CMPU 240

Theory of Computation

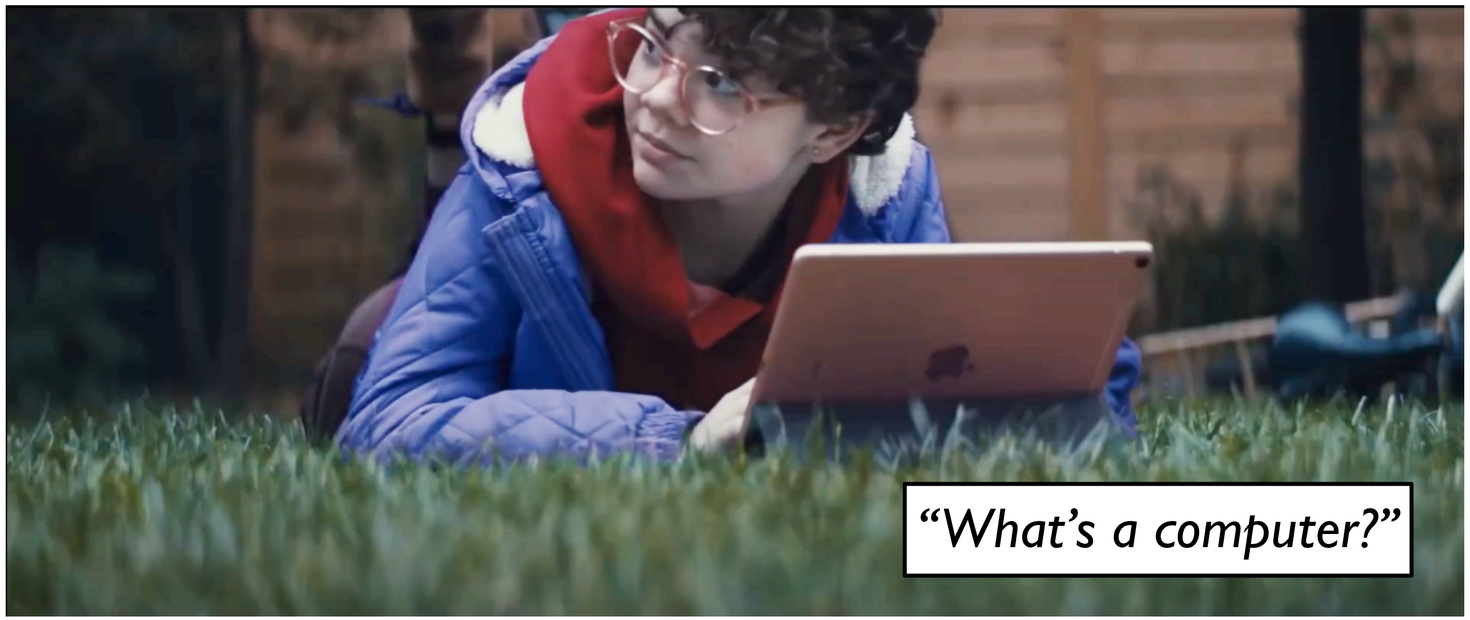
Spring 2026

Before we get started…

In class, I expect you to contribute to an environment that supports everyone's learning.

1. Please raise your hands to ask or answer questions
2. No electronic devices (phones, tablets, laptops, headphones) unless you require an accommodation.
3. No food in class!

What’s a computer?



Apple ~~Computer~~ Inc.,

2017

Seriously, what *is* a computer?

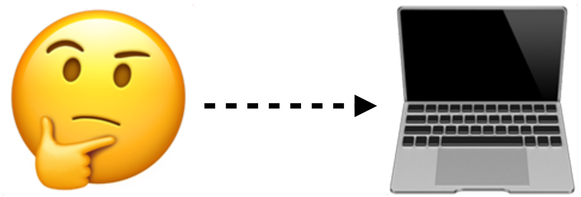
314

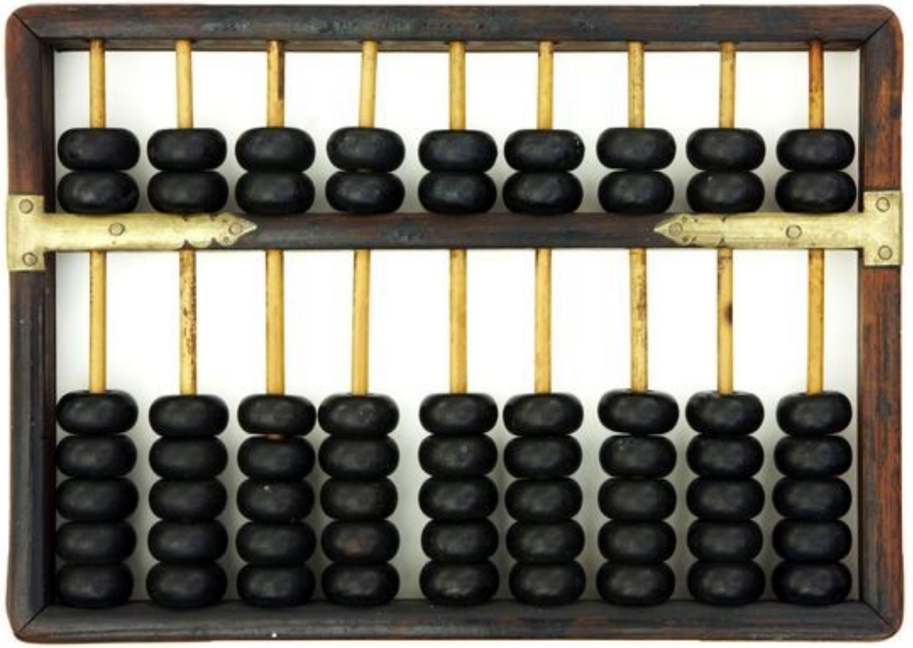
+ 159

$$\begin{array}{r} {\scriptstyle 1} \\ 314 \\ +\ 159 \\ \hline 3 \end{array}$$

$$\begin{array}{r} {\scriptstyle 1} \\ 314 \\ +\ 159 \\ \hline 73 \end{array}$$

$$\begin{array}{r} {\scriptstyle 1} \\ 314 \\ +\ 159 \\ \hline 473 \end{array}$$





A small suan pan (*Chinese abacus*)

Photo from the Computer History Museum

Hand-cranked Curta calculator, c. 1950

Photo from the Computer History Museum

Some kinds of computers have more computational power than others.

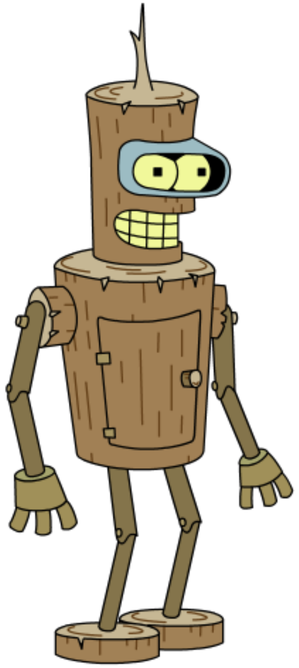
We can abstract devices of the “same kind” to produce *models* of computers and ask what kinds of problems can be solved under a particular model.

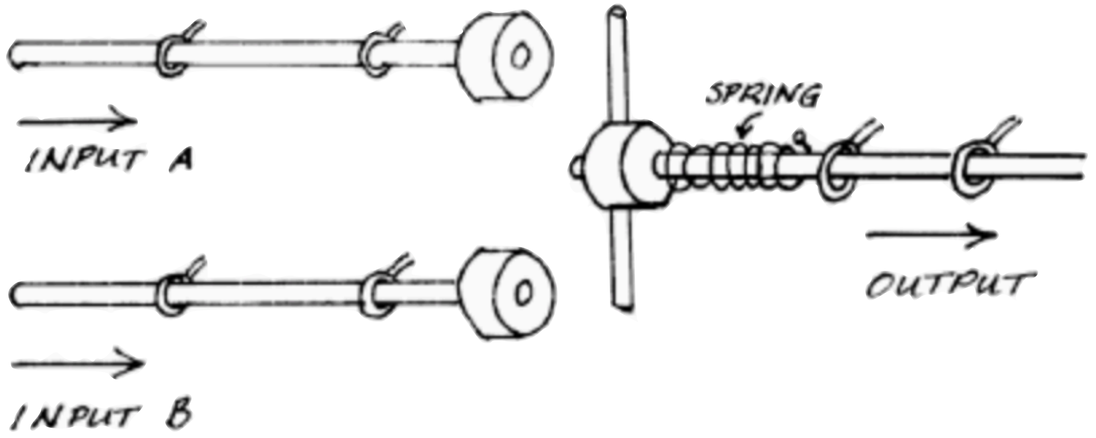
“One of the most remarkable things about computers is that their essential nature transcends technology.”

W. Daniel Hillis, *The Pattern on the Stone*, 1998

Present-day computers are built out of transistors and wires.

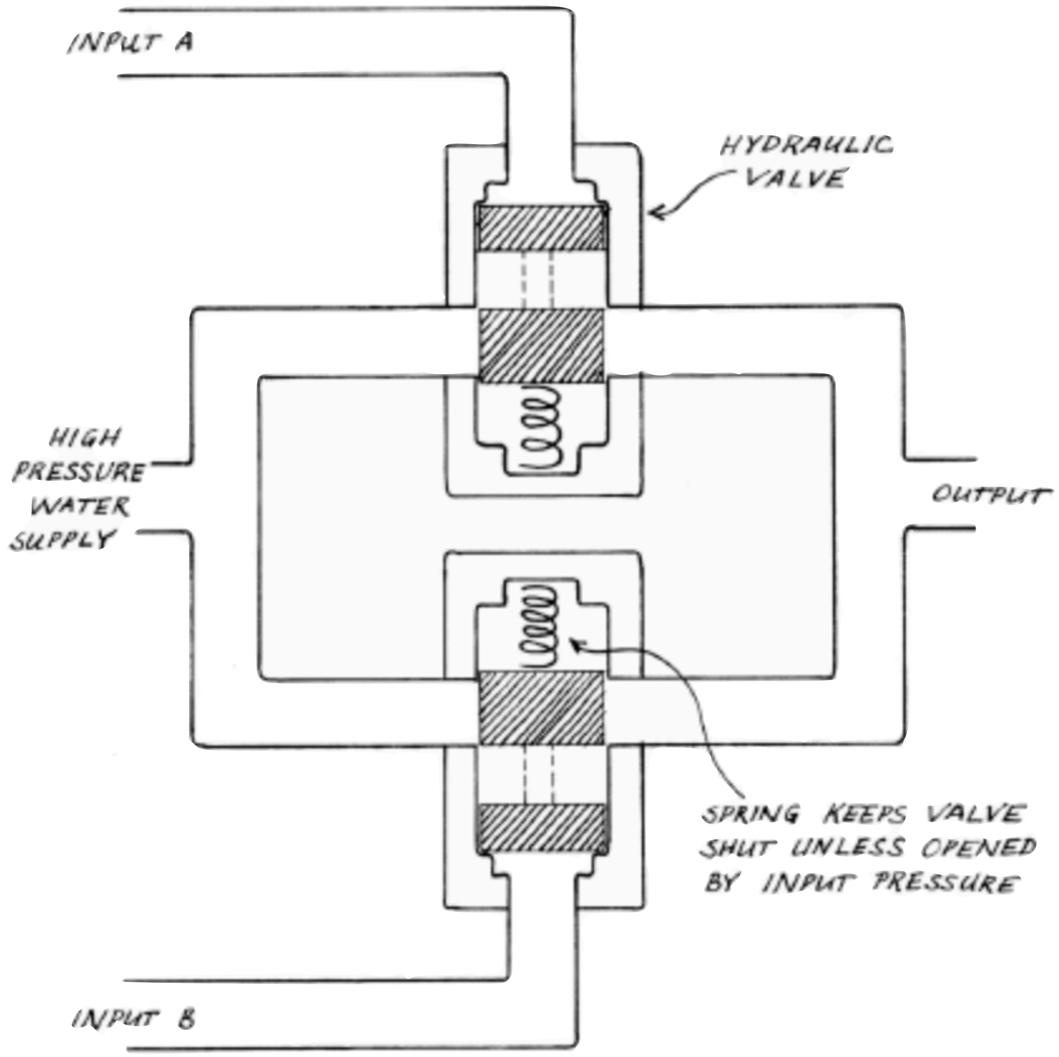
But they could be built – according to the same principles – from valves and water pipes or from sticks and strings.





W. Daniel Hillis,
The Pattern on the Stone,
1998

Mechanical implementation of the **`or`** function



W. Daniel Hillis,
The Pattern on the Stone,
1998

Hydraulic implementation of the **or** function

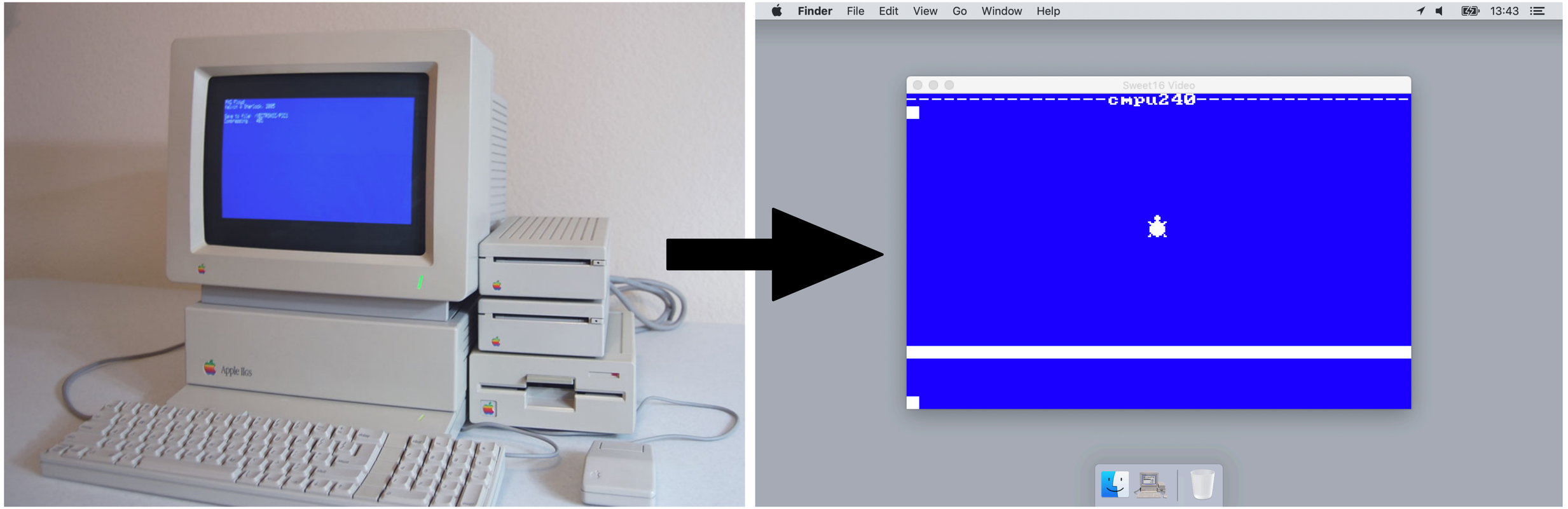
Computer technology changes quickly.

Studying theory enables you to understand the underlying models of *all* computation – not just technical details that become outdated in a few years.

A central idea in the theory of computation is that of a *universal computer*, a computer powerful enough to simulate any other computing device.

Most computers we encounter in everyday life are universal computers.

With the right software – and enough time and memory – they can simulate any other type of computer…

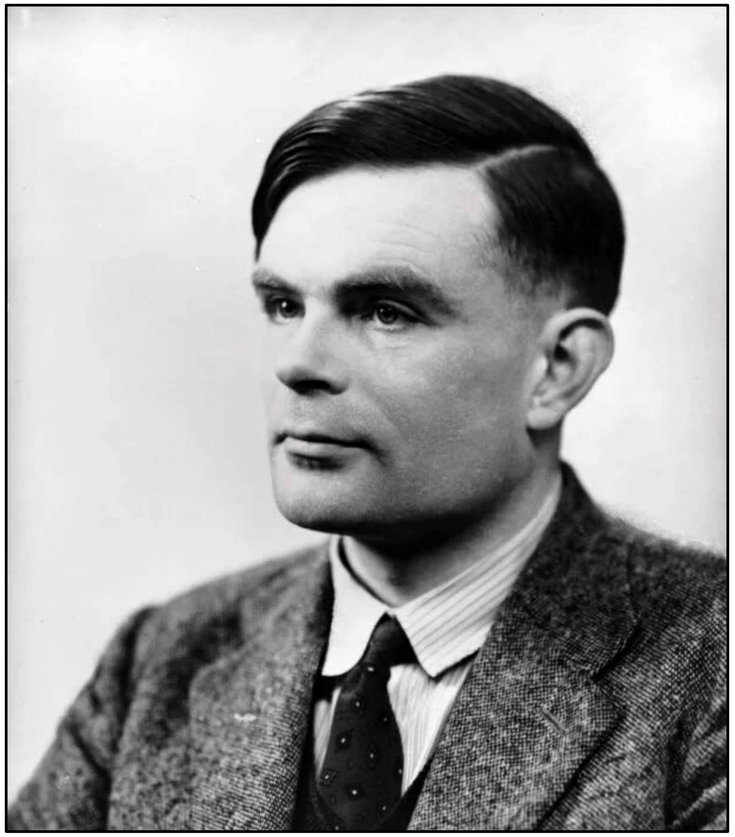

Finder File Edit View Go Window Help
13:43
Sweet16 Video
cmpu240
Apple IIgs

Universal computers

The idea of a universal computer was recognized and described in 1937 by Alan Turing.[1]

He called it a “universal machine” since at the time, “computer” still meant “a person who performs computations”.

1 Poor Alonzo Church is a footnote. Where’s his movie?



The idea of a universal computer was recognized and described in 1937 by Alan Turing.[1]

He called it a “universal machine” since at the time, “computer” still meant “a person who performs computations”.

1 Poor Alonzo Church is a footnote. Where’s his movie?

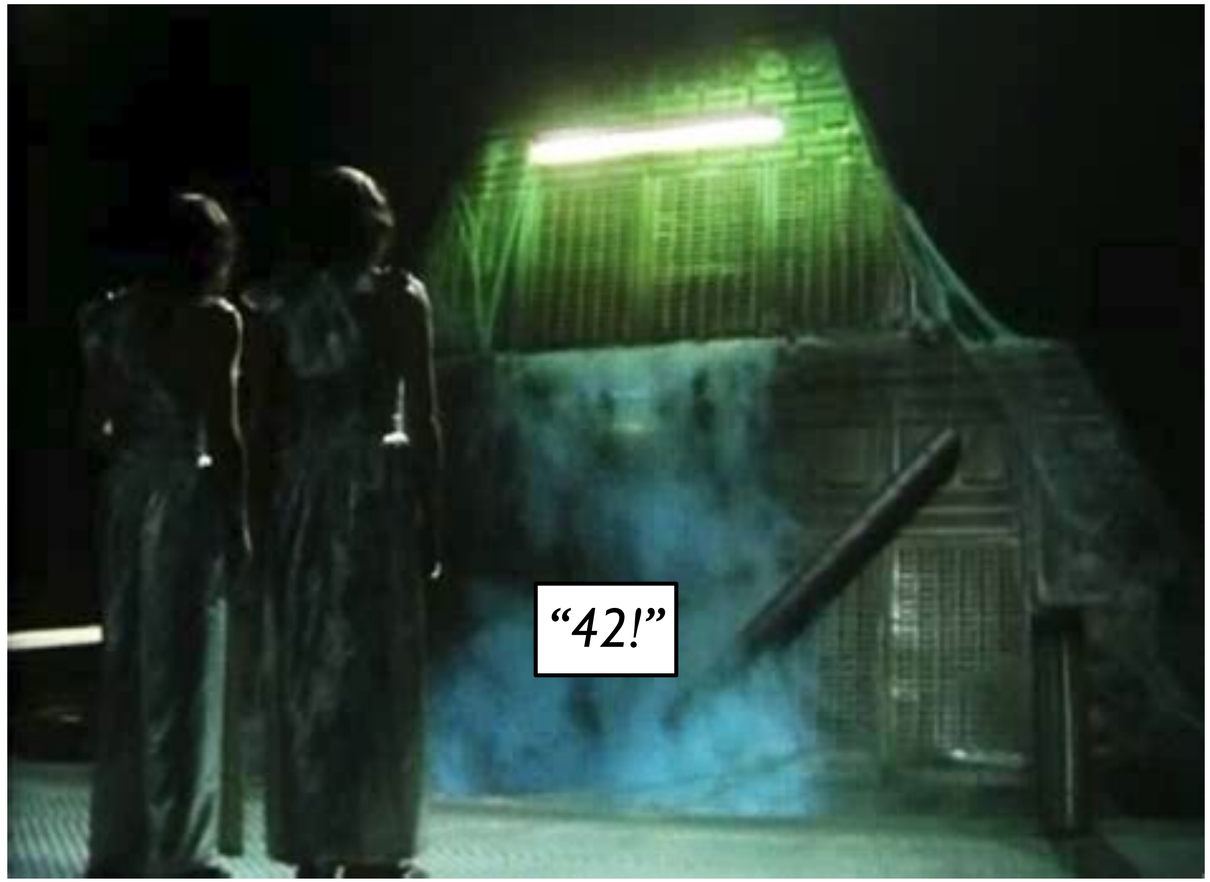
While a universal computer can compute anything that can be computed any other computing device, there are some things that are just impossible to compute.

Questions for which we lack data

“What is the winning number in tomorrow’s lottery?”

Vague questions

"What is the meaning of life?"



The Hitchhiker's Guide to the Galaxy, BBC, 1981

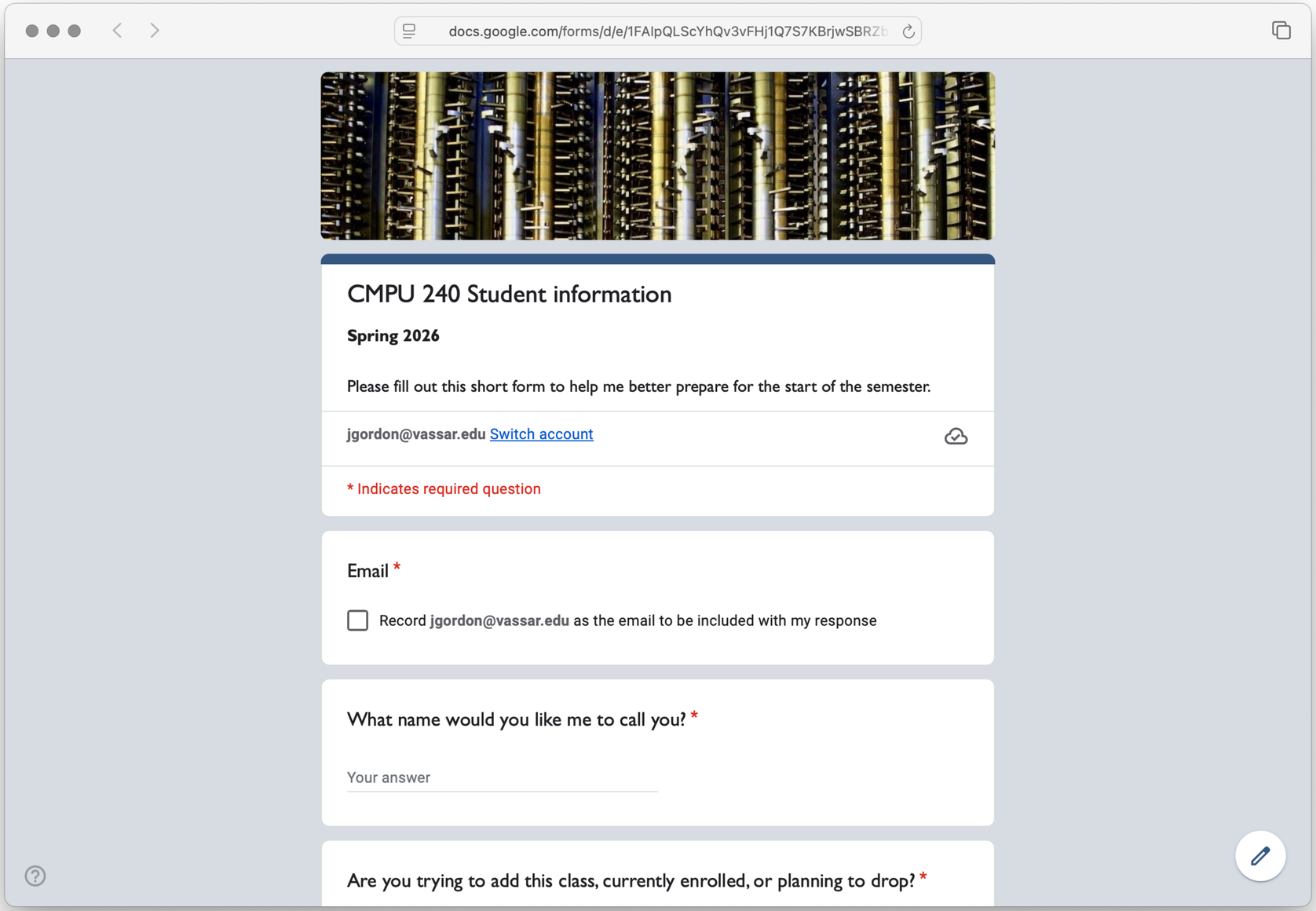
But there are also flawlessly defined computational problems that are impossible to solve.

We call these problems *noncomputable*.

What exactly are the limits to what a computer can do?

We'll work to an answer of this over the semester, taking us through philosophically interesting topics including nondeterminism, Turing machines, computability, and Gödel's incompleteness theorem.

Course information

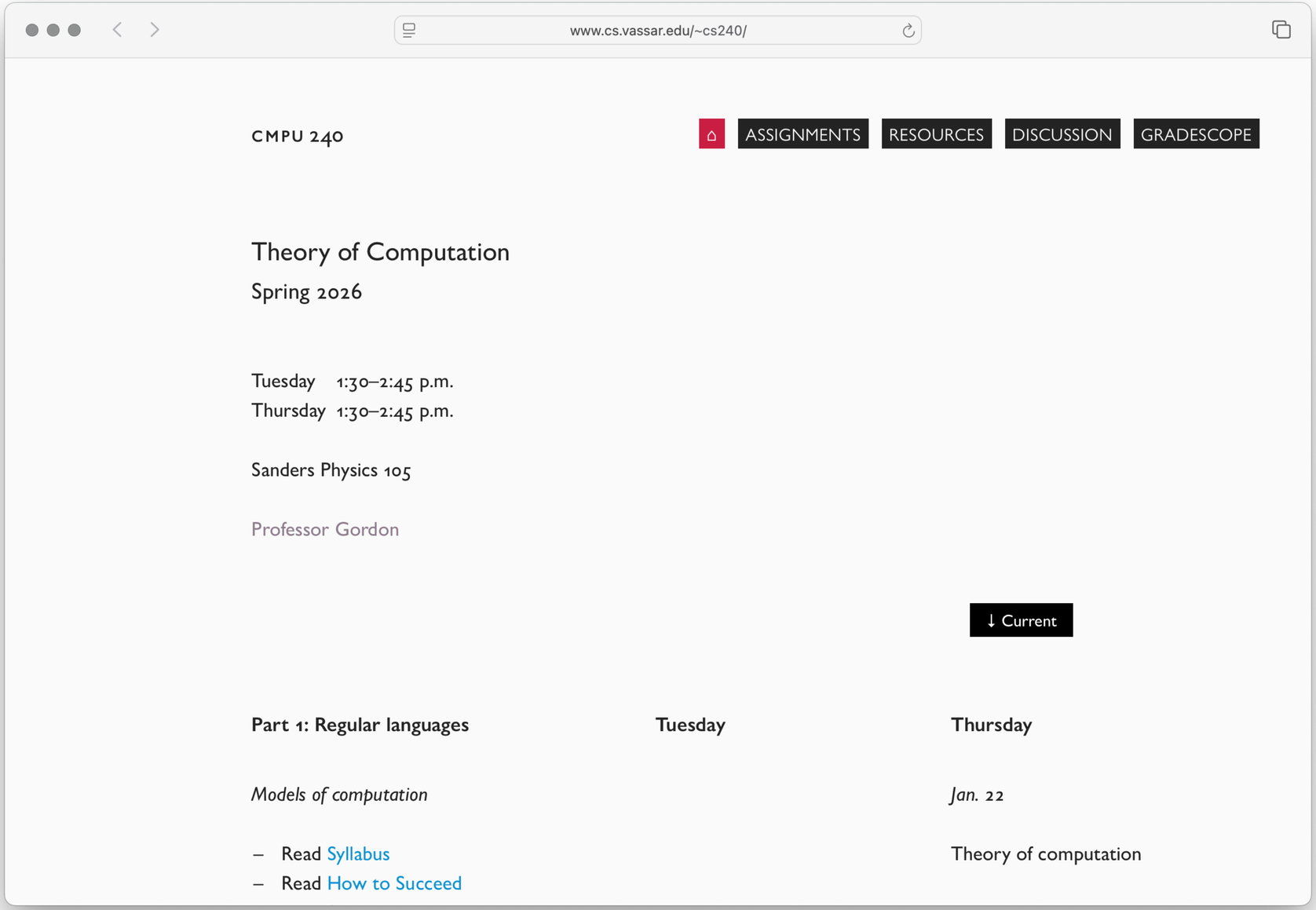


forms.gle/12LDiQRazPiKRY8w7

Prerequisites

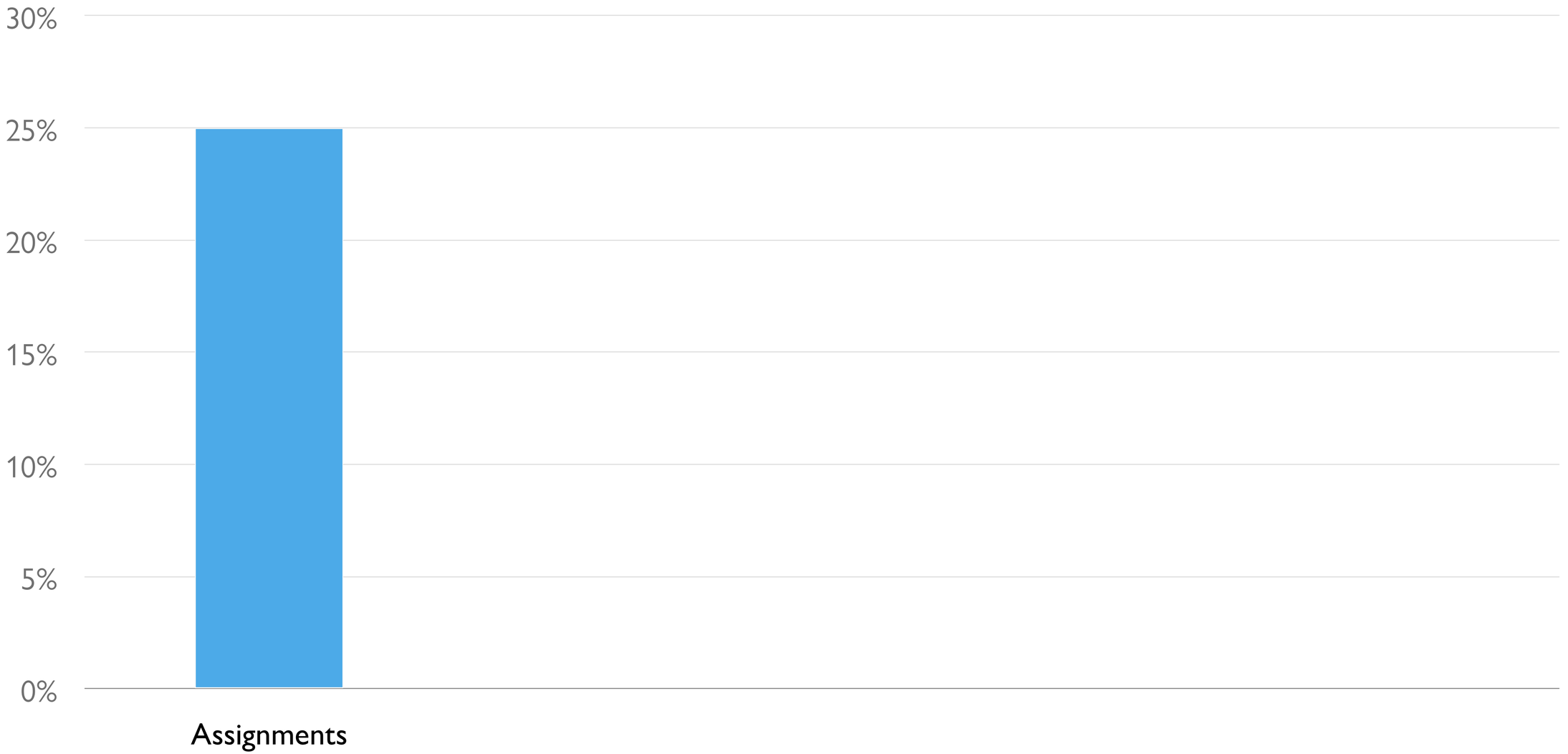
CMPU 102: Data Structures and Algorithms

CMPU 145: Foundations of Computer Science

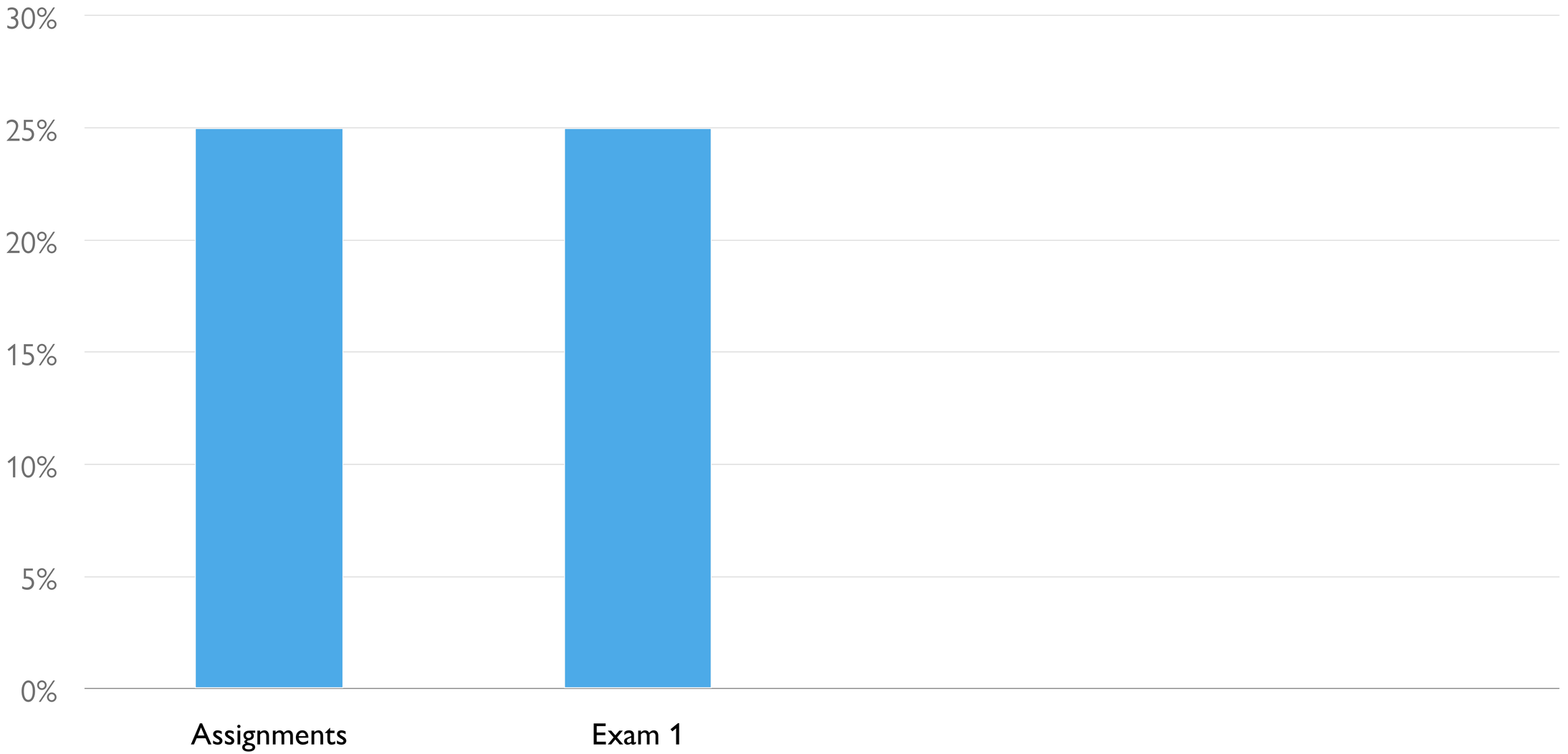


cs.vassar.edu/~cs240

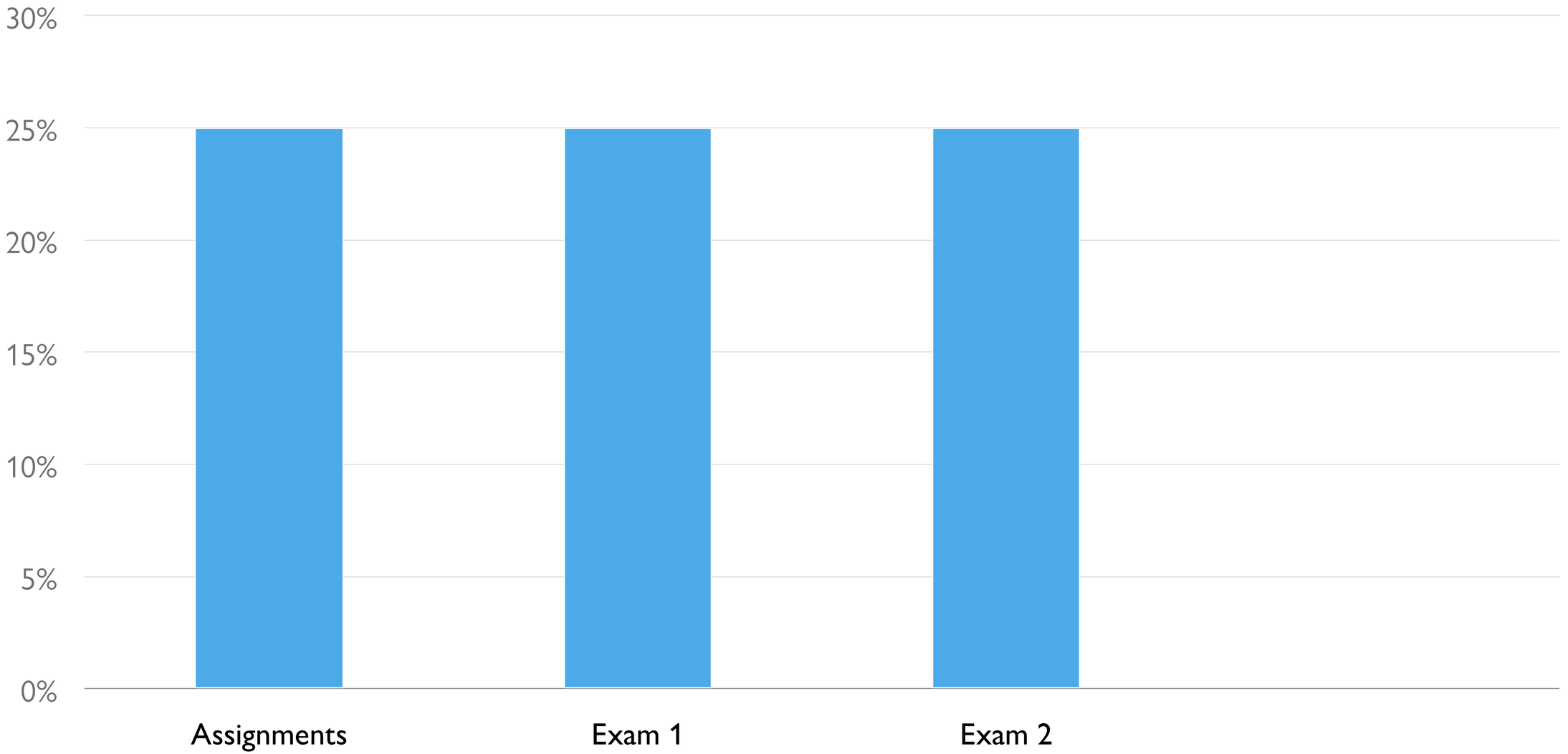
Grading



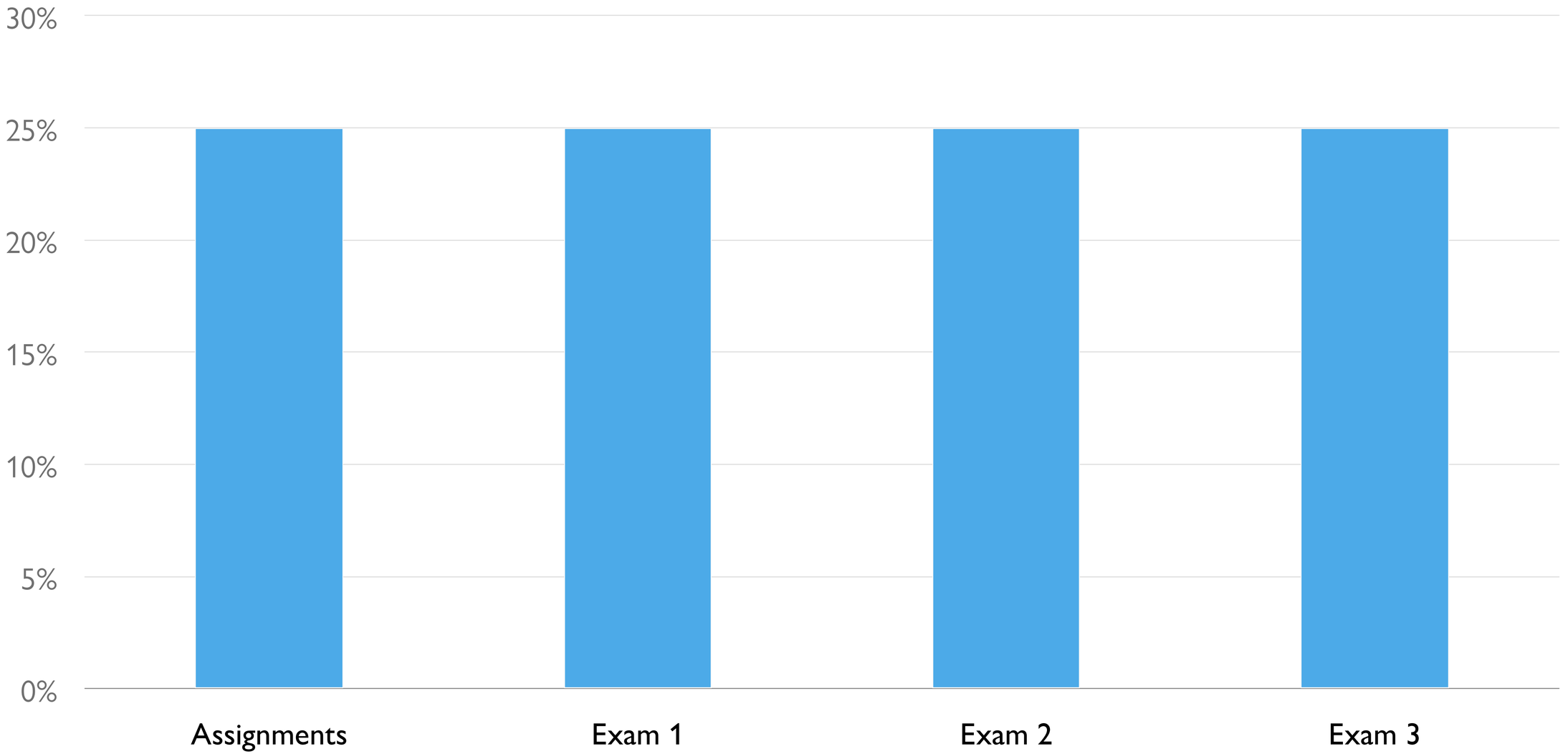
Grading



Grading

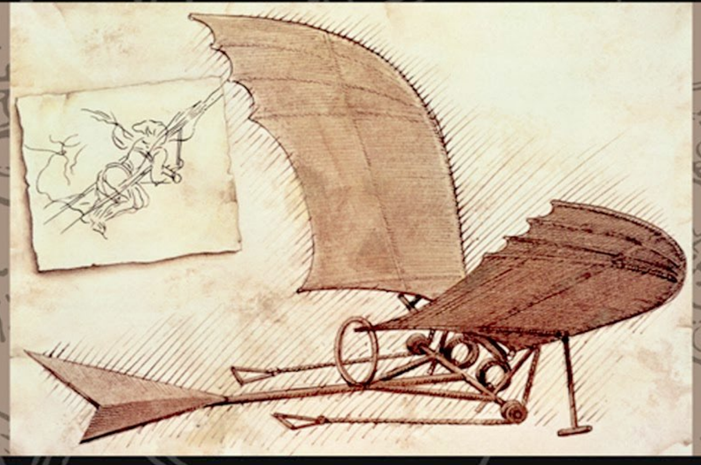


Grading



Introduction to the Theory of COMPUTATION

THIRD EDITION



MICHAEL SIPSER

bncvirtual.com/vb_buy2.php?ACTION=chooseAdoptions&CSID=2T2(

VASSAR AT THE JULIET COLLEGE STORE

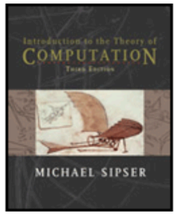
Log In to Your Account ☰ Menu Cart

Your Materials

We've made it easy to get your course materials for each class. Just review and select below.

PRICE MATCH GUARANTEE IS HERE! Excludes Marketplace items **Learn more**

Course 1 of 1: Spring 2026 • Vassar College • CMPU 240 51 Theory of Computation • 01/21/2026 - 05/19/2026 Hide



REQUIRED

Introduction to the Theory of Computation (Hardback) 3RD 13

Author:	**Sipser, Michael**
ISBN-13:	**978-1-133-18779-0**
ISBN-10:	**1-133-18779-X**
Edition/Copyright:	**3RD 13**
Publisher:	**Course Technology, Inc.**

Summary

Digital Rights

Requirements

Print

- ◉ **$274.25** Used Print
- ○ **$365.75** New Print

Digital Digital Rights

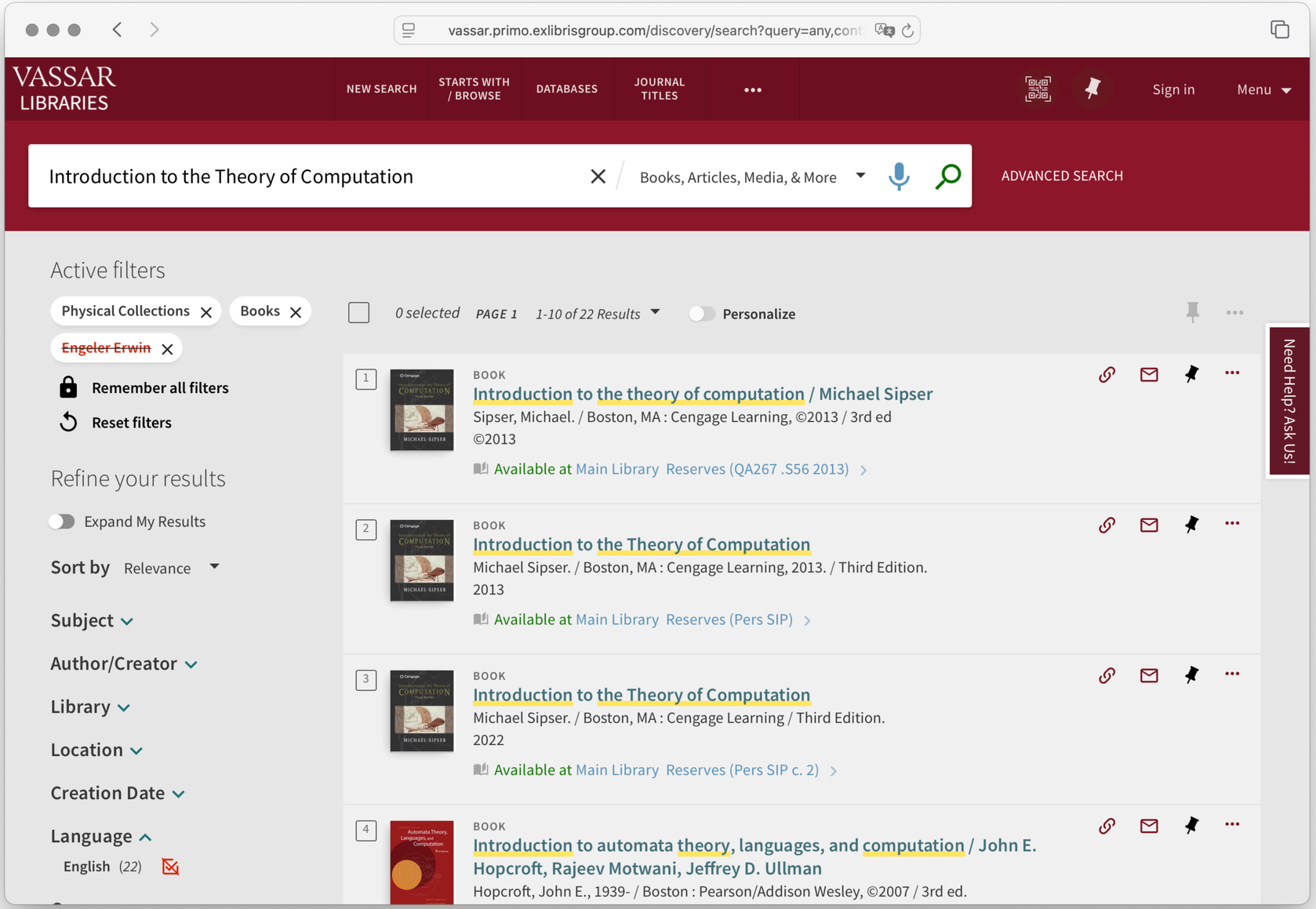
A digital delivery fee of $3.99 will be applied to each applicable digital material.

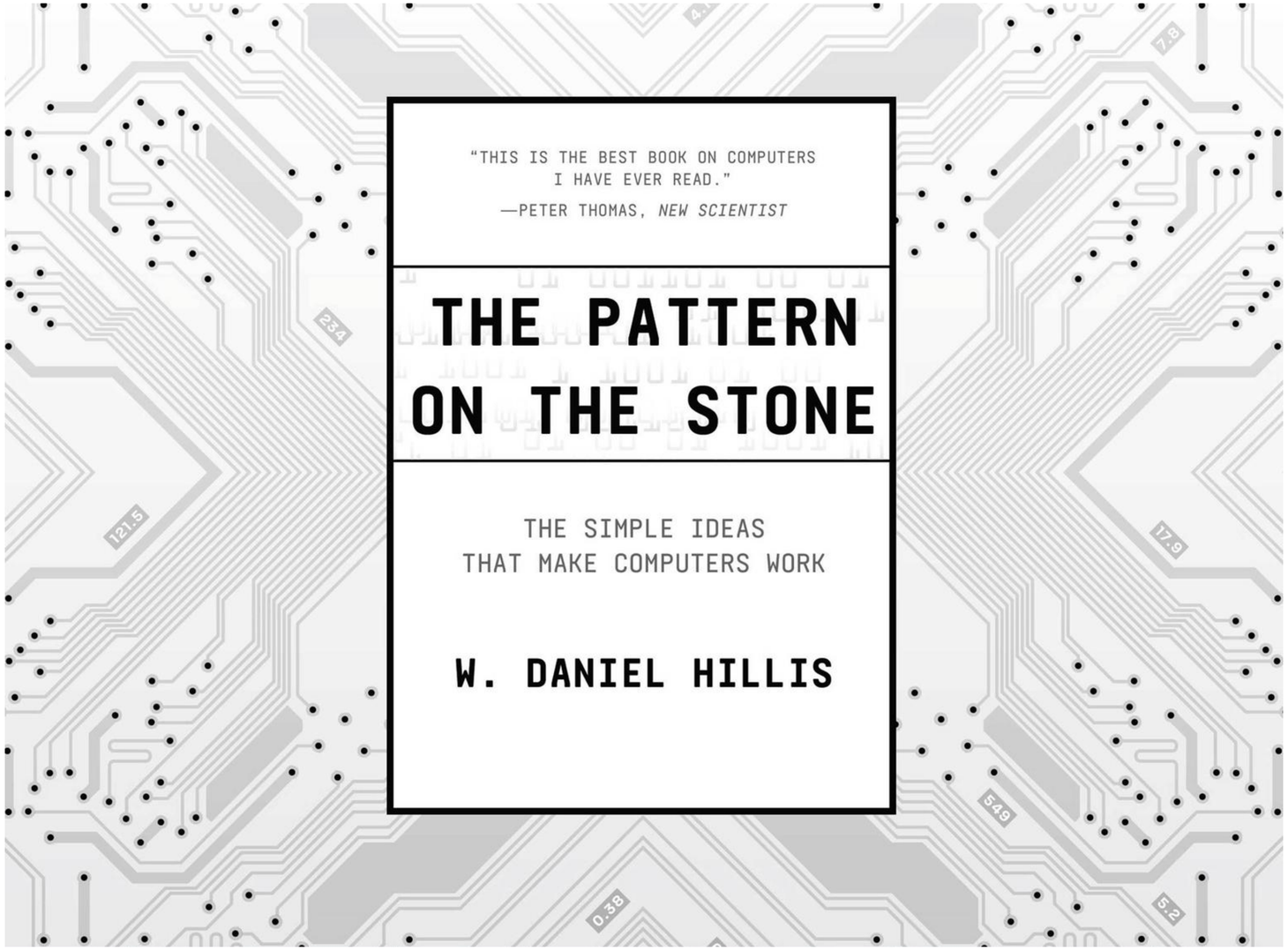
- ○ **$63.99** 180-Day eRental
- ○ **$87.99** 365-Day eRental
- ○ **$111.99** Perpetual Offline Access

Rental

- ○ **$158.38** Used Print Rental
  Return by 05/26/2026.
- ○ **$246.37** New Print Rental
  Return by 05/26/2026.

Marketplace Sellers


vassar.primo.exlibrisgroup.com/discovery/search?query=any,cont
VASSAR LIBRARIES
NEW SEARCH
STARTS WITH / BROWSE
DATABASES
JOURNAL TITLES
Sign in
Menu
Introduction to the Theory of Computation
Books, Articles, Media, & More
ADVANCED SEARCH
Active filters
Physical Collections
Books
Engeler Erwin
Remember all filters
Reset filters
Refine your results
Expand My Results
Sort by Relevance
Subject
Author/Creator
Library
Location
Creation Date
Language
English (22)
0 selected
PAGE 1
1-10 of 22 Results
Personalize
1
BOOK
Introduction to the theory of computation / Michael Sipser
Sipser, Michael. / Boston, MA : Cengage Learning, ©2013 / 3rd ed
©2013
Available at Main Library Reserves (QA267 .S56 2013)
2
BOOK
Introduction to the Theory of Computation
Michael Sipser. / Boston, MA : Cengage Learning, 2013. / Third Edition.
2013
Available at Main Library Reserves (Pers SIP)
3
BOOK
Introduction to the Theory of Computation
Michael Sipser. / Boston, MA : Cengage Learning / Third Edition.
2022
Available at Main Library Reserves (Pers SIP c. 2)
4
BOOK
Introduction to automata theory, languages, and computation / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman
Hopcroft, John E., 1939- / Boston : Pearson/Addison Wesley, ©2007 / 3rd ed.
Need Help? Ask Us!


"THIS IS THE BEST BOOK ON COMPUTERS
I HAVE EVER READ."
—PETER THOMAS, NEW SCIENTIST
THE PATTERN
ON THE STONE
THE SIMPLE IDEAS
THAT MAKE COMPUTERS WORK
W. DANIEL HILLIS

Hillis's Connection Machine CM-2a

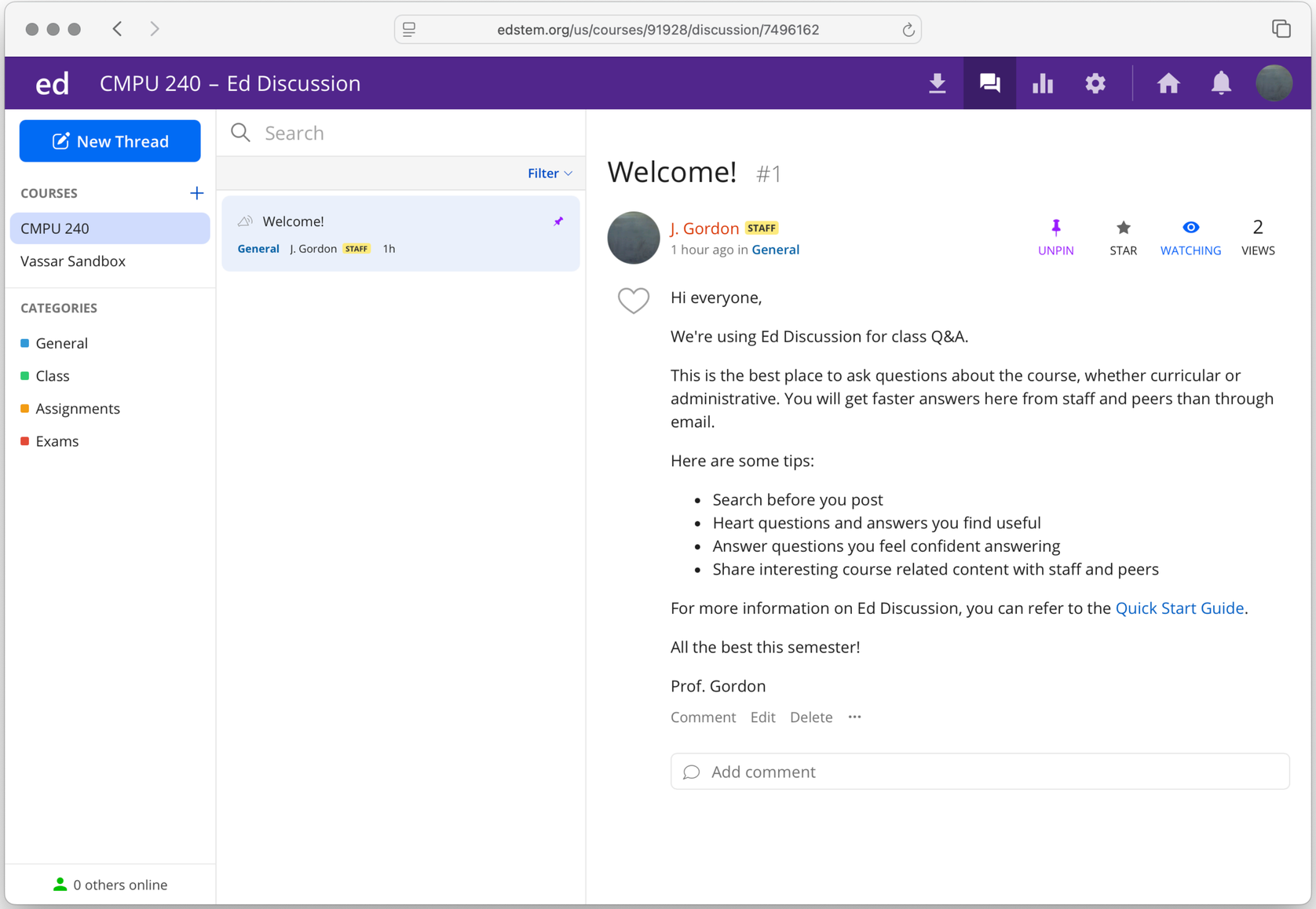
Photo by Steve Grohe for Thinking Machines Corporation

Hillis's Connection Machine CM-5 in Jurassic Park

Beware of strange computers offering you candy

You can use ChatGPT, Claude, Gemini, etc. when studying – but realize that they *will* sometimes give you incorrect explanations and proofs!

You *cannot* use it on assignments or exams – these are expected to be your own work, and copying or adapting answers from generative AI is no different than copying from a friend or a random website.


edstem.org/us/courses/91928/discussion/7496162
ed
CMPU 240 – Ed Discussion
New Thread
Search
COURSES
CMPU 240
Vassar Sandbox
CATEGORIES
General
Class
Assignments
Exams
Filter
Welcome!
General J. Gordon STAFF 1h
Welcome! #1
J. Gordon STAFF
1 hour ago in General
UNPIN
STAR
WATCHING
2
VIEWS
Hi everyone,
We're using Ed Discussion for class Q&A.
This is the best place to ask questions about the course, whether curricular or administrative. You will get faster answers here from staff and peers than through email.
Here are some tips:
Search before you post
Heart questions and answers you find useful
Answer questions you feel confident answering
Share interesting course related content with staff and peers
For more information on Ed Discussion, you can refer to the Quick Start Guide.
All the best this semester!
Prof. Gordon
Comment Edit Delete
Add comment
0 others online

edstem.org

cs.vassar.edu/~cs240/resources/syllabus.pdf

CMPU 240

Theory of Computation

Spring 2026

Tuesday 1:30–2:45 p.m.
Thursday 1:30–2:45 p.m.

Sanders Physics 105

Professor Gordon

cs.vassar.edu/~cs240

Overview

Are there fundamental restrictions to what computers can and cannot do? If so, what do these restrictions look like? And what would such restrictions mean for our ability to computationally solve meaningful problems?

In CMPU 240, we'll explore the answers to these important questions. We'll consider what is an appropriate mathematical model of a computer and what types of computations are and are not possible

What problems can we solve with a computer?

What problems can we solve with a computer?

What kind of computer?

Two challenges

Computers are dramatically better now than they've ever been, and we expect that trend to continue!

Writing proofs on formal definitions is hard – and today's computers are already *way* more complicated than the sets, graphs, or functions you wrote proofs about in CMPU 145.

How can we prove what computers can and can’t do…

…so our results are still true in 20 years?

…without multi-hundred-page proofs?

Enter automata

An *automaton* is a mathematical model of a computing device.

It's an abstraction of a real computer, like how graphs are abstractions of social networks, transportation grids, etc.

The preferred plural is automata rather than automatons.

The automata we'll explore are

Powerful enough to capture huge classes of computer devices, but

Simple enough that we can reason about them in a small space!

What do these automata look like?

A tale of two computers

A tale of two computers

A tale of two computers

Basic Calculator	*Laptop*
Small amount of memory	Large amount of memory
Fixed set of functions	Reprogrammable; run lots of different programs

Basic Calculator

Small amount of memory

Fixed set of functions

Laptop

Large amount of memory

Reprogrammable; run lots of different programs

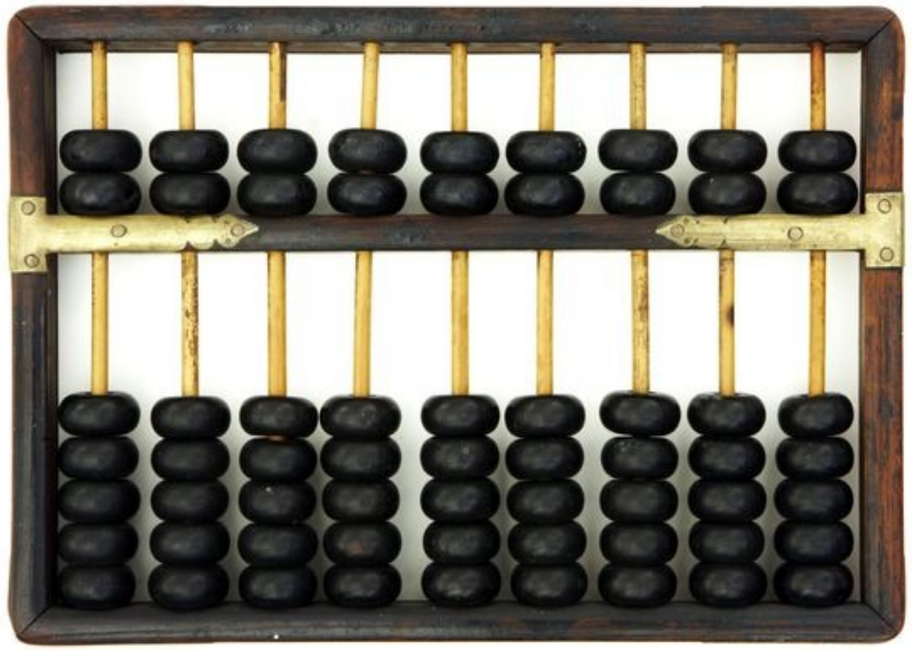
Computing with finite memory


%
÷
×
7
8
9
4
5
6
1
2
3
0
•
=
+
−
C/CE

Data stored electronically

Algorithm is in silicon

Memory limited by the display!



Data stored electronically

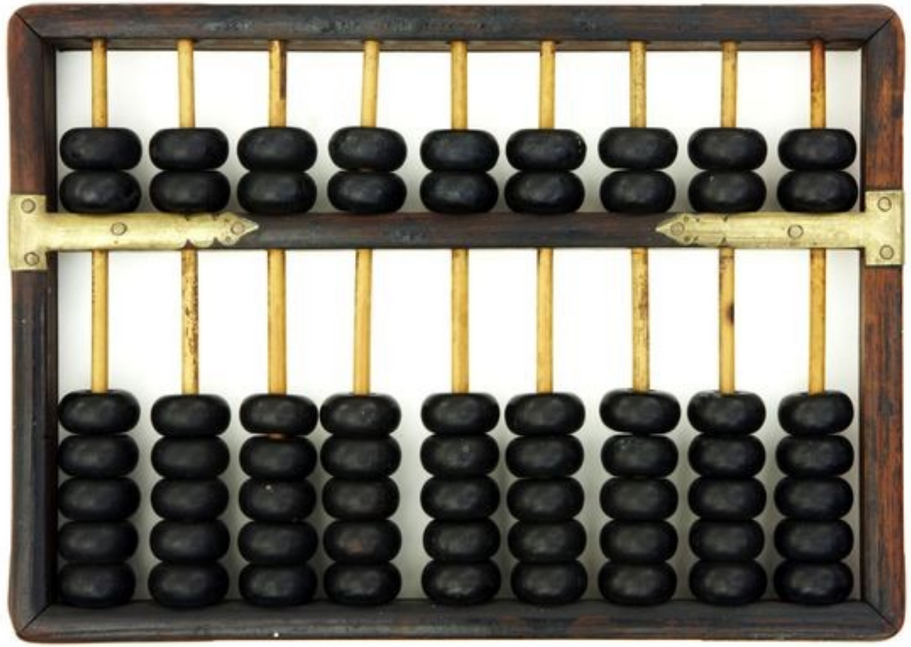
Algorithm is in silicon

Memory limited by the display!

Data stored electronically

Algorithm is in silicon

Memory limited by the display!

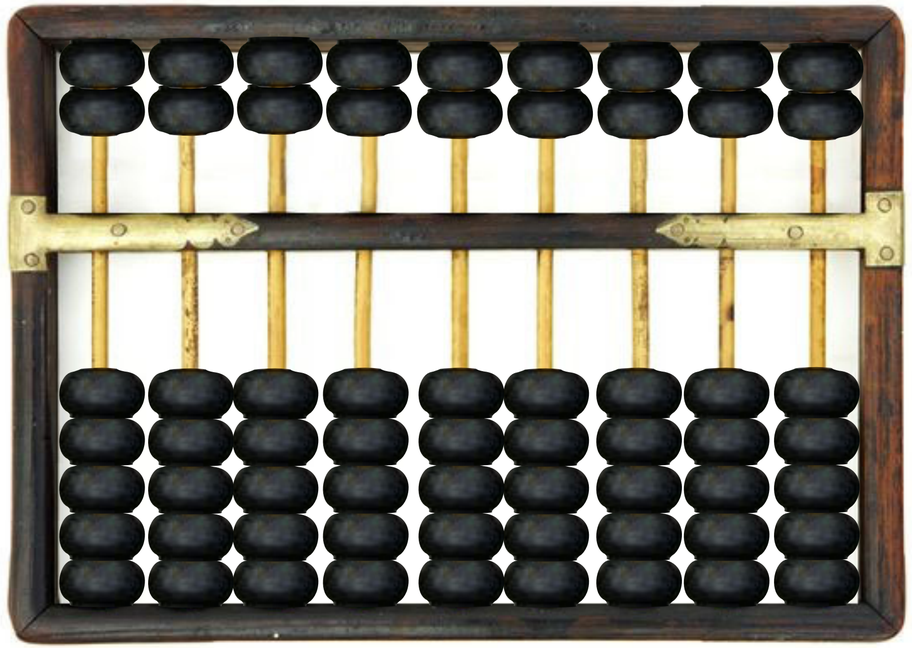


Data stored in wood

Algorithm is in your brain

Memory limited by the number of beads!

How do we model “memory” and “an algorithm”
when they can take on so many forms?



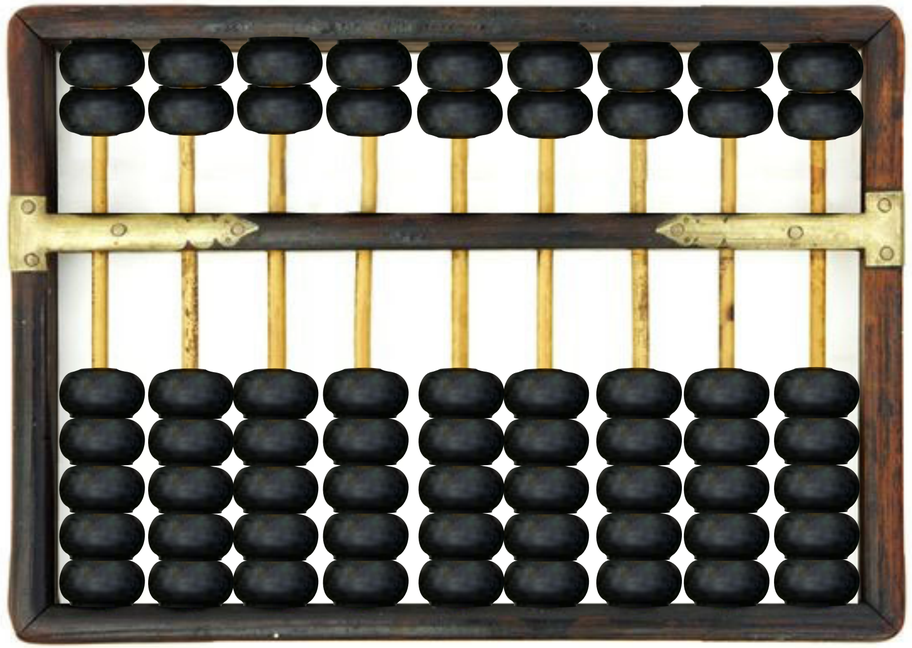
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



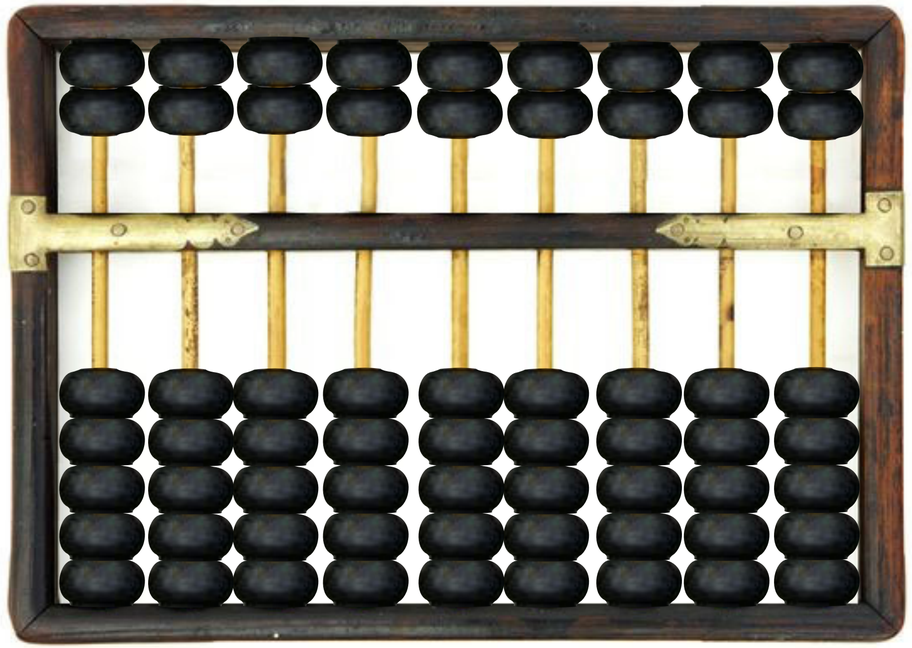
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



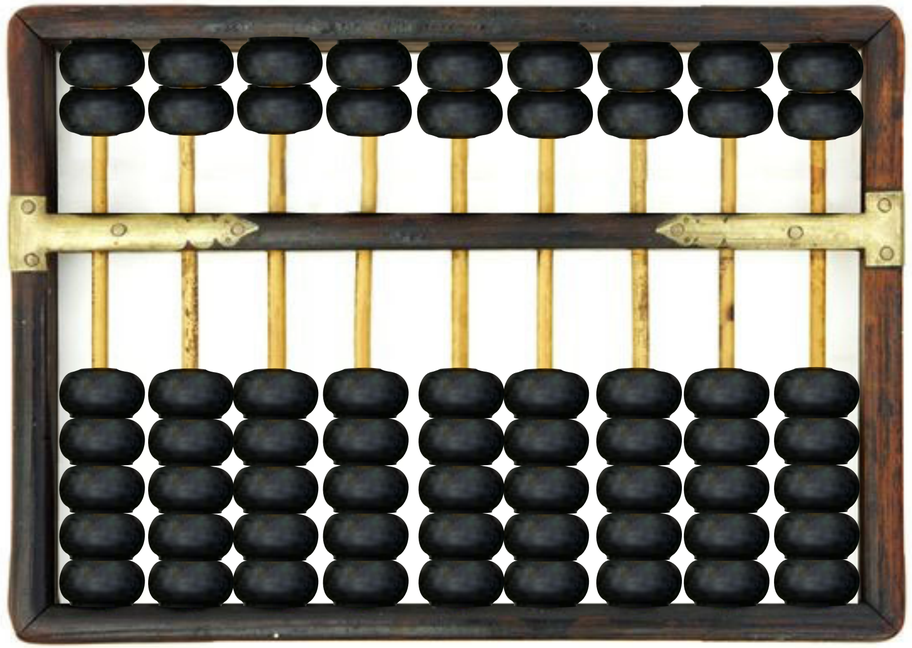
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



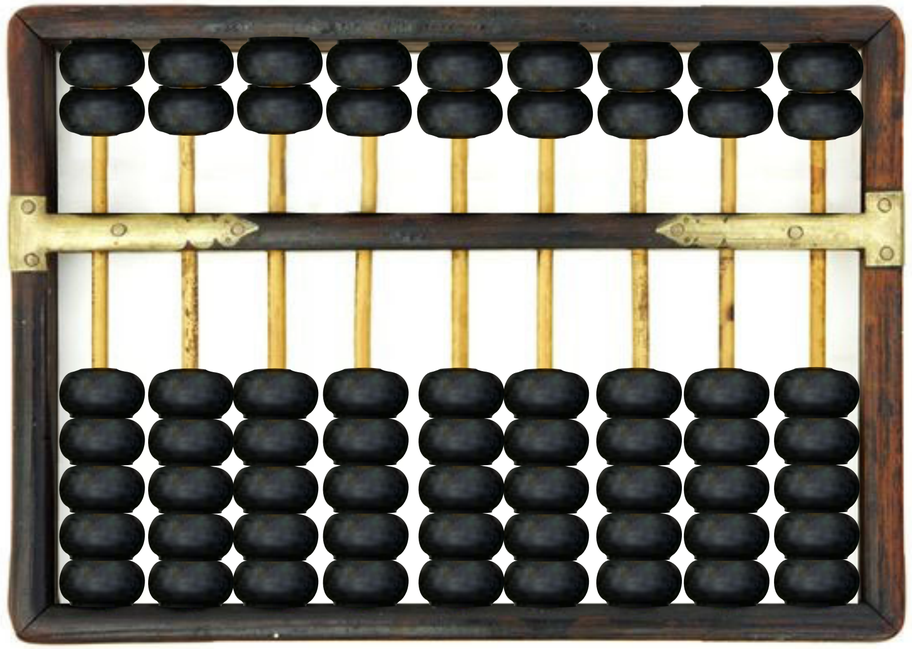
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



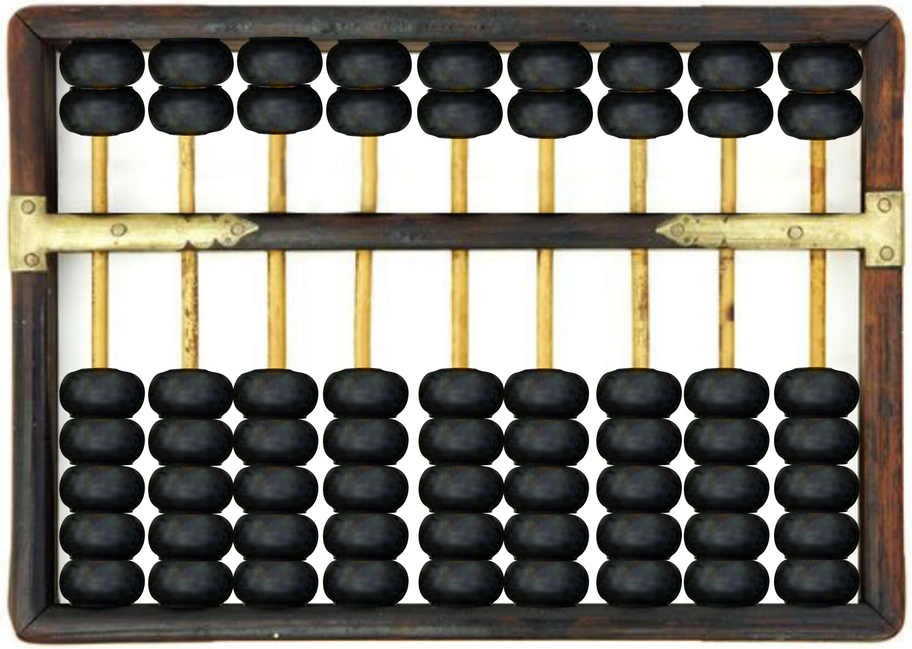
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



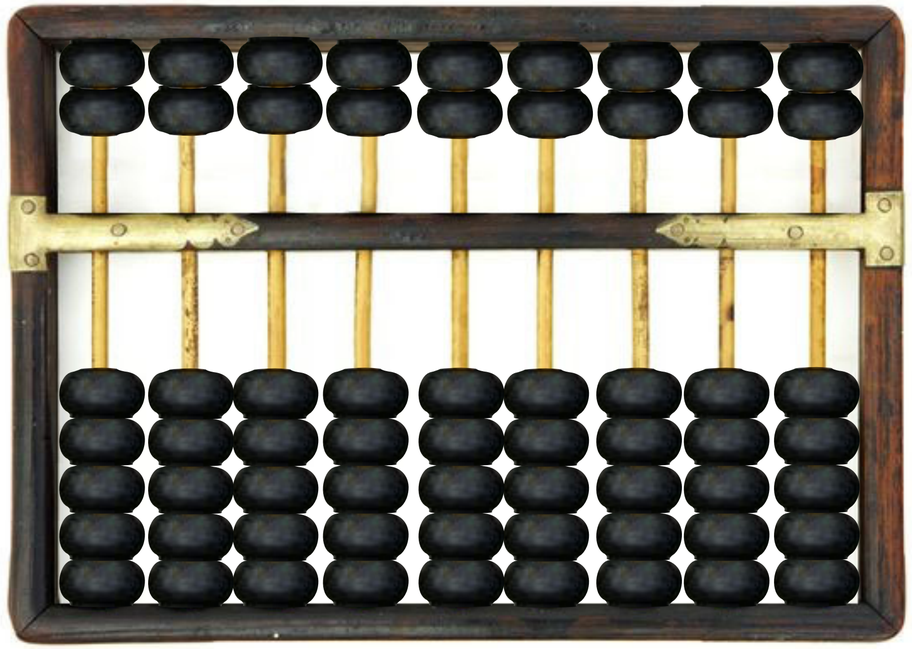
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



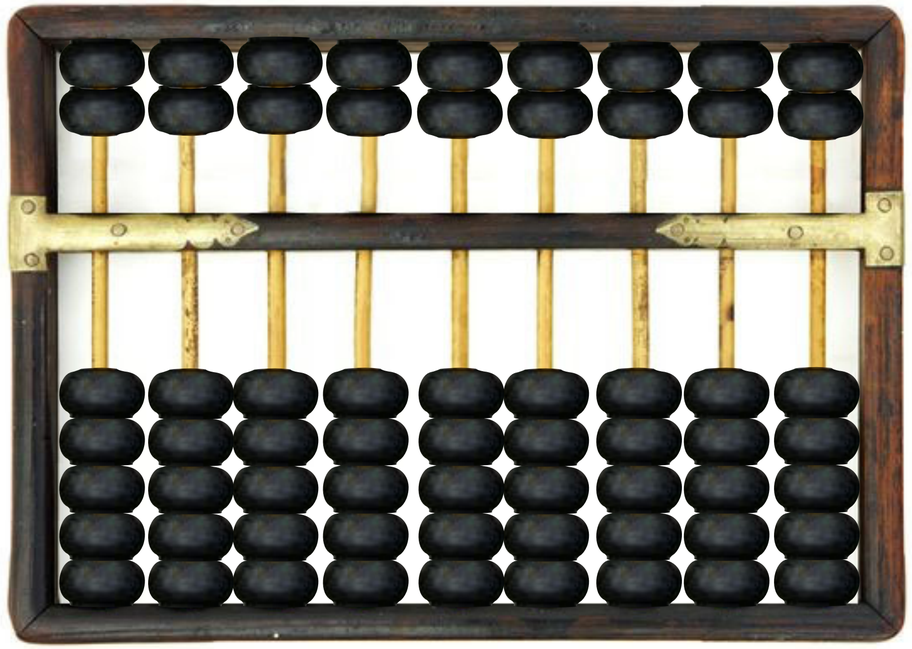
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



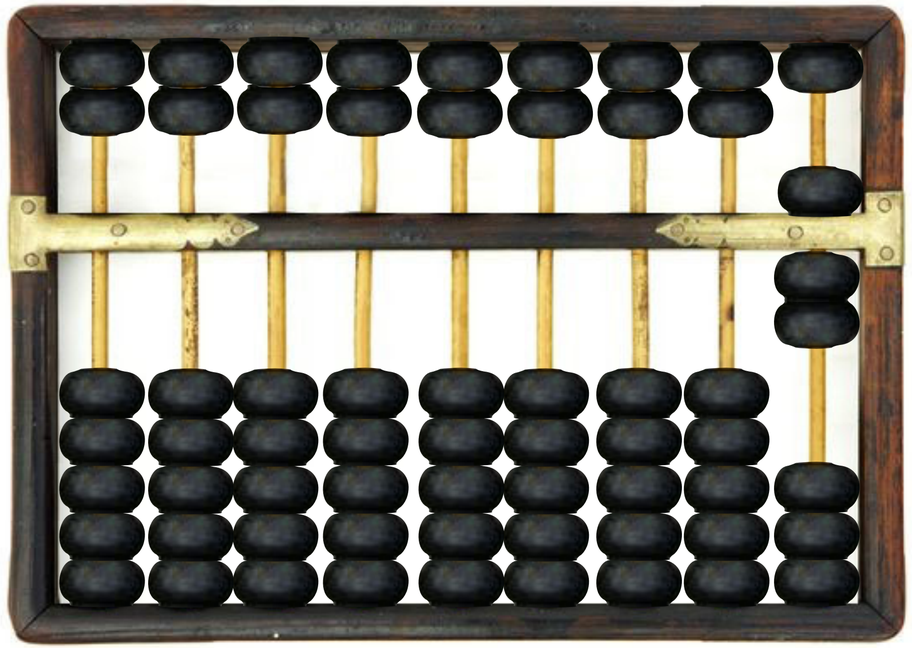
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



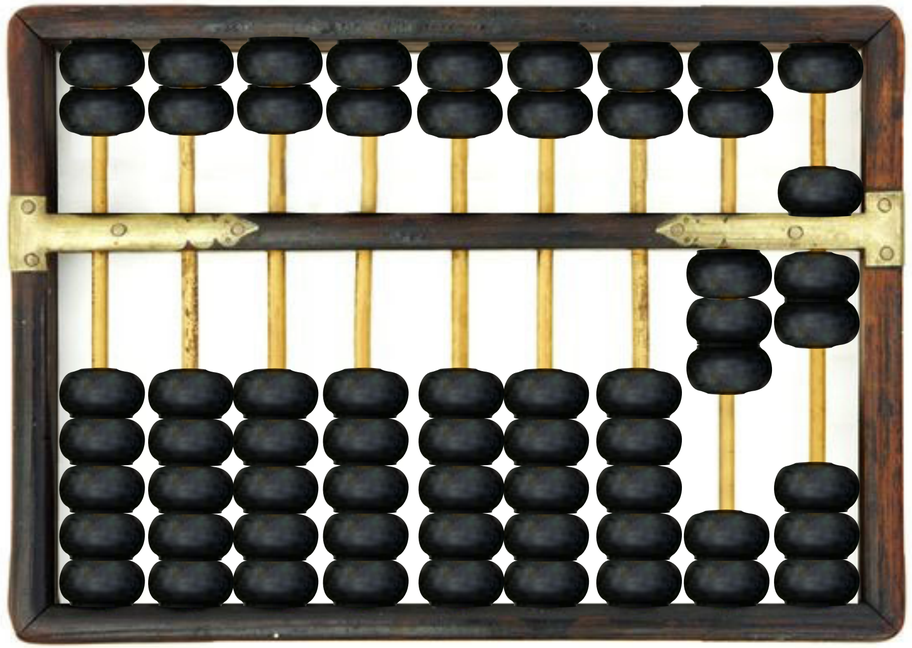
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



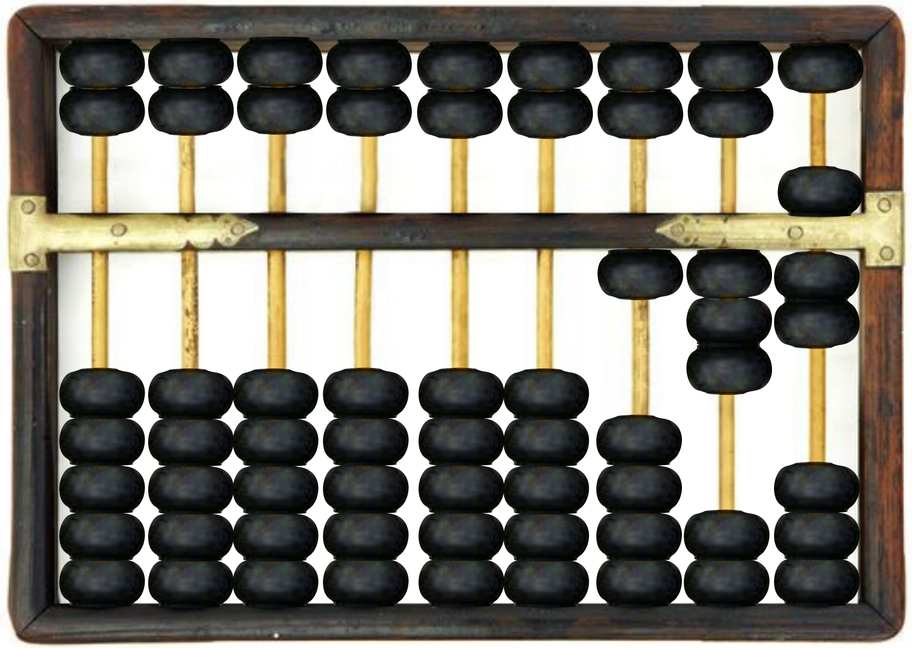
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



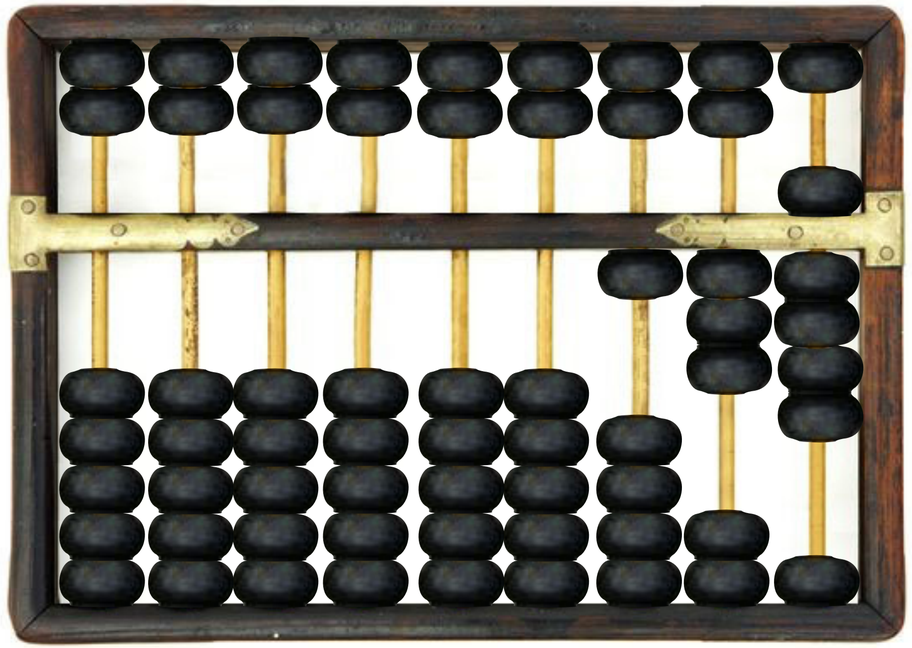
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



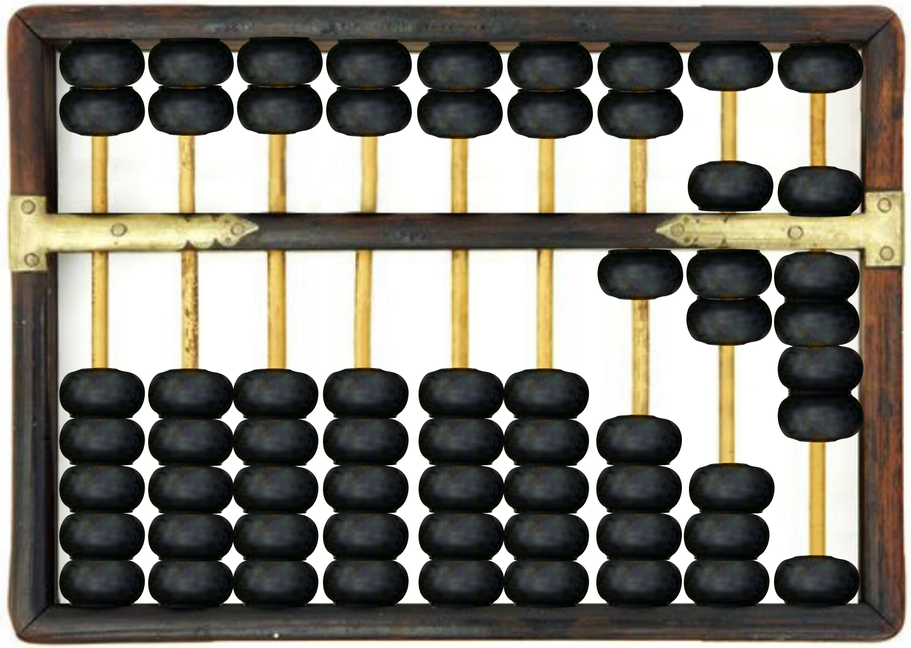
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



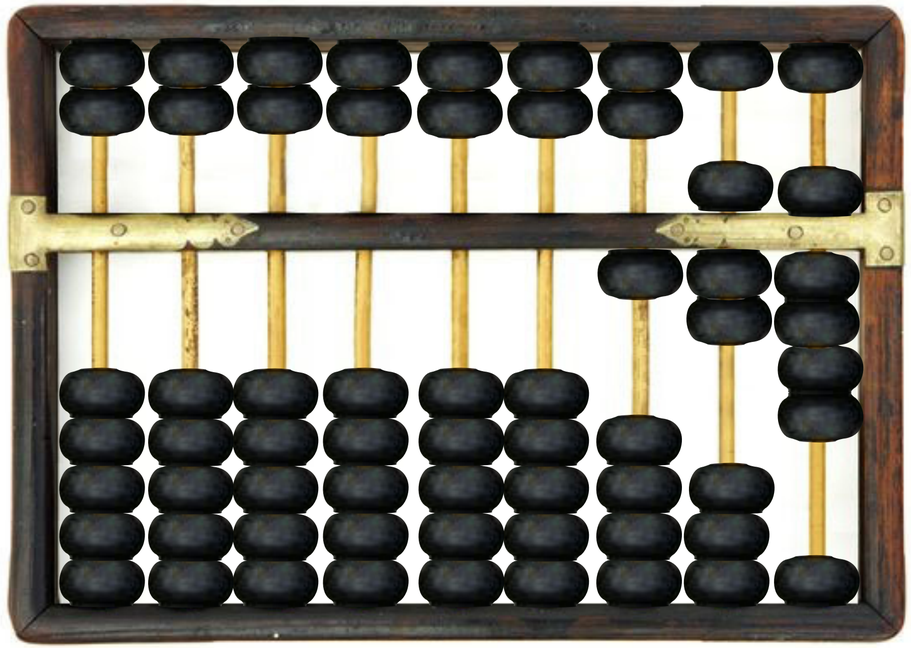
Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

This change, big or small, is where the computation happens!



Focus on what's in common:

The machines *receive input* from an external source.

That input is provided *sequentially*, one discrete unit at a time.

Each input causes the device to *change configuration*.

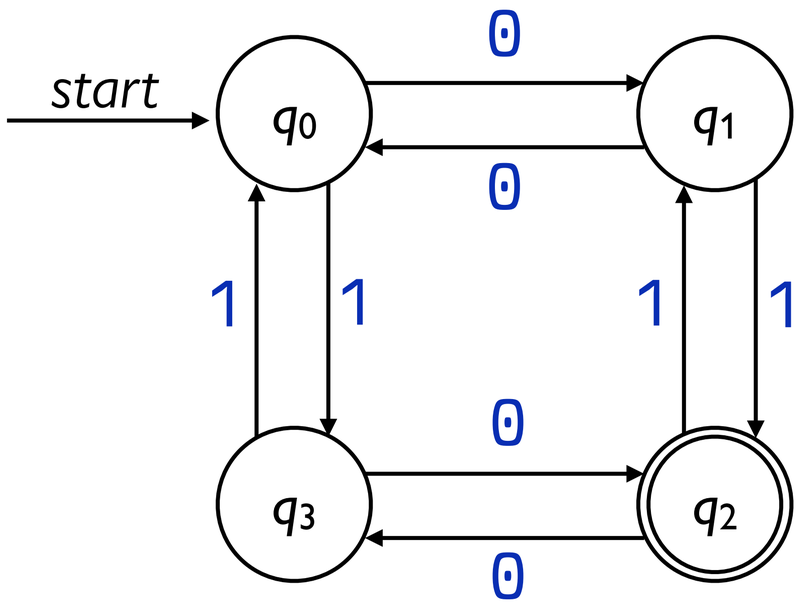
This change, big or small, is where the computation happens!

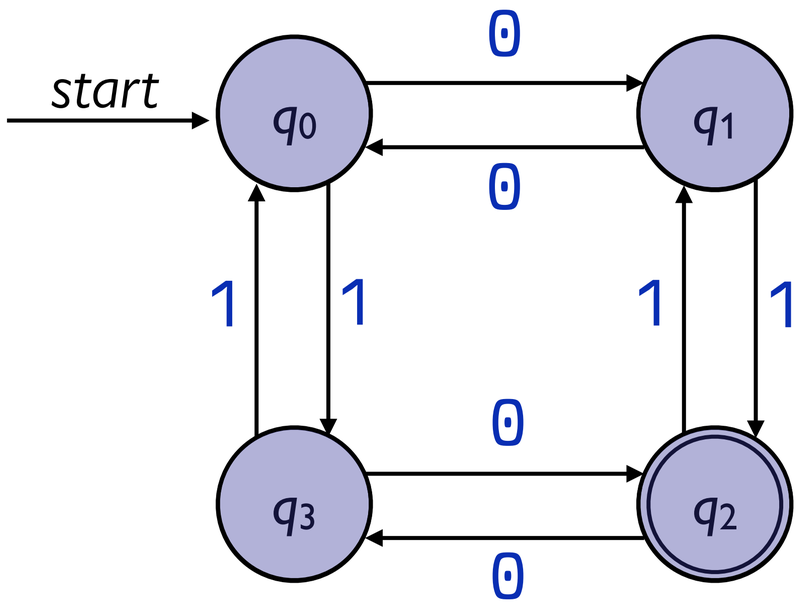
Once all input is provided, we can *read off an answer* based on the configuration of the device.

Finite automata

A *finite automaton* – also called a *finite-state machine* – is the simplest model of a computer that's still interesting to study.

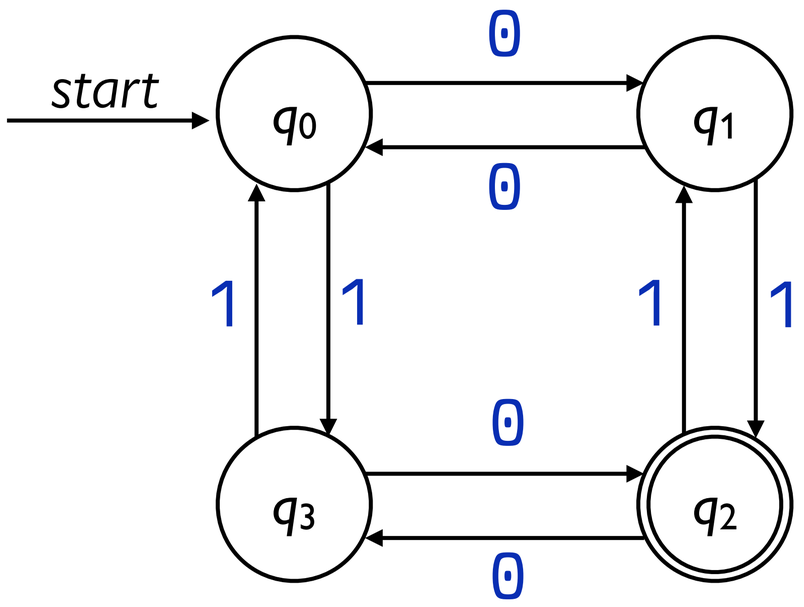
A finite automaton consists of a set of *states* connected by *transitions*.

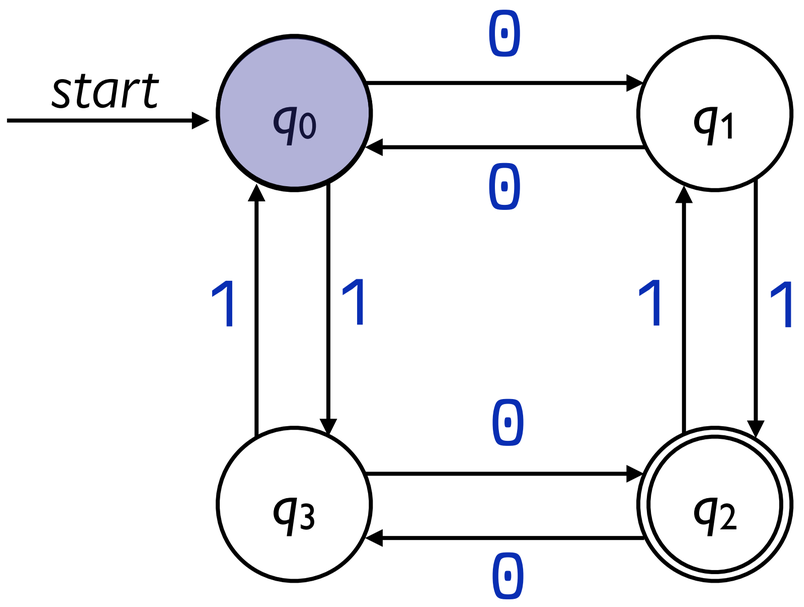

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0



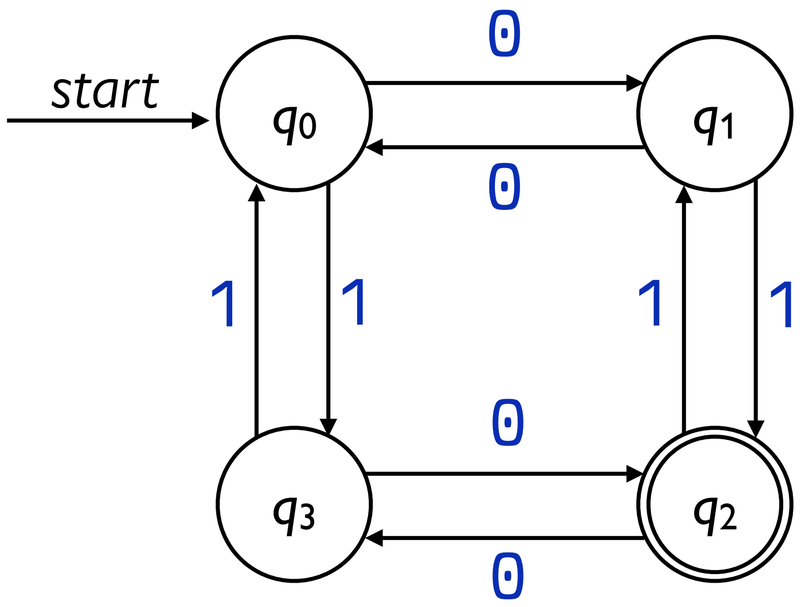
Each circle represents a

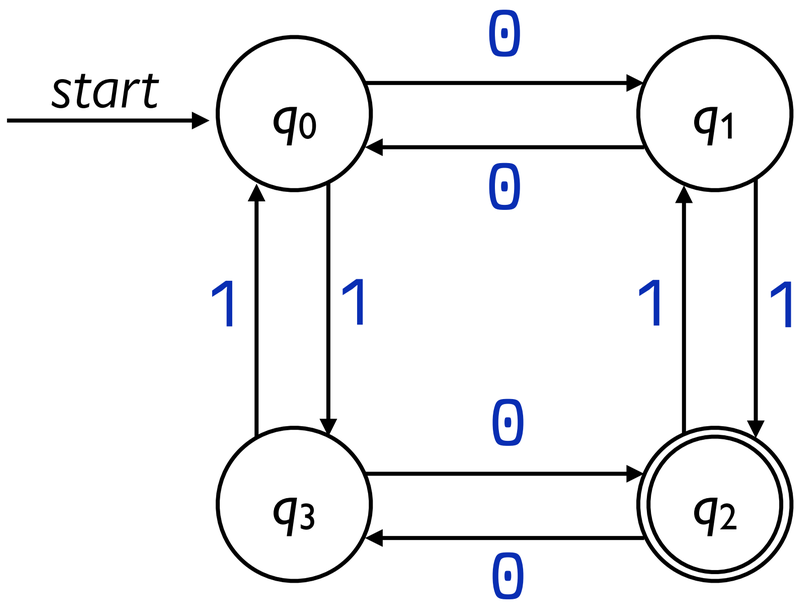
***state** of the automaton*


start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

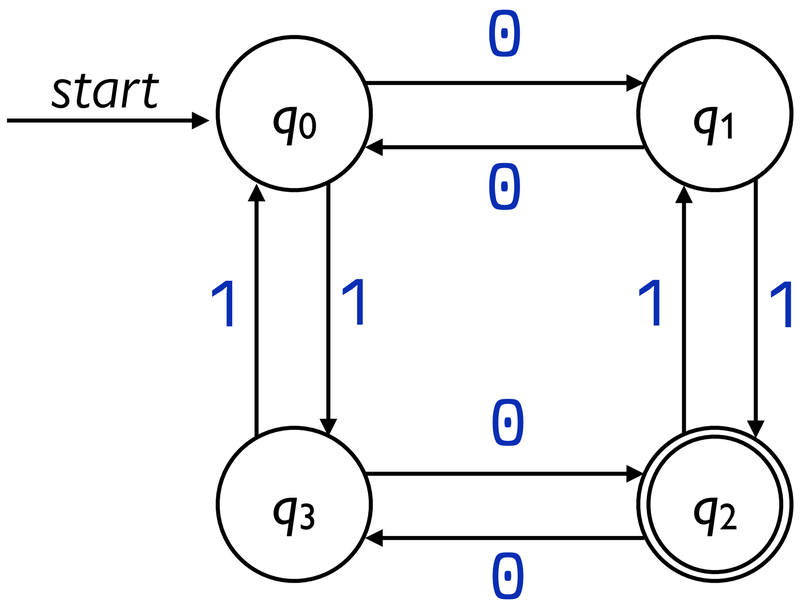


One state is designated as the ***start state****, indicated by an arrow*


start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

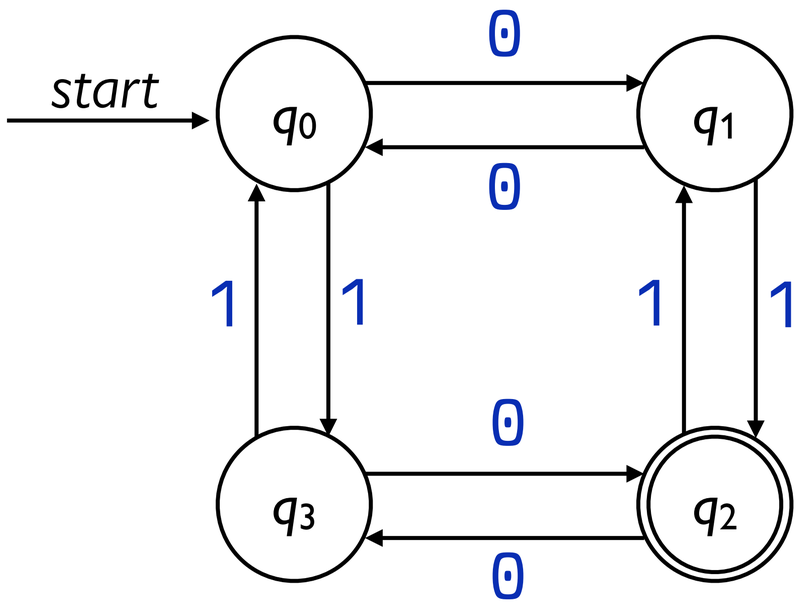


0 1 0 1 1 0

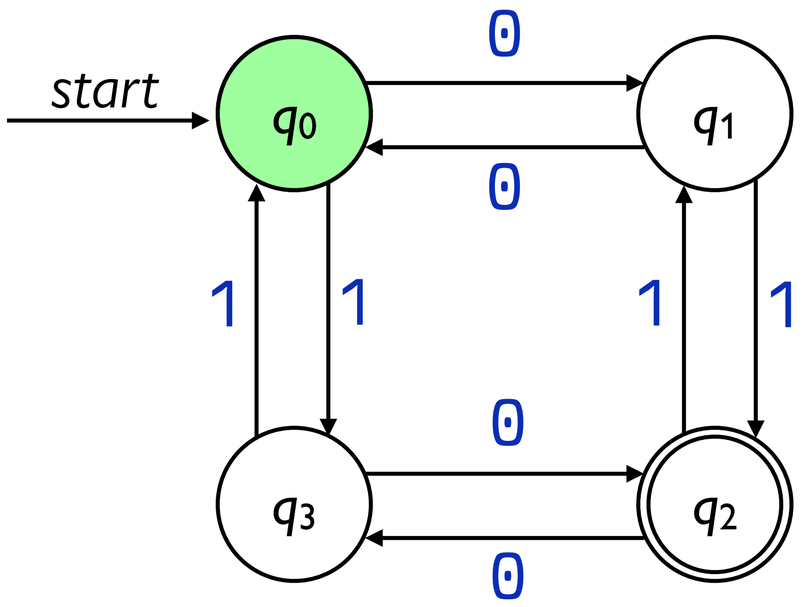


0 1 0 1 1 0

*The automaton is run on an **input string**.*

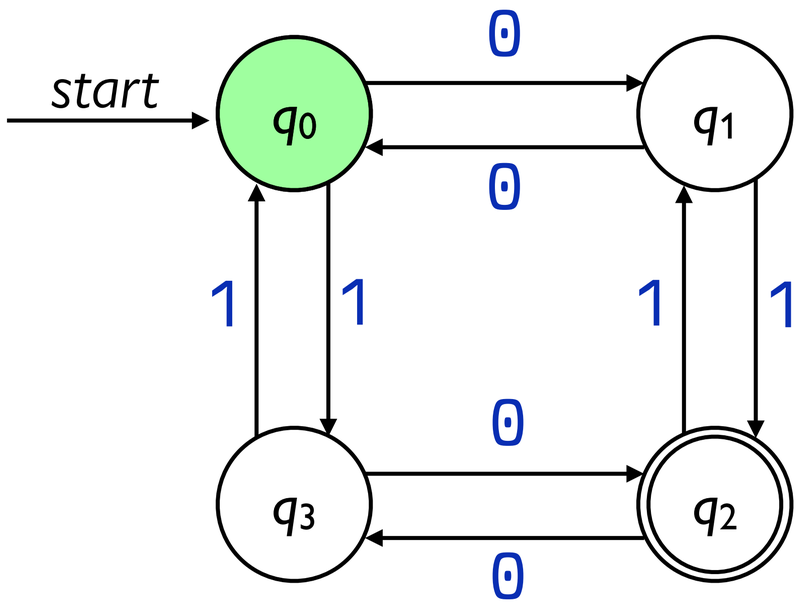


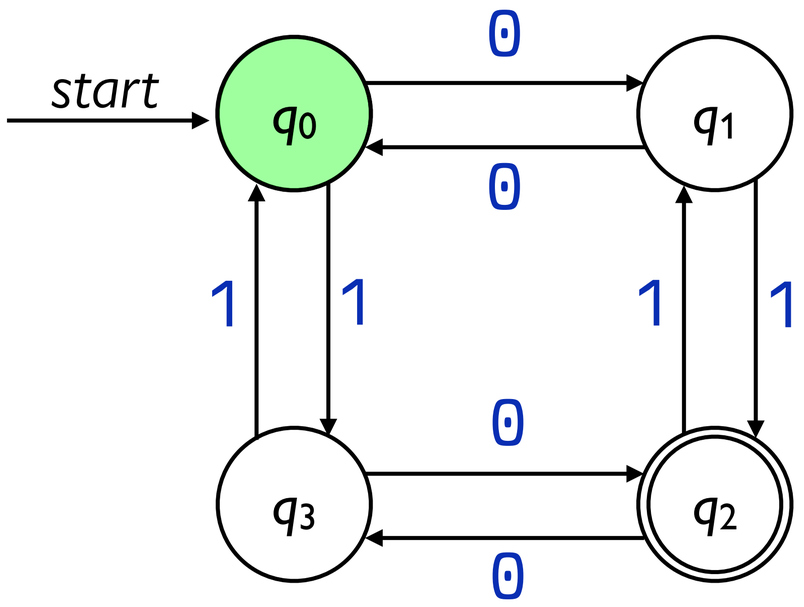
0 1 0 1 1 0



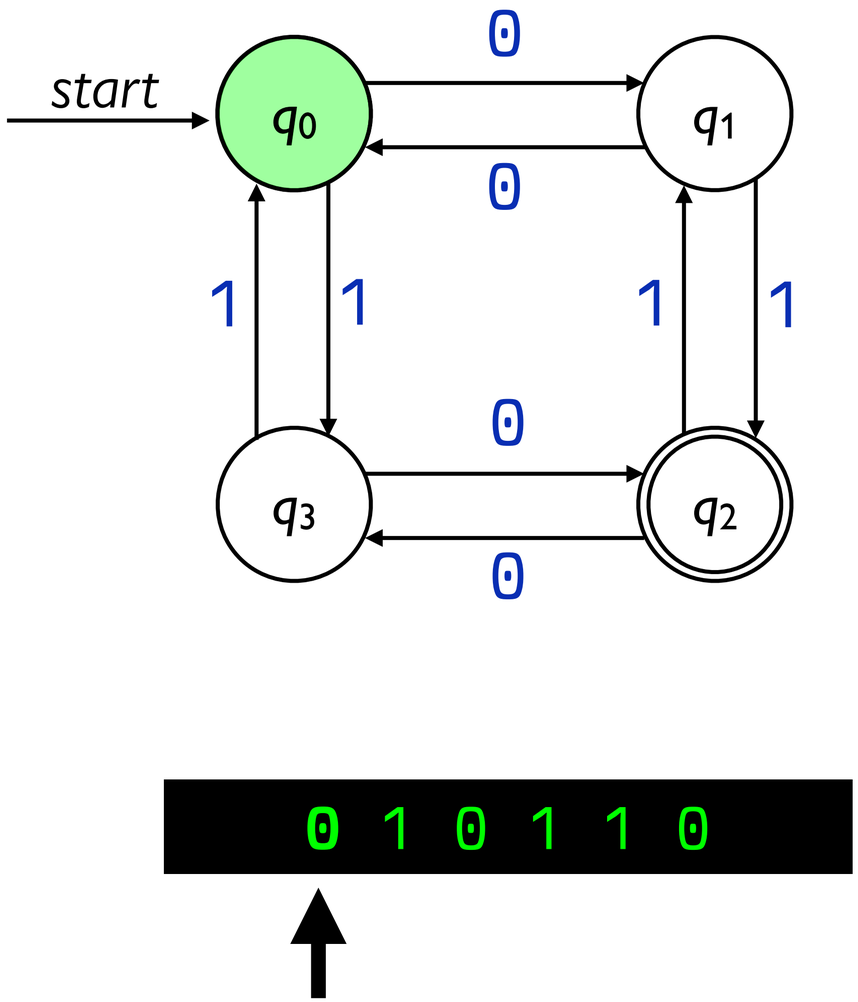
0 1 0 1 1 0

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

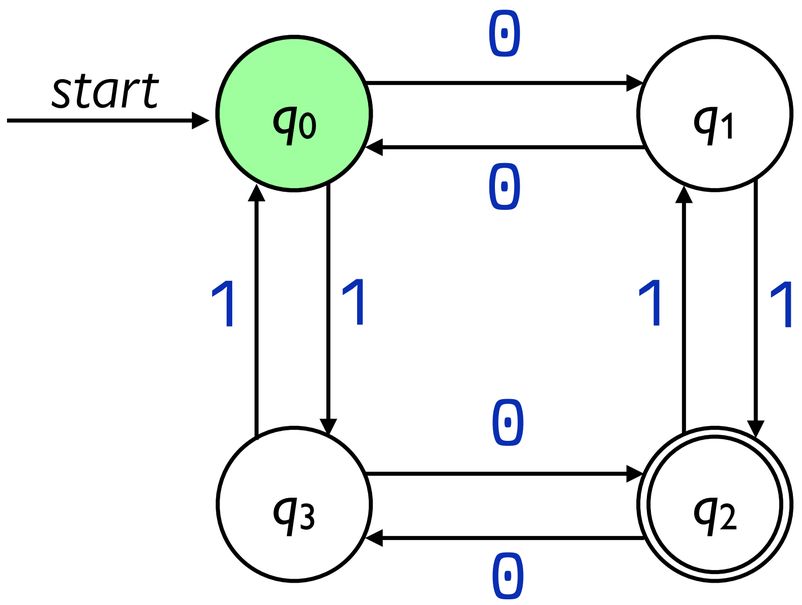


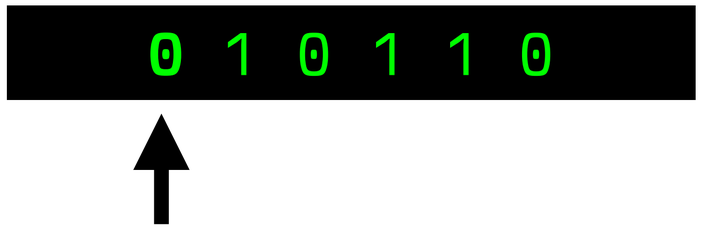


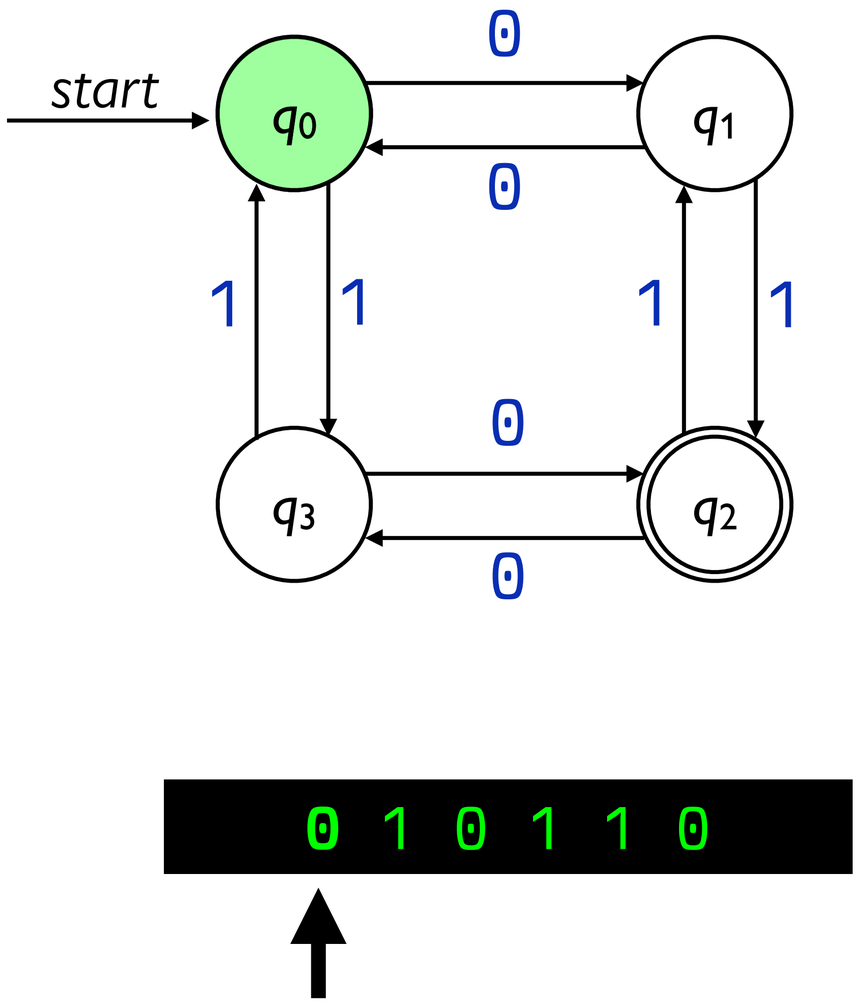
0 1 0 1 1 0

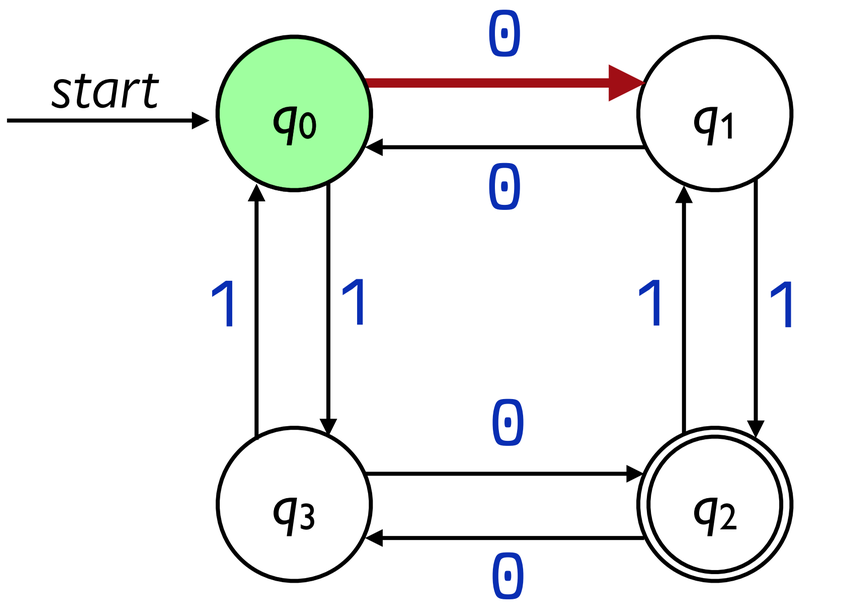

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

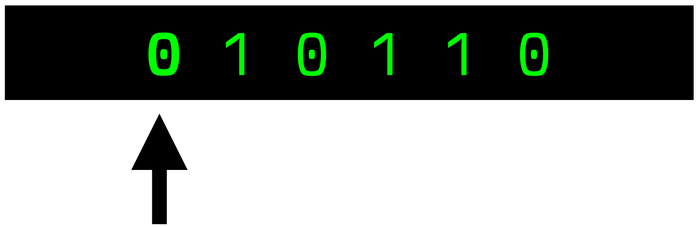
The automaton now begins processing characters in the order in which they appear.

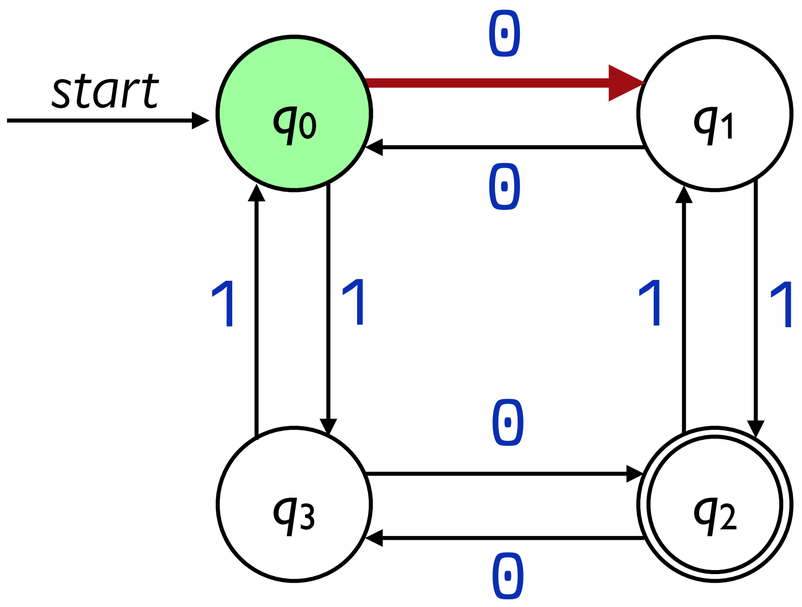


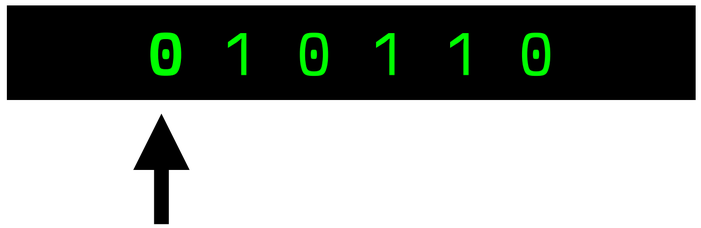



start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

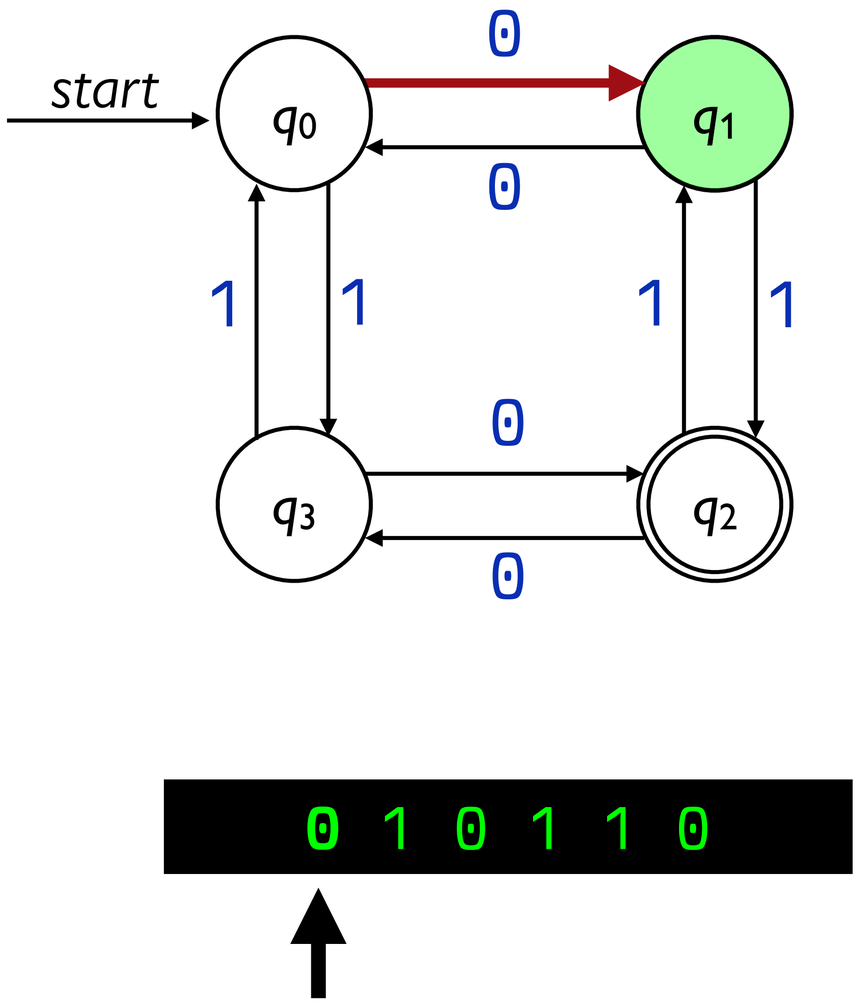

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

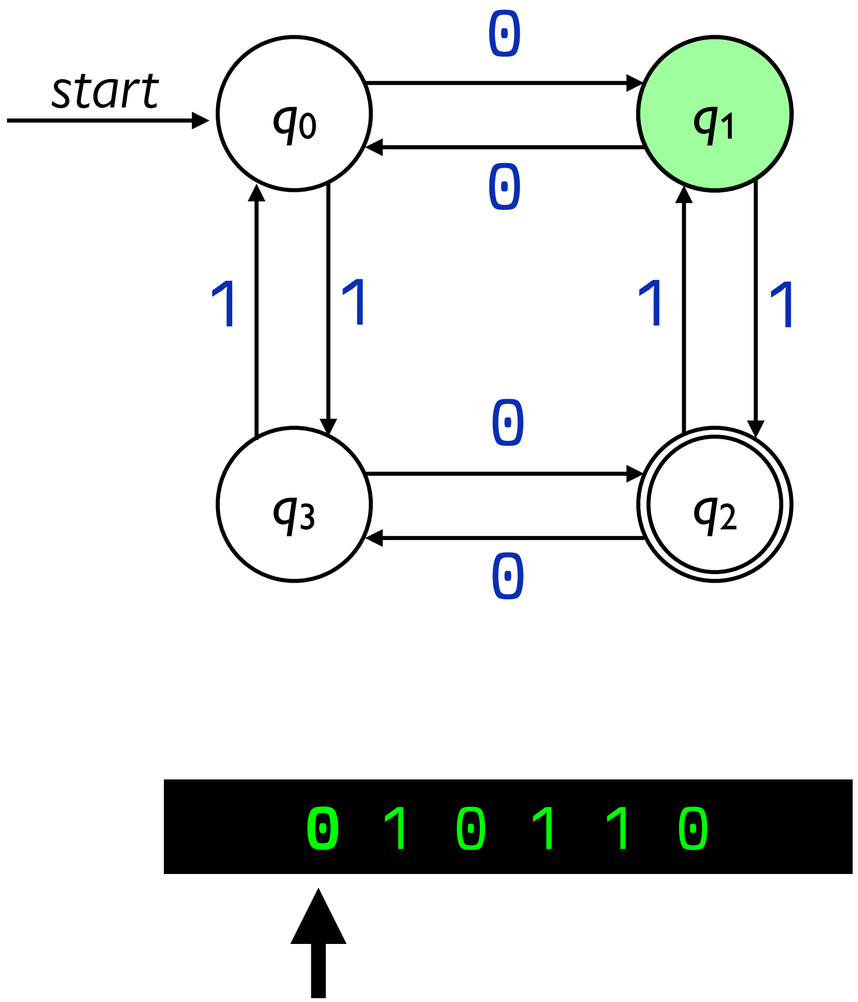

0 1 0 1 1 0

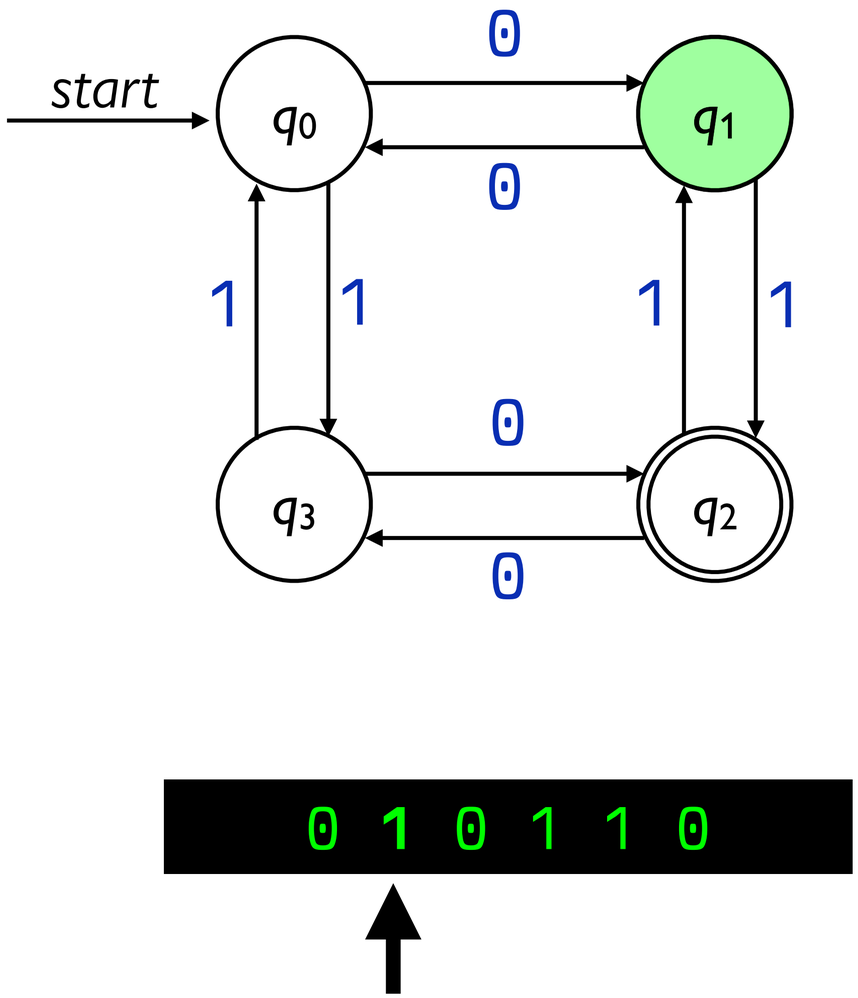


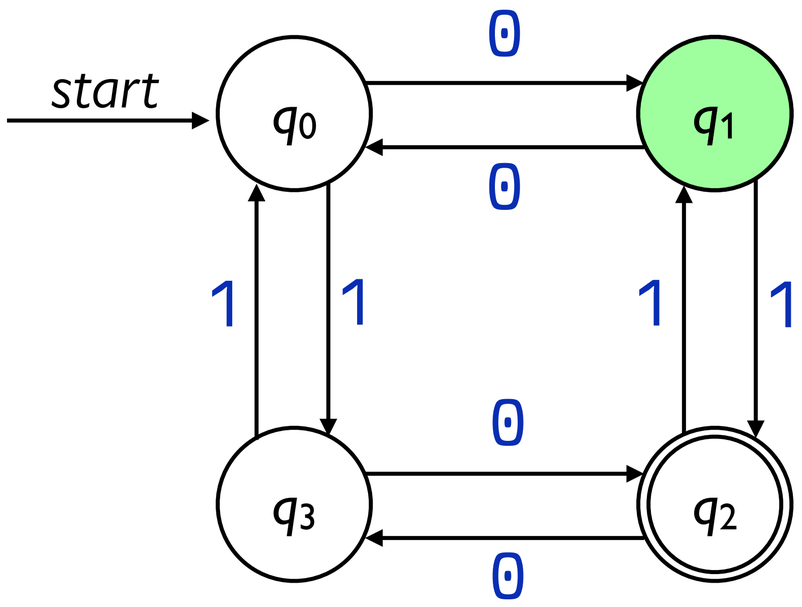


Each arrow in this diagram represents a ***transition****. The automaton always follows the transition for the symbol being read.*

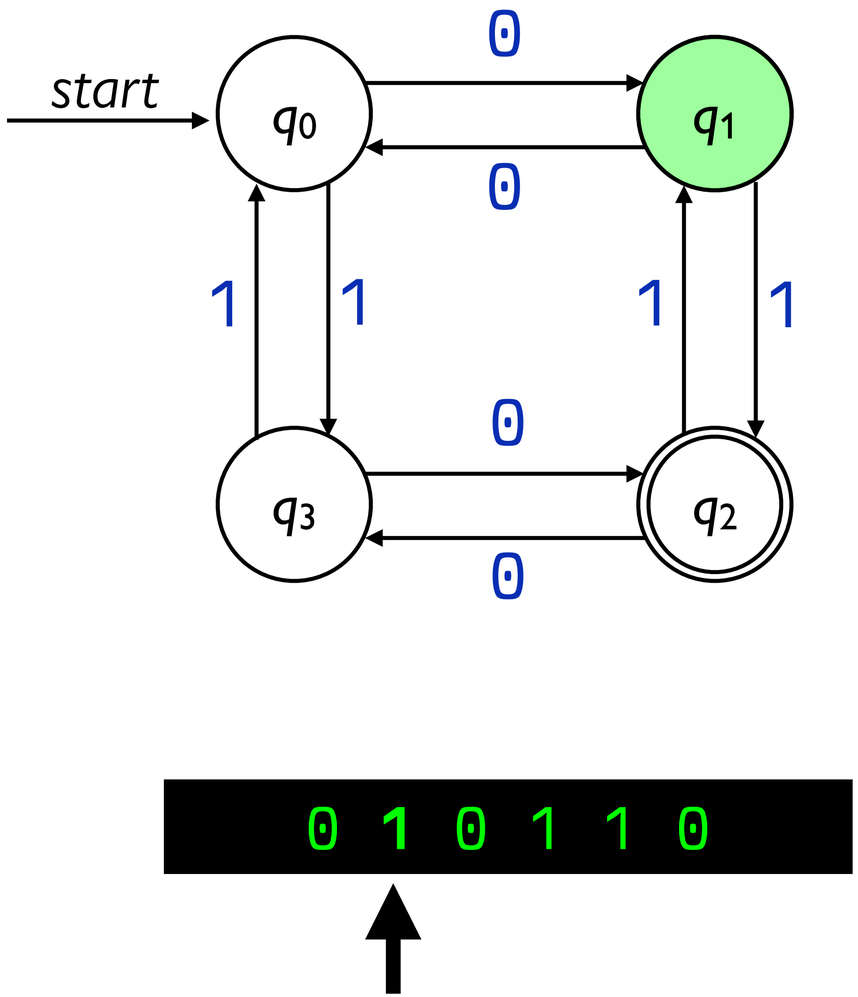

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

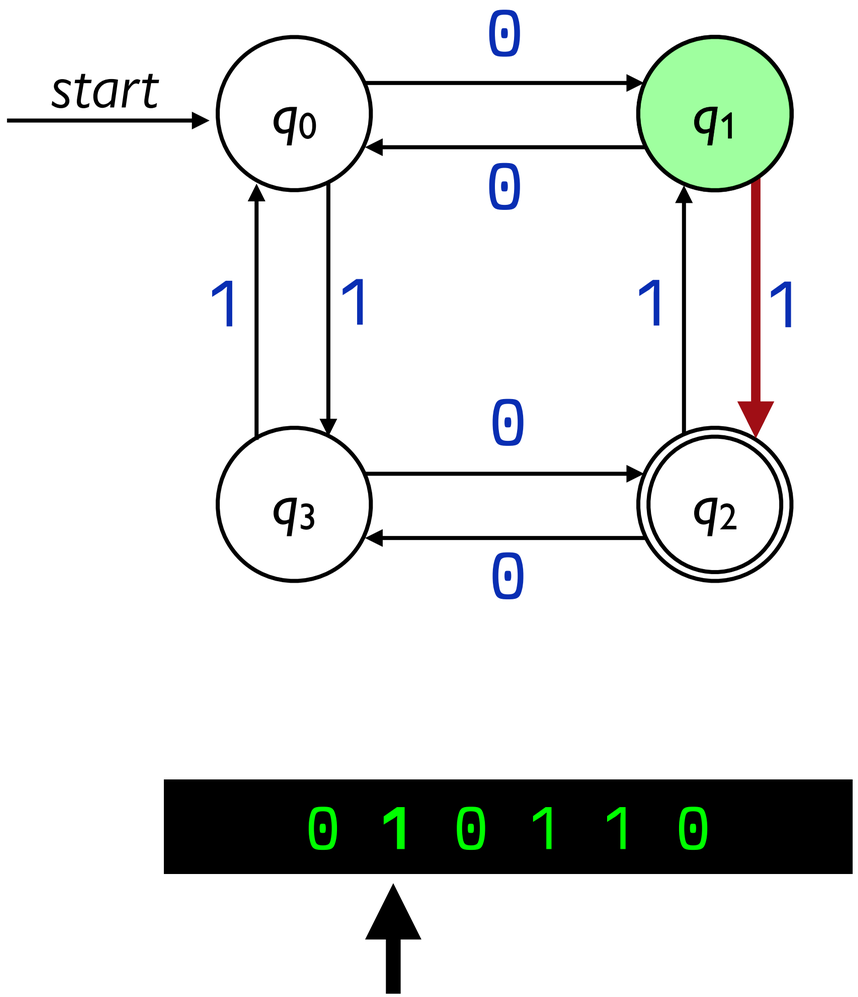

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

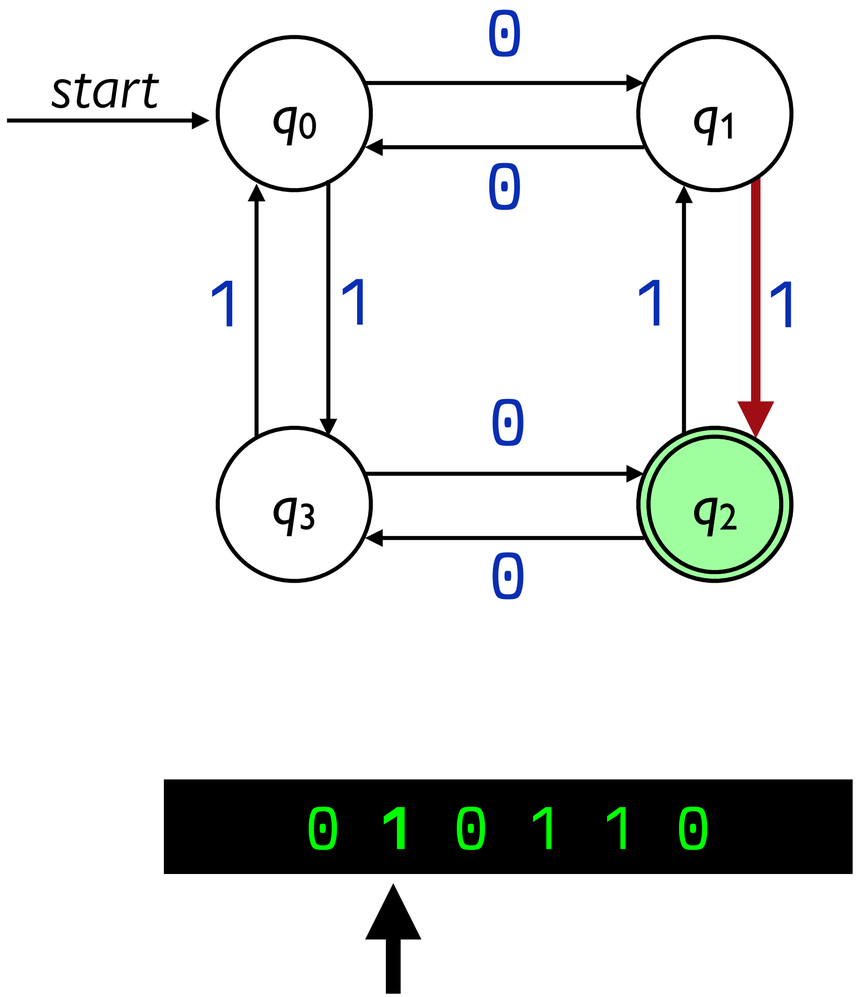

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

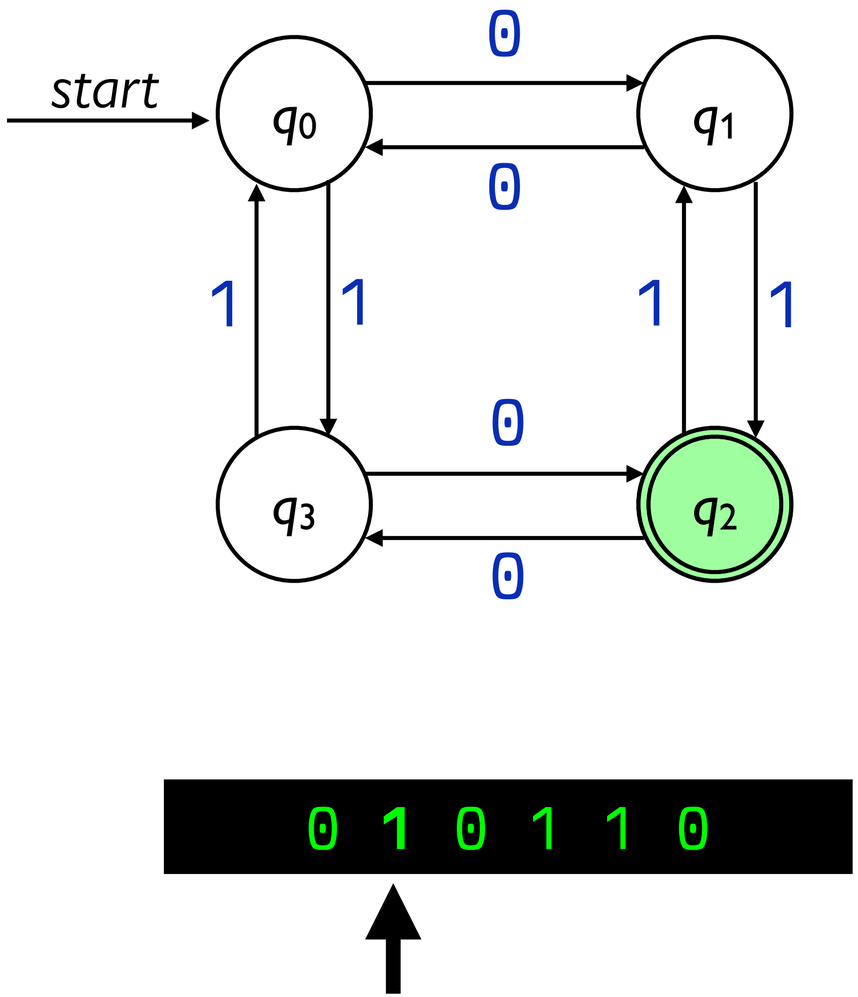


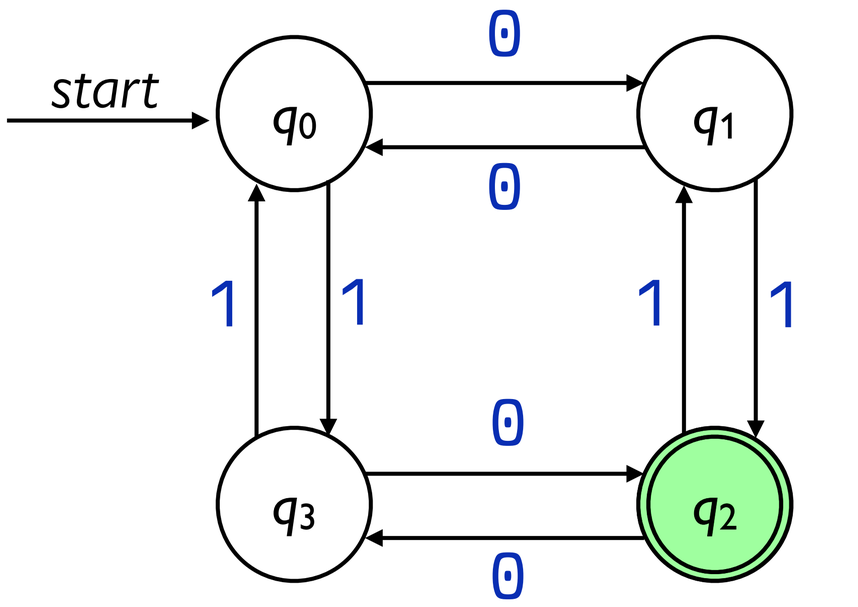
After transitioning, the automaton considers the next symbol in the input.

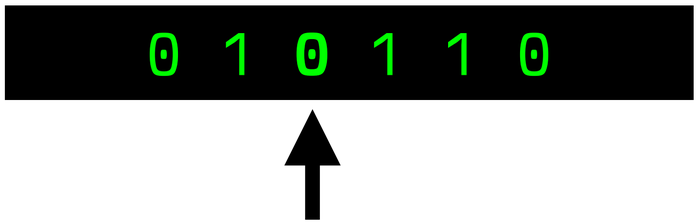

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

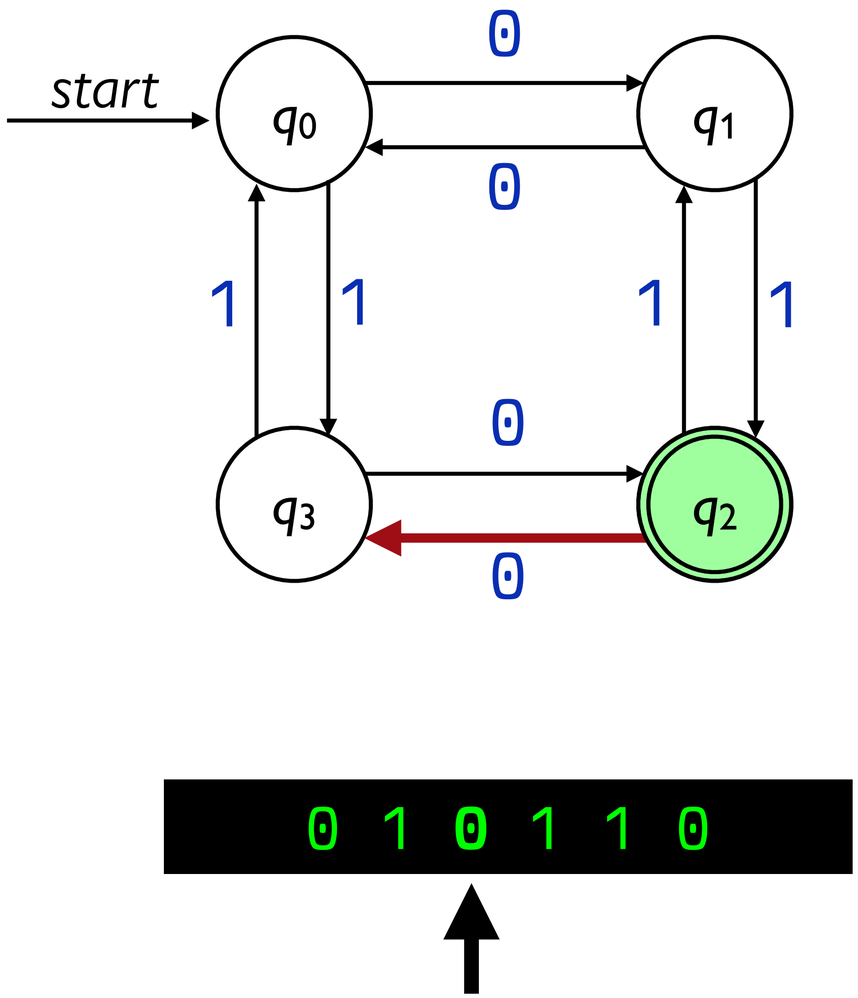

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

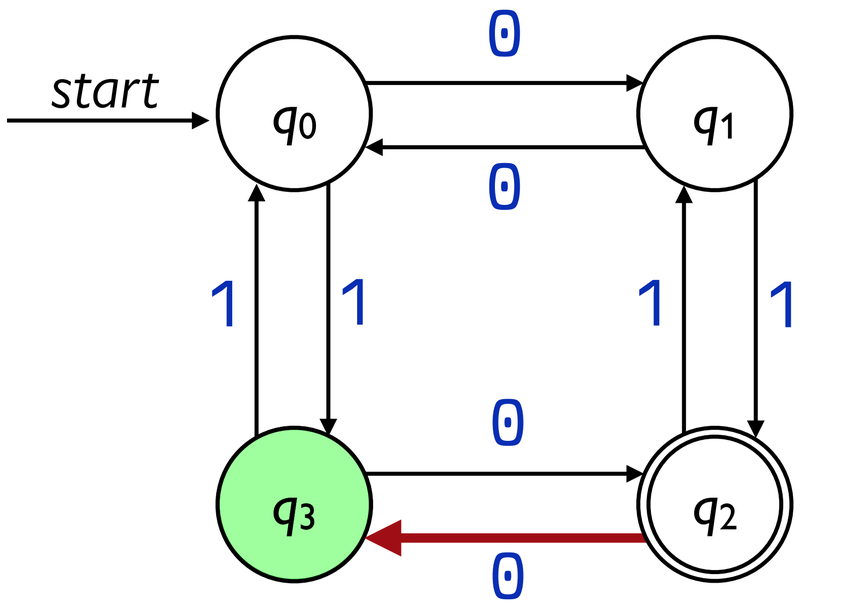

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

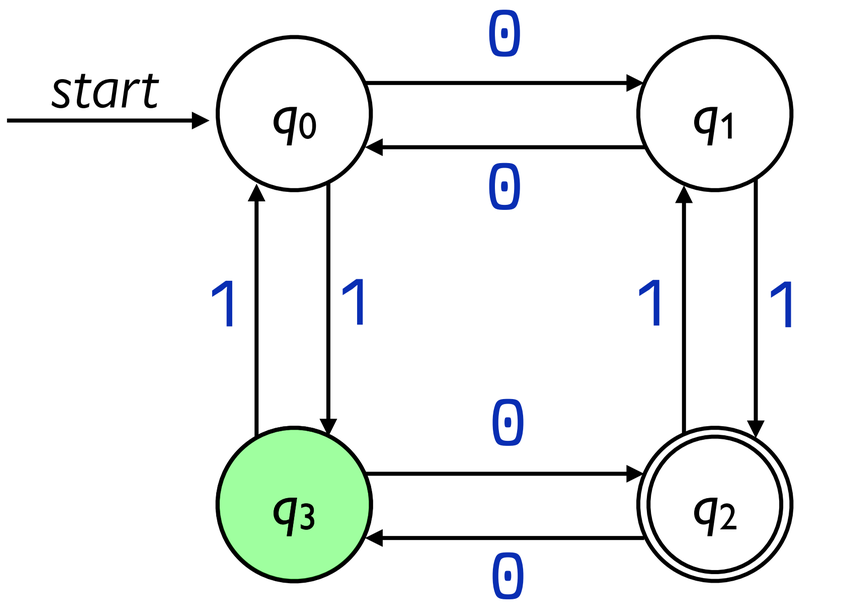

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

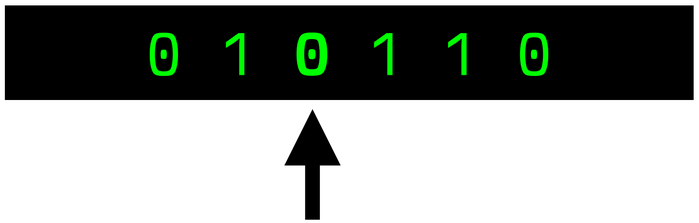

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

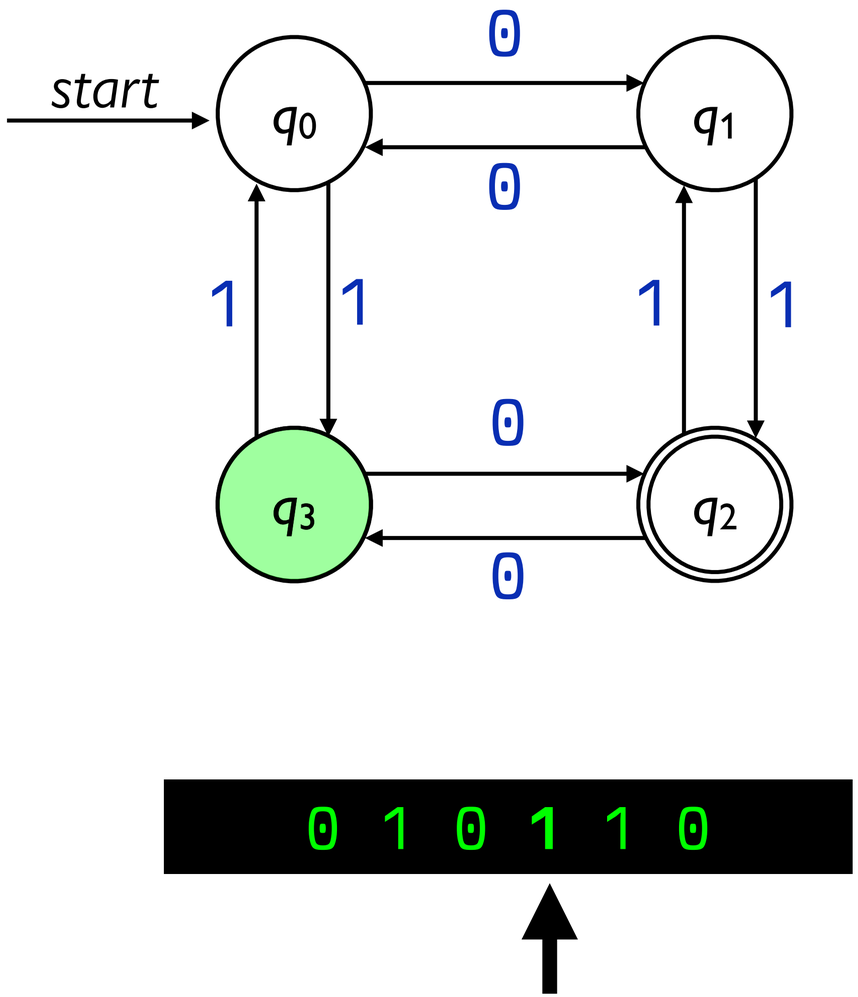

0 1 0 1 1 0

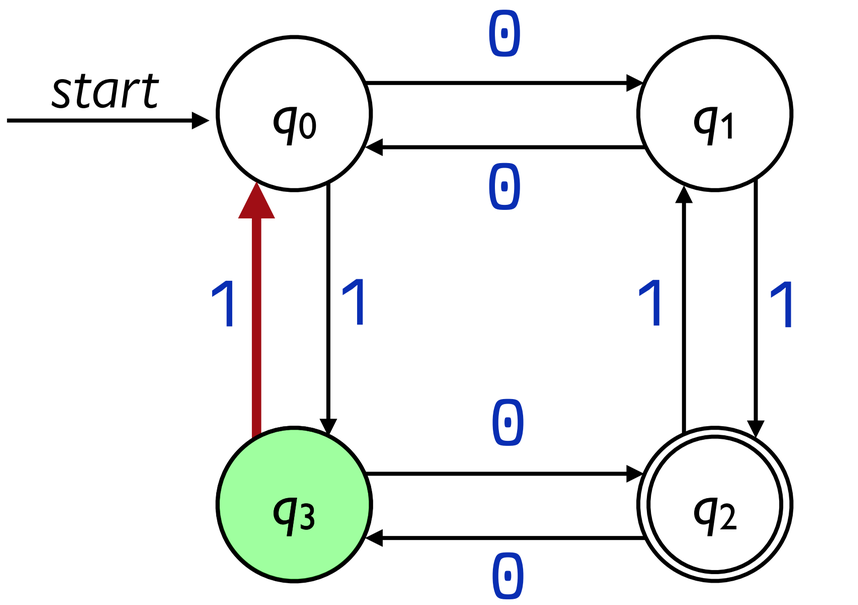

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

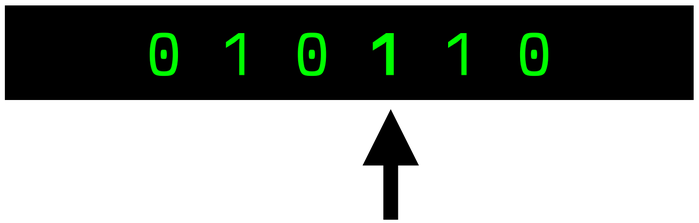

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

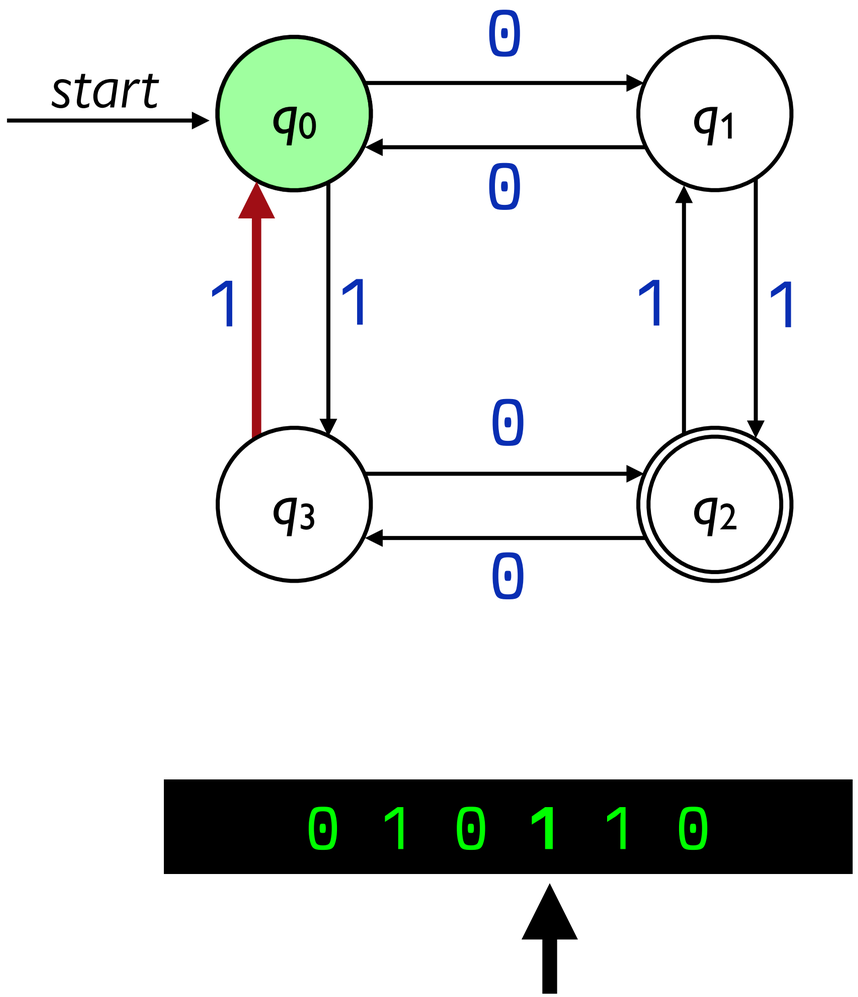

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

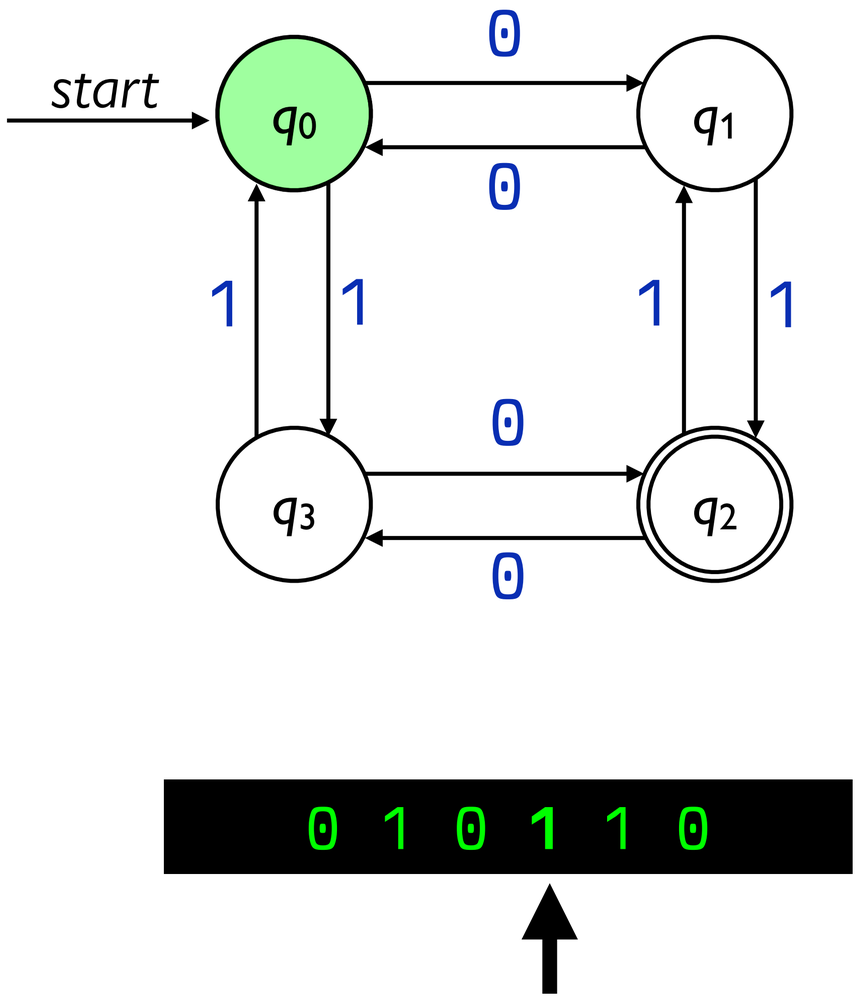

0 1 0 1 1 0

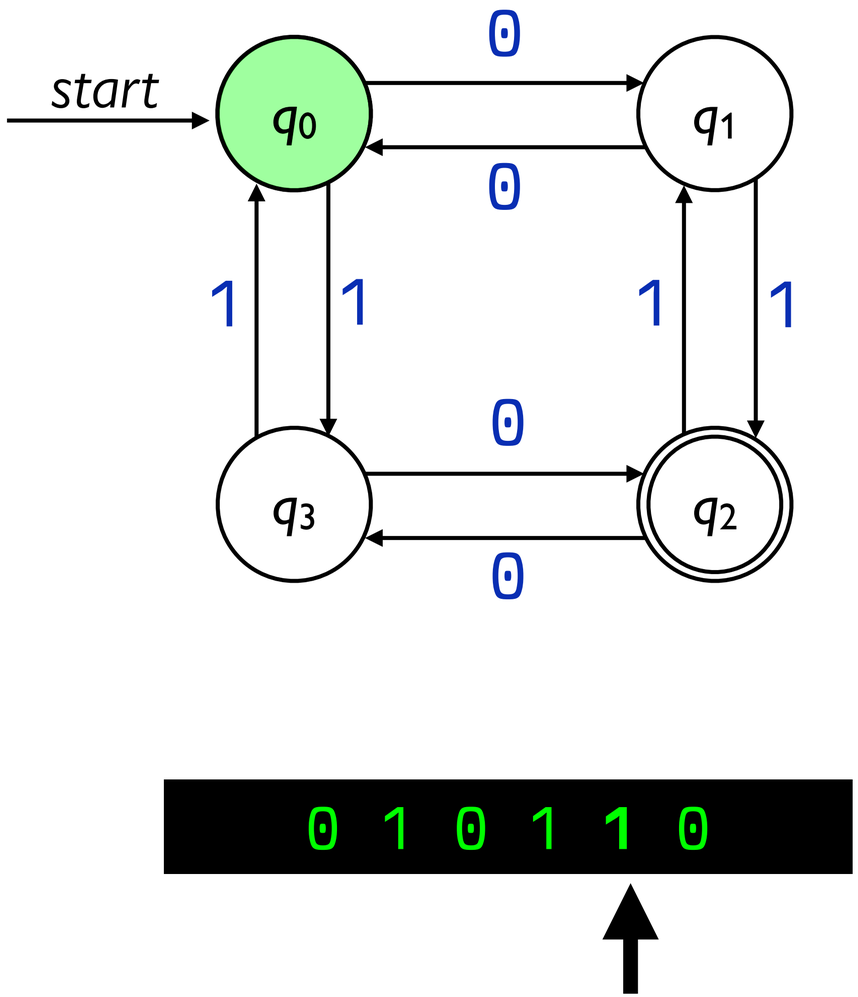

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

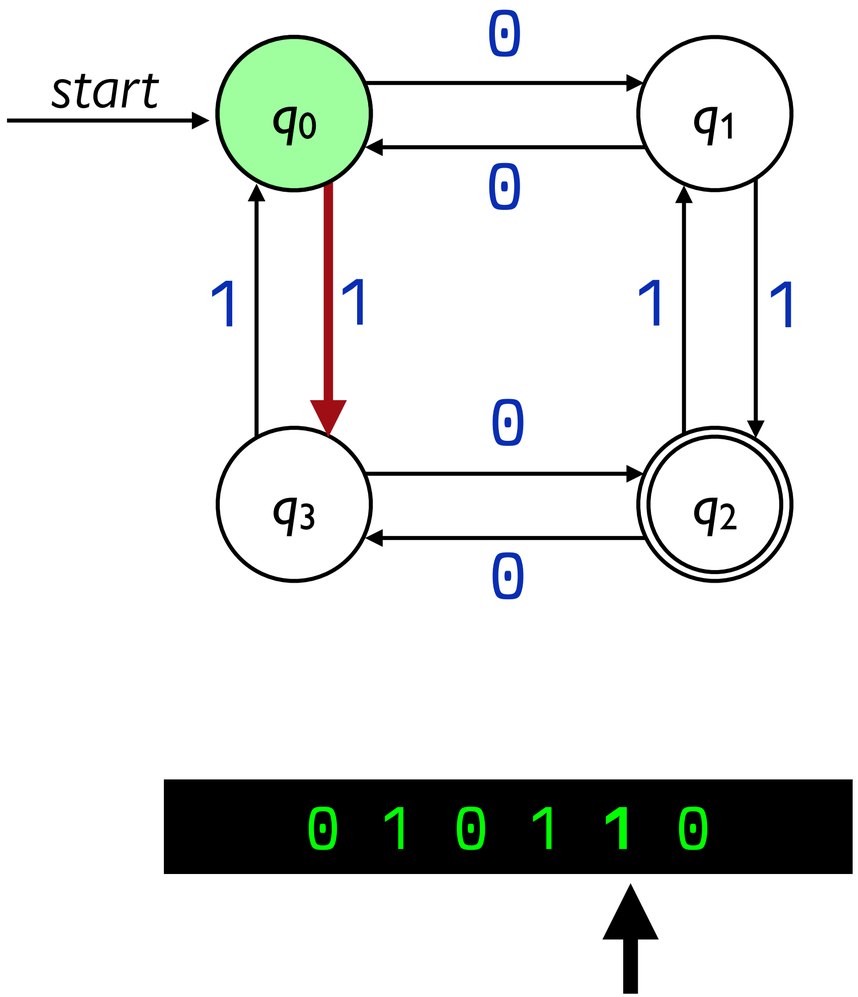

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

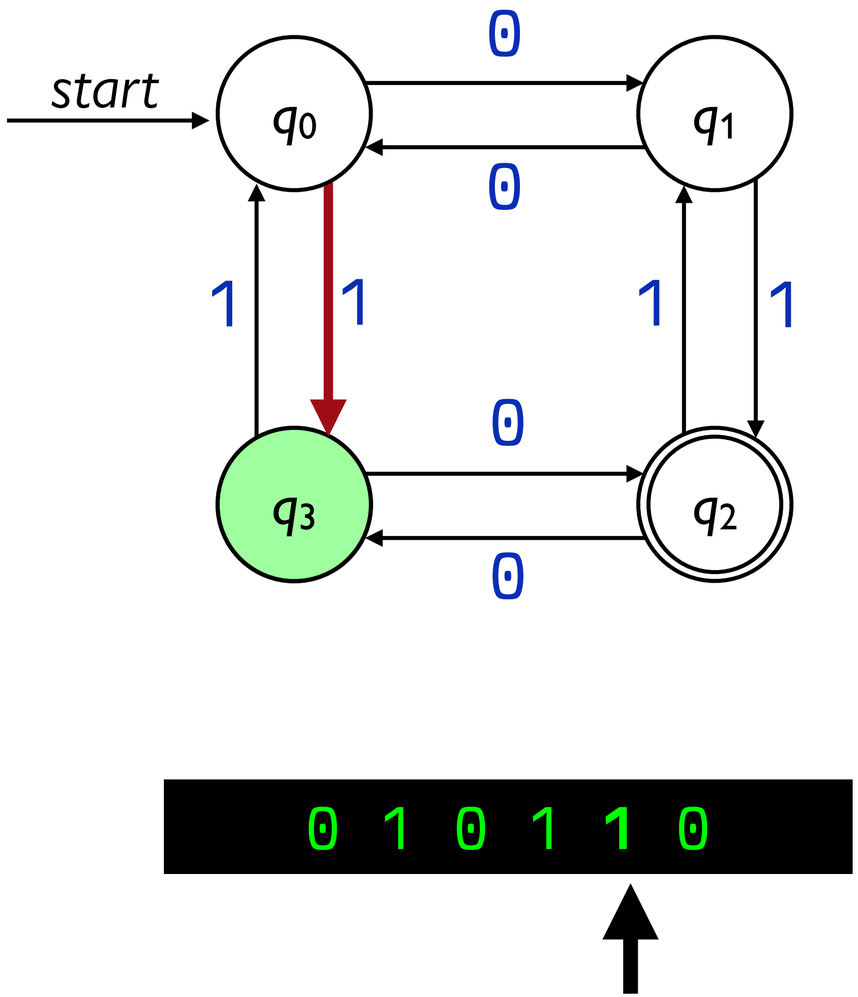

0 1 0 1 1 0

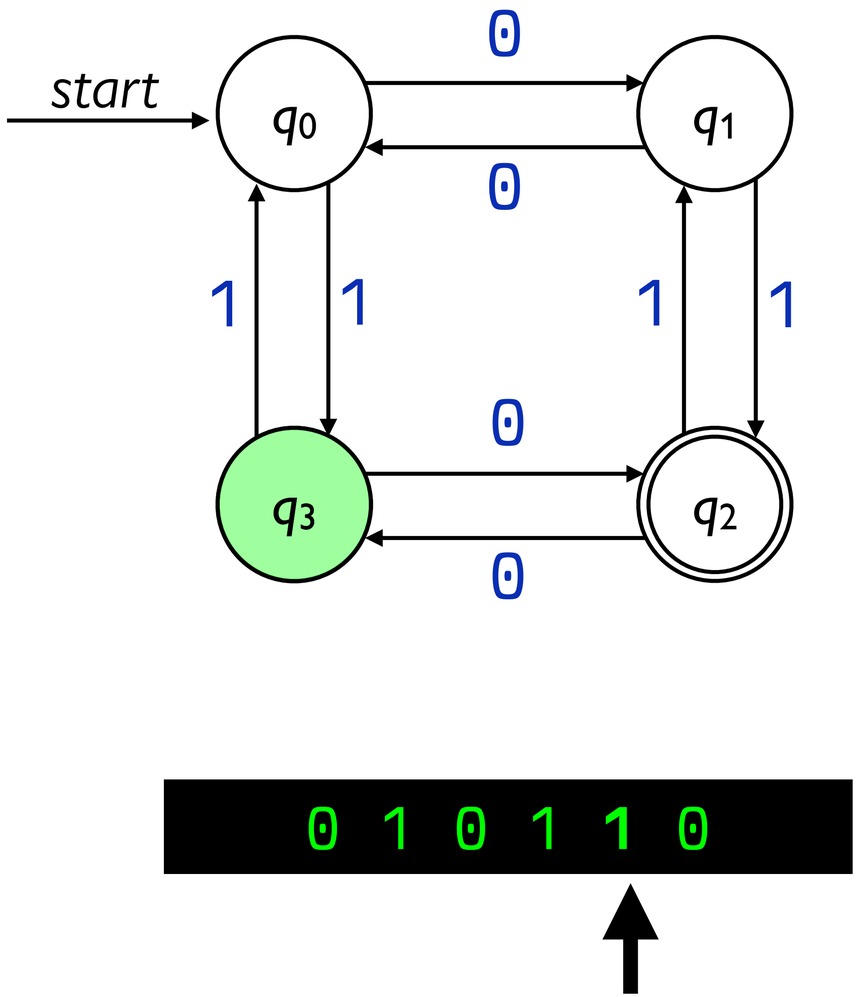

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

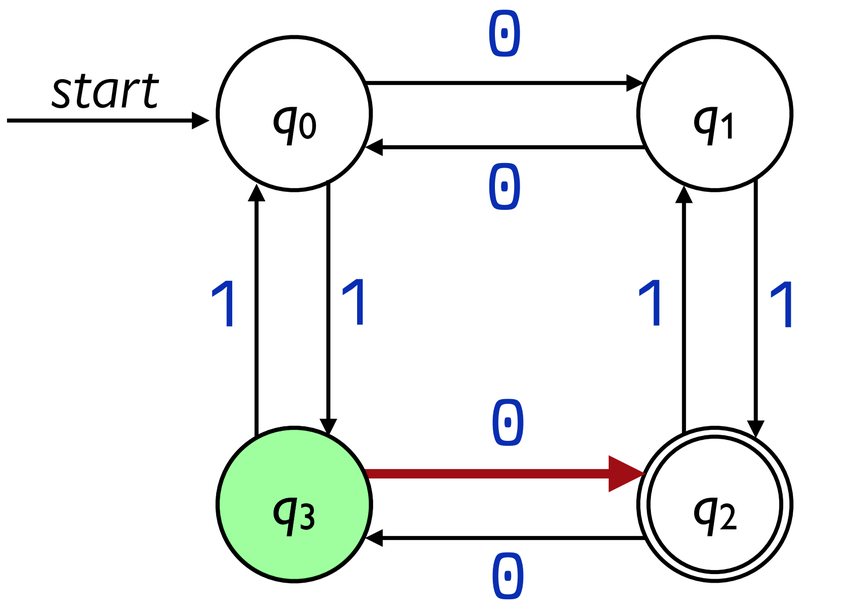

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

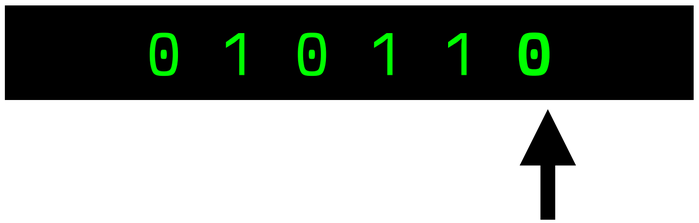

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

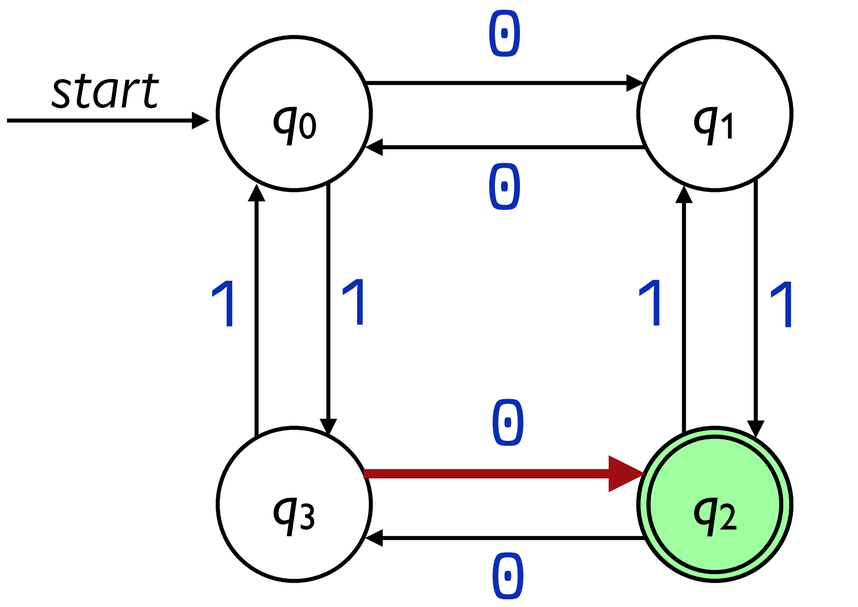

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

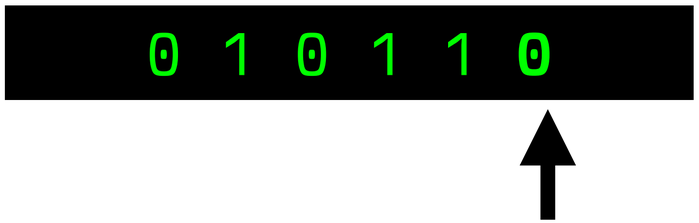

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

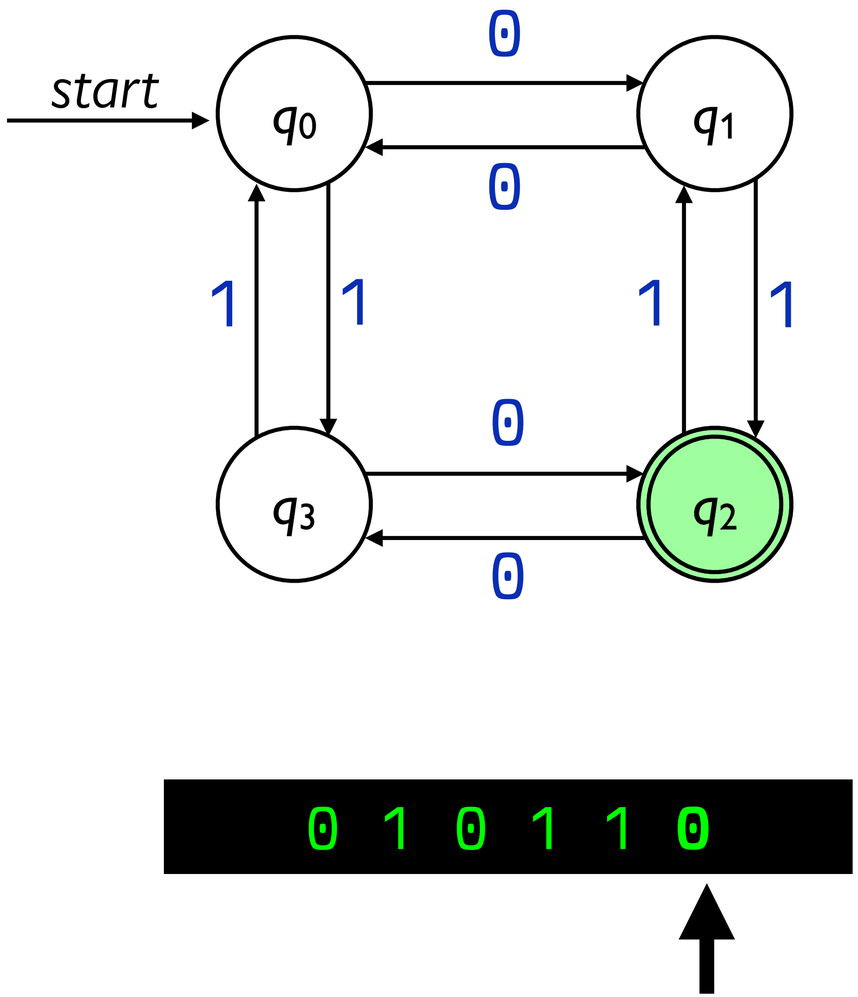

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

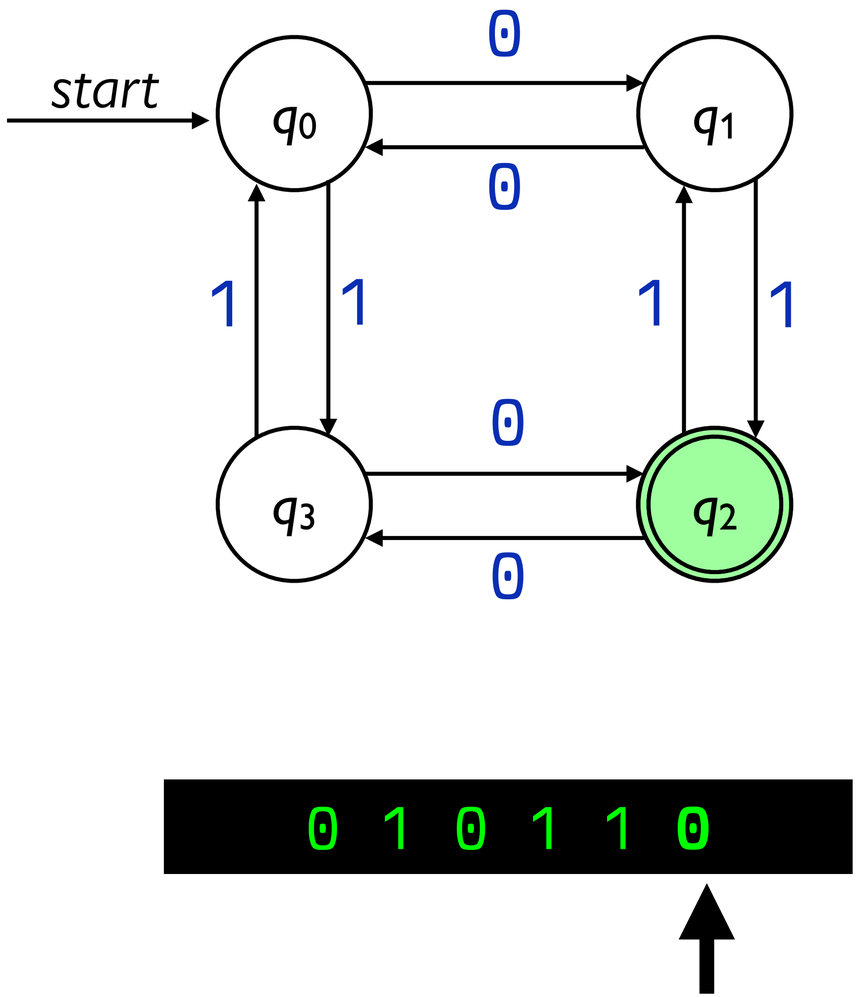

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0


0 1 0 1 1 0

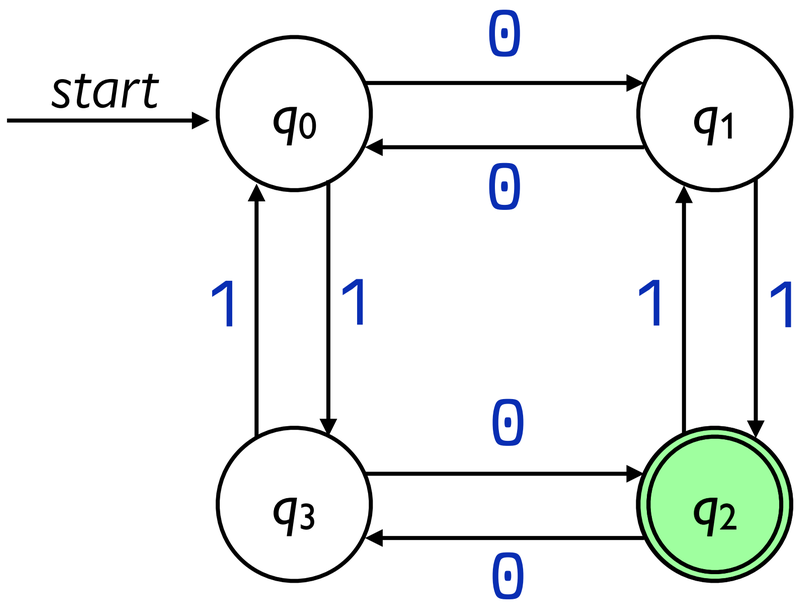

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

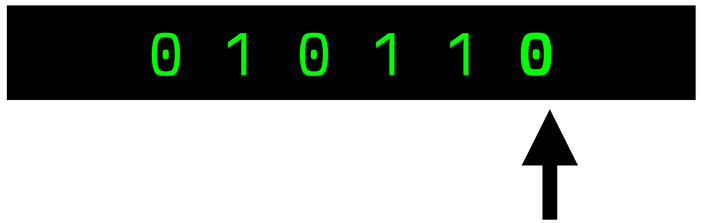

0 1 0 1 1 0


start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

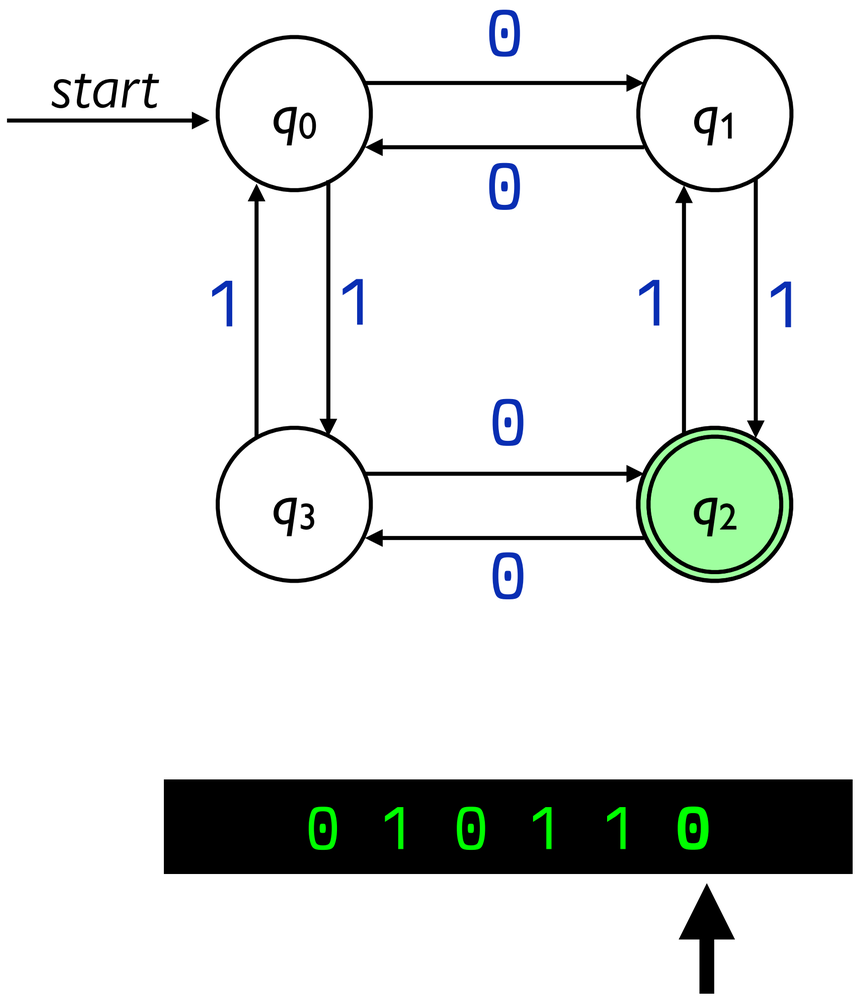

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
0 1 0 1 1 0

Now that the automaton has read all of the input, it can decide whether to accept or reject

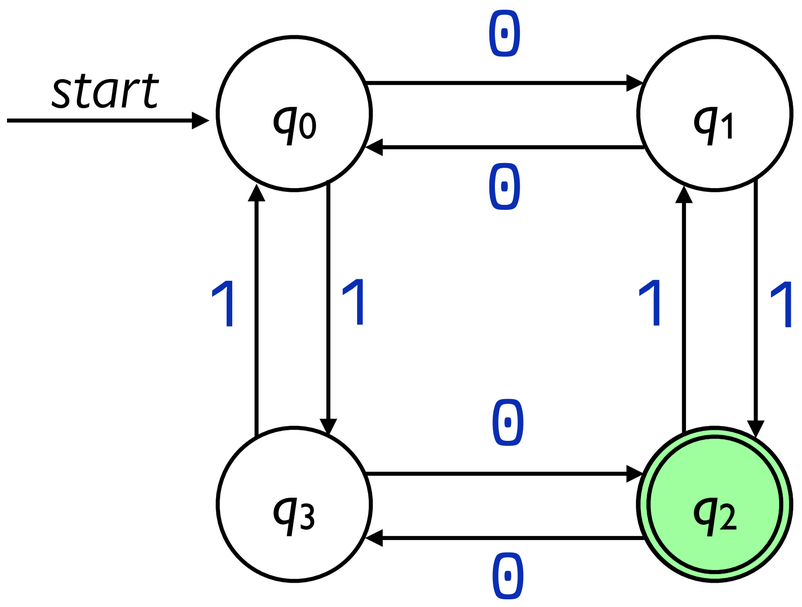




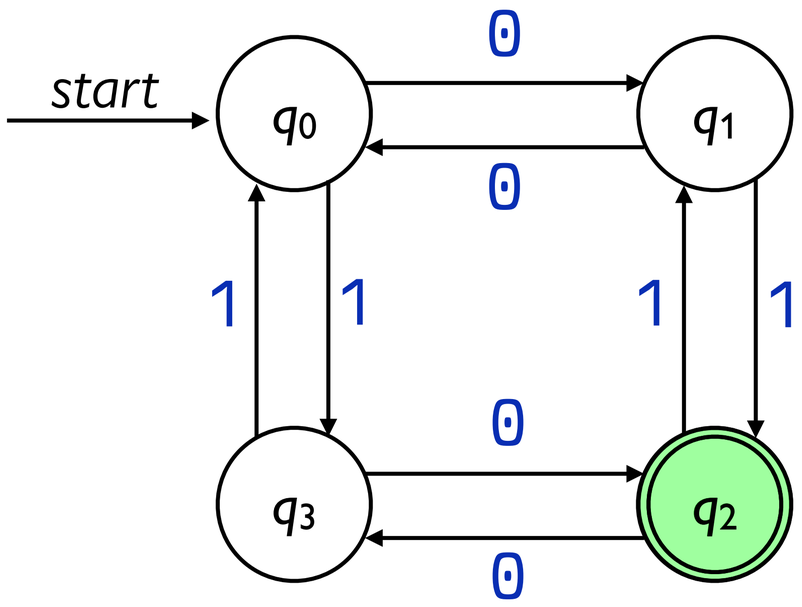
Now that the automaton has read all of the input, it can decide whether to accept or reject

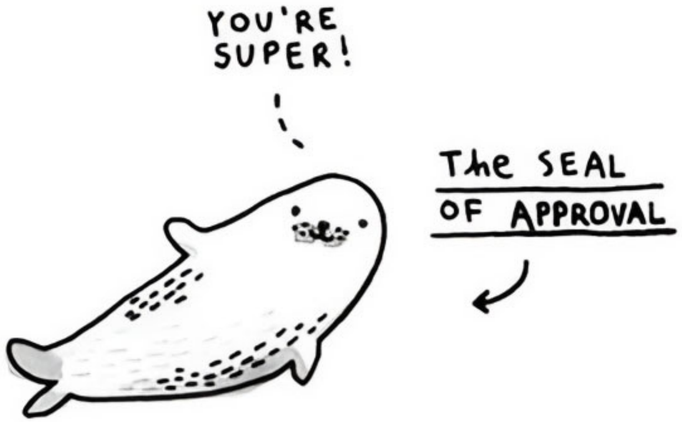


The double circle indicates this is an ***accept state****, so it accepts!*



0 1 0 1 1 0

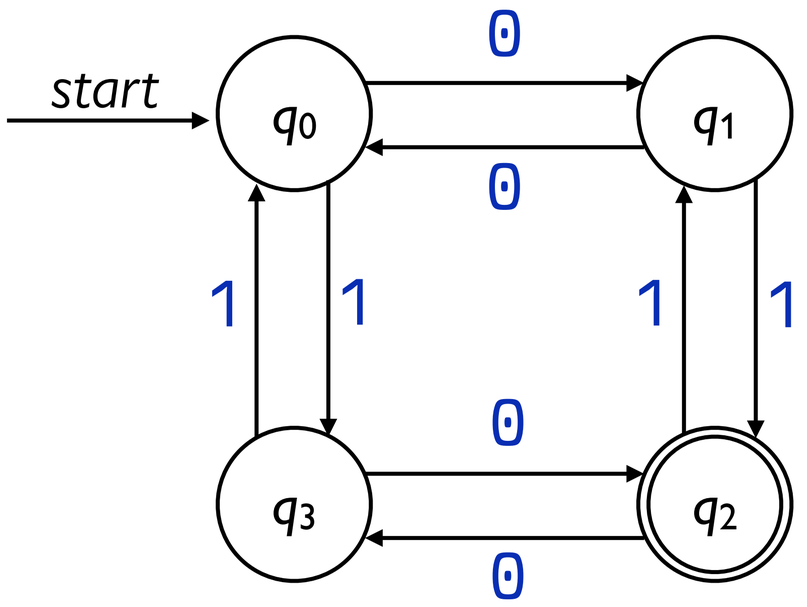




0 1 0 1 1 0

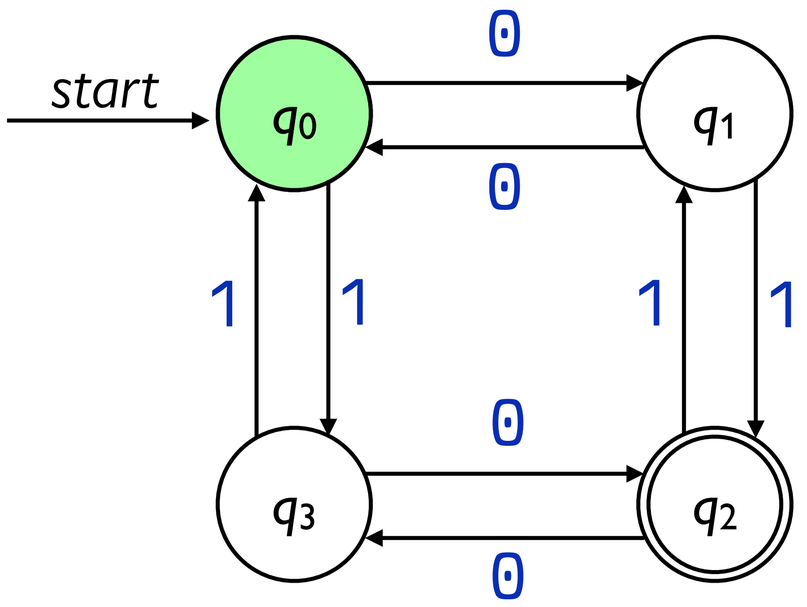
Illustration by Gemma Correll

Let's try another input

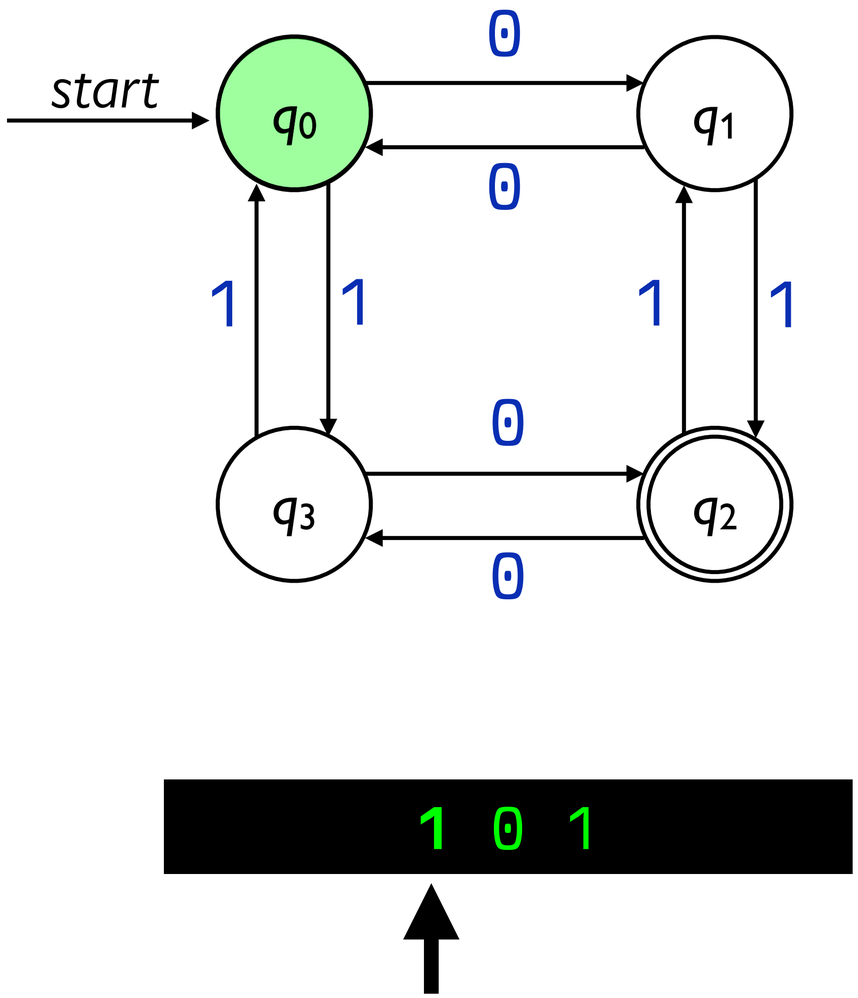


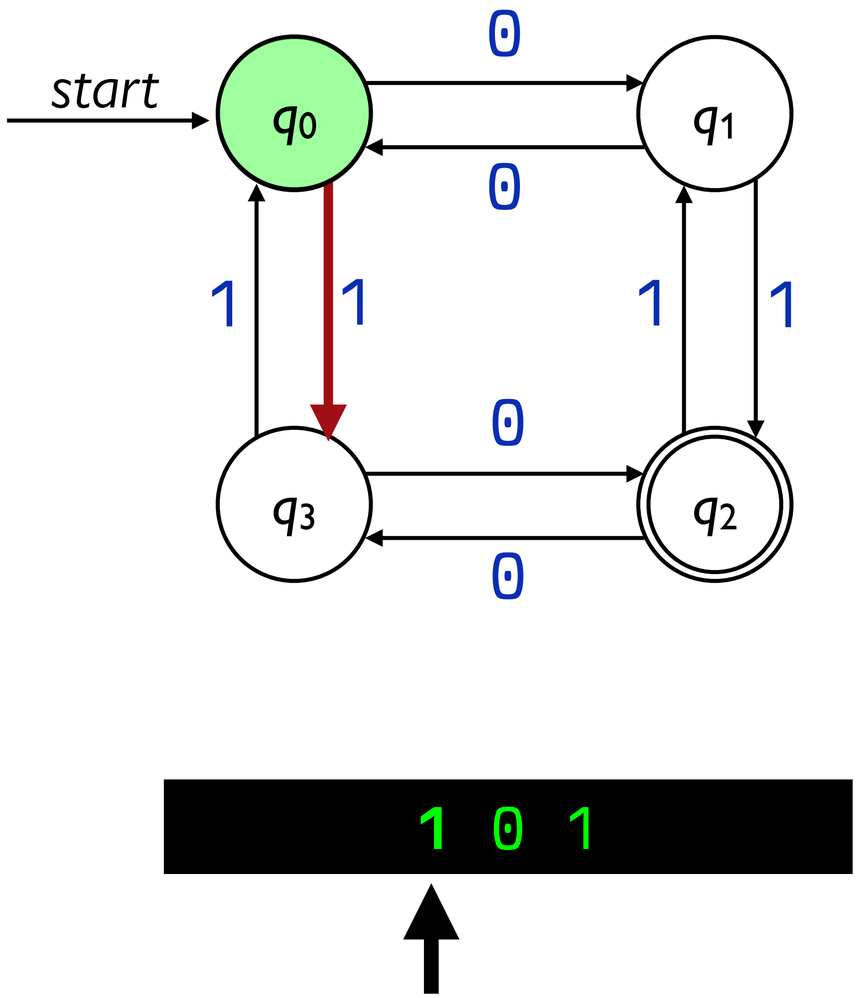
1 0 1

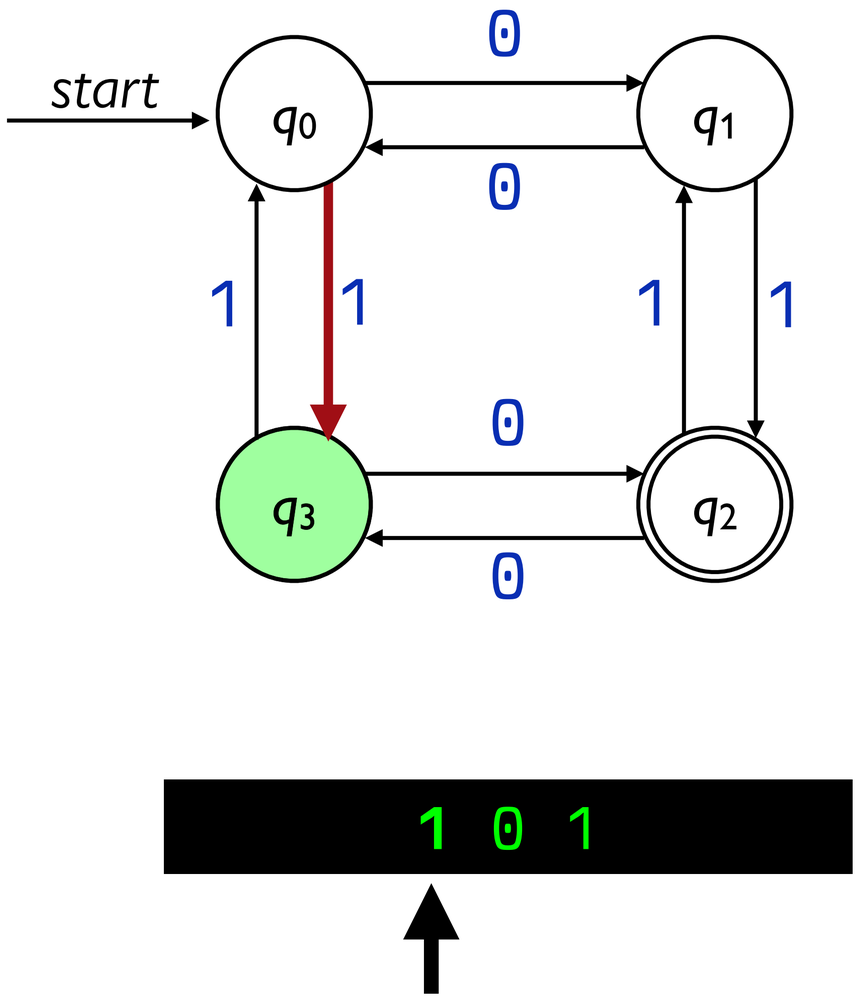
Let's try another input

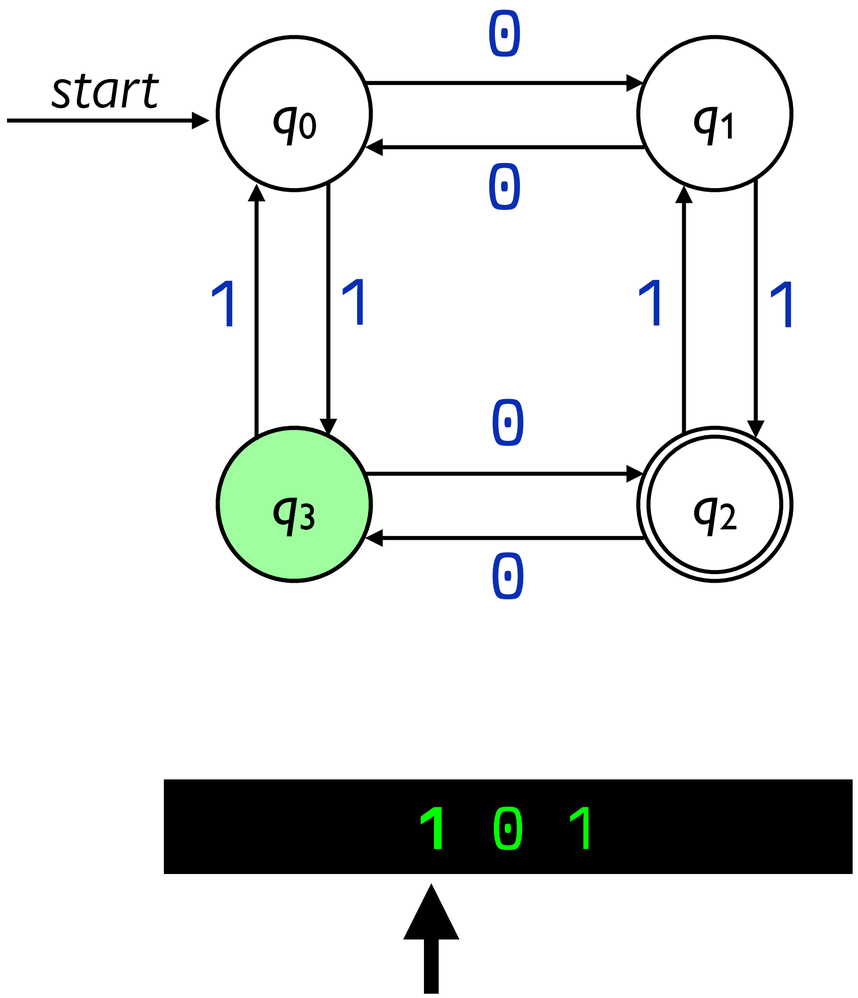


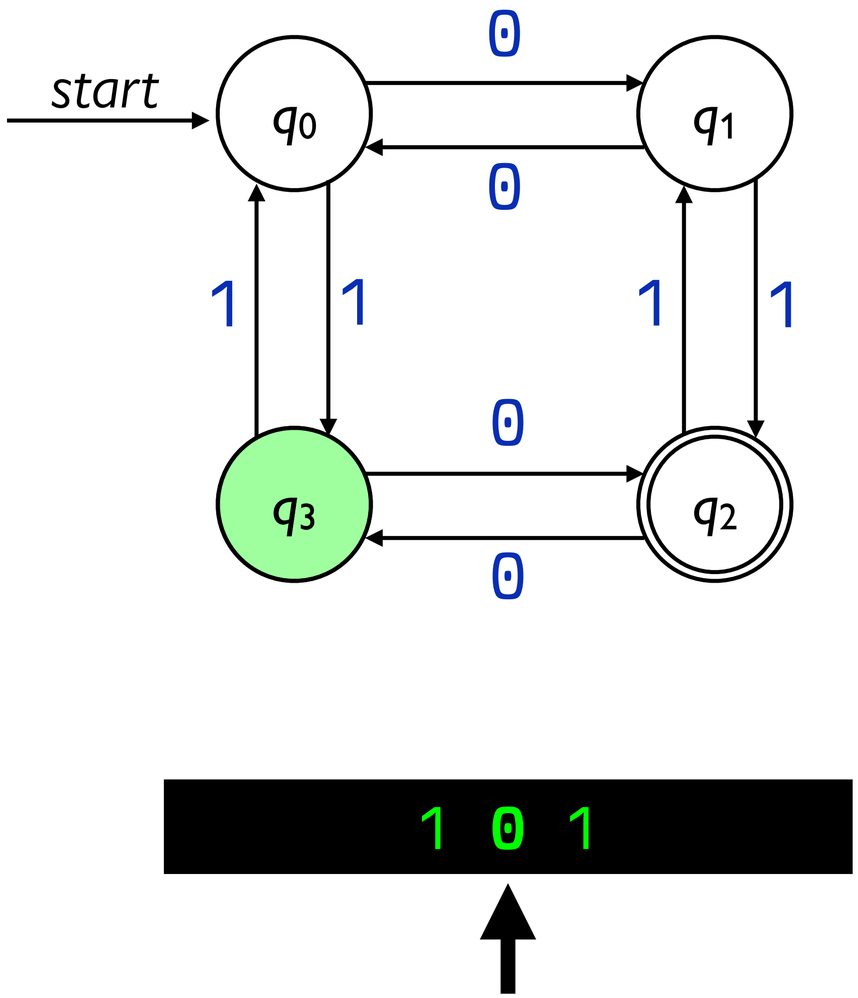
1 0 1

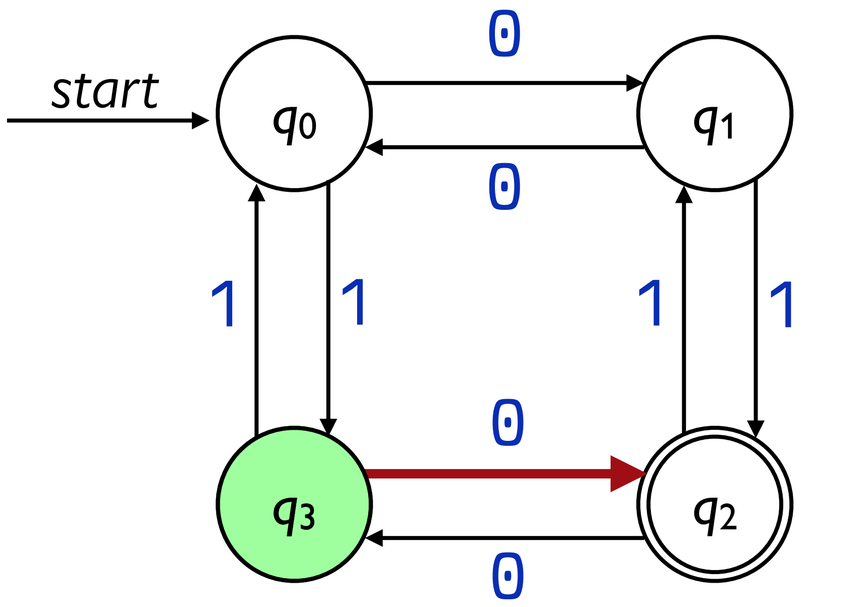

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

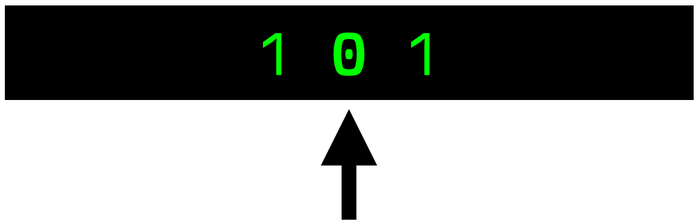

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

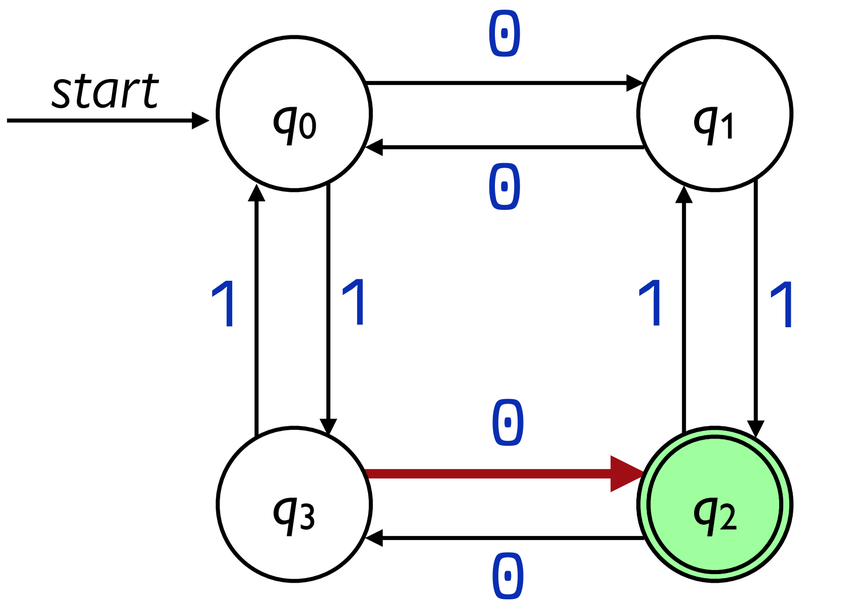

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

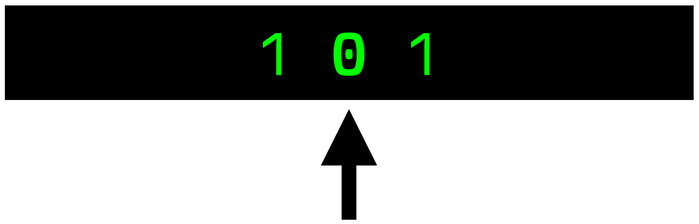

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

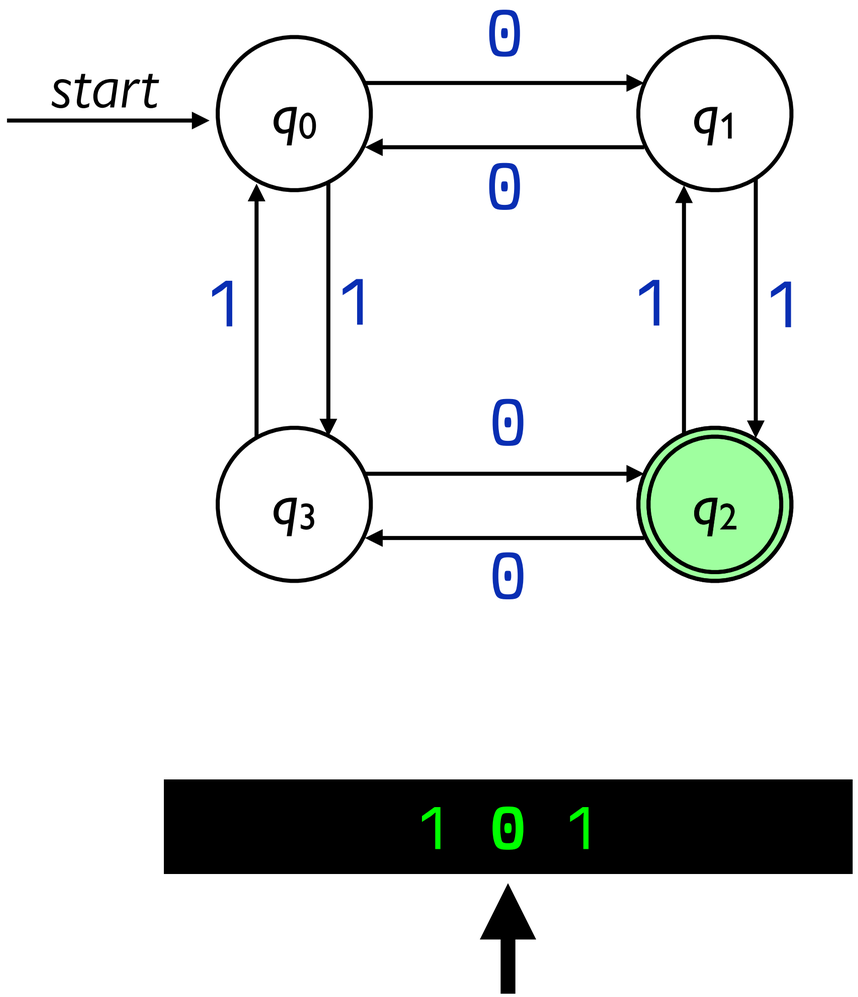

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

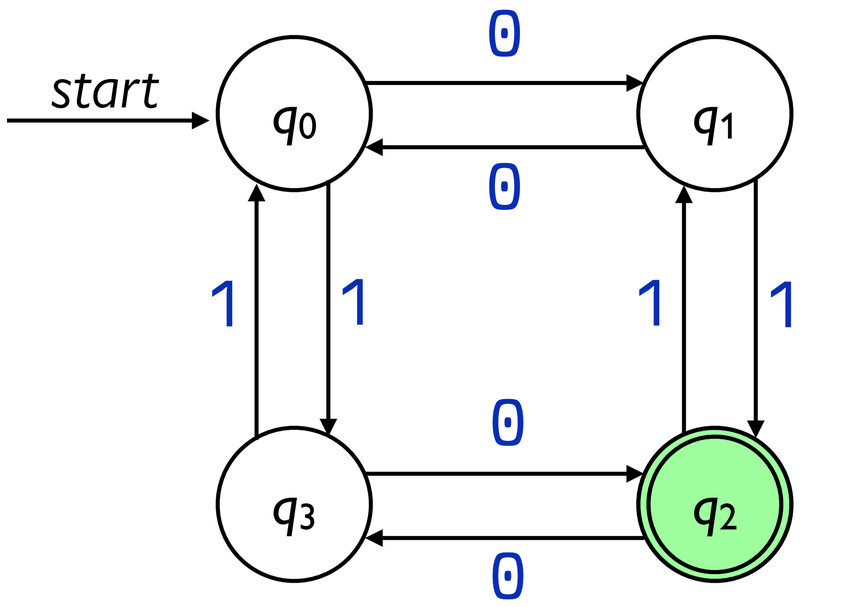

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

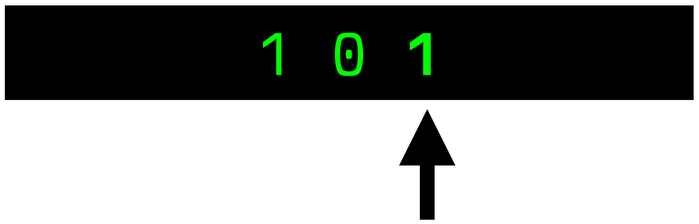

1 0 1

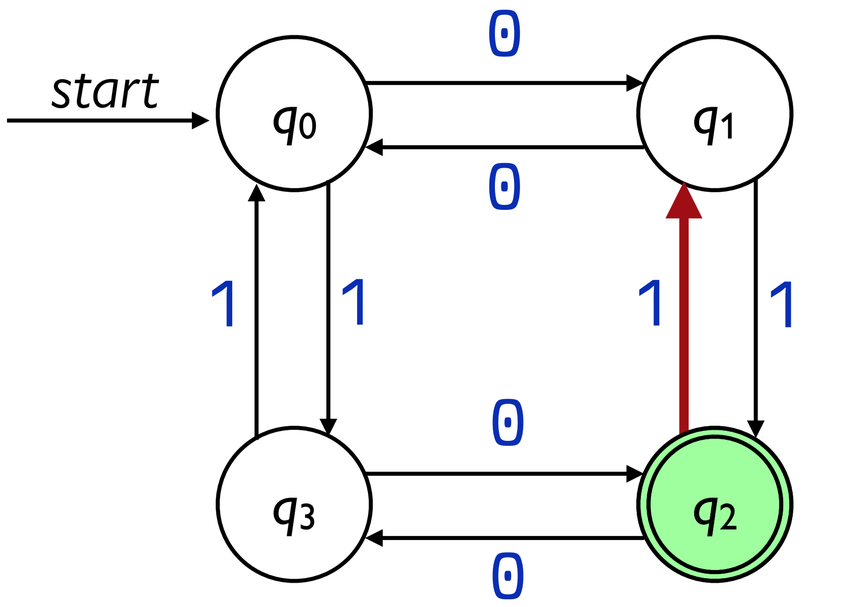

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

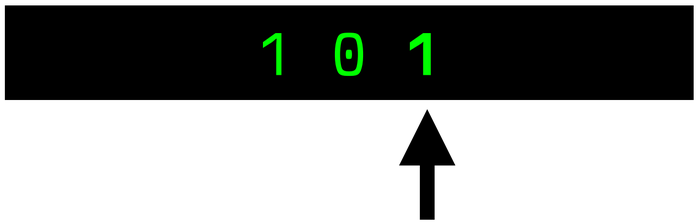

1 0 1

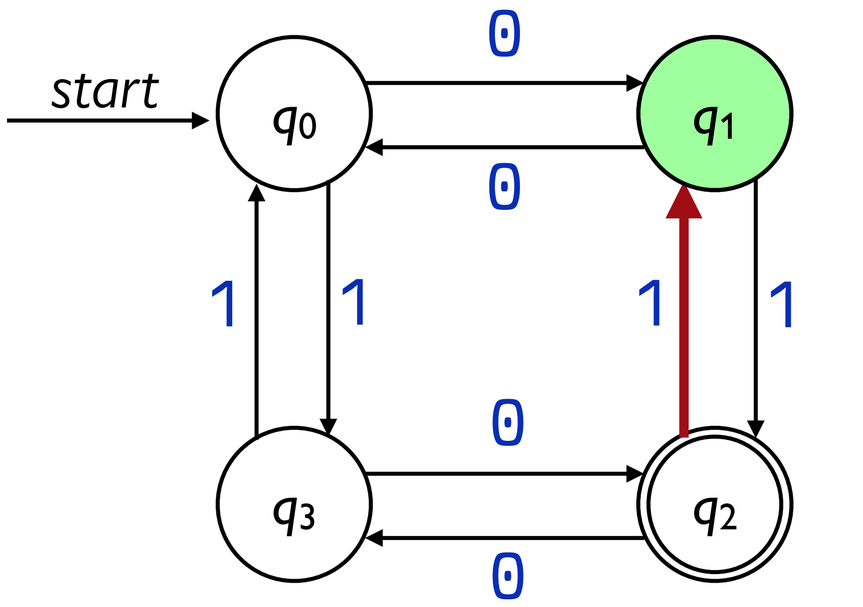

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

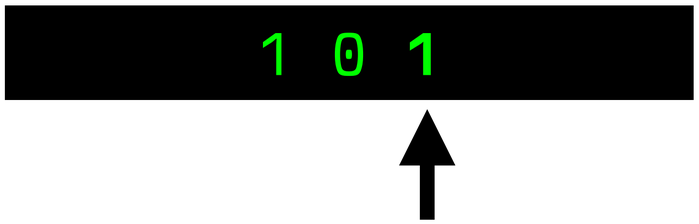

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

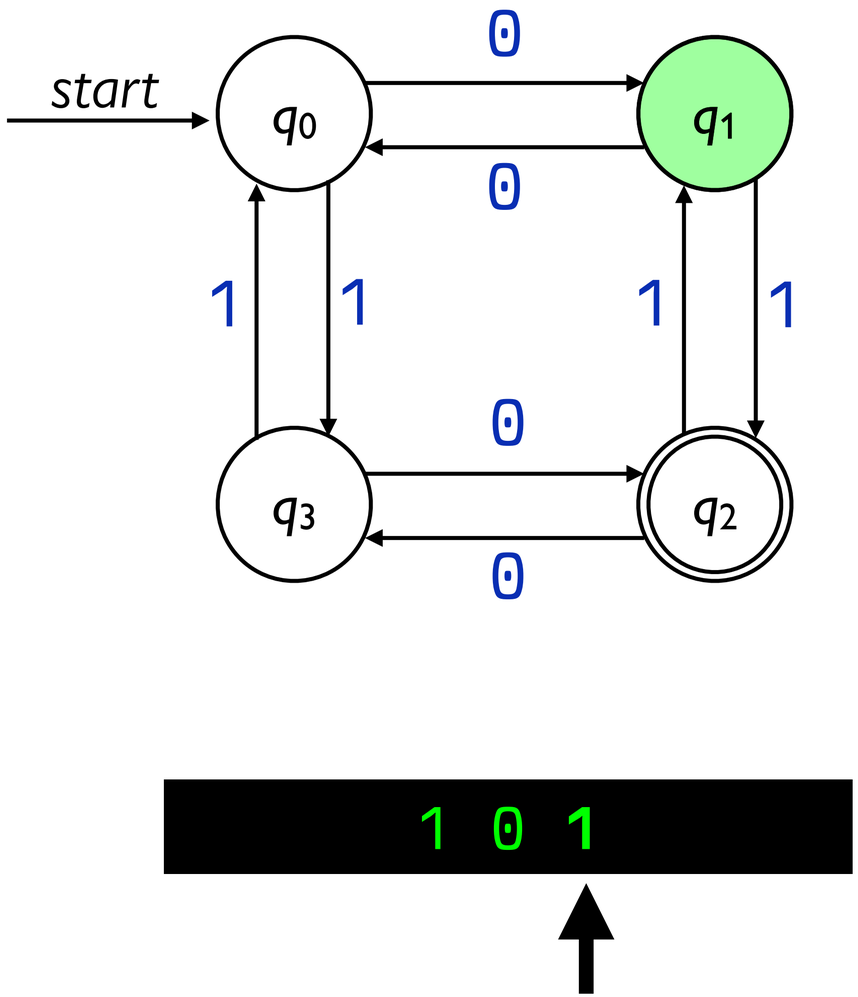

1 0 1

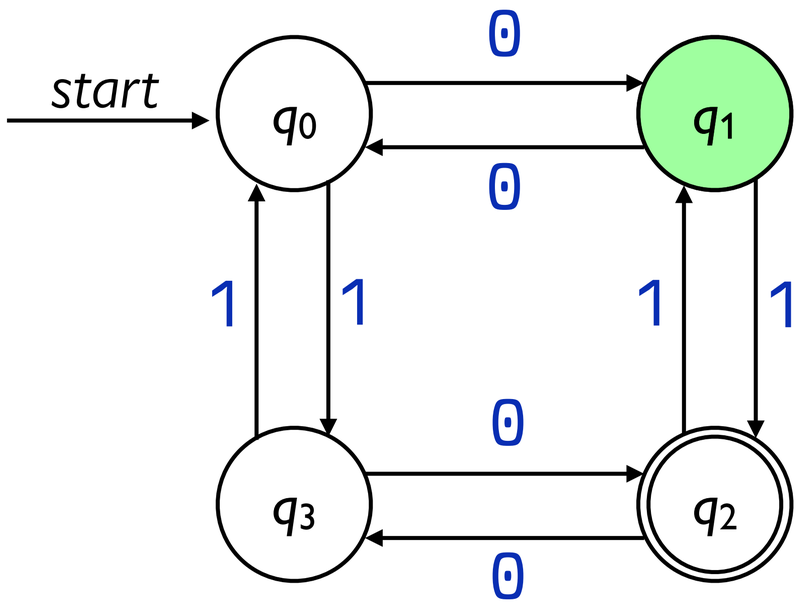

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0


1 0 1

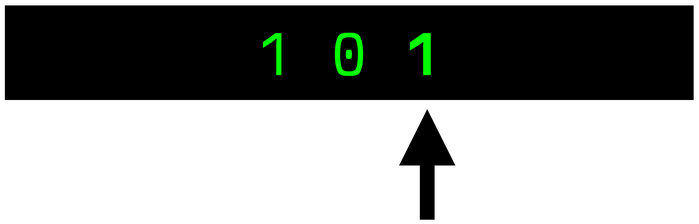

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

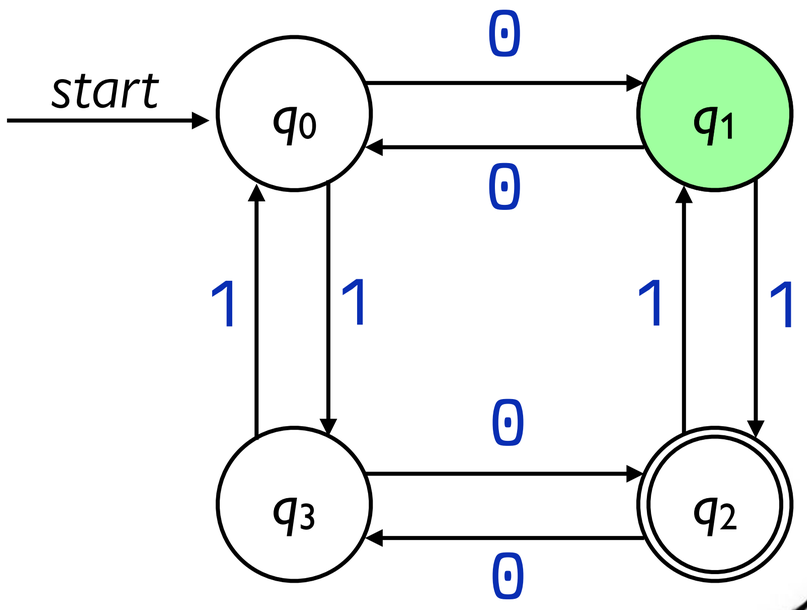

1 0 1


start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0
1 0 1

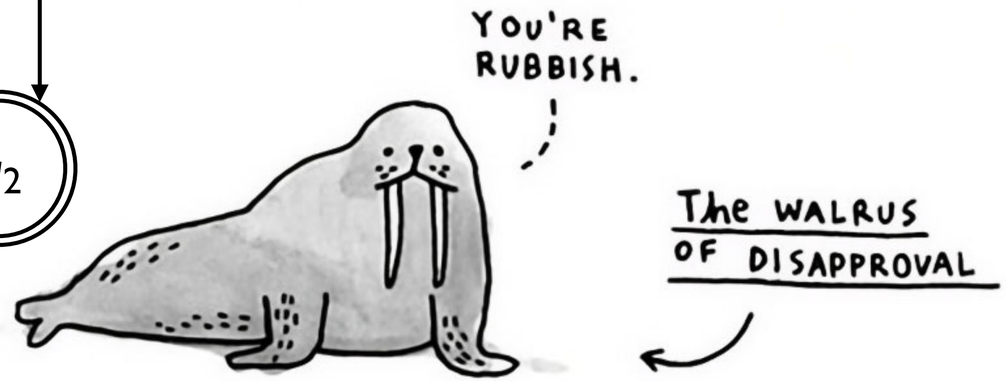


This state is not an accept state, so the automaton rejects.





This state is not an accept state, so the automaton rejects.



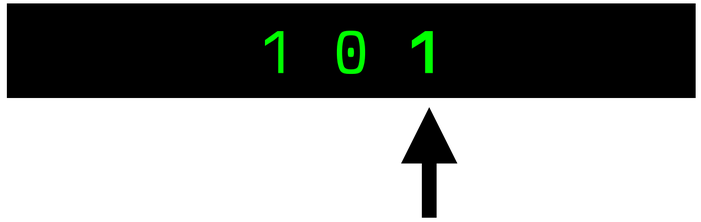


Illustration by Gemma Correll

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

- One state is designated the *start state*.
- Some states are *accept states*.
- Transition arcs are labeled with one or more symbols from some alphabet.

An automaton processes a string by beginning in the start state and following the indicated transitions.

The new state is completely determined by the current state and the symbol it just read.

When the input is exhausted,

If the automaton is in an accept state, it *accepts* the input.

Otherwise, it *rejects* the input.

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

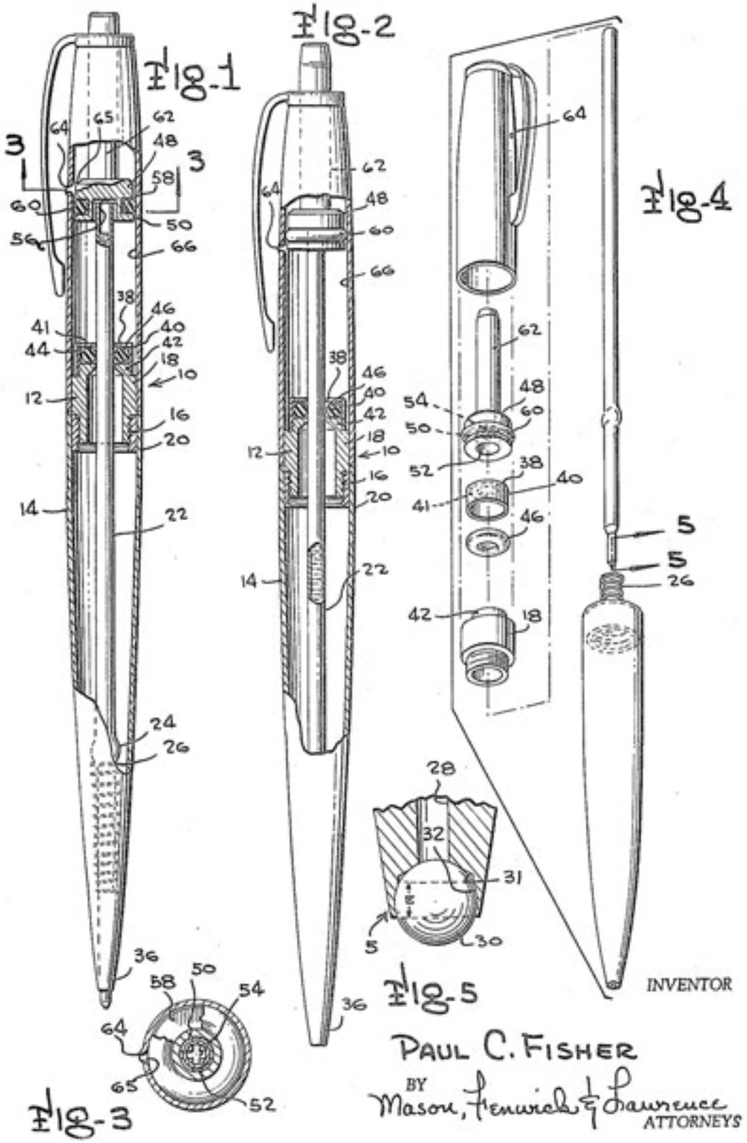
It only accepts if it *ends* in an accept state.

Finite-state machines are all around us.

Nov. 15, 1966 P. C. FISHER 3,285,228

ANTI-GRAVITY PEN

Filed May 19, 1965



INVENTOR

PAUL C. FISHER

BY Mason, Fenwick & Lawrence

ATTORNEYS

United States Patent [19]

Bako

[11] **3,894,415**

[45] **July 15, 1975**

[54] **COMBINATION PADLOCK**

[75] Inventor: **Lazlo Bako**, Woodcliff Lake, N.J.

[73] Assignee: **Presto Lock Company, Division of Walter Kidde & Company, Inc.**, Elmwood Park, N.J.

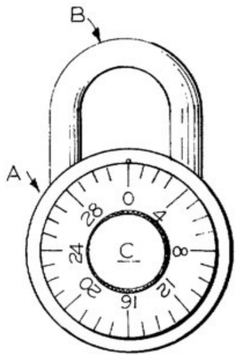
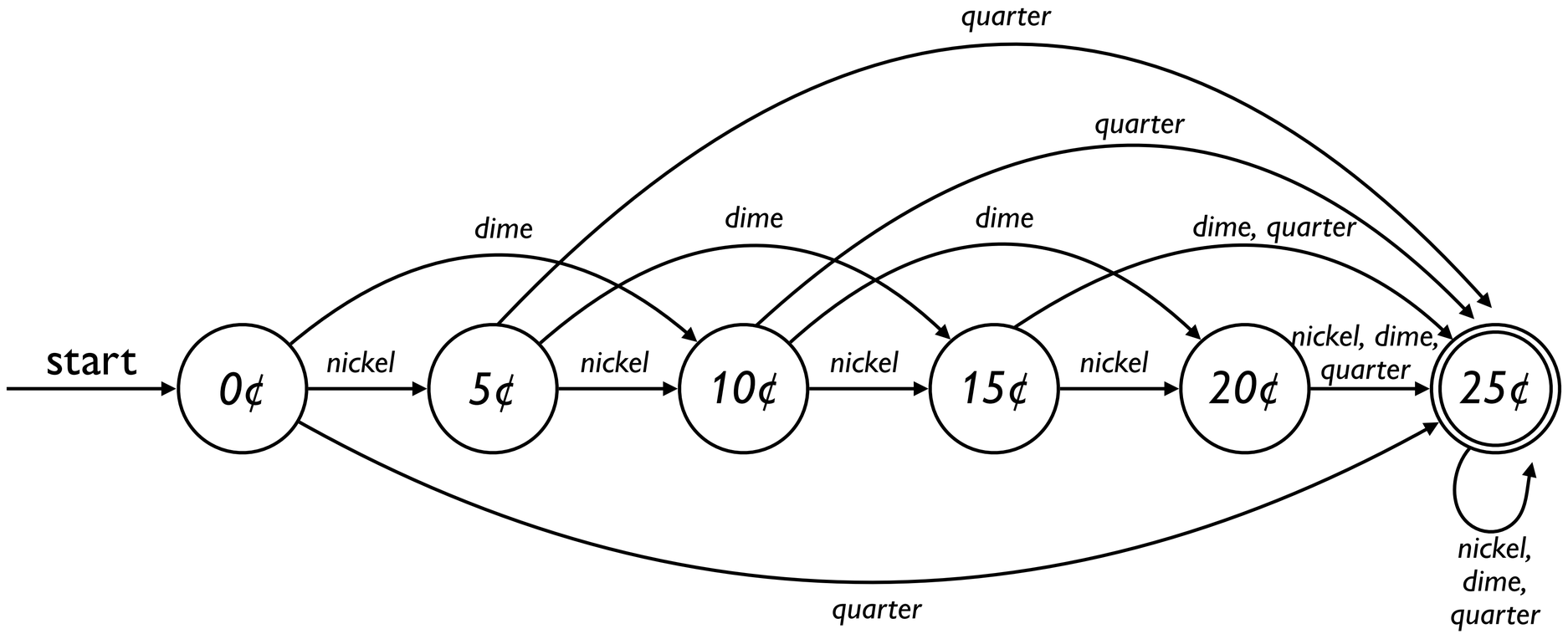


FIG.1

FIG.2

Evan Grothjan for The New York Times

Finite automaton for a newspaper vending machine



Other real-world finite automata?

Finite automata are used in

- Text editors and search engines for pattern matching
- Compilers for lexical analysis
- Web browsers for HTML parsing

They also serve as the control unit in many physical systems, including

- Elevators, traffic signals, vending machines
- Computer microprocessors
- Network protocol stacks and old VCR clocks

A final note

Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame

W. Daniel Hillis, *The Pattern on the Stone*

John Hopcroft, Cornell University

Nancy Ide, Vassar College

Keith Schwarz, Stanford University

Michael Sipser, *Introduction to the Theory of Computation*