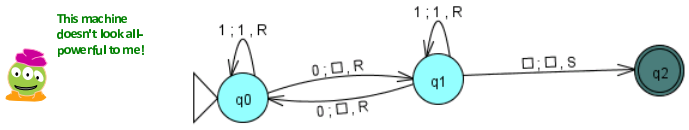
CS 5 today: **More machines!**

Final ideas... Turing Machines and the **MANY** things computers can't compute... !



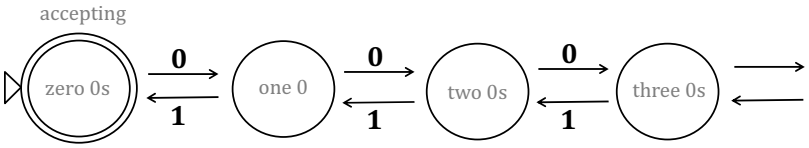
Final projects...

- Final tutoring hours & labs coming up
- HW12 & milestone due on Monday, 4/26
- 5 finite state machines due as part of HW 12

Turing machines extra!

State-machine ***limits***?

You don't need three eyes to see some problems here!



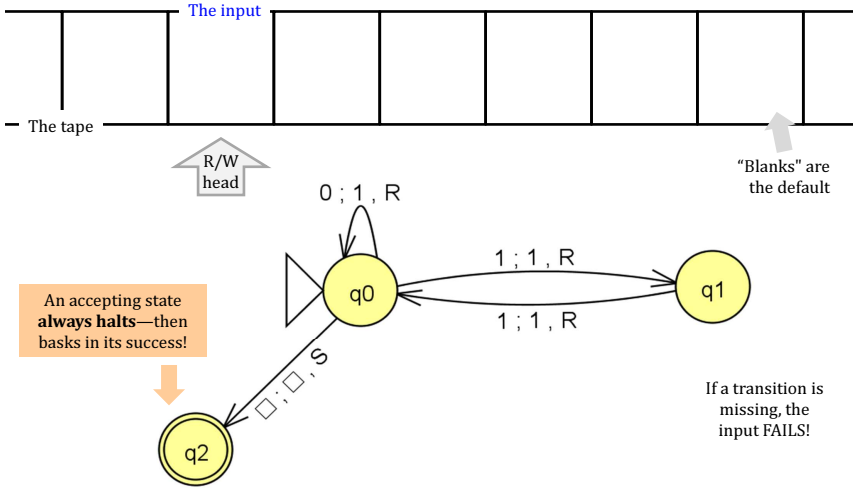
Let's build an FSM that accepts strings with ***any*** **# of 0s** followed by the ***same*** **# of 1s**

rejected
011
001
11100
00110

FSMs "can't count"

At least, not arbitrarily high

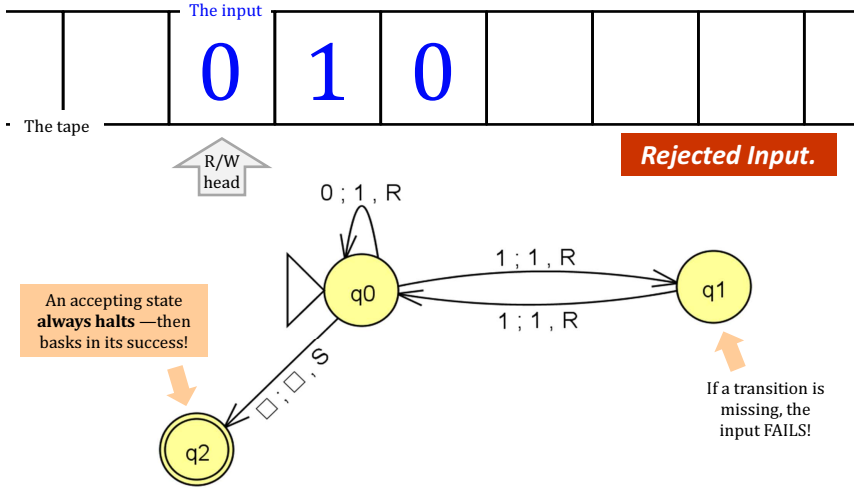
accepted
000111
0011
01
λ



A Turing Machine rule: 0 ; 1 , R

READ WRITE MOTION

Try it in JFLAP...



A Turing Machine rule: 0 ; 1 , R

READ WRITE MOTION

Try it in JFLAP...

Name: _____

The tape

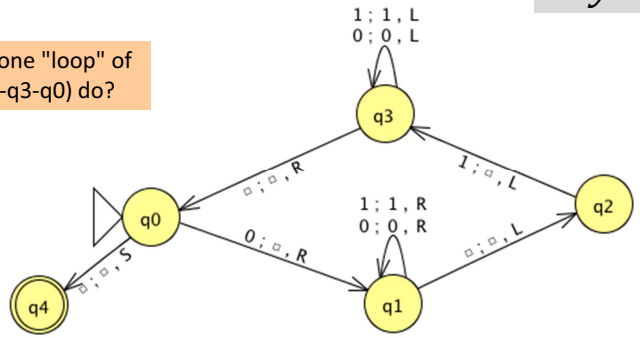
	0	0	1	1	1		

start

Is *this* input accepted or rejected by this TM?

Try it!

What does one "loop" of (q0-q1-q2-q3-q0) do?



What inputs are accepted *in general*?

Extra: How could you change this TM to accept *palindromes?*
(thought experiment and ex.cr.)

Keep this page – don't hand in...

The tape

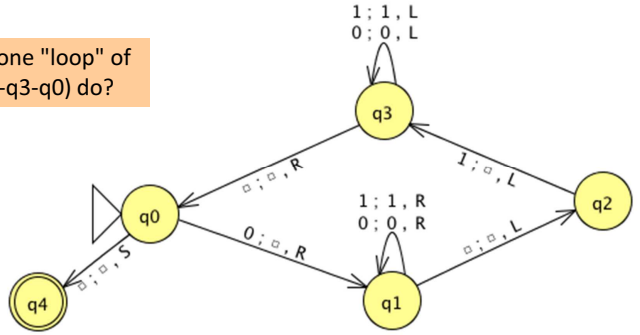
	0	0	1	1	1		

start

Is *this* input accepted or rejected by this TM?

With room to work...

What does one "loop" of (q0-q1-q2-q3-q0) do?



What inputs are accepted *in general*?

Extra: How could you change this TM to accept *palindromes?*
(thought experiment and ex.cr.)

Unprogrammable *functions?*

There are **well-defined mathematical functions** that no **computer program** or TM can compute

even with **any** *amount of memory!*

Functions

`Programs`

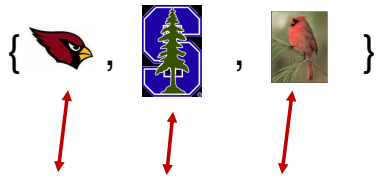
$$f(x) = \begin{cases} 1 & \text{if x is odd} \\ 0 & \text{if x is even} \end{cases}$$

```
def prog1(x):
    return x%2
```

Different infinities!?

There are *infinitely many* functions & programs…

…*but not all infinities are created equal!*



Two sets have equal size if their elements have a *one-to-one matching*

This matching is called a ***bijection***.

These sets have the same ***cardinality***

One-t… zed
infinit… ing!

```
def n2z(n):
    return (-1)**(n%2)*((n - (n%2))//2)

def z2n(z):
    return abs(z)*2 + (z<=0)
```

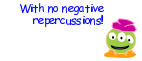
```
def e2n(e):
    return e//2

def n2e(n):
    return 2*n
```

Positive evens **E** = { 2, 4, 6, 8, 10, … }

Positive integers **N** = { 1, 2, 3, 4, 5, … }

ALL integers **Z** = {…, -3, -2, -1, 0, 1, 2, 3, …}



E and **N** and **Z** *all have the same size!*

Georg Cantor
1845-1918

Cantor Diagonalization

There are always real numbers missing from ***any*** list!

$\mathbb{R}$ The Reals from 0 to 1

Positive integers $\mathbb{N}$

Real #s are always missing r_0 =

Regardless of the matching…

r_1 = .333333333333333… ↔ 1

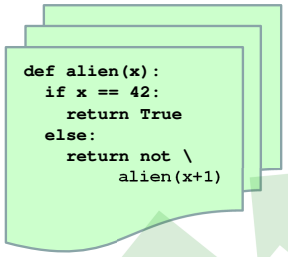
r_2 = .424242424242424… ↔ 2

r_3 = .314159426535897… ↔ 3

r_4 = .090909090909090… ↔ 4

r_5 = … ↔ 5

Programs are integers (and vice-versa)



```
def alien(x):
  if x == 42:
    return True
  else:
    return not \
        alien(x+1)
```

Programs = $\mathbb{N}$ Positive integers

Every program is a **string**.

Every string is just a sequence of **bits**

Every sequence of bits is also an **int**!

For each real number…

There's a (mathematical) function that simply returns that number!

Any real number from 0-1

r_1 = .11111111… $f_1(x) = 1/9$

r_2 = .31415926… $f_2(x) = \pi/10$

r_3 = .42424242… $f_3(x) = .42$ forever

r_4 = .01101010… $f_4(x) = \begin{cases} 1 & \text{if x s prime} \\ 0 & \text{otherwise} \end{cases}$

6 and 8 are *not* prime, so these digits are 0

2 and 3 are prime, so these digits are 1

(Many) more **functions** than **programs** !

The Reals from 0 to 1 $\mathbb{R}$

$\mathbb{N}$ Positive integers

Functions > Programs

Uncountably infinite

Countably infinite