More **bits** of CS

Too many bits? Compress!

Below binary: **physical circuits**

**Circuit design, part 1**

I'd call this a **KNOT gate...**

This circuit was **NOT,** in fact, designed!

<table>
<thead>
<tr>
<th>Jotto Corner</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Hw #5 due Mon. 4/25**

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

pr1 *(lab)* binary ~ decimal

pr2 conversion + compression

extra image processing...

Lots of tutoring hrs - join in... !

**Lab Debriefing & hw5pr2.py**

```python
# This is circuit addition!

def numToBin(N):
    """ converts a decimal int to a binary string """
    if N==0: return ''
    elif N%2==0: return numToBin(N//2) + '0'
    elif N%2==1: return numToBin(N//2) + '1'

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

't101010' out

# This is syntactic addition!

def add10(S,T):
    """ adds the *strings* S and T as decimal numbers """
    S '31'
    T '11'

def add10(S,T):
    """ adds the *strings* S and T as decimal numbers """
    S '31'
    T '11'

# This is circuit addition!

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

- All computation is simply functions of bits.

### Adding strings?
- Adding strings? is circuit addition! syntactic ~ meaning-free
- Multiplying by machine:
  - Multiplying by machine is circuit multiplying! syntactic multiplying!

### Doing anything by machine...
- Doing anything by machine... is circuit interaction! syntactic interaction!

- Means it can be done purely via surface syntax, which means it can be done without thinking...

### Ariane 5
- Ariane 5
- This week’s reading: bits can be vital

**IndexError**  **TypeError**  **HumanError**

- IndexError
- TypeError
- HumanError

### How high can we count...?
- How high can we count...?

<table>
<thead>
<tr>
<th>N bits</th>
<th>1 bit</th>
<th>2 bits</th>
<th>3 bits</th>
<th>4 bits</th>
<th>7 bits</th>
<th>8 bits</th>
<th>31 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>111</td>
<td>1111</td>
<td>111111</td>
<td>1111111</td>
<td>255</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- I can see some patterns here – even with one eye closed!
How high can we count... in 2015?

List of most viewed YouTube videos

Top videos

<table>
<thead>
<tr>
<th>Rank</th>
<th>Video name</th>
<th>uploader / artist</th>
<th>views</th>
<th>upload date</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Baby</td>
<td>Justin Bieber</td>
<td>1,216,729,055</td>
<td>February 19, 2010</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Blank Space</td>
<td>Taylor Swift</td>
<td>1,173,560,710</td>
<td>November 10, 2014</td>
<td></td>
</tr>
</tbody>
</table>

Counting sheep, xkcd style...

How many bits?

Hw5: images are just bits, too! hw5pr3 (extra)

old pixel at 42,42 has
red = 1 (out of 255)
green = 36 (out of 255)
blue = 117 (out of 255)

new pixel at 42,42 has

how many bits represent each color channel?

guesses as to what this transformation was?

Hw5: images are just bits, too!

Encoding as raw bits

10101010
01010101
10101010
01010101
01010101
01010101
01010101
01010101
to 1010101010110110101011010110101010110101010101010101010101010101

especially binary images
likely *compressible* image...

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

**image compression is everywhere!**

Hw5: *lossless* binary image compression

**Binary Image**

**Encoding as raw bits**

- 00000000
- 00000000
- 11111111
- 11111111
- 00000000
- 00000000
- 00000000
- 00000000
- 00000000
- 00000000

- 0 is the first digit
- and there are 1,098,188 of them.

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

- 0100001100000111001100

- our algorithm:
  - bit #repeats

- could be misinterpreted!
fixed-width compression

We need fixed-width blocks:

<table>
<thead>
<tr>
<th>bit</th>
<th>#repeats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7 bits: # of repeats</td>
</tr>
<tr>
<td>8</td>
<td>8-bit total data block</td>
</tr>
</tbody>
</table>

8-bit data block 8-bit data block 8-bit data block

000100001001000010010001001000011010010100111001000010010000100

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

7 bits?

B bits?

What if you need a larger # of repeats?

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

    a binary image, I

    "00101010100111111"

    the "compressed" output returned by compress(I)

Use this!

Use this!

def frontNum(S):
    if len(S) <= 1:
        return len(S)
    elif len(S) == 0:
        return 0

    if S == "0" or S == '0':
        if the first two bits DO match...
    else:
        if the first two bits DON'T match...
What are the **BEST** and the **WORST** compression results you can get for an 8x8 image input (64 bits)?

**BEST**

![Best image](image1)

**WORST**

![Worst image](image2)

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 *work*?

Impossible! Provable!

All computation boils down to manipulating bits!
In a computer, each bit is represented as a voltage (1 is +5v and 0 is 0v). Computation is simply the deliberate combination of those voltages!

101010 (1) set input voltages

But what's this green thing?

Our building blocks: logic gates

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputs 1 only if ALL inputs are 1</td>
<td>outputs 1 if ANY input is 1</td>
<td>reverses its input</td>
</tr>
</tbody>
</table>

AND

OR

NOT

These circuits are physical functions of bits...

... and all mathematical functions can be built from them!

From gates to circuits...

What inputs make this circuit output 1?

What inputs make this circuit output 0?

from circuit design...

next 2 weeks... to a full computer!
def compress(I):
    """ returns the RLE of the input binary image, I """

    a 64-bit binary image, I

    "00000000000000111111111111111111111111111111111111111100000000000000000000000001111111111"

    12 zeros  20 ones  21 zeros  11 ones

    compress(I)

    the "compressed" output returned by compress(I)

    Then, discuss ...

    What helper function would be useful for compress?

    What's an image I whose compressed output gets larger, not smaller? (Aargh!)
    • What are the BEST-compressible / WORST-compressible 64-bit images?
    • How could you improve the algorithm so that it always compresses?!!