Reminders

- Project plan

  due Tuesday Oct 30, 9 AM

  Presentations: 10 min, Oct 30 and Nov 1, in class

- Next Tuesday: no lecture

  Use time to work on project

- Next Thursday: Oct 25, guest lecture

Denver Airport Baggage System

- Billed as the most advanced system in the world.
- Goal: automate baggage handling through the entire airport.
- Building the system resulted in the newly completed airport sitting idle for 16 months while engineers worked on getting the baggage system to work.
  - delay added half a billion dollars to the cost

Denver Airport Baggage System

- Final system far smaller than original design.
- Does not integrate concourses.
- Supports outbound flights only on one concourse.
- Everything else done manually, with hand tagging.

Denver Airport Baggage System

- The one-concourse part that was kept:

  scrapped soon after because of $1 million monthly maintenance cost

http://calleam.com/WTPF/?page_id=2086

Los Angeles Air-Traffic Control System

- Voice communication system shut down unexpectedly
- 400 airplanes lost contact
- Several were headed toward one another
- Pilots had to self-direct
- We got lucky
- A counter in the voice system counted down from 2^{32} ms (~50 days). Maintenance routinely rebooted the system once per month.

http://spectrum.ieee.org/aerospace/aviation/lost-radio-contact-leaves-pilots-on-their-own

London Ambulance Service

- Computer-assisted dispatch system automated deployment of ambulances
- Resulted in up to 11 hour waits
- 30 people died
- Shut down 2 days later
- Reason: system got out of sync with reality

- Tried again in 2011 and 2012 with a brand new $29million system
  - ~50 minute delays caused reverting to manual system

http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/las/lascase0.9.pdf

Why do these failures matter?

- Design, requirements, specification matter.
- But in the end, many of errors in those phases can be caught during testing.
- Catching errors in testing has a high cost, but much lower than in deployment.

Questions?

Computability

How many apples?

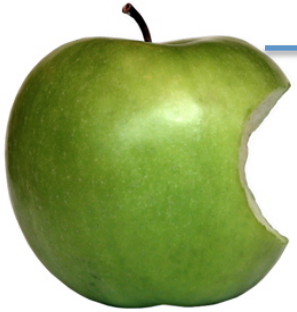
Can you answer without counting?

What does “5 apples” mean?

Let's try an easier question: Same number of apples as cats?

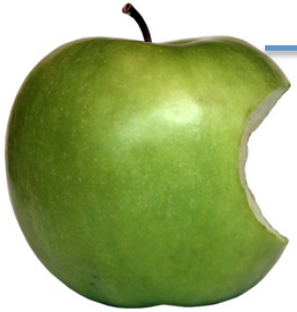
No counting!

Same number of apples as cats? How do you know?



No counting!

What about now?



?

Let's define "equal-size sets"

- Two sets are equal-sized if there exists a 1-to-1 mapping of elements of one set to the elements of the other set
  - No element maps to 2 or more elements
  - No element maps to 0 elements
  - There may be other mappings, but we only care if one good one exists

Why is Yuriy asking these stupid questions?

- Comparing sizes of sets turns out to be very important in computing
- How many different programs can we write?
- How many different problems exist?

- We have to tackle infinite sets

Some infinite sets

- How many integers are there?
- How many even numbers are there?
- How can we compare two infinite sets?

Are there the same number of integers and odd numbers?

Let’s forget about non-positive numbers for now

Integers and Odds

Integers	Odds
1	1
2	3
3	5
4	7
5	9
6	11
7	13
8	15
n	2n-1

What about other sets?

- Integers and primes?
- Evens and odds?
- Perfect squares and perfect cubes?

Are all infinite sets the same size?

What about integers and rationals?

Are all infinite sets the same size?

What about integers and reals?

How many reals are there?

- Let's say you come up an ordered list of all real numbers

  (or a program that on input i prints a unique real number)

- I will now, write down on the board, a real number that's not on your list

  (or that your program never prints)

Sizes of infinity:

- Countably infinite: $\aleph_0$

  integers, naturals, rationals, evens, odds, primes, etc.

- Uncountably infinite: $2^{\aleph_0}$

  irrationals, reals, reals between 0 and 1

There is an infinite number of different infinities!

Let’s get back to programming

How many different computational problems are there?

- Let's come up with a simple definition of a problem:

  A subset of the integers

- So a problem may be to compute:
  - integers
  - odds
  - evens
  - primes

- Clearly this is a subset of all possible problems, so we are underestimating the total number

How many different computational problems are there?

same answer as:

How many different subsets of integers are there?

Counting sets of integers

- Let's set we are looking at subsets of {1,2,3,4,5}. How many subsets are there?

- 2 options: 1 is in the set or it is not in the set
- 2 options: 2 is in the set or it is not in the set
- …
- So $2^5 = 32$

So what about subsets of all integers?

How many different programs are there?

Let's come up with a simple definition of a program that would solve our type of problem:

- Takes as input a single integer (no limit)
- Does whatever computation it wants

  (let's say Java)
- Outputs "yes" or "no"

Then a program can be characterized by the set of integers it says "yes" to.
In other words, the problem it solves.

How many programs are there?

- Each program can be compiled down to a binary.
- A binary is just a series of bits. 01010110100101010111010
- How many different binaries are there?

not all binaries fit our form of a program, so we are overestimating the number of programs.

How many programs are there?

same answer as:

How many different binary numbers are there?

Mismatch!

- There are $\aleph_0$ (or fewer) different programs we can write.
- There are (at least) $2^{\aleph_0}$ different problems we can attempt to solve.

So there must be lots of problems for which we cannot write programs!

The overwhelming majority of problems are undecidable

- Undecidable means no computer program can solve this problem.
- Example:

Consider “deciding” if a program ever halts.

Halting problem

- Let's say you wrote a program P that takes as input a program Q and prints 1 if Q ever halts, and 0 if Q gets stuck in an infinite loop.

- Let me write a new program R:

```
Input: program Q
   simulate P on Q
   if P prints 0, halt
   if P prints 1, loop forever
```

What is P(R)?

Next class is Thursday

- Good luck on your projects!