Finite Automata

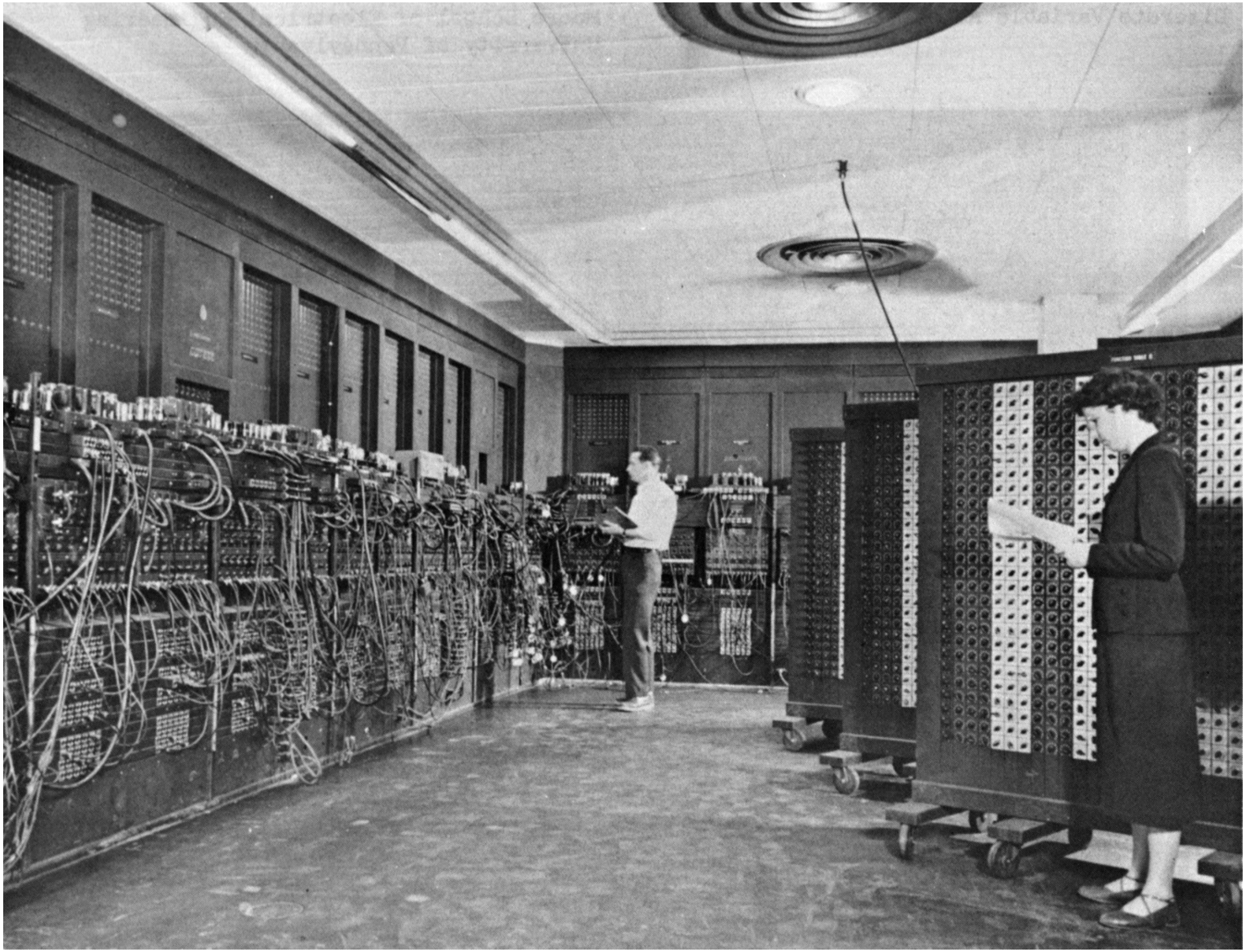
Part One

Computability Theory

What problems can we solve with a computer?

What kind of computer?

Computers are Messy



http://en.wikipedia.org/wiki/File:Eniac.jpg

Computers are Messy

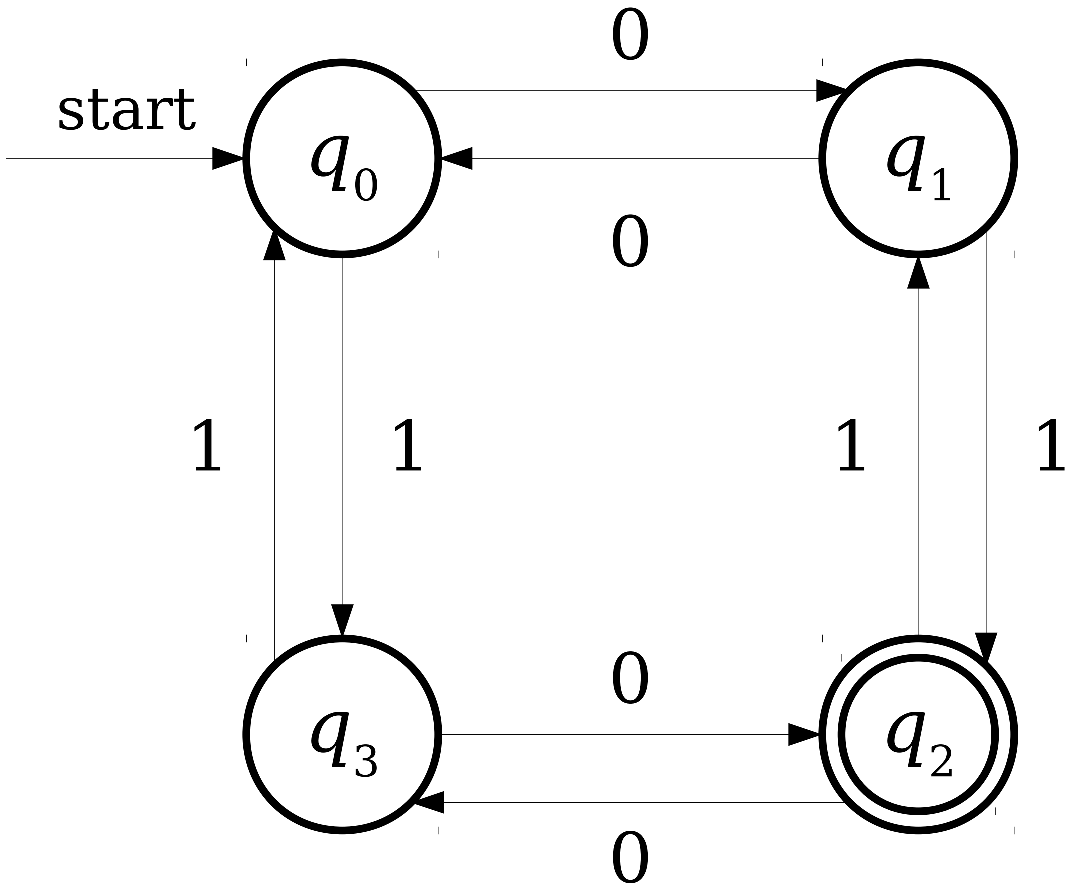
That messiness makes it hard to *rigorously* say what we *intuitively* know to be true: that, on some fundamental level, different brands of computers or programming languages are more or less equivalent in what they are capable of doing.

C vs C++
vs Java
vs
Python

We need a simpler way of discussing computing machines.

An ***automaton*** (plural: ***automata***) is a mathematical model of a computing device.

Automata are Clean



Why Build Models?

- ***Mathematical simplicity.***
  - It is significantly easier to manipulate our abstract models of computers than it is to manipulate actual computers.
- ***Intellectual robustness.***
  - If we pick our models correctly, we can make broad, sweeping claims about huge classes of real computers by arguing that they're just special cases of our more general models.

Why Build Models?

- The models of computation we will explore in this class correspond to different conceptions of what a computer could do.
- ***Finite automata*** (next two weeks) are an abstraction of computers with finite resource constraints.
  - Provide upper bounds for the computing machines that we can actually build.
- ***Turing machines*** (later) are an abstraction of computers with unbounded resources.
  - Provide upper bounds for what we could ever hope to accomplish.

What problems can we solve with a computer?

What is a "problem?"

Problems with Problems

- Before we can talk about what problems we can solve, we need a formal definition of a "problem."
- We want a definition that
  - corresponds to the problems we want to solve,
  - captures a large class of problems, and
  - is mathematically simple to reason about.
- No one definition has all three properties.

Formal Language Theory

Strings

- An ***alphabet*** is a finite, nonempty set of symbols called ***characters***.
  - Typically, we use the symbol **Σ** to refer to an alphabet.
- A ***string over an alphabet Σ*** is a finite sequence of characters drawn from Σ.
- Example: If Σ = {**a**, **b**}, here are some valid strings over Σ:

  a **aabaaabbabaaabaaaabbb** **abbababba**

- The ***empty string*** has no characters and is denoted **ε**.
- Calling attention to an earlier point: since all strings are finite sequences of characters from Σ, you cannot have a string of infinite length.

Languages

- A ***formal language*** is a set of strings.
- We say that L is a ***language over Σ*** if it is a set of strings over Σ.
- Example: The language of palindromes over Σ = {**a**, **b**, **c**} is the set
  - {ε, **a**, **b**, **c**, **aa**, **bb**, **cc**, **aaa**, **aba**, **aca**, **bab**, … }
- The set of all strings composed from letters in Σ is denoted **Σ***.
- Formally, we say that L is a language over Σ if $L \subseteq \Sigma^*$.

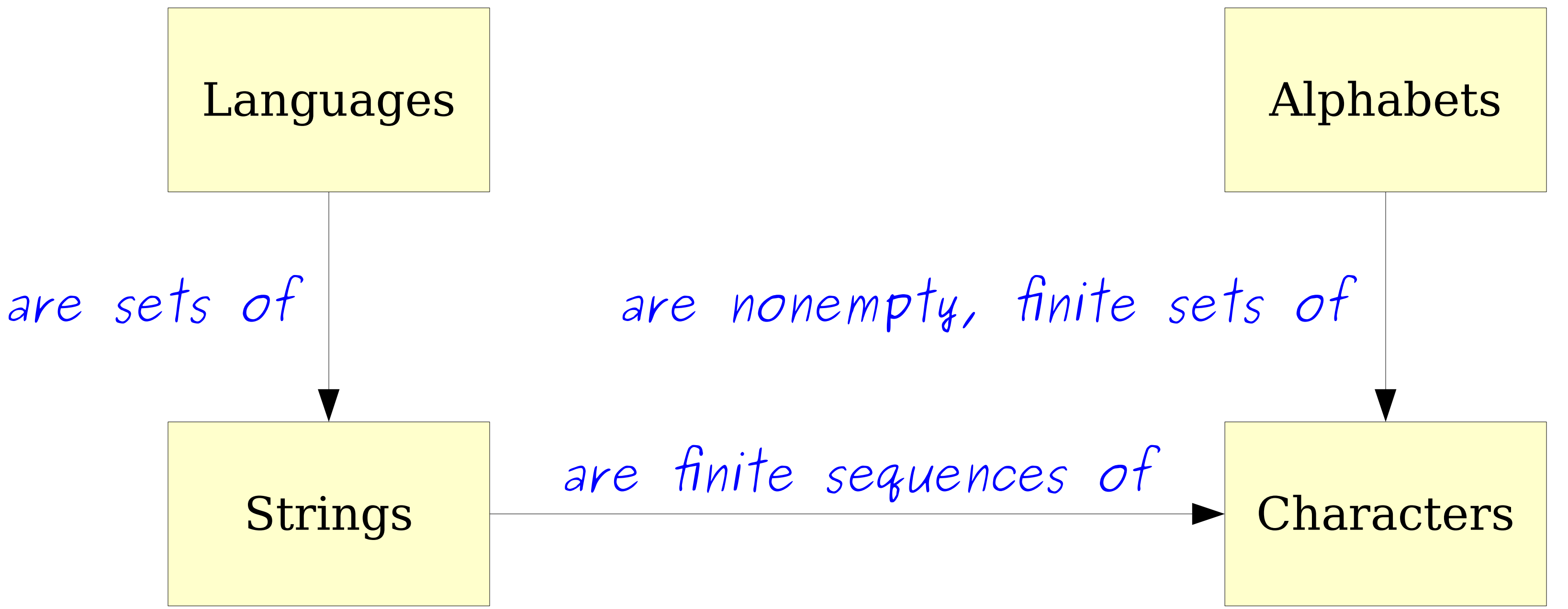
How many of the following statements are true?

- ***Alphabets*** are sequences of characters.
- ***Languages*** are sets of strings.
- ***Strings*** are sets of characters.
- ***Characters*** are individual symbols.
- ***Languages*** are sequences of characters.

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **a number**.

To Recap

- ***Languages*** are sets of strings.
- ***Strings*** are sequences of characters.
- ***Characters*** are individual symbols.
- ***Alphabets*** are sets of characters.



The Model

- ***Fundamental Question:*** For what languages L can you design an automaton that takes as input a string, then determines whether the string is in L?
- The answer depends on the choice of L, the choice of automaton, and the definition of "determines."
- In answering this question, we'll go through a whirlwind tour of models of computation and see how this seemingly abstract question has very real and powerful consequences.

To Summarize

- An ***automaton*** is an idealized mathematical computing machine.
- A ***language*** is a set of strings, a ***string*** is a (finite) sequence of characters, and a ***character*** is an element of an ***alphabet***.
- ***Goal:*** Figure out in which cases we can build automata for particular languages.

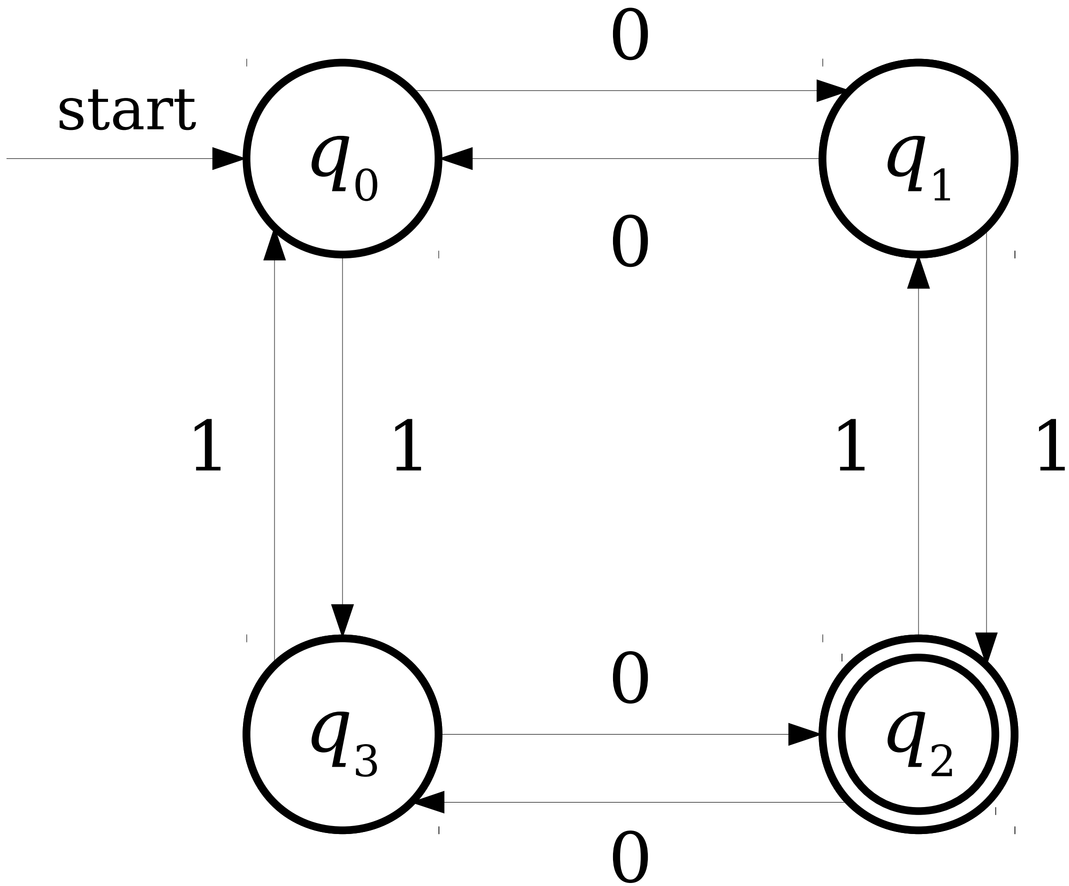
What problems can we solve with a computer?

Finite Automata

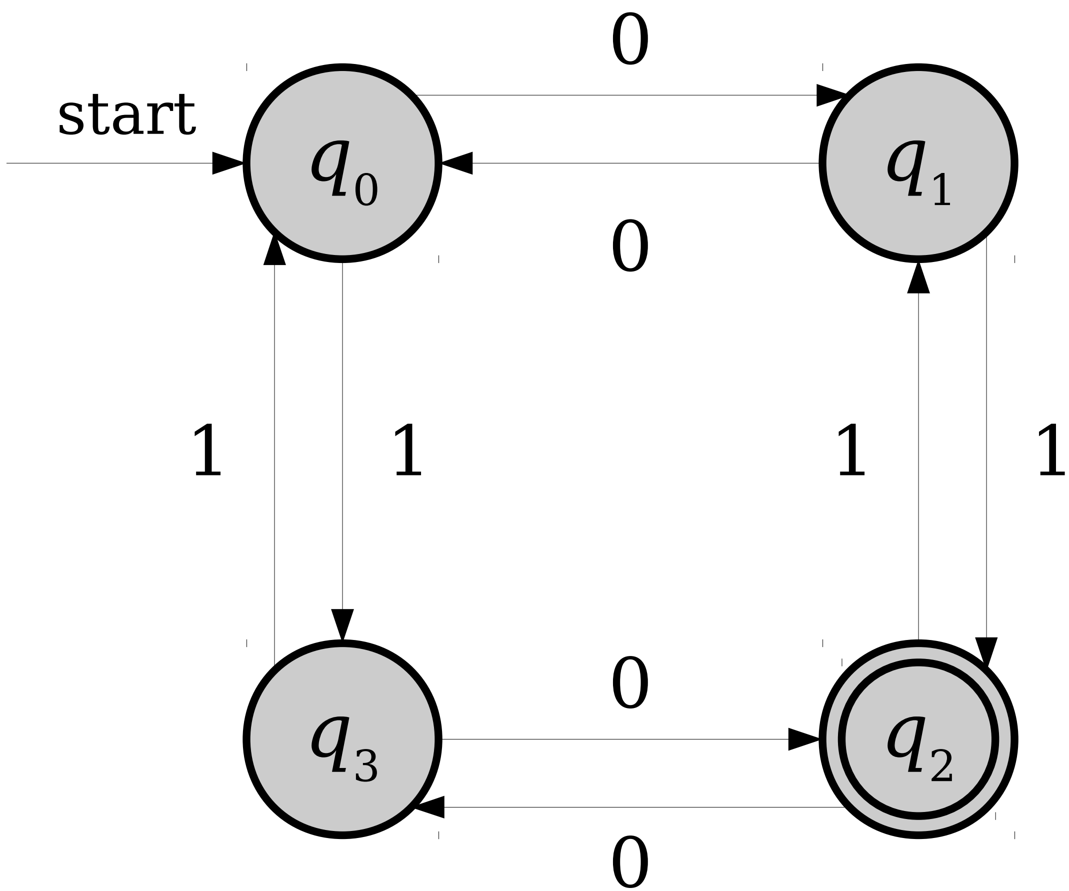
A ***finite automaton*** is a simple type of mathematical machine for determining whether a string is contained within some language.

Each finite automaton consists of a set of ***states*** connected by ***transitions***.

A Simple Finite Automaton

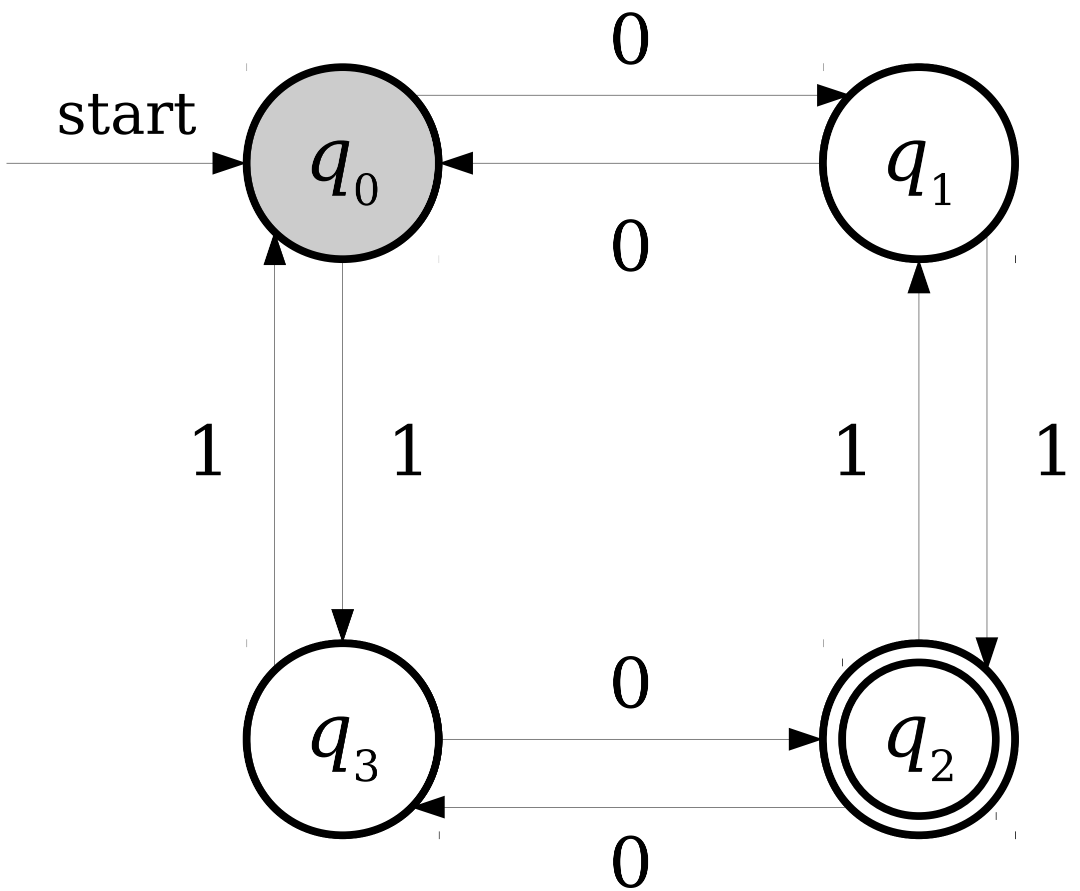


A Simple Finite Automaton



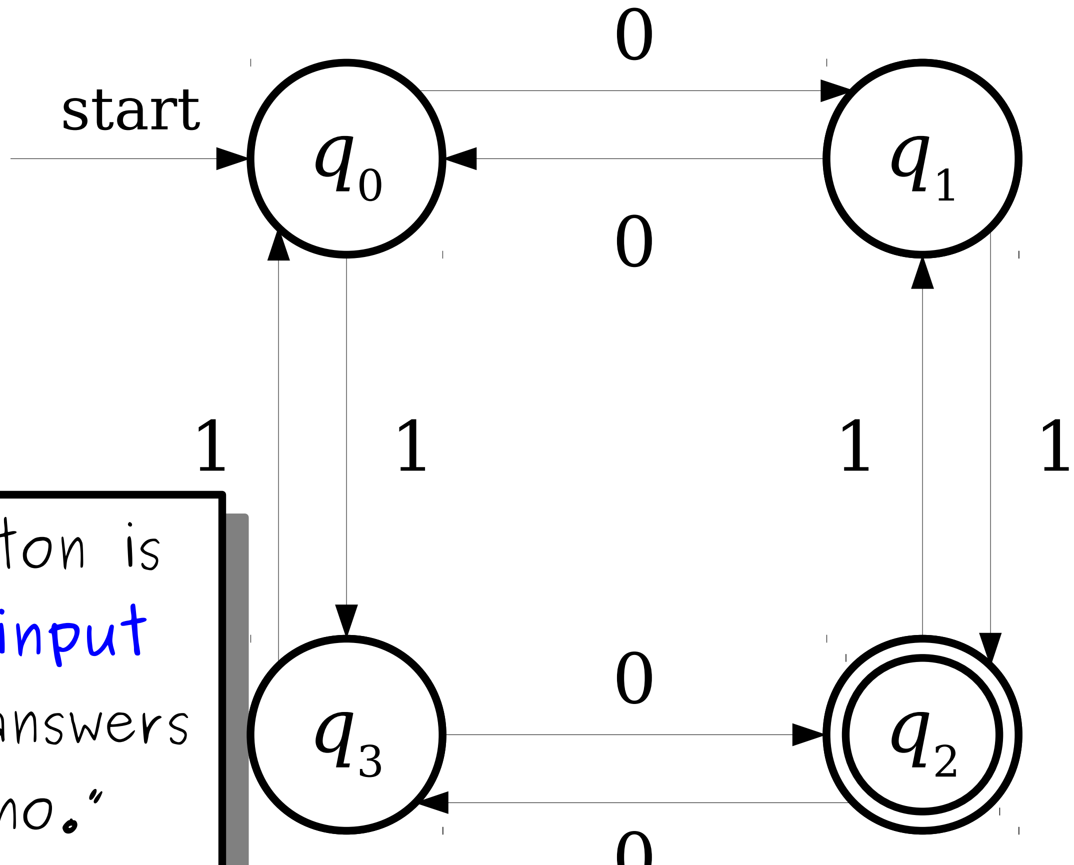
Each circle represents a state of the automaton.

A Simple Finite Automaton



One special state is designated as the **start state**.

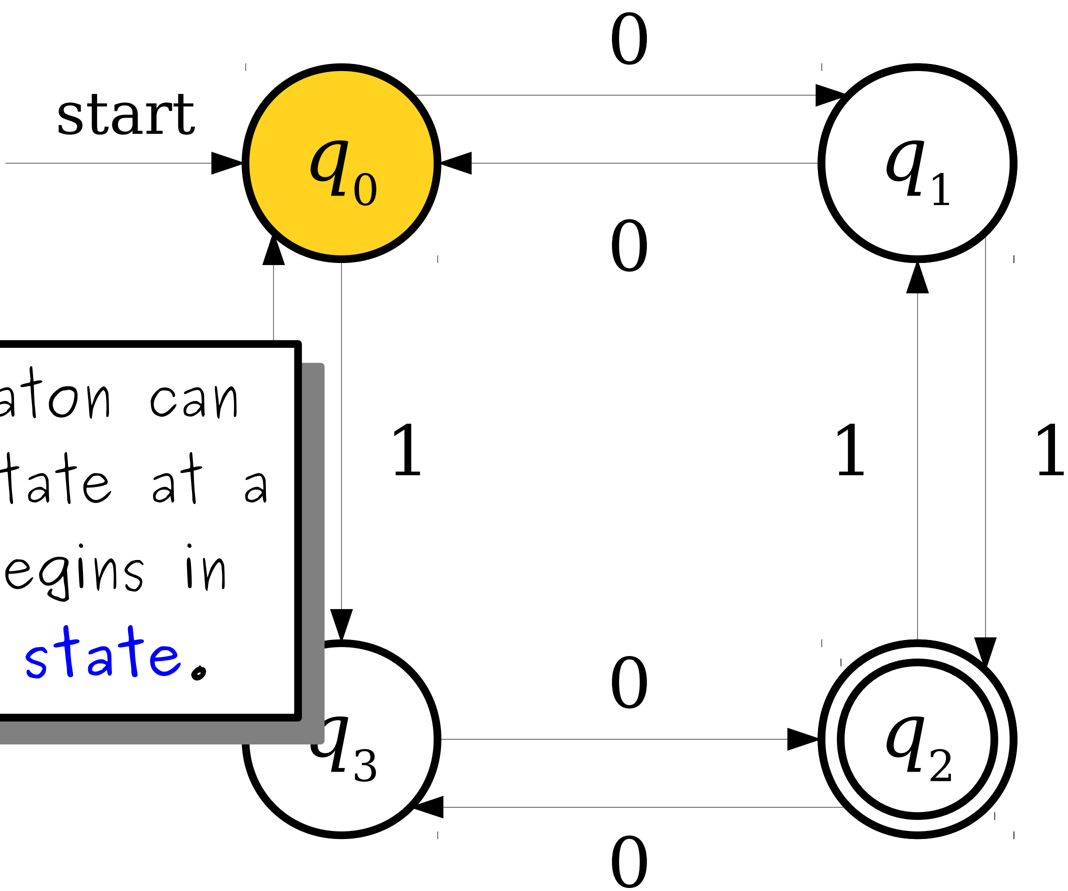
A Simple Finite Automaton



The automaton is run on an input string and answers "yes" or "no."

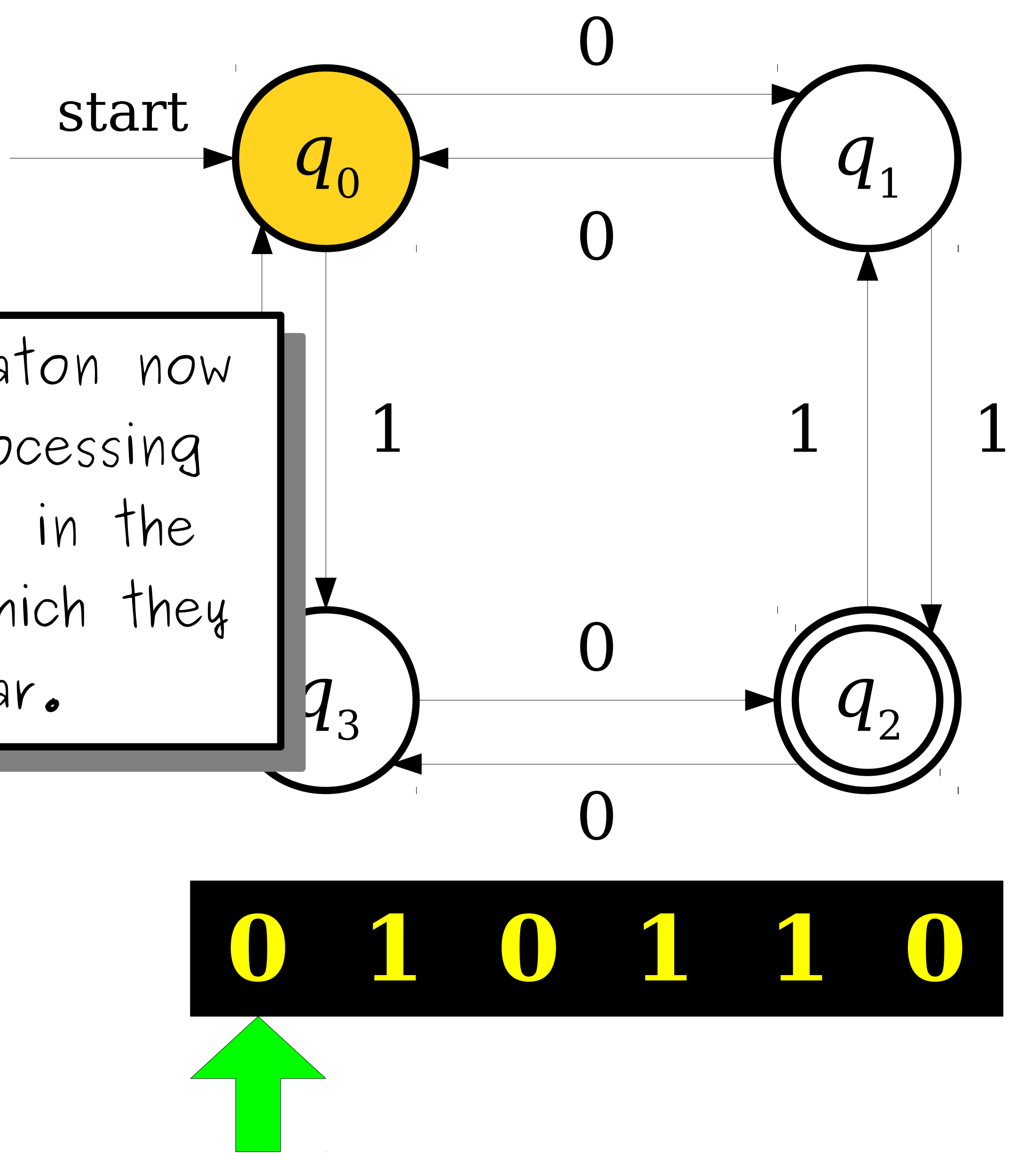
0 1 0 1 1 0

A Simple Finite Automaton



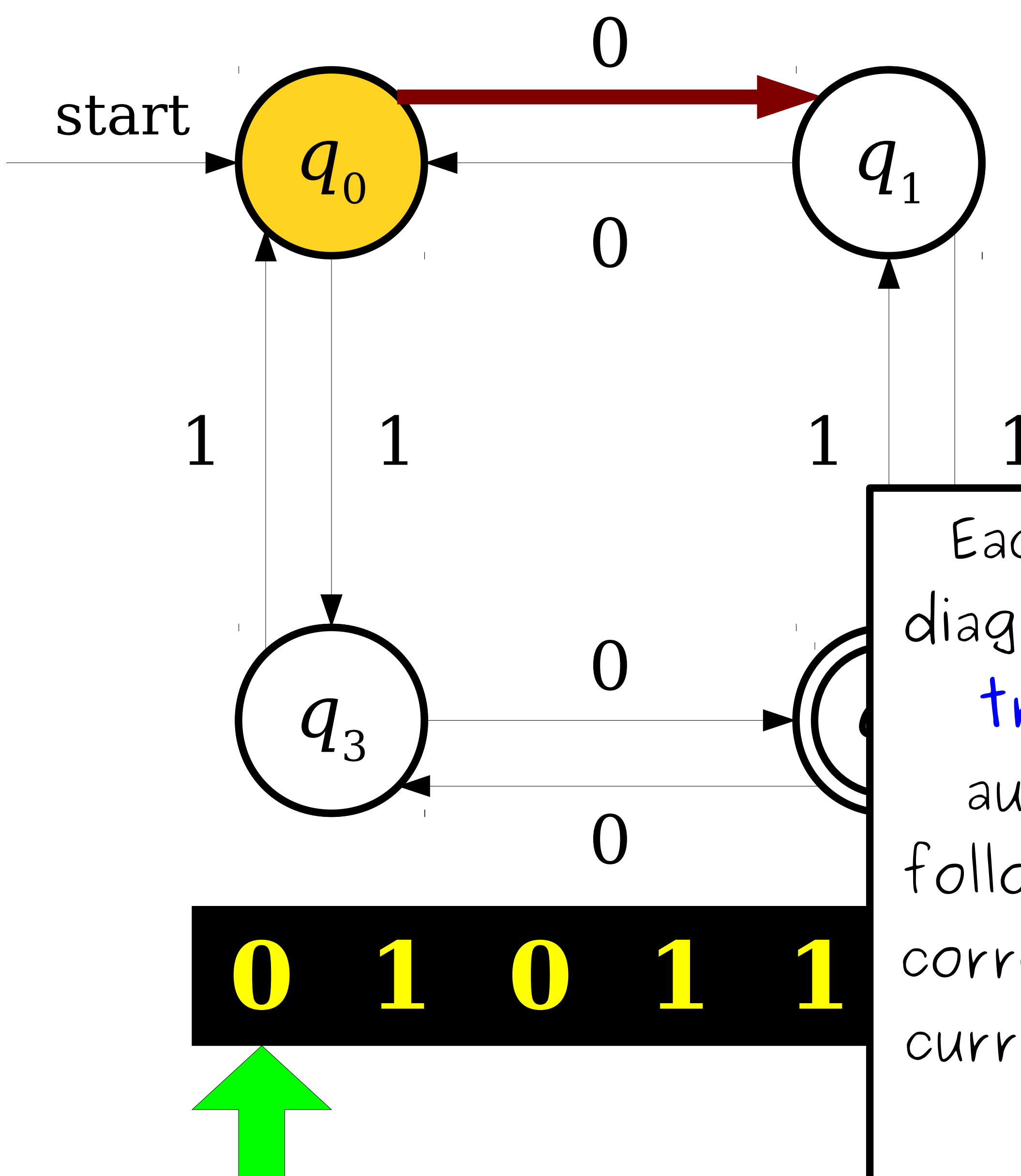
The automaton can be in one state at a time. It begins in the start state.

A Simple Finite Automaton



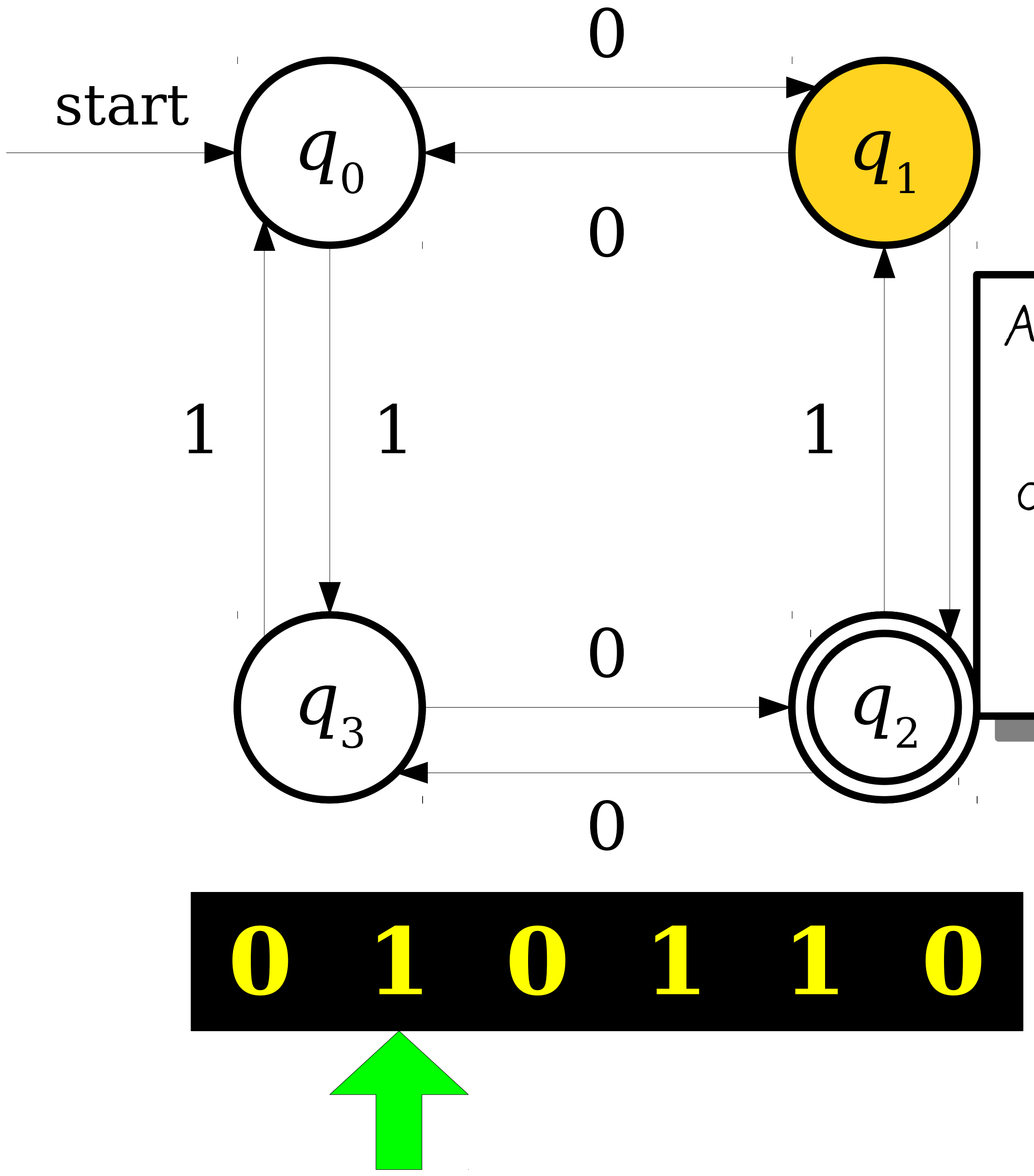
The automaton now begins processing characters in the order in which they appear.

A Simple Finite Automaton



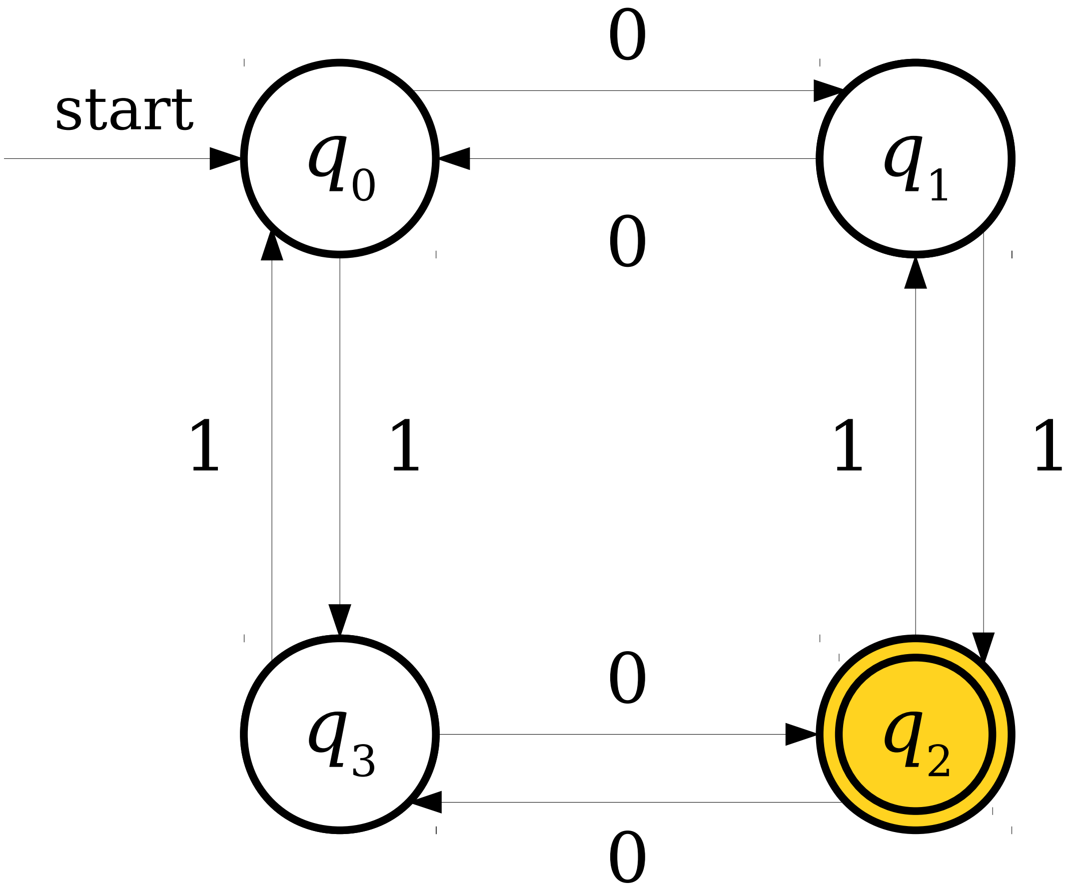
Each arrow in this diagram represents a **transition**. The automaton always follows the transition corresponding to the current symbol being read.

A Simple Finite Automaton



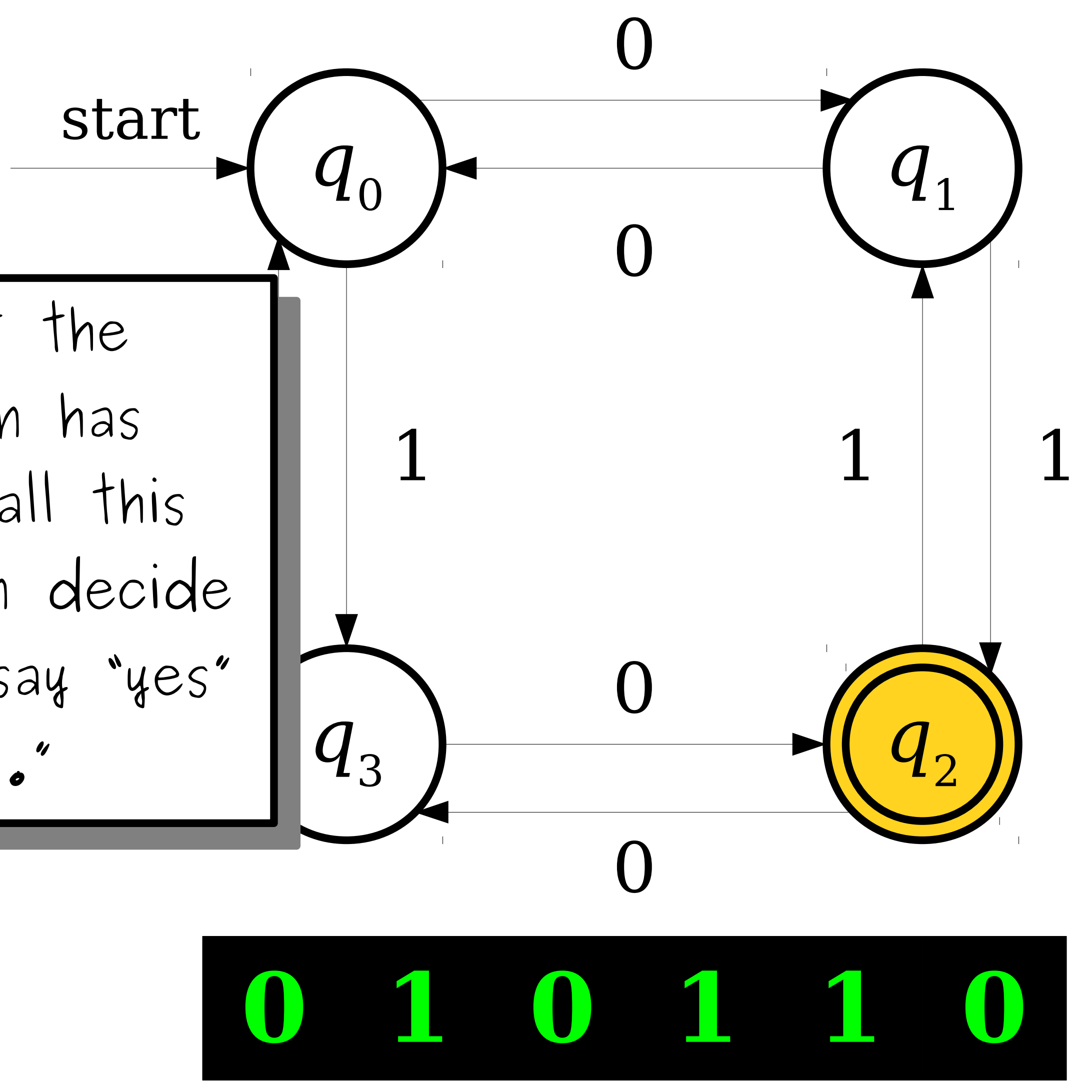
After transitioning, the automaton considers the next symbol in the input.

A Simple Finite Automaton



0 1 0 1 1 0

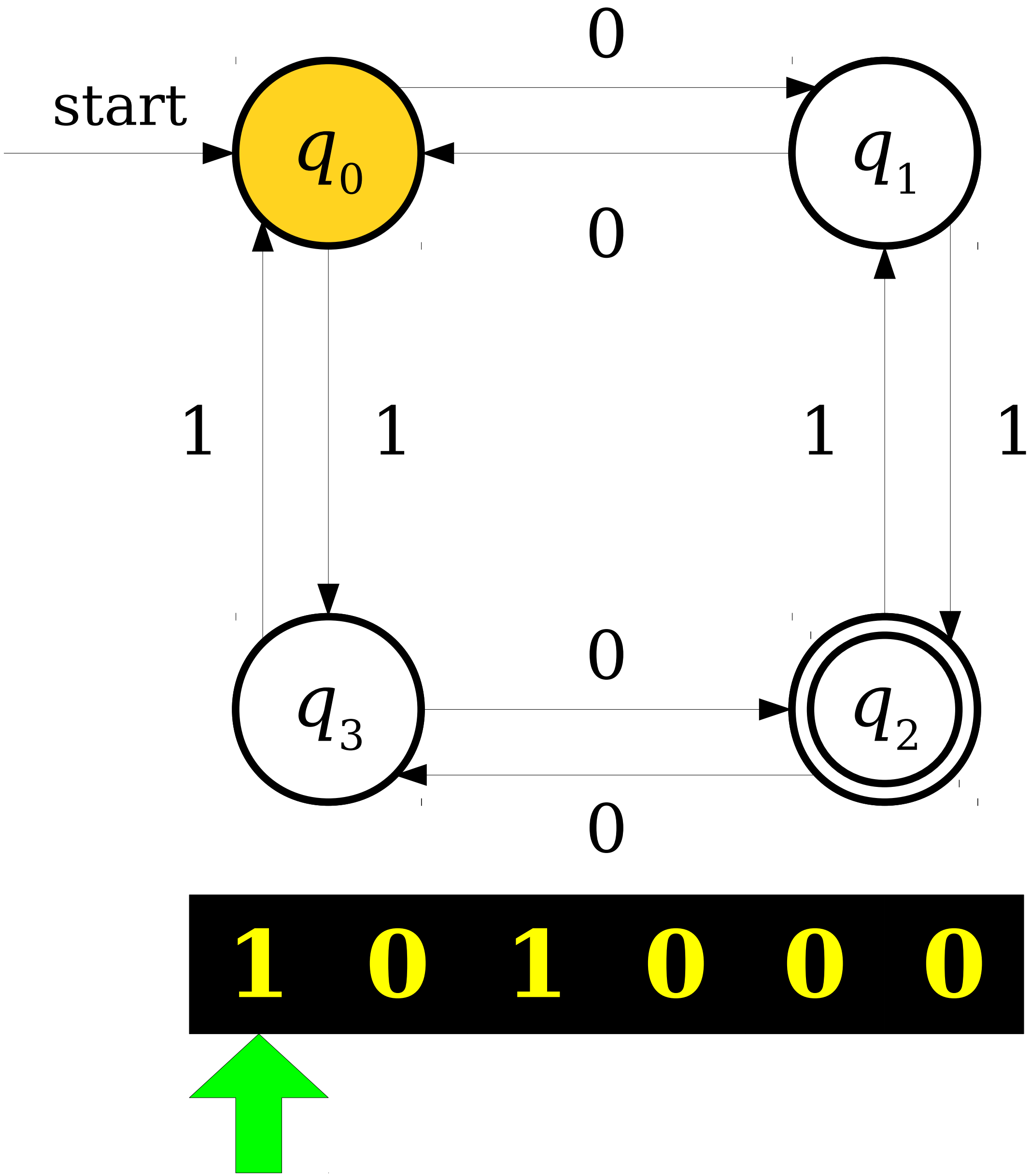
A Simple Finite Automaton



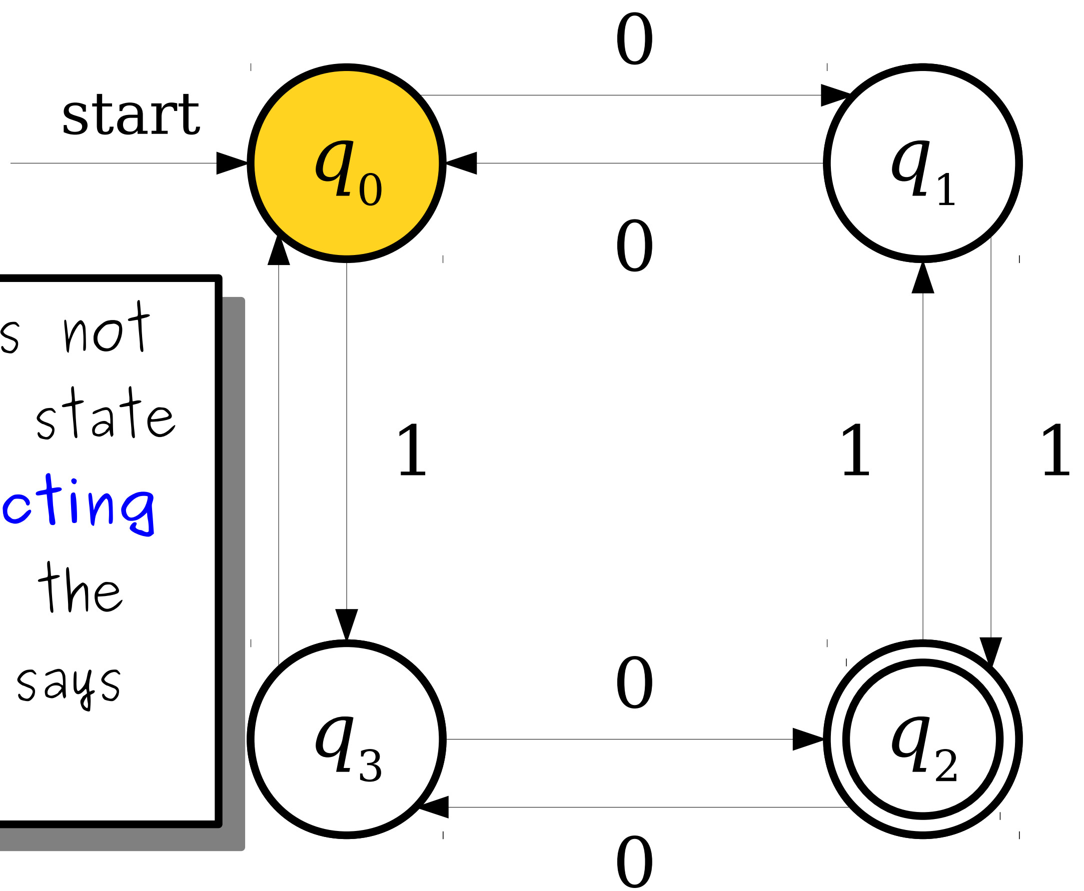
Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

The double circle indicates that this state is an **accepting state**, so the automaton outputs "yes."

A Simple Finite Automaton



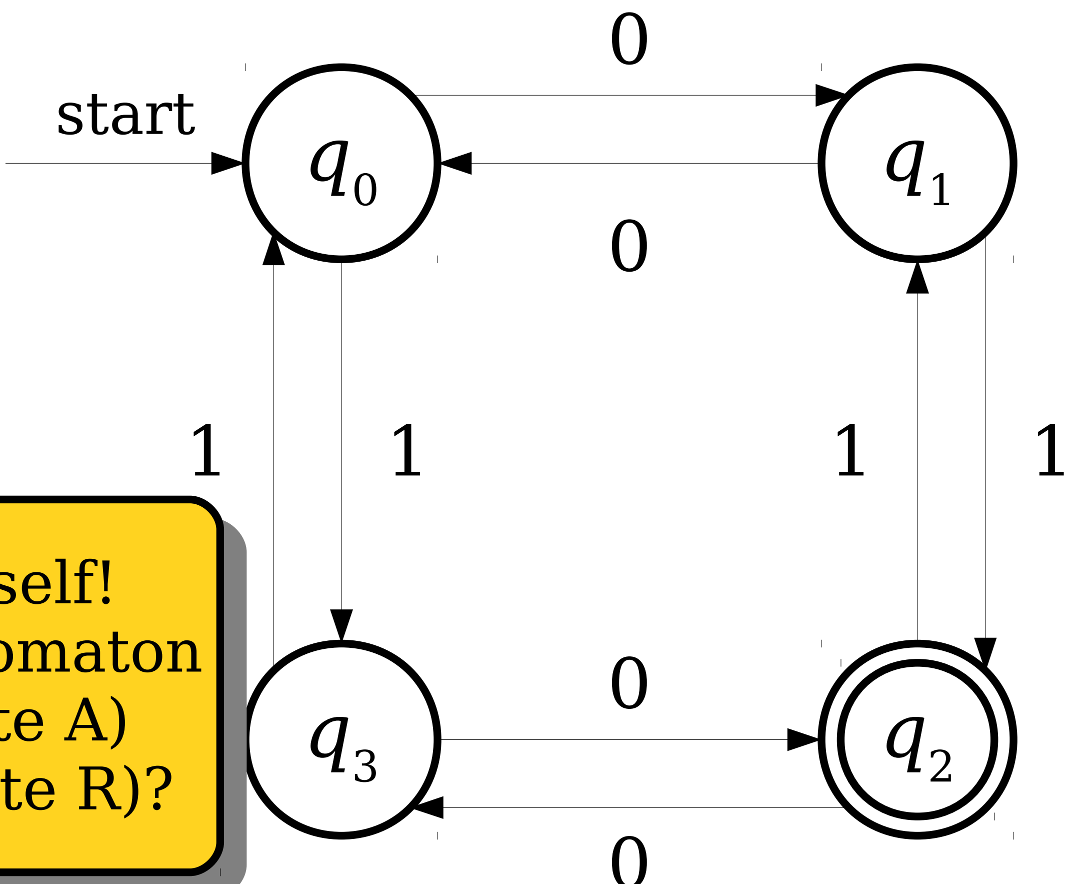
A Simple Finite Automaton



This state is not an accepting state (it's a **rejecting state**), so the automaton says "no."

1 0 1 0 0 0

A Simple Finite Automaton



Try it yourself!
Does this automaton ***accept*** (vote A)
or ***reject*** (vote R)?

1 1 0 1 1 1 0 0

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **A or R**.

The Story So Far

- A ***finite automaton*** is a collection of ***states*** joined by ***transitions***.
- Some state is designated as the ***start state***.
- Some states are designated as ***accepting states***.
- The automaton processes a string by beginning in the start state and following the indicated transitions.
- If the automaton ends in an accepting state, it ***accepts*** the input.
- Otherwise, the automaton ***rejects*** the input.

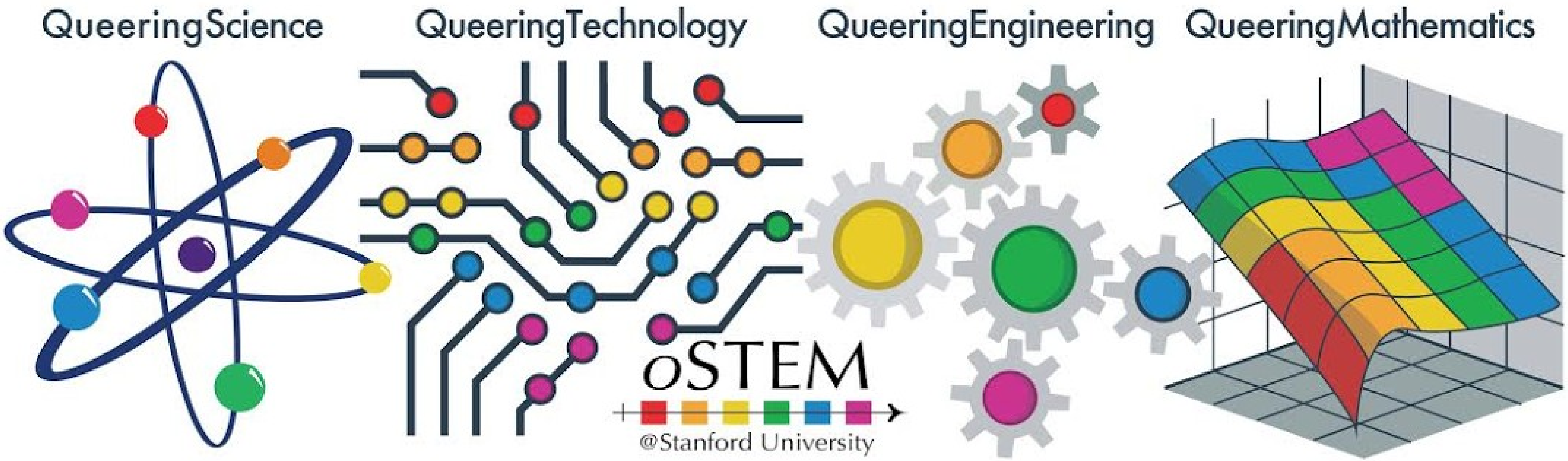
Time-Out For Announcements!

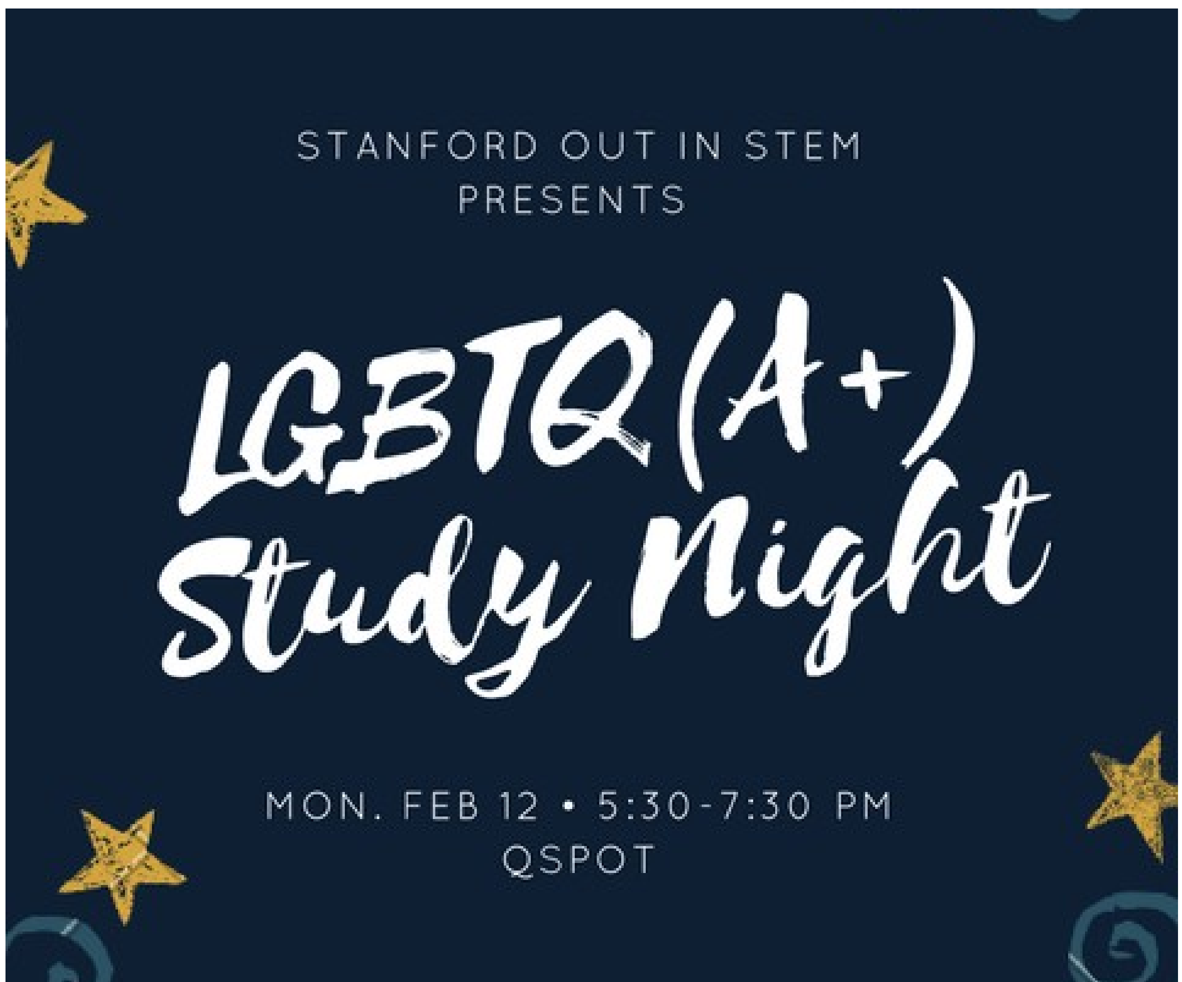
Girl Code @Stanford

- This summer, I'll be running our sixth iteration of Girl Code @Stanford from July 9th – July 20st.
- We invite high-school girls (primarily from low- to middle-income schools in majority-minority areas) to come to campus for two weeks to learn CS, meet researchers, and talk to folks from industry.
- We're looking for Stanford students to serve as "Workshop Assistants" during the program. We pay competitively (roughly $3,000 over two weeks).
- Interested? Learn more and apply using this link:

https://goo.gl/forms/Y76akbVWUYV0NPpR2

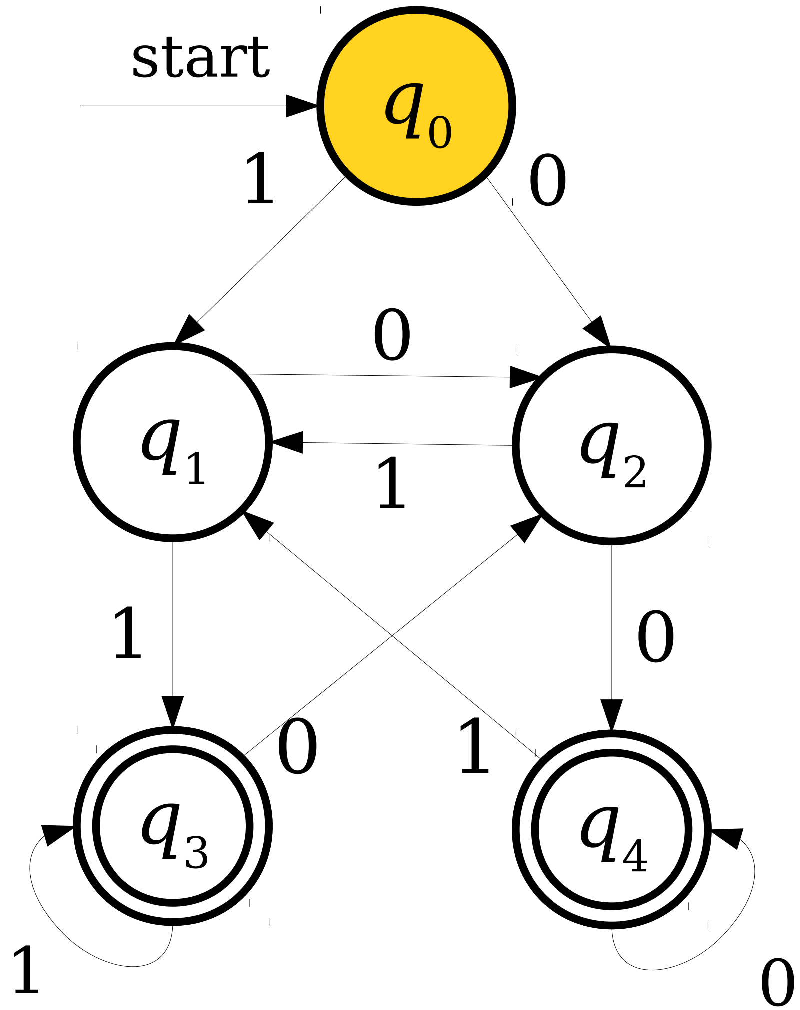
All current Stanford students are invited to apply. Feel free to forward this link around!


QueeringScience
QueeringTechnology
QueeringEngineering
QueeringMathematics
oSTEM
@Stanford University

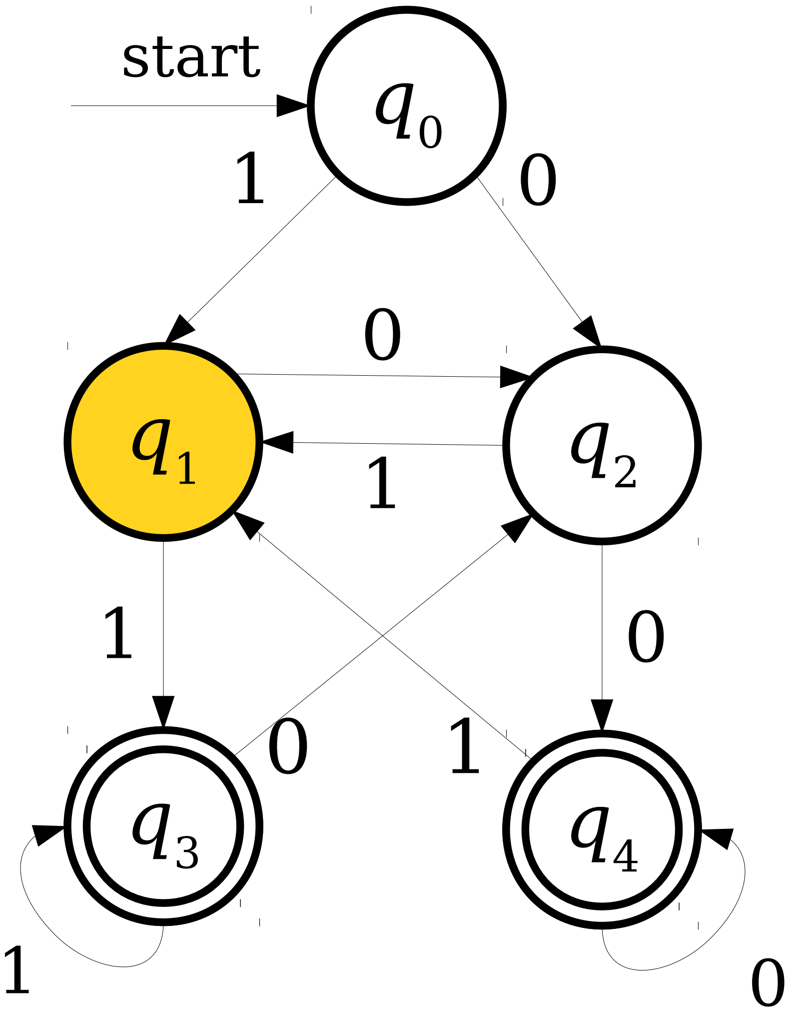

STANFORD OUT IN STEM
PRESENTS
LGBTQ(A+)
Study Night
MON. FEB 12 • 5:30-7:30 PM
QSPOT

Back to CS103!

Just Passing Through



Just Passing Through

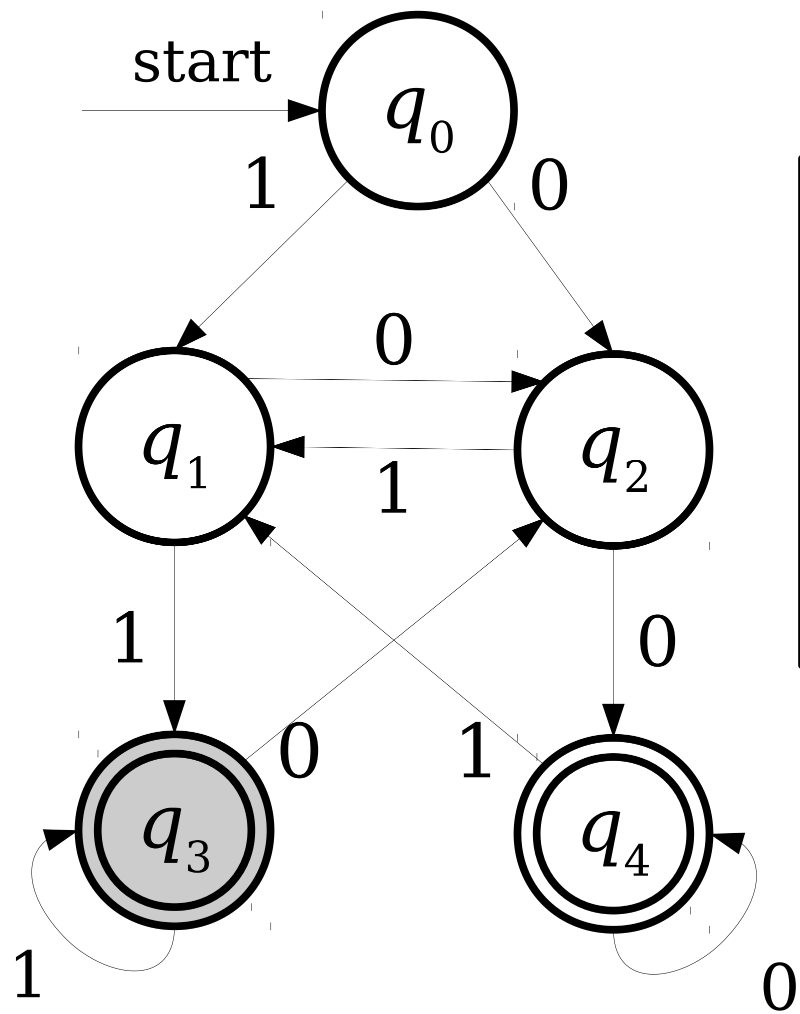


1 1 0 1

A finite automaton does ***not*** accept as soon as it enters an accepting state.

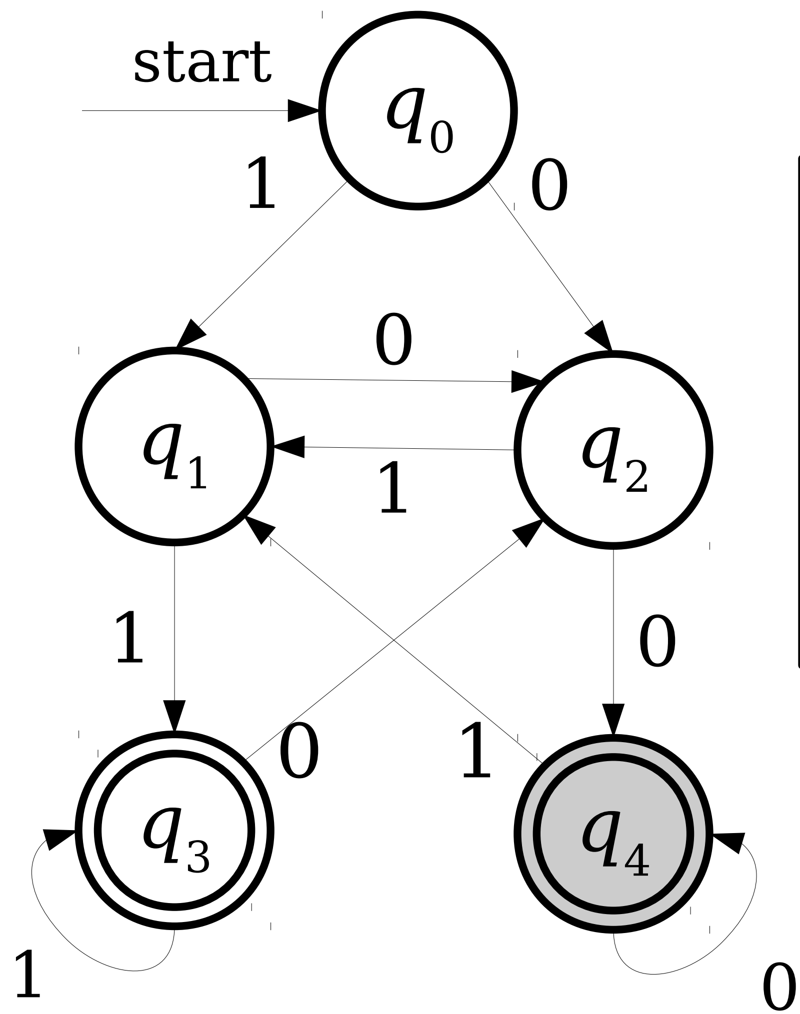
A finite automaton accepts if it ***ends*** in an accepting state.

What Does This Accept?



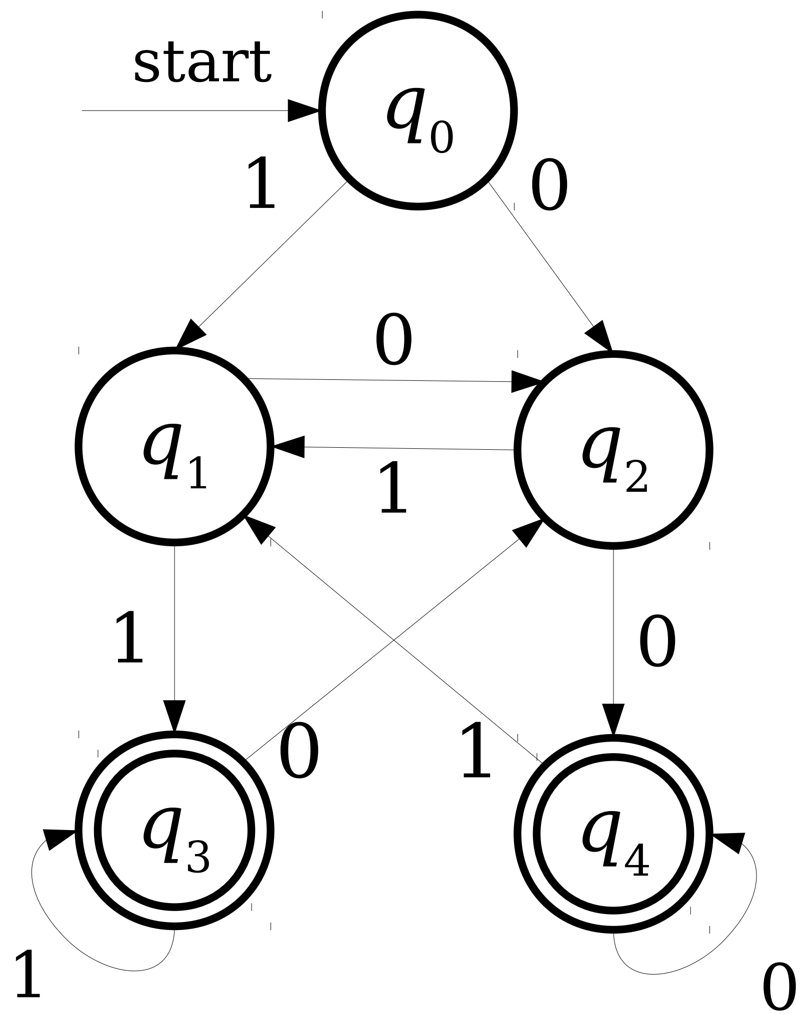
No matter where we start in the automaton, after seeing two 1's, we end up in accepting state q_3.

What Does This Accept?



No matter where we start in the automaton, after seeing two 0's, we end up in accepting state q_5.

What Does This Accept?



This automaton accepts a string in {0, 1}* iff the string ends in 00 or 11.

The ***language of an automaton*** is the set of strings that it accepts.

If D is an automaton that processes characters from the alphabet Σ, then $\mathscr{L}(\boldsymbol{D})$ is formally defined as

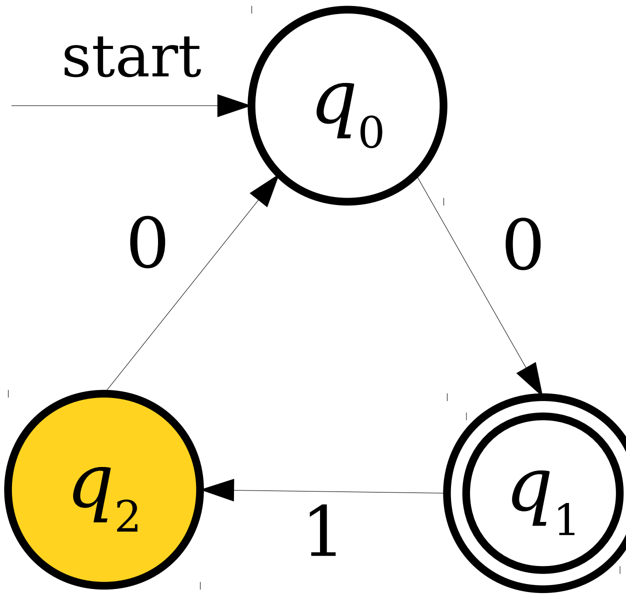
$$\mathscr{L}(\boldsymbol{D}) = \{\ \boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{D} \text{ accepts } \boldsymbol{w}\ \}$$

How many of the following statements are true?

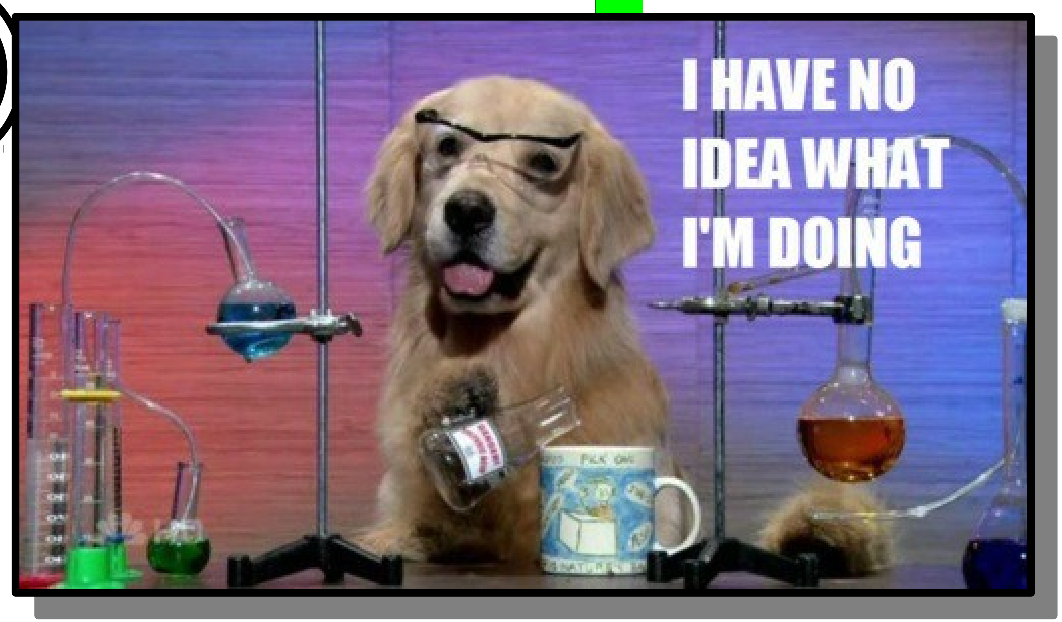
- ***A language*** of an automaton can have an infinitely long string (or many of them) in it.
- ***A language*** of an automaton can contain infinitely many strings.
- ***A language*** of an automaton can contain no strings.

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **a number**.

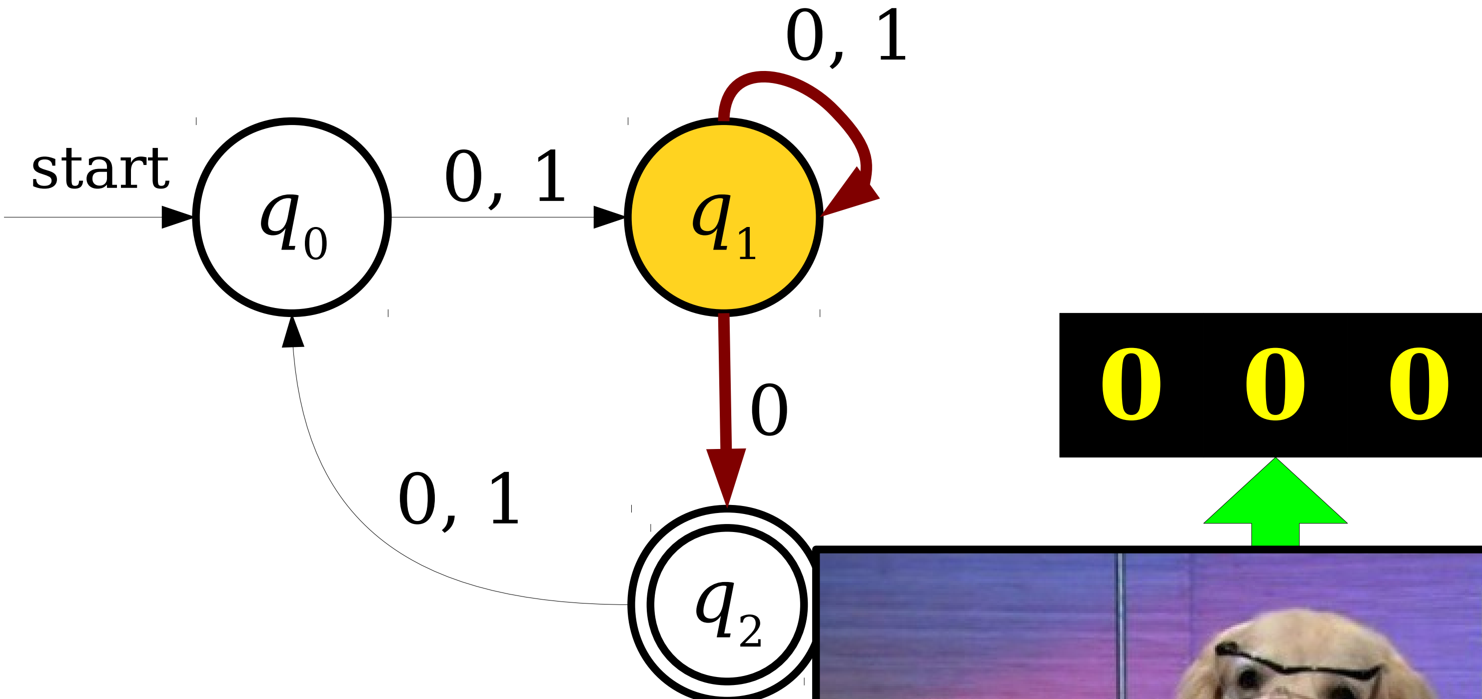
A Small Problem

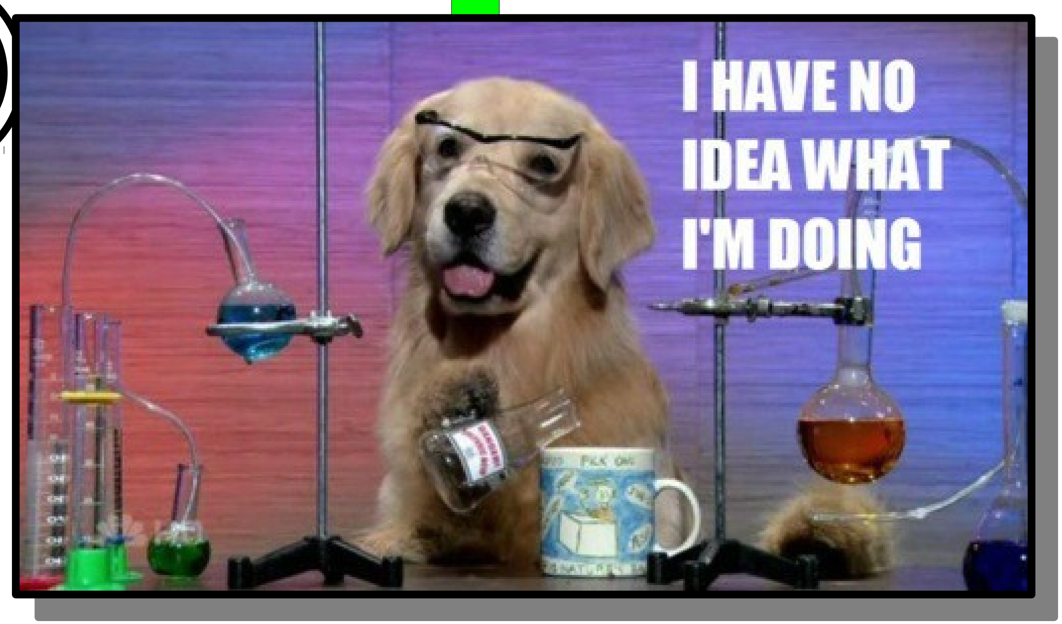


0 1 1 0



Another Small Problem





The Need for Formalism

- In order to reason about the limits of what finite automata can and cannot do, we need to formally specify their behavior in *all* cases.
- All of the following need to be defined or disallowed:
  - What happens if there is no transition out of a state on some input?
  - What happens if there are *multiple* transitions out of a state on some input?

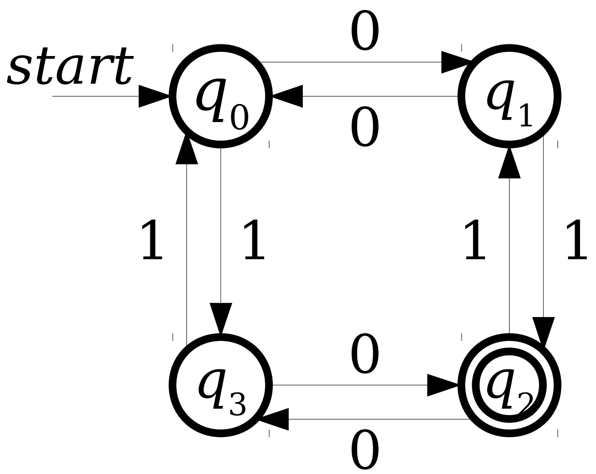
DFAs

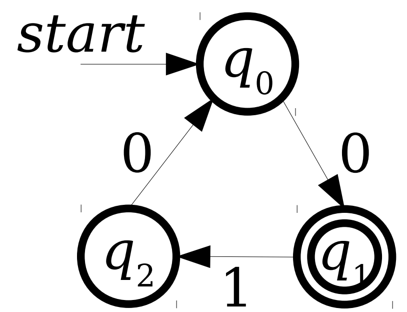
- A ***DFA*** is a
  - ***D***eterministic
  - ***F***inite
  - ***A***utomaton
- DFAs are the simplest type of automaton that we will see in this course.

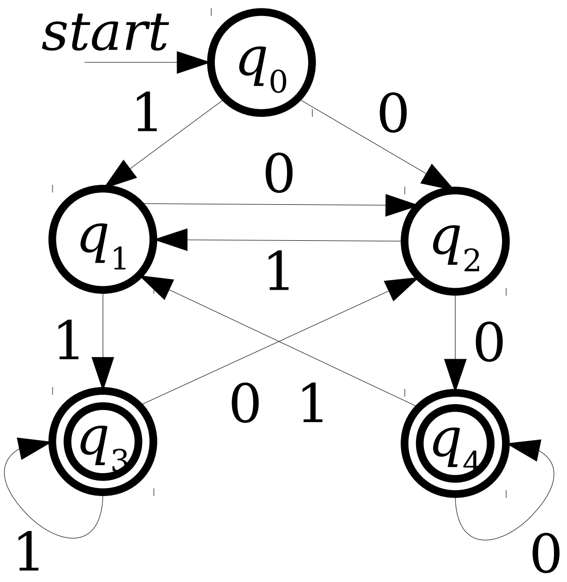
DFAs

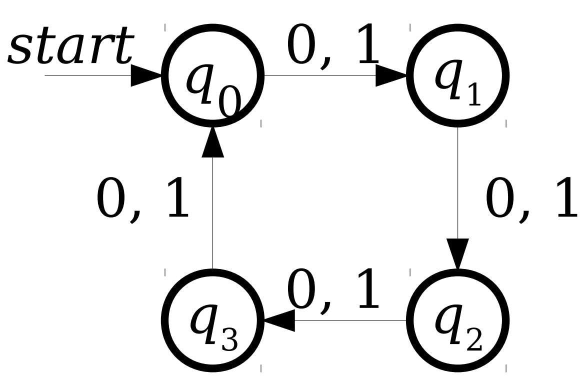
- A DFA is defined relative to some alphabet Σ.
- For each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ.
  - This is the "deterministic" part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

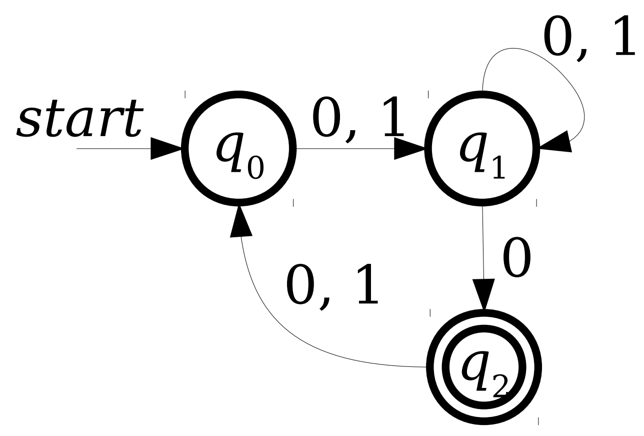
How many of these are DFAs over {0, 1}?











Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **a number**.

Is this a DFA?

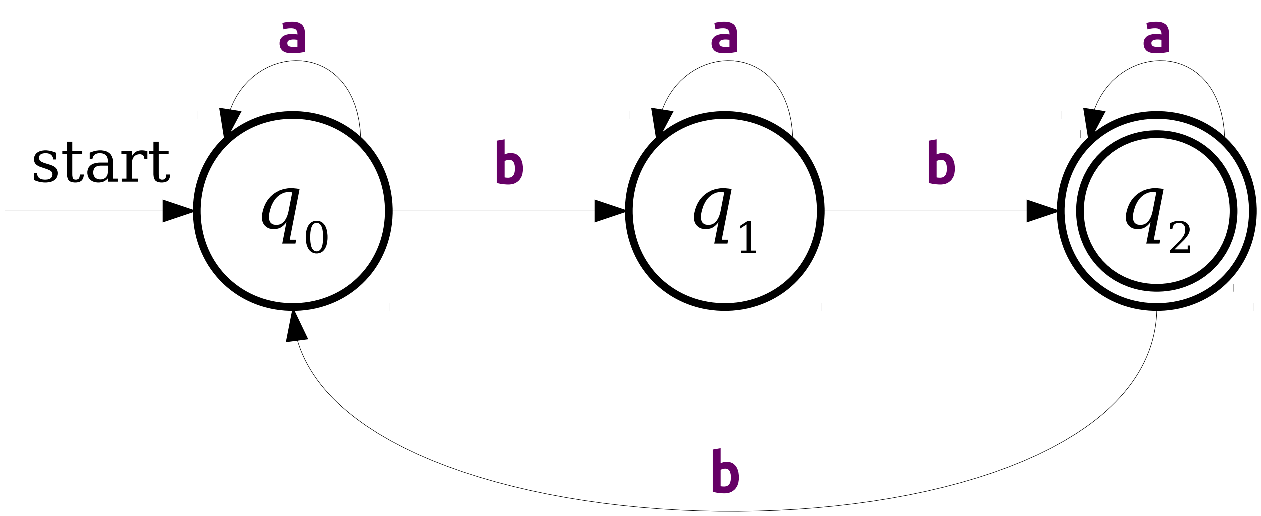
Drinking **F**amily of **A**ardvarks

Designing DFAs

- At each point in its execution, the DFA can only remember what state it is in.
- ***DFA Design Tip:*** Build each state to correspond to some piece of information you need to remember.
  - Each state acts as a "memento" of what you're supposed to do next.
  - Only finitely many different states means only finitely many different things the machine can remember.

Recognizing Languages with DFAs

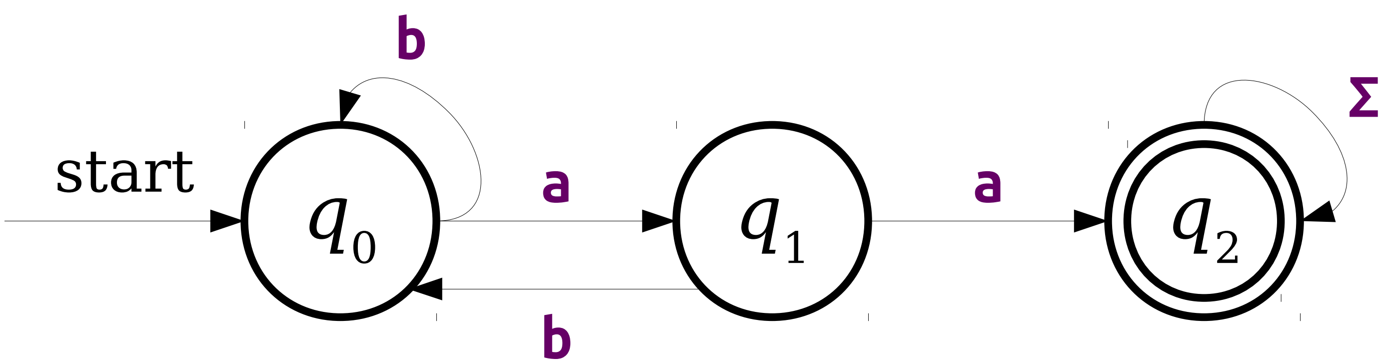
$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid$ the number of **b**'s in w is congruent to two modulo three $\}$



Each state remembers the remainder of the number of **b**s seen so far modulo three.

Recognizing Languages with DFAs

$L = \{\ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ contains } \texttt{aa} \text{ as a substring } \}$



More Elaborate DFAs

$L = \{ w \in \{$**a**, *****, **/**$\}^* \mid w$ represents a C-style comment $\}$

Let's have the **a** symbol be a placeholder for "some character that isn't a star or slash."

Try designing a DFA for comments! Here's some test cases to help you check your work:

Accepted:	Rejected:
/*a*/	**/****
//**	**/**/a/*aa*/**
/*/**	**aaa/**/aa**
/*aaa*aaa*/	**/*/**
/*a/a*/	**/**a/**
	//aaaa

More Elaborate DFAs

$L = \{ w \in \{$**a**, *****, **/**$\}^* \mid w$ represents a C-style comment $\}$