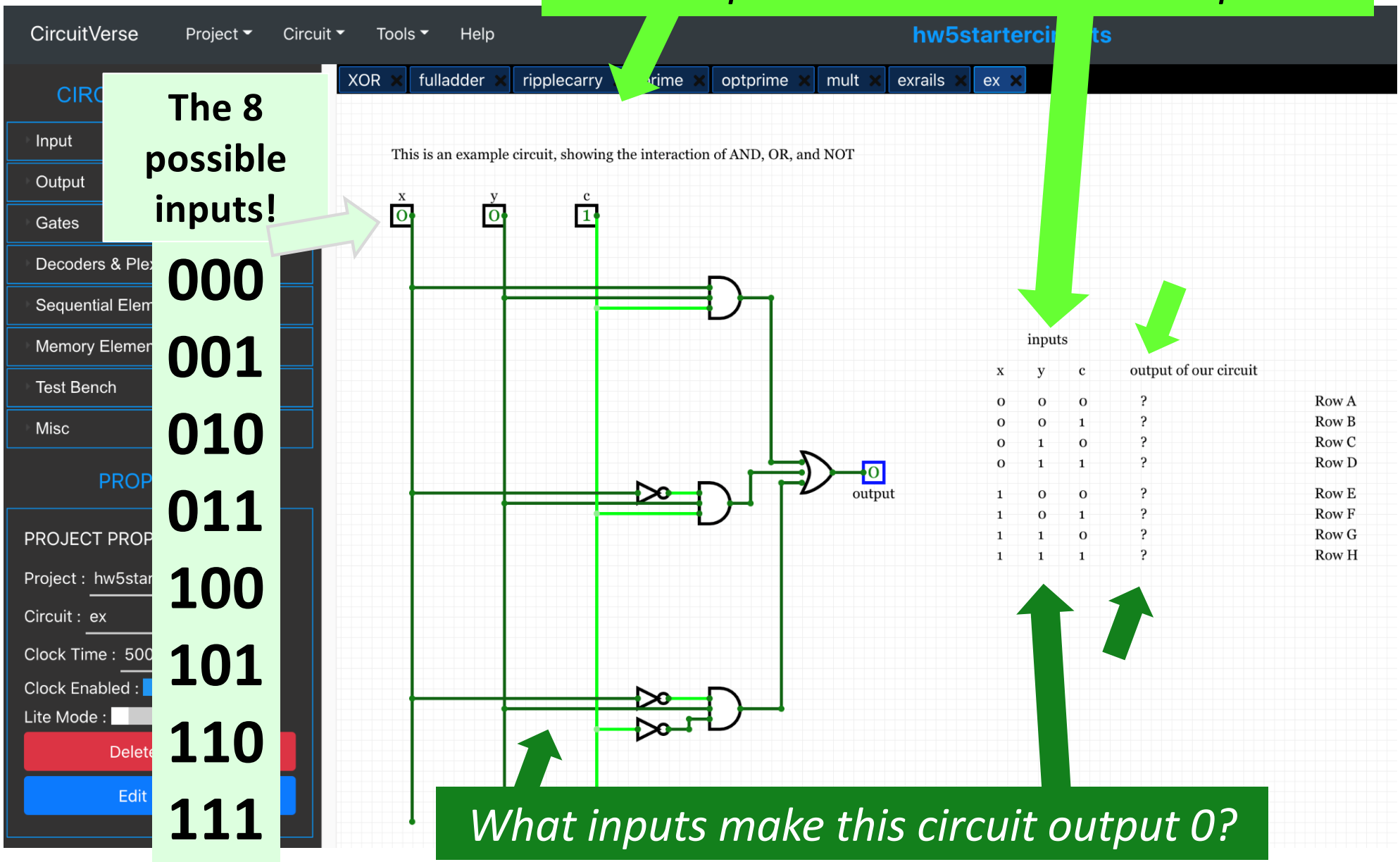
NOT

ALL circuits are **ANY** tions of bits…

… and *all* mathematical functions can be built from them!

From gates to ***circuits***...

What inputs make this circuit output 1?



The 8 possible inputs!

000
001
010
011
100
101
110
111

inputs

x	y	c	output of our circuit	
0	0	0	?	Row A
0	0	1	?	Row B
0	1	0	?	Row C
0	1	1	?	Row D
1	0	0	?	Row E
1	0	1	?	Row F
1	1	0	?	Row G
1	1	1	?	Row H

What inputs make this circuit output 0?

from circuit design…

next 2 weeks

…to a full computer!

Have an outst ingly

gl ious weekend!

Why ?!

AND

OR

NOT

Yo

HARVEY MUDD COLLEGE

Resources for Students Faculty and Staff Alumni Parents Mudd A–Z

Admission and Financial Aid Academics Clinic Research Student Life About Giving

/ Academics / Hixon Center for Climate and the Environment

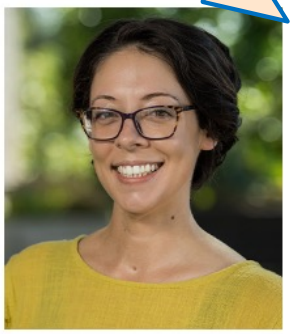
Hixon Center Menu

Hixon Center for Climate and the Environment

n

e

black!



Sarah Kavassalis

Assistant Professor of Climate and Chemistry

Olin 1257A

skavassalis@g.hmc.edu

Prof. Kavassalis!

Prof. Medero!