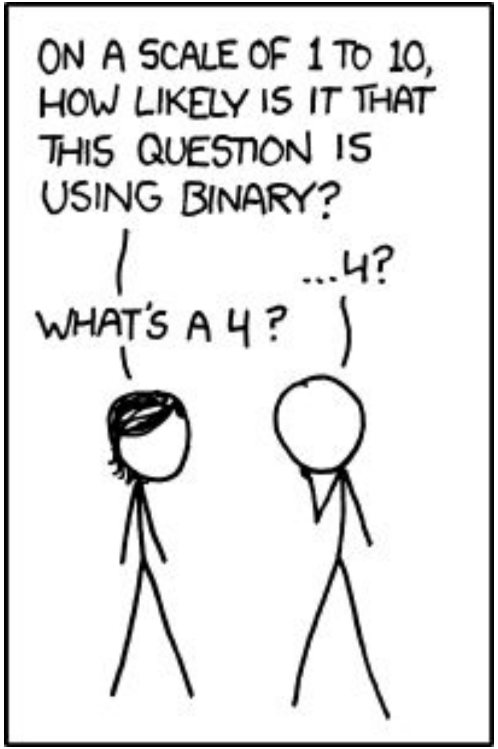
Lecture 2: Binary and Python



Announcements and reminders

- Enroll in the class Moodle -- enrollment keys are:

  csci2824-Tony or *csci2824-Rachel*

- Make sure you can access Piazza -- https://piazza.com/colorado/fall2018/csci2824

Warm-up problem

Example: Express the decimal number 30 in binary.

Example: Express the binary number 1011 in decimal.

Representing fractions as binary

We can!

- First bit right of the “radix point” represents ½.
- Second bit on the right represents ¼.
- Third bit represents ⅛.
- … and so on.

Representing fractions as binary

We can!

- First bit right of the "radix point" represents ½.
- Second bit on the right represents ¼.
- Third bit represents ⅛.
- … and so on.

Example: Convert 0.75 from decimal to binary.

Representing fractions as binary

We can!

- First bit right of the “radix point” represents ½.
- Second bit on the right represents ¼.
- Third bit represents ⅛.
- … and so on.

Example: Convert 0.75 from decimal to binary.

$0.75 = \frac{3}{4} = \frac{1}{2} + \frac{1}{4}$

$0.75_{10} = 0.11_2$

Representing fractions as binary

A tougher example: Convert 0.875 from decimal to binary.

- We need a better algorithm, like for the big numbers

Representing fractions as binary

A tougher example: Convert 0.875 from decimal to binary.

- We need a better algorithm, like for the big numbers
  - Multiply N by 2. Next bit is the digit in the ones place.
  - Proceed with new N as the part to the right of the decimal point.
  - Continue until you have 0 remaining.

Representing fractions as binary

A tougher example: Convert 0.875 from decimal to binary.

- We need a better algorithm, like for the big numbers
  - Multiply N by 2. Next bit is the digit in the ones place.
  - Proceed with new N as the part to the right of the decimal point.
  - Continue until you have 0 remaining.
- **Quick check**: does this work for $0.75_{10} = 0.11_2$?

Representing fractions as binary

A tougher example: Convert 0.875 from decimal to binary.

- We need a better algorithm, like for the big numbers:
  - Multiply N by 2. Next bit is the digit in the ones place.
  - Proceed with new N as the part to the right of the decimal point.
  - Continue until you have 0 remaining.
- 0.875 * 2 = 1.75, so first bit right of the radix point is a 1.
  - Continue with 0.75.
- 0.75 * 2 = 1.5, so next bit is a 1.
  - Continue with 0.5.
- 0.5 * 2 = 1.0, so next bit is a 1.
  - Continue with 0. → Exit → Left with $0.875_{10} = 0.111_2$

Representing fractions as binary

Example: Convert 123.321 from decimal to binary.

- Do the part left of the decimal first.

Representing fractions as binary

Example: Convert 123.321 from decimal to binary.

- Do the part left of the decimal first.
  - 123 → Odd, so first bit is a 1. Proceed with (123-1)/2 = 61.

Representing fractions as binary

Example: Convert 123.321 from decimal to binary.

- Do the part left of the decimal first.
  - 123 → Odd, so first bit is a 1. Proceed with (123-1)/2 = 61.
  - 61 → Odd, so next bit is a 1. Proceed with (61-1)/2 = 30.
  - 30 → Even, so next bit is 0. Proceed with 30/2 = 15.
  - 15 → Odd, so next bit is a 1. Proceed with (15-1)/2 = 7.
  - 7 → Odd, so next bit is a 1. Proceed with (7-1)/2 = 3.
  - 3 → Odd, so next bit is a 1. Proceed with (3-1)/2 = 1.
  - 1 is a 1, so last bit is a 1.
  - Put it all together: $123_{10} = 1111011_2$

Representing fractions as binary

Example: Convert 123.321 from decimal to binary.

- Do the part right of the decimal now.
  - 0.321 * 2 = 0.642 → first bit is a 0

Representing fractions as binary

Example: Convert 123.321 from decimal to binary.

- Do the part right of the decimal now.
  - 0.321 * 2 = 0.642 → first bit is a 0
  - 0.642 * 2 = 1.284 → next bit is a 1
  - 0.284 * 2 = 0.568 → next bit is a 0
  - 0.568 * 2 = 1.136 → next bit is a 1
  - 0.136 * 2 = 0.272 → next bit is a 0
  - 0.272 * 2 = 0.544 → next bit is a 0
  - 0.544 * 2 = 1.088 → next bit is a 1
  - … eventually find $0.321_{10} = 0.01010010001_2$
  - … so $\mathbf{123.321_{10} = 1111011.01010010001_2}$

Example: How many bits are needed to encode each lowercase letter of the English alphabet?

Example: How many bits are needed to encode each lowercase letter of the English alphabet?

Solution: There are 26 letters, so we could let A=0, B=1, C=2, … , Z=25.

→ Need as many bits as in the binary representation of Z=25

→ $25_{10} = ?_2$

25 is odd, so first bit is 1; continue with (25-1)/2 = 12

12 is even, so second bit is 0; continue with 12/2 = 6

6 is even, so third bit is 0; continue with 6/2 = 3

3 is odd, so fourth bit is 1; continue with (3-1)/2 = 1

1 is a 1, so last bit is 1 and STOP

→ $25_{10} = 11001_2$

→ **5 bits** are needed to represent all lowercase letters

Binary arithmetic

Example: What is 30 in binary? What is 2 in binary? What is 32 in binary?

Binary arithmetic

Example: What is 30 in binary? What is 2 in binary? What is 32 in binary?

Answer: $30_{10} = 11110_2$ $2_{10} = 10_2$ and $32_{10} = 100000_2$

Adding two numbers in binary: we proceed from right to left just like with adding in decimal.

Binary arithmetic

Example: What is 30 in binary? What is 2 in binary? What is 32 in binary?

Answer: $30_{10} = 11110_2$ $2_{10} = 10_2$ and $32_{10} = 100000_2$

Adding two numbers in binary: we proceed from right to left just like with adding in decimal.

- Decimal: if our column exceeds 10, we **carry a 1 to the left**.
- Binary: if our column exceeds 2, we carry a 1 to the left:

Binary arithmetic

Example: What is 11101_2 - 110_2 ?

Binary arithmetic

Example: What is $11101_2 - 110_2$?

Subtracting numbers in binary: we proceed from right to left just like with decimal.

Binary arithmetic

Example: What is $11101_2 - 110_2$?

Subtracting numbers in binary: we proceed from right to left just like with decimal.

- Decimal: we can take 1 from the column to the left, and bring **10 to the right**
- Binary: we can take 1 from the column to the left, and bring **2 to the right**

Python primer

- **Easiest:** download Anaconda -- https://www.anaconda.com/download
- Lots of tutorials are available...
- … but the best way to get practice is by **doing things:**
  https://www.hackerrank.com/domains/python
- **Note:** we will be using **Python 3** (there are some subtle differences!)

Binary arithmetic and Python primer

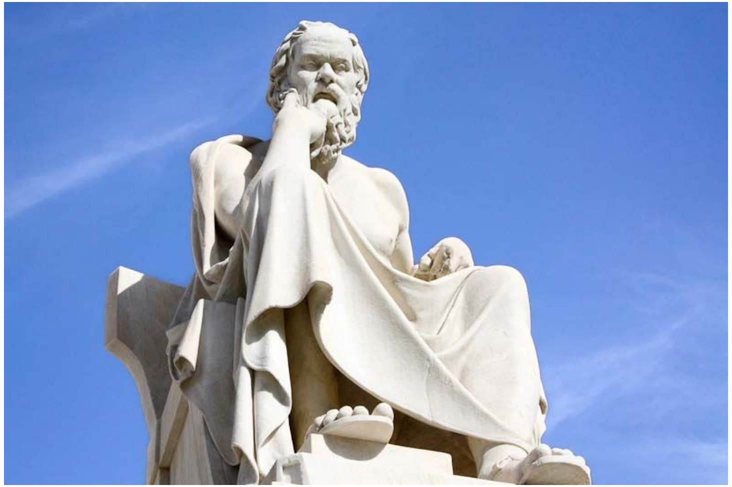
Recap:

Today, we…

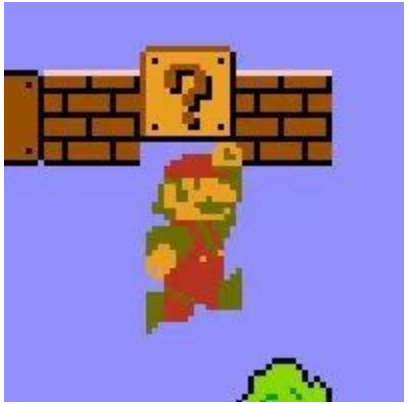
- Learned how to represent fractions in binary
- Learned how to add and subtract numbers in binary
- Messed around in Python

Next time:

- We **think**. Like, ***really* hard**.
  (**logic!**)



Bonus material!



Representing fractions as binary

- **Example:** Convert 0.1 from decimal to binary.

Representing fractions as binary

- **Example:** Convert 0.1 from decimal to binary.
  - 0.1 * 2 = 0.2, so first bit is a 0
  - 0.2 * 2 = 0.4, so next bit is a 0
  - 0.4 * 2 = 0.8, so next bit is a 0
  - 0.8 * 2 = 1.6, so next bit is a 1
  - 0.6 * 2 = 1.2, so next bit is a 1
  - 0.2 * … NOW HOLD ON A SECOND!
    - Restarts the pattern from *here*.

Representing fractions as binary

- **Example:** Convert 0.1 from decimal to binary.
    - 0.1 * 2 = 0.2, so first bit is a 0
    - 0.2 * 2 = 0.4, so next bit is a 0
    - 0.4 * 2 = 0.8, so next bit is a 0
    - 0.8 * 2 = 1.6, so next bit is a 1
    - 0.6 * 2 = 1.2, so next bit is a 1
    - 0.2 * … NOW HOLD ON A SECOND!
        - Restarts the pattern from *here*.
        - So $0.1_{10} = 0.00011001100110011..._2$

Representing fractions as binary

- **Example:** Convert 0.1 from decimal to binary.
  - So $0.1_{10} = 0.00011001100110011..._2$
  - So computers would need to store an infinite number of bits in order to store 0.1_{10} exactly.
  - … obviously, computers can't do that.
  - They truncate at a certain point, leading to some error.

Representing fractions as binary

- **Example:** Convert 0.1 from decimal to binary.
  - So $0.1_{10} = 0.00011001100110011..._{2}$
  - So computers would need to store an infinite number of bits in order to store 0.1_{10} exactly.
  - … obviously, computers can’t do that.
  - They truncate at a certain point, leading to some error.
  - **How bad can this error be?**

Representing fractions as binary

- **How bad can this error be?**
- Consider the function

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + a/2b$$

where a=77617 and b=33096.

Representing fractions as binary

- **How bad can this error be?**

- Consider the function

  $$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + a/2b$$

  where a=77617 and b=33096.

- If you run this on a 64-bit machine, you'll find
  $f(a,b)$ = -44450695952321879337122922496.000000, or maybe
  $f(a,b)$ = -1.180592e+21
  (depending on the way the program you code in truncates)
- But the true answer is more like
  $f(a,b)$ = -0.82739605...