Day Two

Problem Solving

Exploring Computer Science

Unit 1: Human Computer Interaction

Unit 2: Problem Solving

- Days 1-2: Intro to Data collection and problem solving **morning**
- Day 3: Steps in Problem Solving
- Days 4-6: Problem Solving Strategies
- Days 7-9: Reinforcing the phases in the problem solving process
- Days 10-12: Counting in Binary
- Days 13-14: Linear and binary search
- Days 15-16: Lists and sorting
- Days 17: Minimal spanning trees and graphs
- Days 18-21: Final unit projects **afternoon**

Unit 3: Web Design

Unit 4: Introduction to Programming

Unit 5: Computing Applications

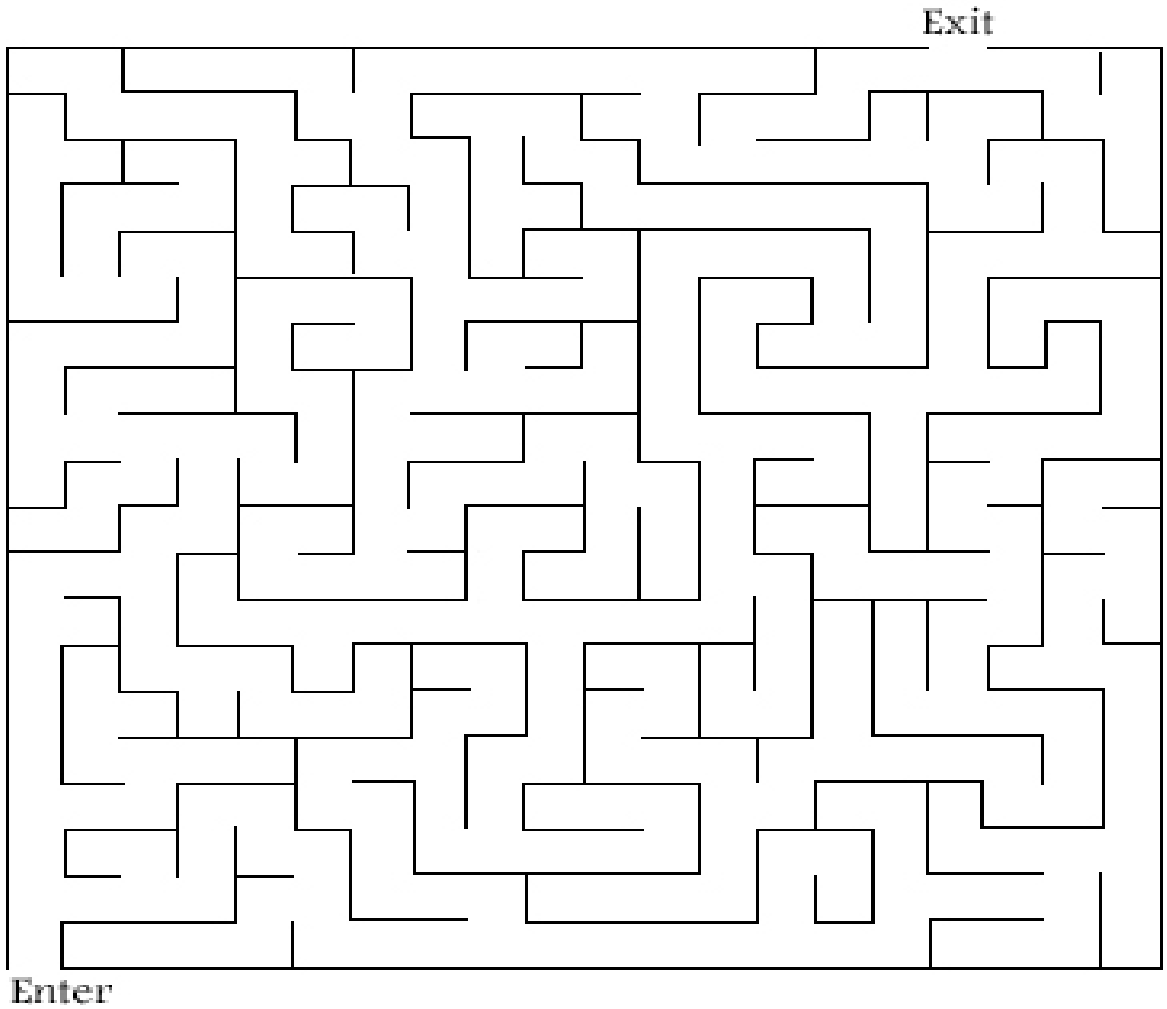
Unit 6: Robotics

No computers!

Problem solving

The purpose of computing
is ***insight***, not numbers.

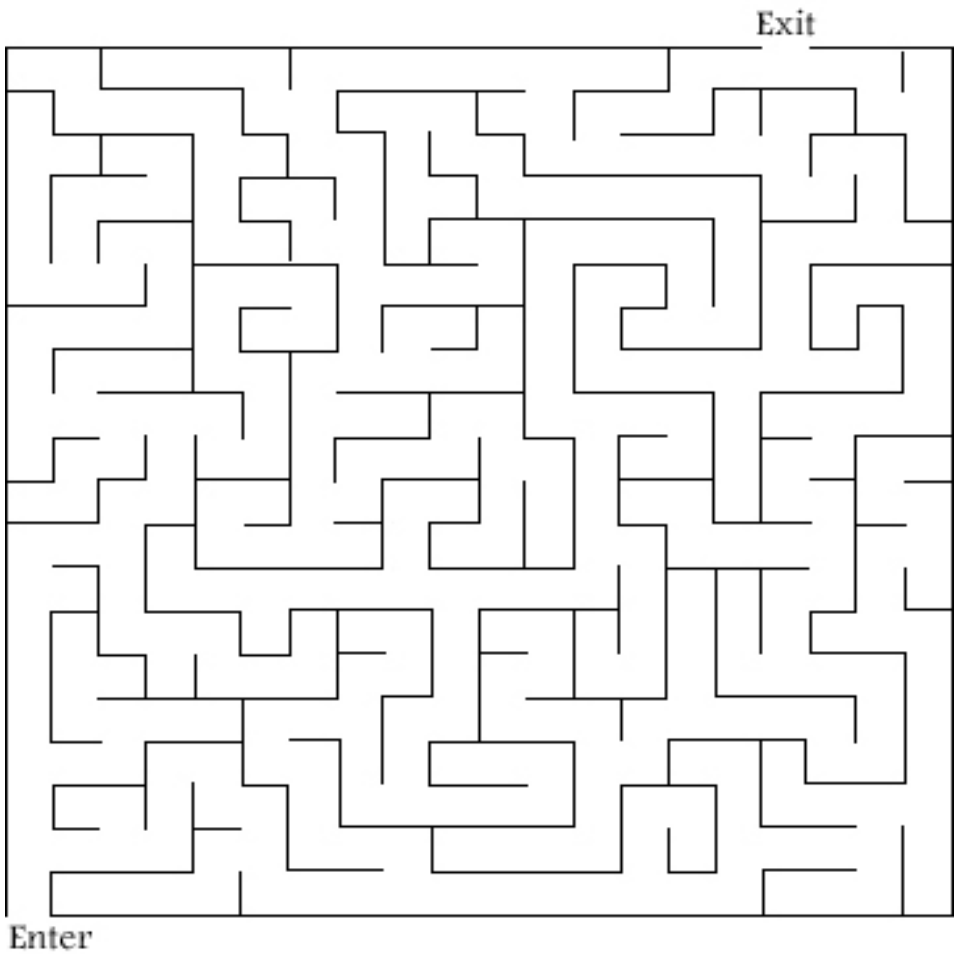
Problem solving *process*



Here's a familiar problem.

How would you solve it?

Problem solving *process*



(1) experiment / understand

-
-
-
-

(2) describe a plan

-
-
-
-

(3) test your plan (*execute* it)

-

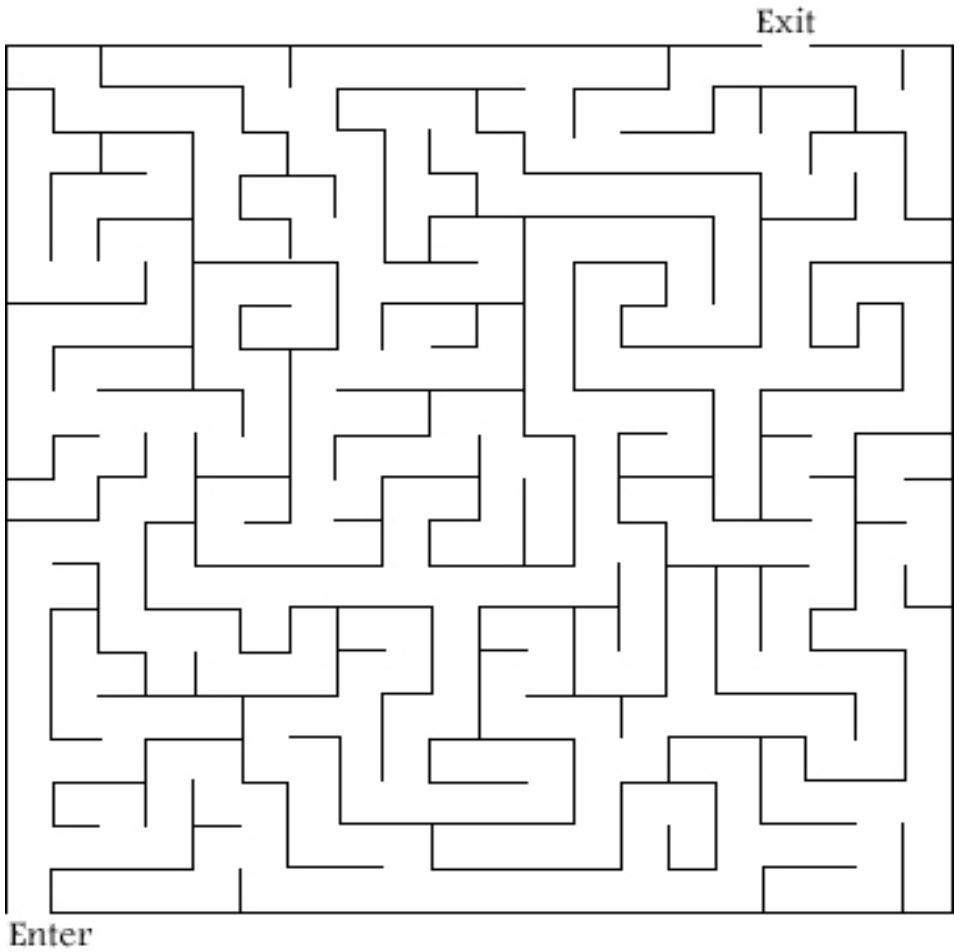
(4) reflect / evaluate

-
-
-
-

The ***process*** is much more useful than the solution!!

solving *one* vs. solving *many*

Problem solving *process*



(1) experiment / understand

- how to start?
- **what's the goal?**
- try a strategy... and/or another
- try smaller pieces

(2) describe a plan

- first step
- next step...
- **are there choices?**
- if so, how do you handle them

(3) test your plan (*execute* it)

- go through the steps in your plan

(4) reflect / evaluate

- does it work in this case?
- would it always work?
- **what are hard or easy cases?**
- **what are the applications of this?**

Problem solving *process*

"rook-jumping maze"

What is the ***path*** from the ring to the G?

Problem solving *process*

What is the ***path***?

(1) experiment / understand

- how to start?
- **what's the goal?**
- try a strategy... and/or another
- try smaller pieces

(2) describe a plan

- first step
- next step...
- **are there choices?**
- if so, how do you handle them

(3) test your plan (*execute* it)

- go through the steps in your plan

(4) reflect / evaluate

- does it work in this case?
- would it always work?
- **what are hard or easy cases?**
- **what are the applications of this?**

Problem solving ***process***

Brick-breaking!

of chocolate...

Q: How many breaks are needed to fully separate the pieces?

Try smaller examples...

# of pieces	# of breaks needed
12	*?*
#	#
#	#
#	#

(1) experiment / understand

may be a ***thought*** experiment (or taste!)

(2) describe a plan

(3) test your plan (*execute* it)

(4) reflect / evaluate

Q: How many breaks are needed?

Other examples in the ECS curriculum

There are 10 people (including you) at a party.

Q: How many handshakes do you need in order to greet everyone?

# in group	***# of h.shakes needed***
10	***?***
#	#
#	#
N	#

There are 10 people (including you) at a party.

Q: How many handshakes are needed so everyone greets everyone?

# in group	***# of h.shakes needed***
10	***?***
#	#
#	#
N	#

Other examples elsewhere...

Recognize this game show?

http://www.youtube.com/watch?v=WKR6dNDvHYQ

Let's Make a Deal...

1 2 3

Choose... Reveal... Switch or stay?

Let's Make a Deal...

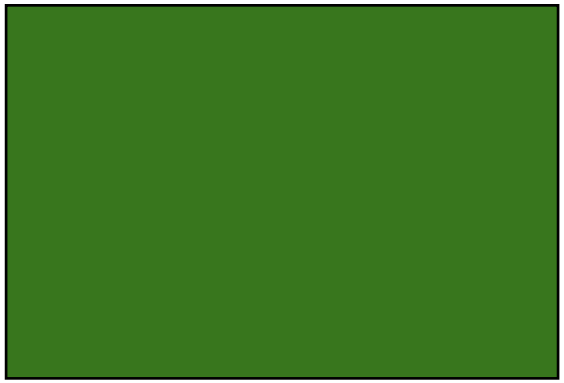
	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										

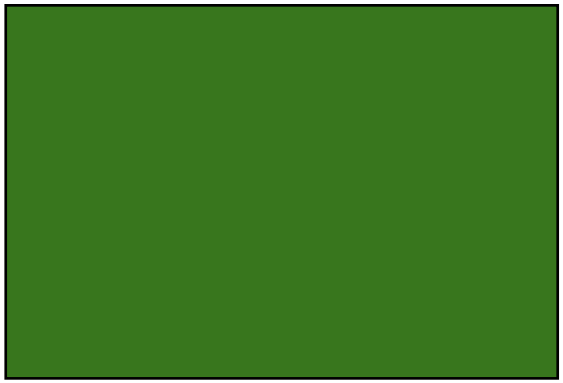
Let's Make a Deal...

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										

Two envelopes...

Both envelopes have some money in them (as a check). You also know that one envelope has double the amount in the other envelope.





Suppose you choose an envelope and
it has M money in it. Say, $10.

Now you have the opportunity to trade.
Should you?

Take-home message

Understanding problems is the key to solving them -- and is ***more important*** than solving a particular one!

CS is the field that analyzes and solves *information-based* problems, i. e., those based on **data**.

But there is ***so much <u>different</u>*** data!

How can we ***represent*** data in a single, consistent way *so that machines can process it* ?

Representing data: ***Binary***

Binary Lab/Activity

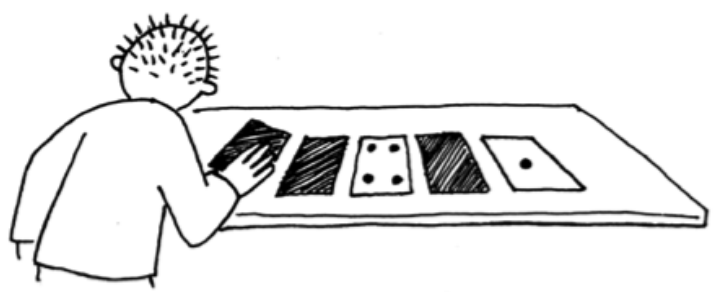
Complete the table by converting from binary to decimal, and from decimal to binary:

Binary	Decimal
101	
0101	
1111	
11110	
	6
	21
	42
	63
	64

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	c	d	e	f	g	h	i	j	k	l	m
14	15	16	17	18	19	20	21	22	23	24	25	26
n	o	p	q	r	s	t	u	v	w	x	y	z

beepin' ...

boopin' ...



Other bases: Hexadecimal

Just like binary is base 2, we can create numbers in any base we want!

Computers often use hexadecimal (base 16) to represent color values.

Since we only have 10 arabic numerals, we also use 6 letters (a-f) as the additional digits needed in this base. (We need to represent 10 through 15 with only one character, so a=10, b=11, c=12, d=13, e=14, f=15)

For example in hexadecimal:

$$ff = f * 16^1 + f * 16^0 = 15*16 + 15 * 1 = 255$$

Other bases: Hexadecimal

Hex	Decimal
10	
31	
A5	
FF	
	20
	40
	100
	254
	0

Colors in Hexadecimal!

Each color has an RGB (Red-Green-Blue) value which represents how much of each primary additive color is needed to make that color (with a value from 0-255).

For example Red = 255, 0, 0,
Black = 0, 0, 0,
White = 255, 255, 255

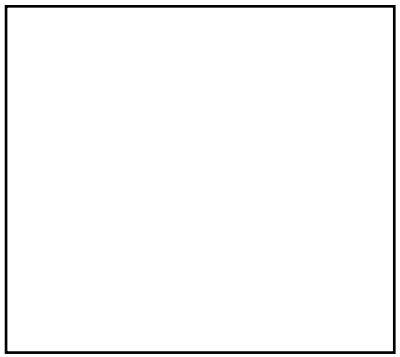
Since these numbers can all be represented by only 2 digits, to save space, computers represent them in hexadecimal.

So, Red = FF0000
Black = 000000
White = FFFFFF

In reality, people just look up hexadecimal color codes online.

RGB values!

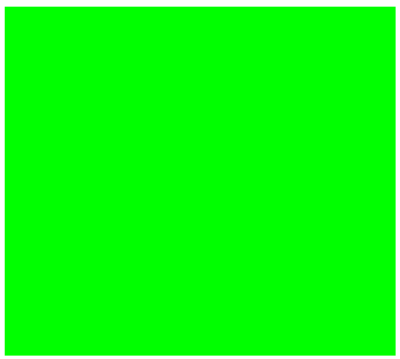
check your guess...



Red Value	Green Value	Blue Value
255 (FF)	0 (00)	0 (00)
255 (FF)	255 (FF)	255 (FF)
0 (0)	0 (0)	0 (0)

Estimate the RGB values!

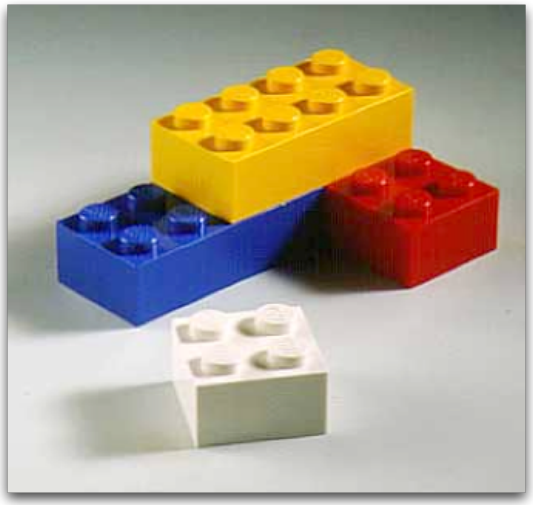
check your guess...



Red Value	Green Value	Blue Value
0	0	255
	255	0
50	50	50

Coffee Break

Unit 2 Lab: Lego Encoding (Part 1)



Challenge:

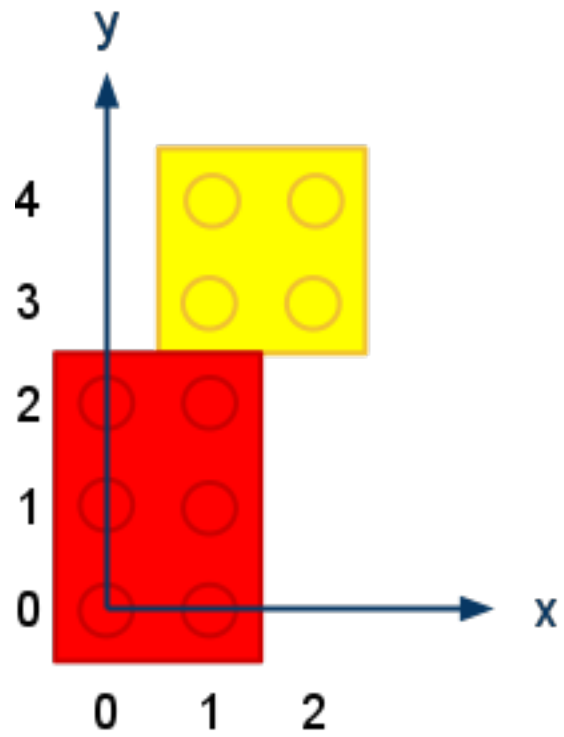
To encode a lego structure with binary numbers.

Then to decode those numbers to build the structure.

Lego example!

If we want to encode the blocks at right:

x- coordinate	y-coordinate	color	brick type	orientation
0	0	red	2X3	vertical
1	3	yellow	2X2	horizontal



x- coordinate	y-coordinate	color	brick type	orientation
00	00	0	0	1
01	11	1	1	0

Legend:

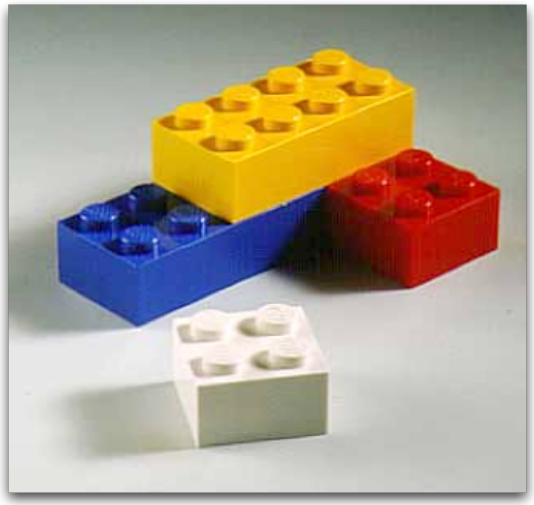
color	
red	0
yellow	1

brick type	
2X3	0
2X2	1

orientation	
horizontal	0
vertical	1

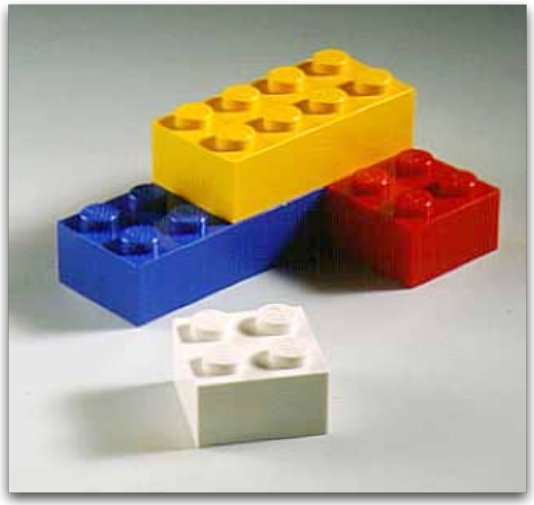
Unit 2 Lab: Lego Encoding (Part 1)

- Make a structure out of your 8 legos.



- REMEMBER the order in which you placed the legos!

Unit 2 Lab: Lego Encoding (Part 1)



- Make a structure out of your 8 legos.
- In binary, encode the information to build the structure, so that it is reproducible.
- Swap instructions and try to build another group's structure.

You will have to supply the position (in xyz coordinates), color, brick type, and rotation of each lego.

Let red = 0, yellow = 1, white = 2, blue = 3. Convert these numbers into binary.

Unit 2 Lab: Legos!

color	
red	000
yellow	001

Finish filling in the legend tables, and encode your tower!

brick type	
2X4	0000
2X2	0001

orientation	
horizontal	0
vertical	1

Then, another group will decode your instructions and try to build your tower.

Brick	x-coordinate	y-coordinate	color	brick type	orientation

Pause. Thoughts... ?

Exploring Computer Science

Unit 1: Human Computer Interaction

Unit 2: Problem Solving

- Days 1-2: Intro to Data collection and problem solving **morning**
- Day 3: Steps in Problem Solving
- Days 4-6: Problem Solving Strategies
- Days 7-9: Reinforcing the phases in the problem solving process
- Days 10-12: Counting in Binary
- Days 13-14: Linear and binary search
- Days 15-16: Lists and sorting
- Days 17: Minimal spanning trees and graphs
- Days 18-21: Final unit projects **afternoon**

Unit 3: Web Design

Unit 4: Introduction to Programming

Unit 5: Computing Applications

Unit 6: Robotics

No computers!

Unit 2 Lab: Treasure Island Game

We will demonstrate a shortened version of the game:

Starting at Jolly Roger Bay, you must discover directions that will get you to Treasure Island without a map. (by guessing)

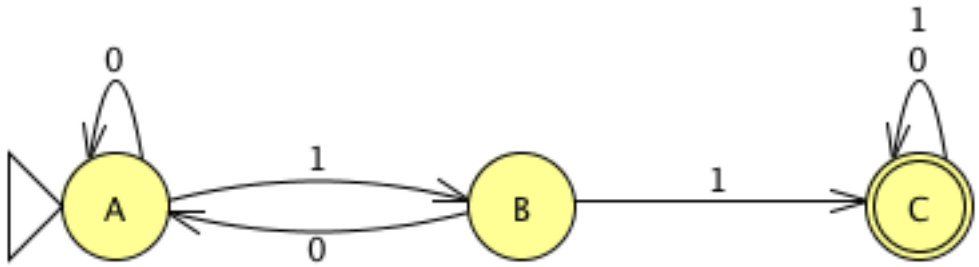
As you watch, think about the following questions,

- Is there only one set of "correct" directions?
- How can you get to Treasure Island in the least number of steps?

and write your observations in your journal.

After, you will have an opportunity to share your thoughts.

Unit 2 Lab: Treasure Island Solution



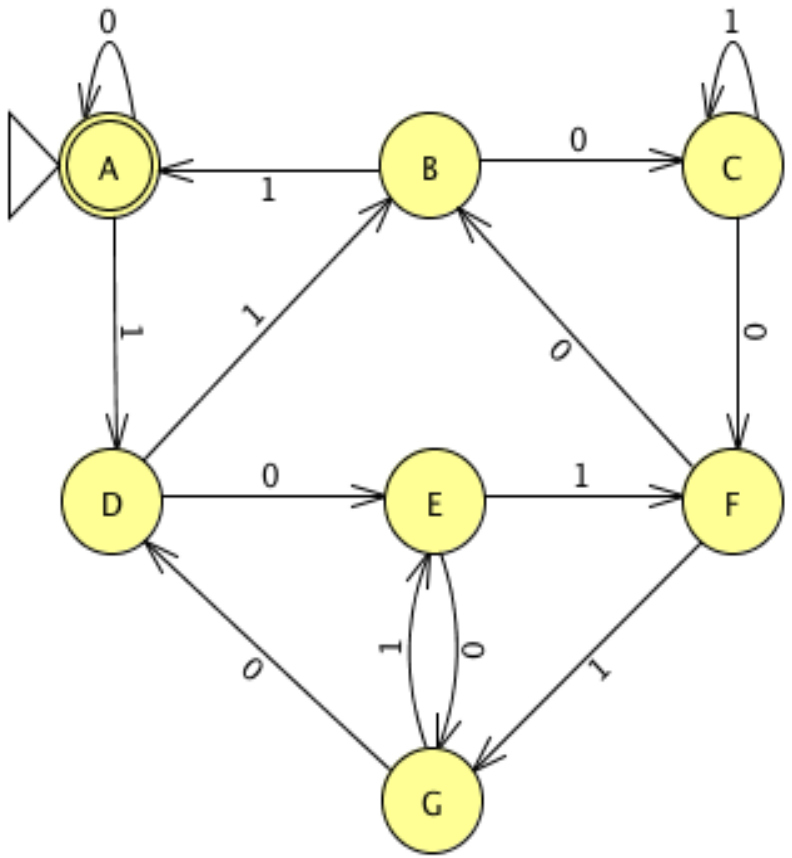
This set of islands can be represented by what we call a **Finite State Machine**.

Which circle represents Jolly Roger Bay? How is it marked as the start of your journey?

Which circle represents Treasure Island? How is it marked as your desired destination?

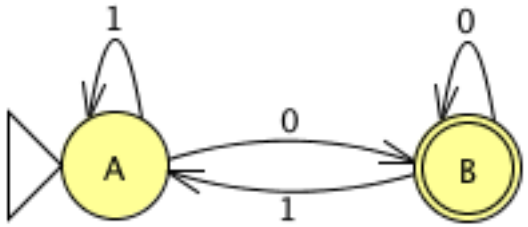
Unit 2 Lab: Finite State Machines (Part 2)

- Take an input (instructions) and then either accept or reject the input.
- Each circle represents a "state."
- Here, the input is a binary string.
- Start at triangle, follow the arrows.
- Double circles accept, single circles reject.
- Does this FSM accept or reject the string 1101001?



Mystery FSM

Input	Accepted?
0	Yes
1	No
01	No
11	No
10	Yes
111	No
101	No
110	Yes
010	Yes
011	No
010	Yes

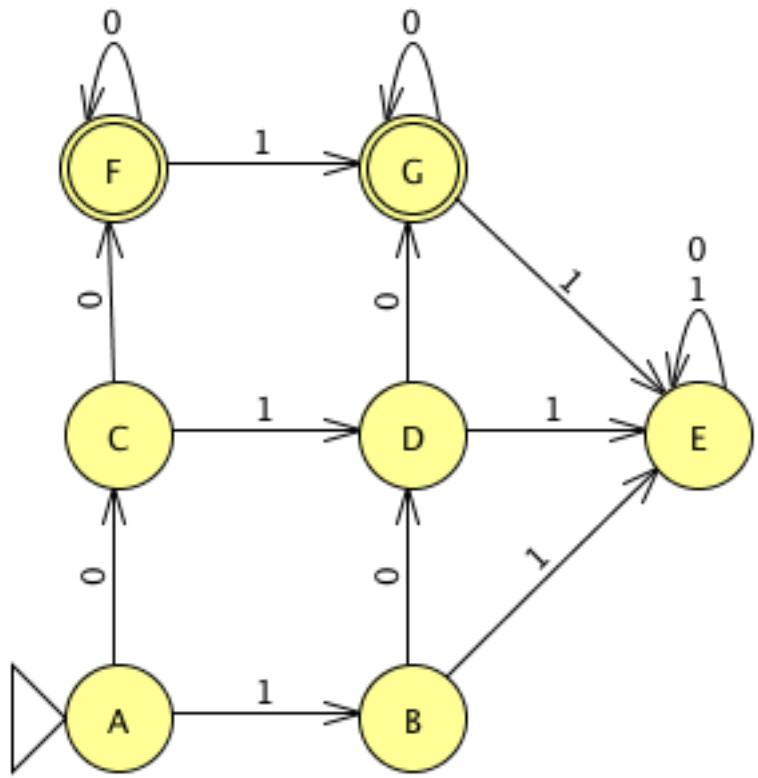


Which strings does this FSM accept, generally?

What do the states (circles) represent?

Mystery FSM 2!

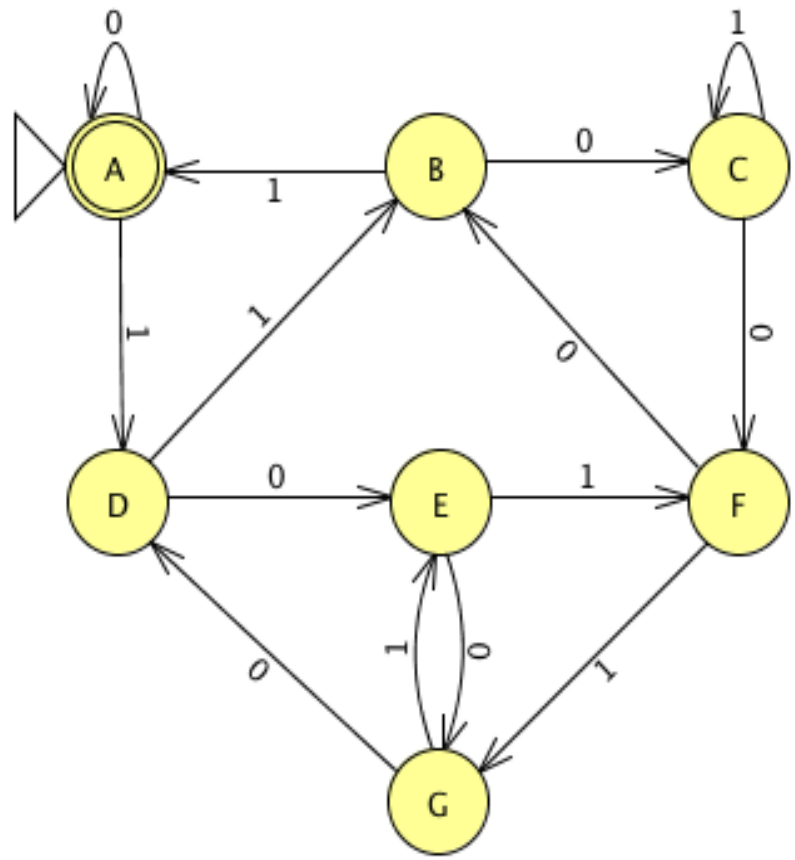
Input	Accepted?
0	
1	
10	
111	
100	
101	
110	
111	
1000	
1001	
1010	
1011	



Which strings does this FSM accept, generally?

Mystery FSM 3! (from the first slide)
Robyn & Johnny

Input	Accepted?
0	Yes
1	
10	
111	
100	
101	
110	
111	
1000	
10010.	
1010	
1011	
1100	
1101	
1110	
1111	
10000	
101010	

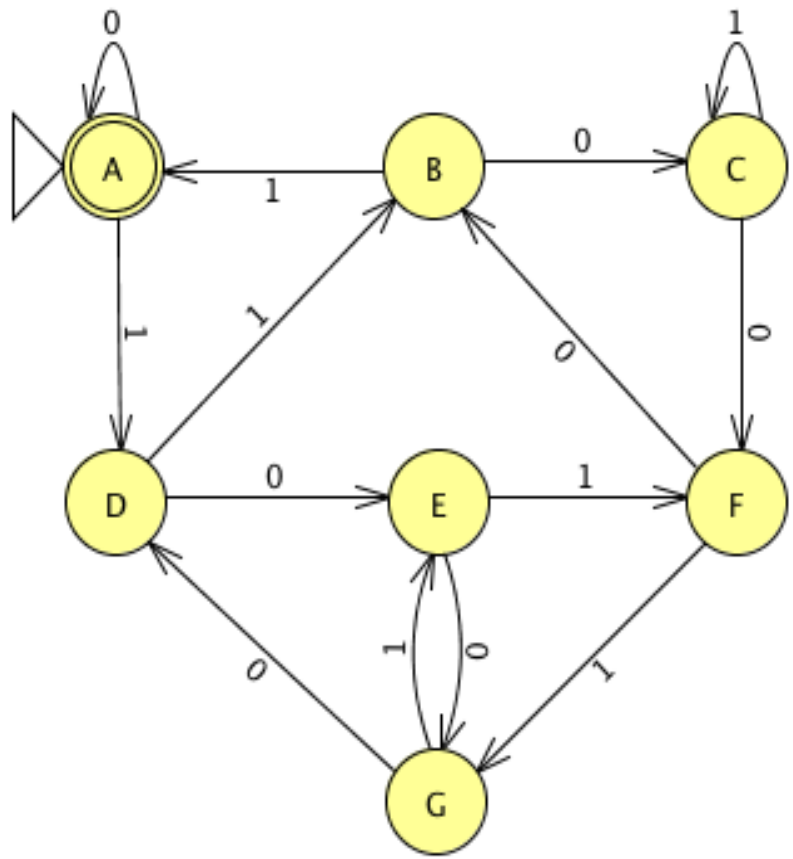


Which strings does this FSM accept, generally?

Mystery FSM 3! (from the first slide)

Patrick Susan

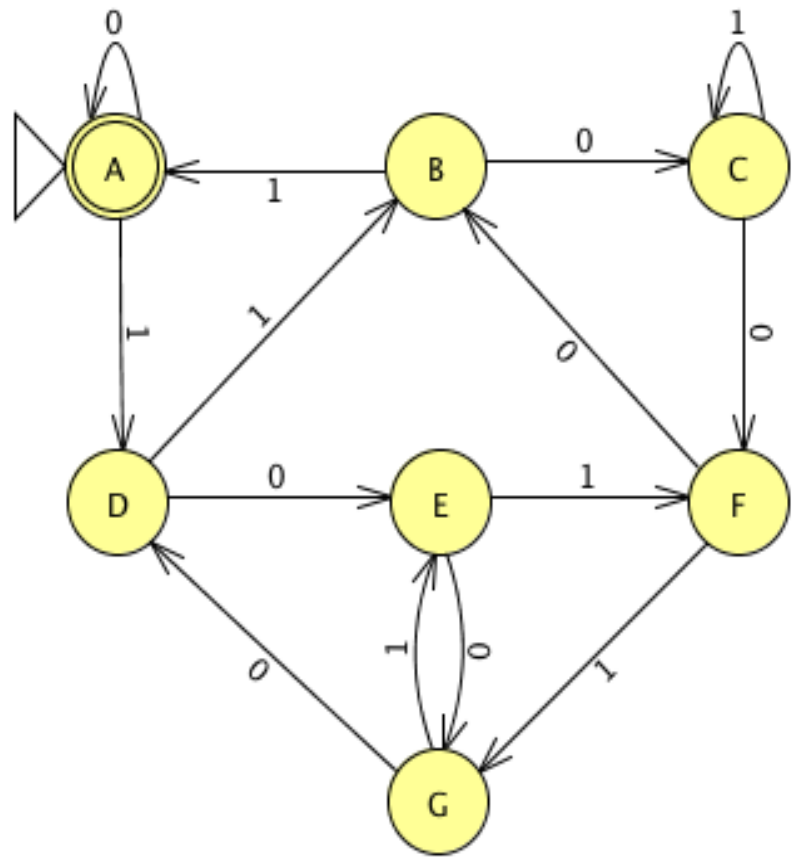
Input	Accepted?
0	yes
1	
10	
111	
100	no
101	no
110	
111	yes
1000	no
1001	no
1010	no
1011	no
1100	no
1101	no
1110	yes
1111	no
10000	no
101010	yes



Which strings does this FSM accept, generally?

Mystery FSM 3! (from the first slide)
Cheryl and Jen!

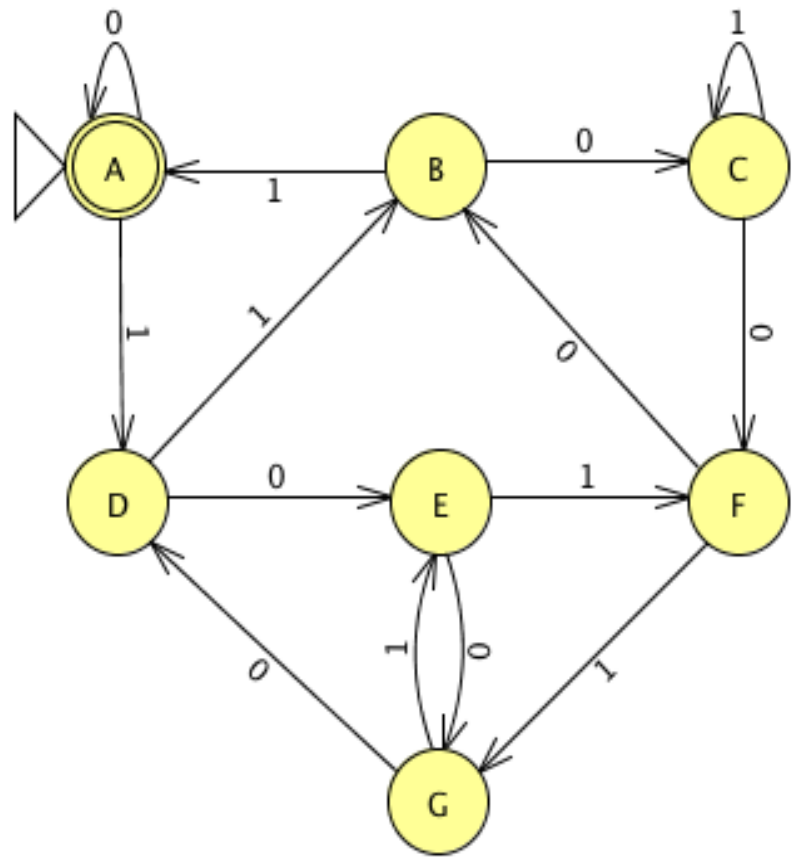
Input	Accepted?
0	y
1	
10	
111	
100	n g
101	n f
110	
111	YES
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	
10000	
101010	



Which strings does this FSM accept, generally?

ours

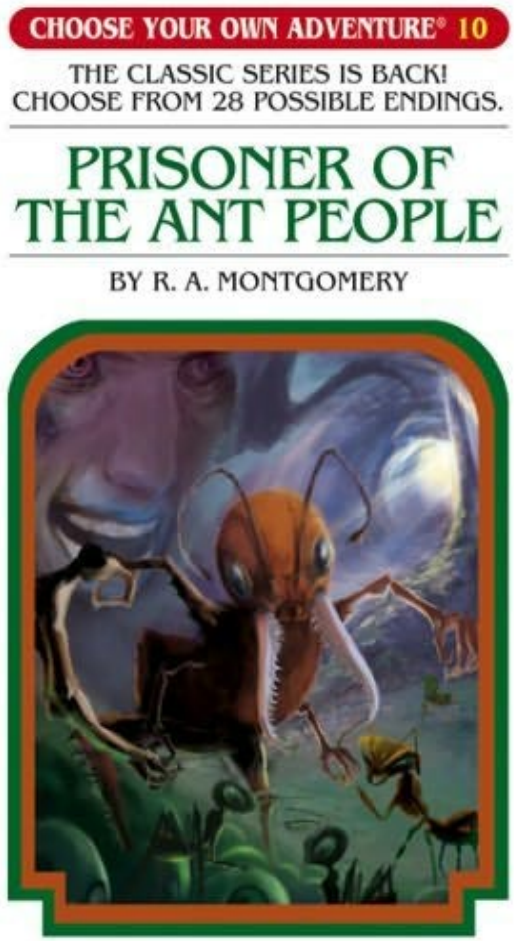
Input	Accepted?
0	yes
1	
10	
111	
100	
101	
110	
111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	
10000	
101010	

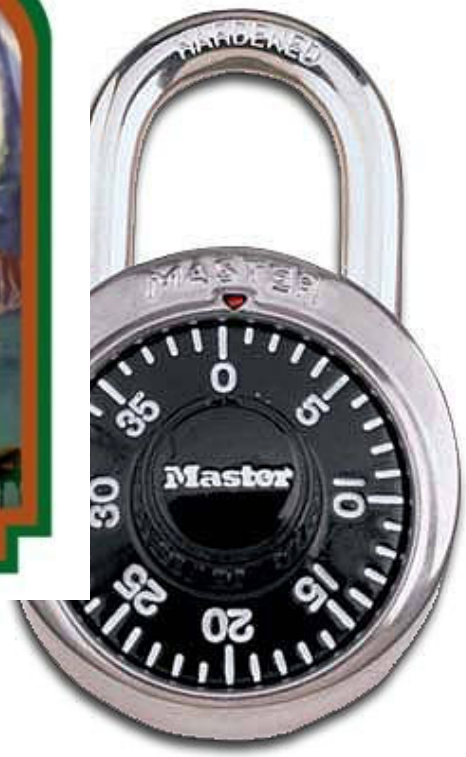


Which strings does this FSM accept, generally?

Applications of Finite State Machines

The principles of finite state machines show up everywhere in our everyday life. Guess which of these use finite state machines to operate.

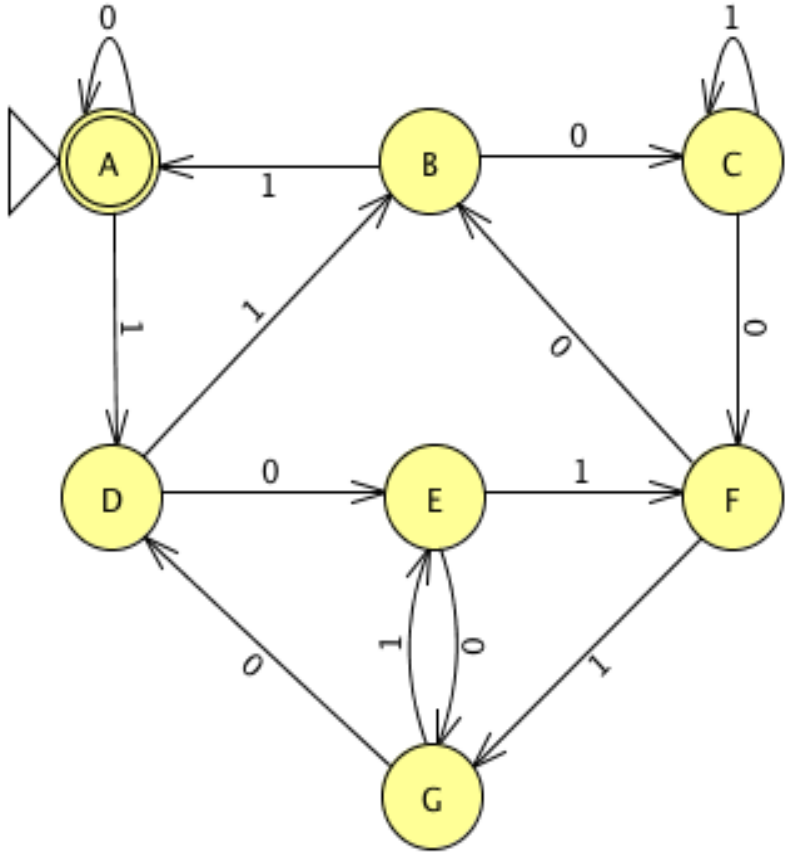




JFLAP

How did we make these beautiful FSMs?

JFLAP is a cross-platform utility for creating and testing finite state machines (and lots of other abstract machines!)



Lunch Videos

The Creation of the Computer- middle section

Data

Up until now, we've been looking at ones and zeros. These are really abstract pieces of data!

All of the little ones and zeroes (bits) are used to represent pieces of information that are useful to us, like bank records, or grade books.

Lots of compelling problem solving in Computer Science happens at the **data structure** level.

One of the simplest data structures we can imagine is a **list**.

Tower Building Activity

Prompt:

Donald Trump wants to build a 100 meter high tower as quickly as possible. He has unlimited resources and unlimited budget and is willing to spend any amount to get the job done.

He has chosen to build the tower with blocks that are only 1 meter tall. The blocks interlock on top and bottom (like legos). They cannot be stacked sideways.

Using special lifters, putting one block on top of another block takes one week regardless of how high the stacks are.

What is the shortest amount of time that it will take to build the tower?

Tower Building

Think about your tower building strategy.
How many weeks will it take to build the tower?

Can you think of any faster strategies?
(you can use as many machines as you want!)

Tower-building strategies

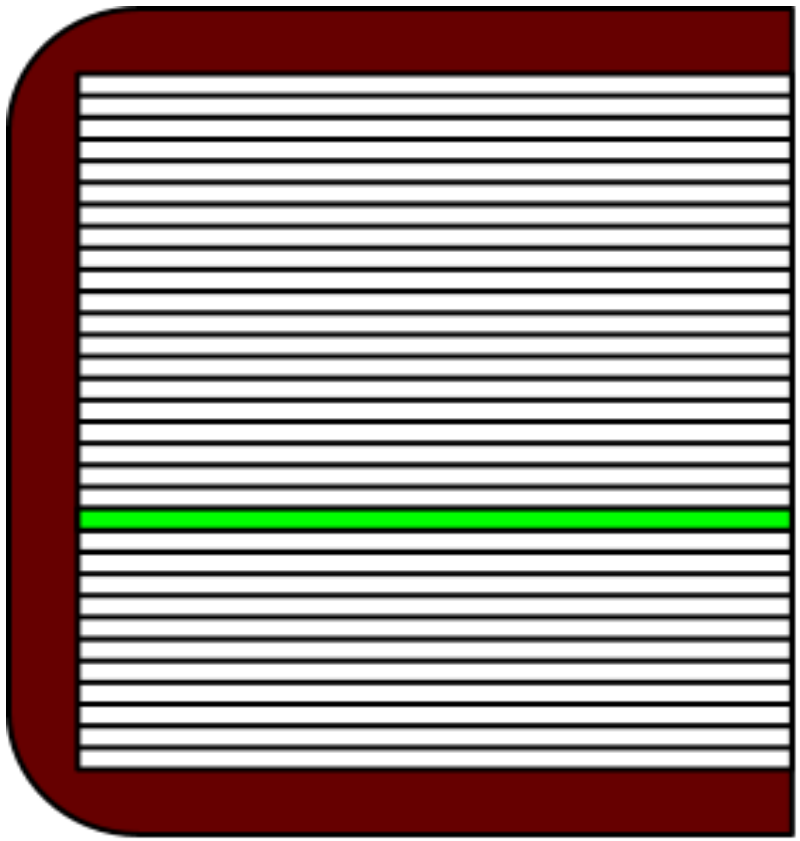
What are different strategies that were used to solve the tower building problem?

What's the **slowest** (straightforward) way?

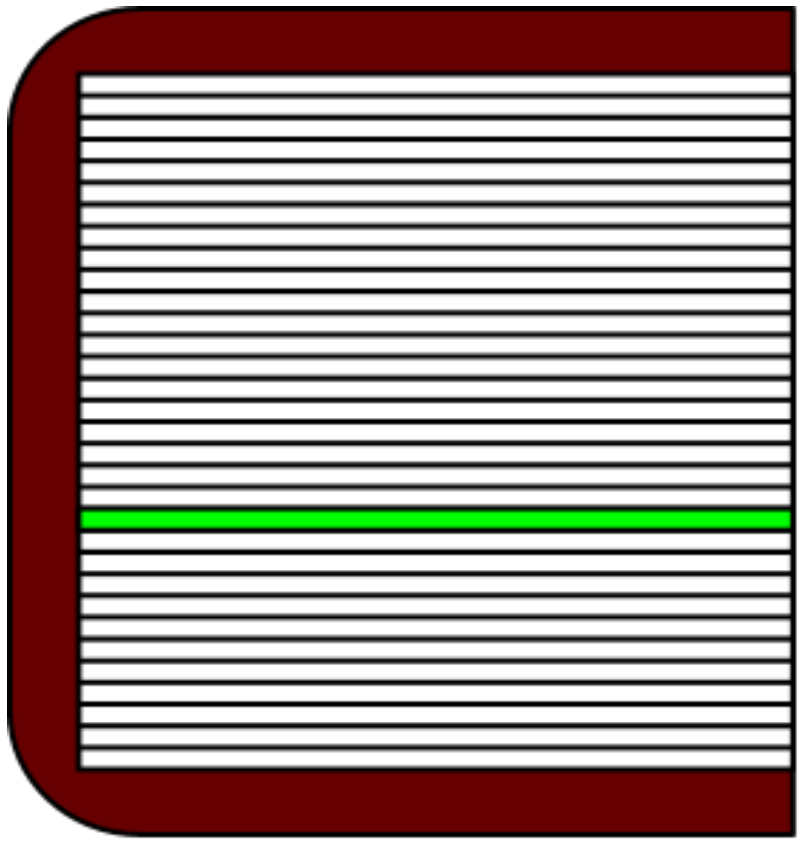
What's the **fastest** way to build the tower?

Searching!

"spam"



Wubster's
Dictionary

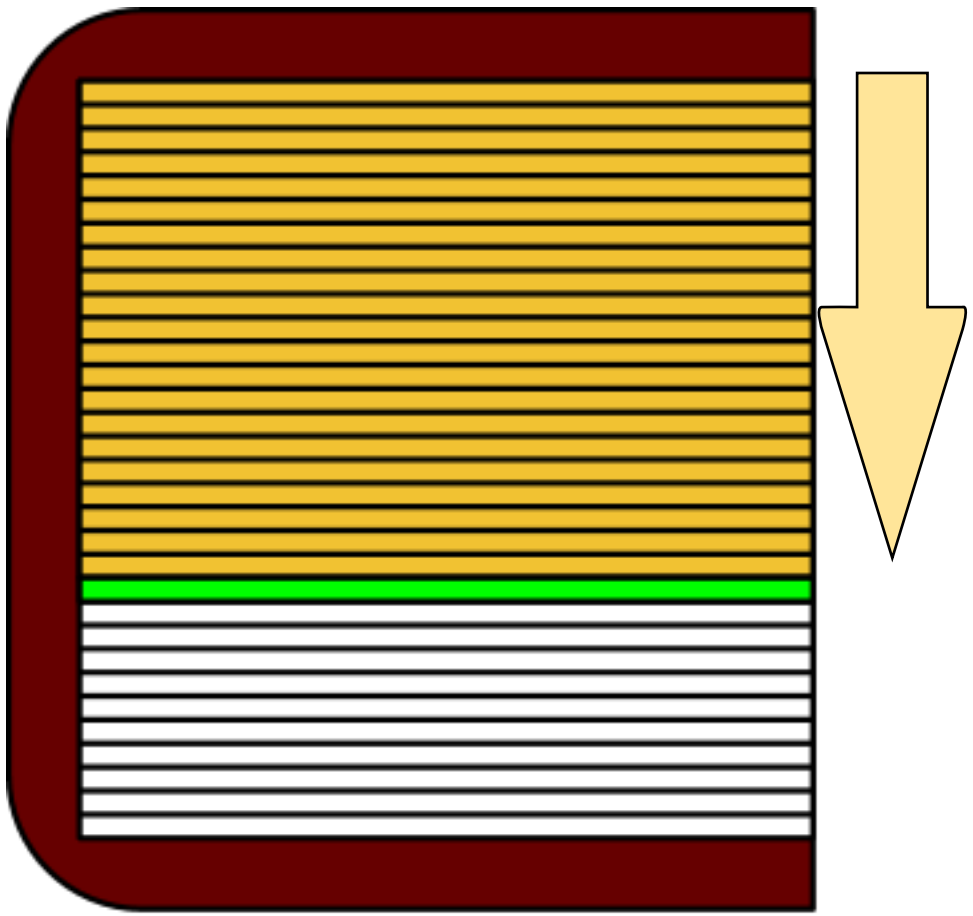


Webster's
Dictionary

What ***algorithm*** do we use to find "spam"?

Linear Search

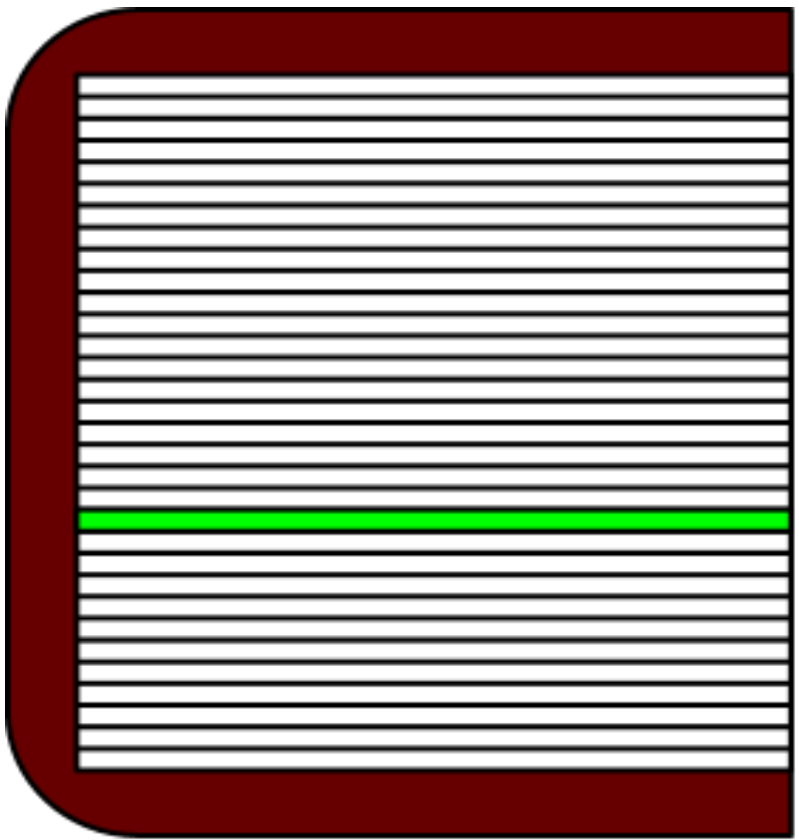
- Start at the first word.
- "Is this the word we want?"
- If it's not, go to the next word.



How many words did we have to look at before we got to the right one?

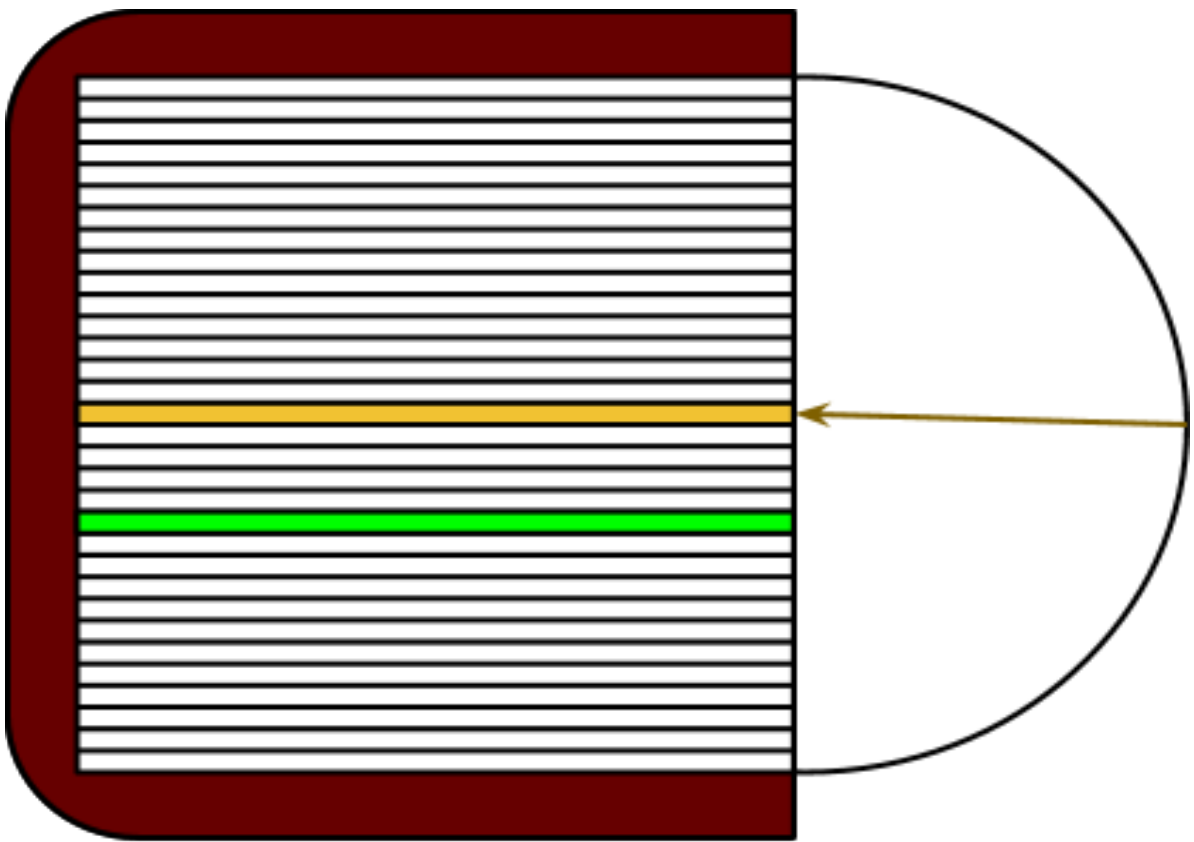
Binary Search

- Start in the middle.
- "Is this the word we want?"
  - If it's before the word we want, go to the middle of the second half.
  - If it's after the word we want, go to the middle of the first half.



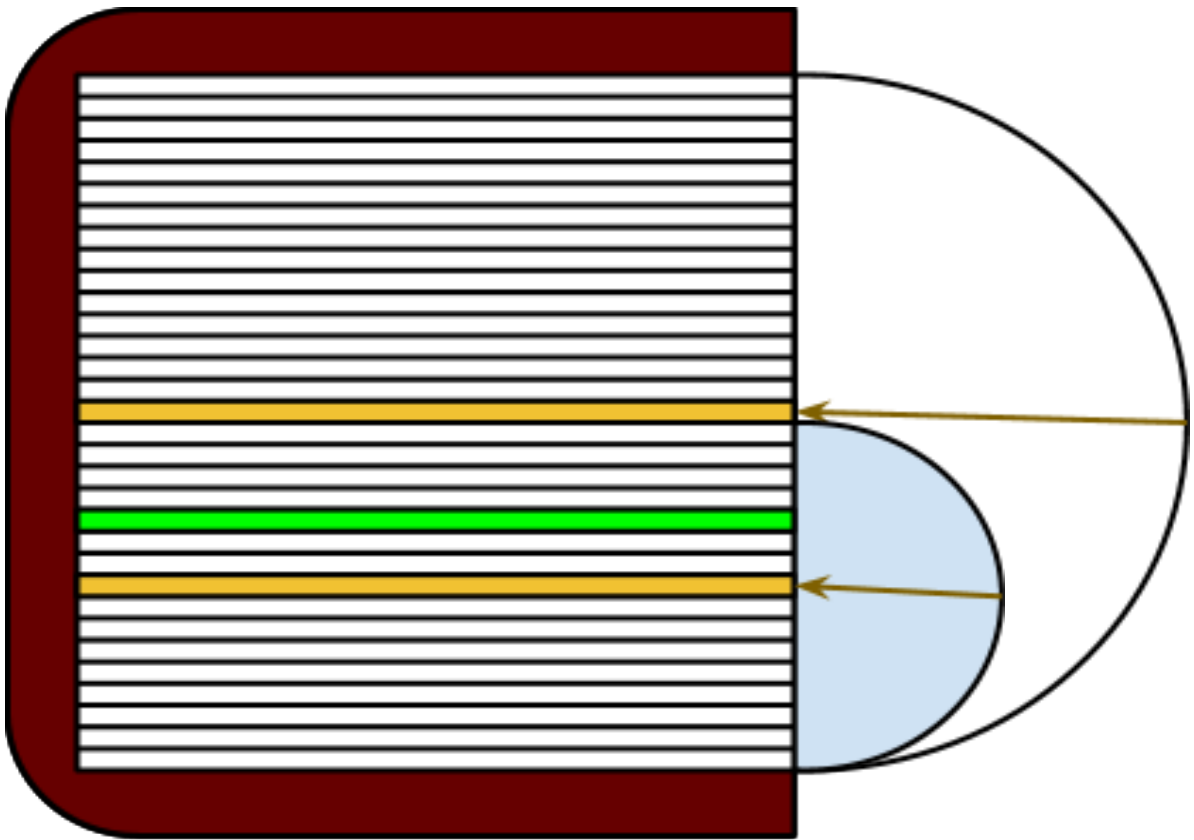
Binary Search

- Start in the middle.
- "Is this the word we want?"
  - If it's before the word we want, go to the middle of the second half.
  - If it's after the word we want, go to the middle of the first half.



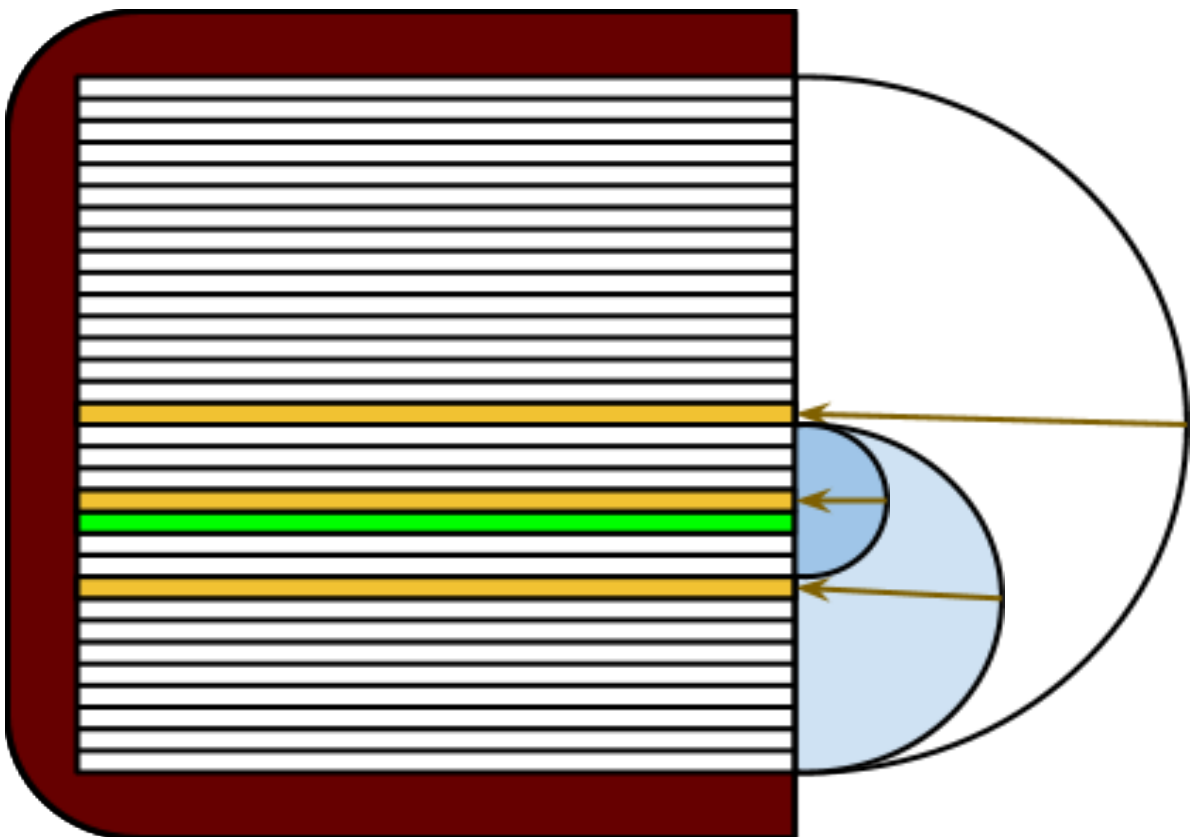
Binary Search

- Start in the middle.
- "Is this the word we want?"
  - If it's before the word we want, go to the middle of the second half.
  - If it's after the word we want, go to the middle of the first half.



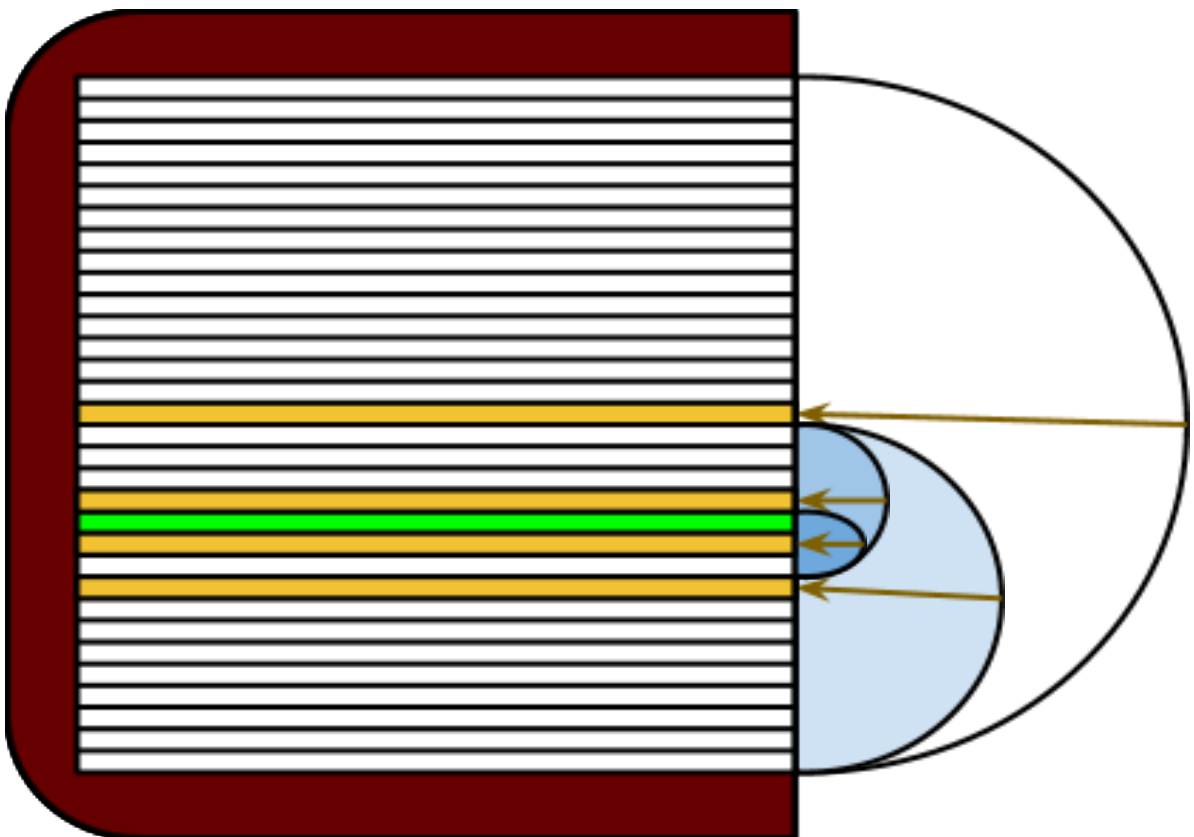
Binary Search

- Start in the middle.
- "Is this the word we want?"
  - If it's before the word we want, go to the middle of the second half.
  - If it's after the word we want, go to the middle of the first half.



Binary Search

- Start in the middle.
- "Is this the word we want?"
  - If it's before the word we want, go to the middle of the second half.
  - If it's after the word we want, go to the middle of the first half.



Is this **binary searching** going to be better than linear searching?

If the dictionary had 100 words in it, how many words did we have to look at before we got to the right one?

Linear vs. Binary Search

With a large amount of ***sorted*** data, binary search is almost always more efficient.

But ***how*** do we get our data sorted?

Sorting

Challenge: Sort a list of numbers in increasing order using as few comparisons at possible.

Let's sort!

Thinking Like A Computer

Put your blinders on!

We can only compare two elements at a time.

Computer's can't tell when a list is sorted "by inspection" like we can. They need a concrete set of conditions that lets them know that a list is sorted.

Binary vs. linear sorting ?!?

Let's try two sorting strategies:

(1) "Minsort"

(2) "Mergesort"

We'll keep track of the number of comparisons.

Stop!

Photo Op!

Binary vs. linear sorting ?!?

How did they compare?

(1) "Minsort"

(2) "Mergesort"

How does this relate to dictionary-searching?

ECS sorting material

Video

- minsort/maxsort
- bubble sort
- merge sort
- quick sort

Different sorting algorithms

- minsort/maxsort
- bubble sort
- merge sort
- quick sort

Which of these do you think is the fastest (requires the least comparisons)?

Minsort

The vague, handwavey idea:

Search for the first element, then the second, and so on...

Minsort

- Compare elements pairwise and find the minimum element in the list. Remove this element from the unsorted list.
- Place this element at the end of the sorted list.

Repeat to find the second, third...

- We know we are finished when there are no elements in the unsorted list. (This means all of the elements are sorted, in the other list!)

Minsort

Let's try it!

How many comparisons did minsort take?

Bubble Sort

The vague, handwavey idea:

Sort the entire list at once by switching elements if we notice them out of order.

Bubble Sort

- Compare each set of adjacent elements.
- If we find two elements that are "out of order", swap them.

Repeat.

- We know we are finished when there are no elements "out of order".

Bubble Sort

Let's try it!

How many comparisons did bubble sort take?

Merge Sort

Can we sort faster than this?

When we have two lists that are already sorted, making one larger sorted list from them is very easy.

Merge Sort

To merge two sorted lists:

- Compare the first element of each list, and remove the lesser.
- This element should be the first element of the sorted list.

  Repeat to find the second, third, ...

- When there are no elements in one of the lists, we can simply append the other list to the sorted list.

Merge Sort

- Split the list in half(ish) again and again until you have a bunch of lists of one element.
  - One element lists are *already sorted.*
- Merge these one-element sorted lists to get two-element sorted lists.
- Merge the two-element sorted lists to get four-element sorted lists...
- And so on, until all of your lists have been merged into one big sorted list!

Merge Sort

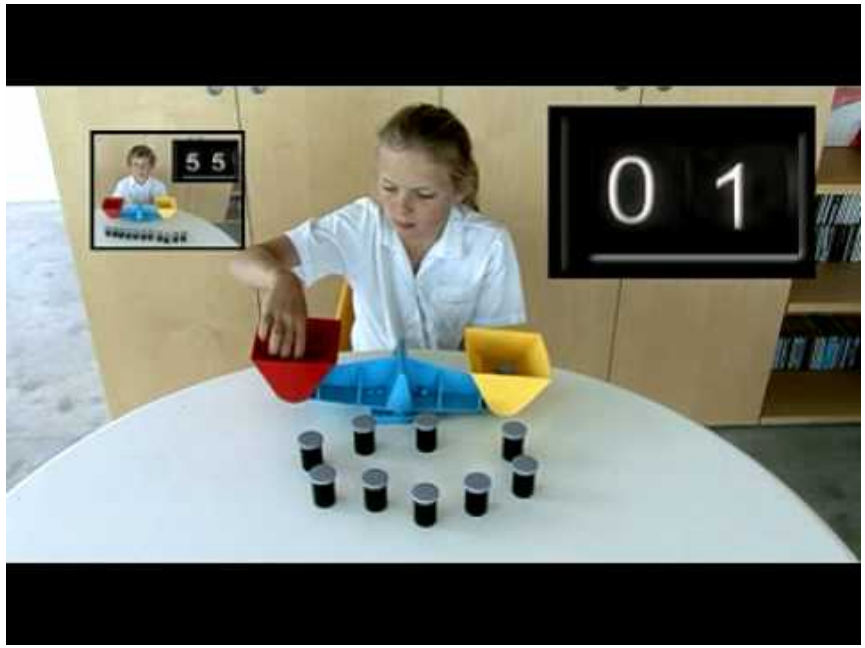
Let's try it!

How many comparisons did merge sort take?

Quick Sort

- Pick an element in the list. (Any element will do.)
  - "Weirdo"
- Compare each element in the list to the Weirdo.
  - If the element is greater than the Weirdo, put it to the Weirdo's right.
  - If it's less than the Weirdo, put it to the Weirdo's left.
- We now have two smaller unsorted lists!
- Sort the smaller lists.
  - How? Using quick sort!

Sorting CSUnplugged Video

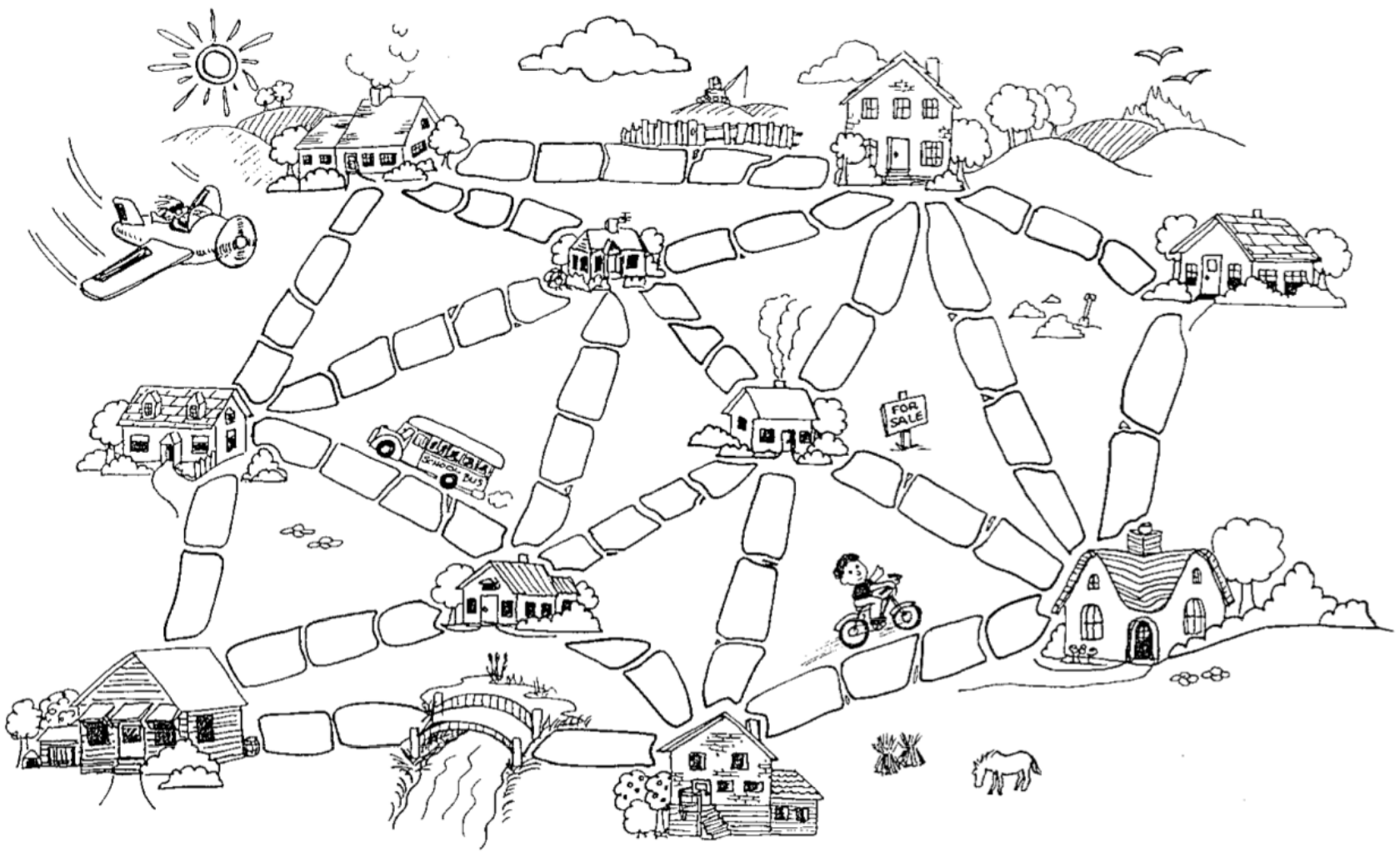


Which sorting algorithm is your favorite? Why?

View animations of different sorting algorithms here:
http://commons.wikimedia.org/wiki/Category:Animations_of_sort_algorithms

Muddy City

The Muddy City Activity from CS Unplugged

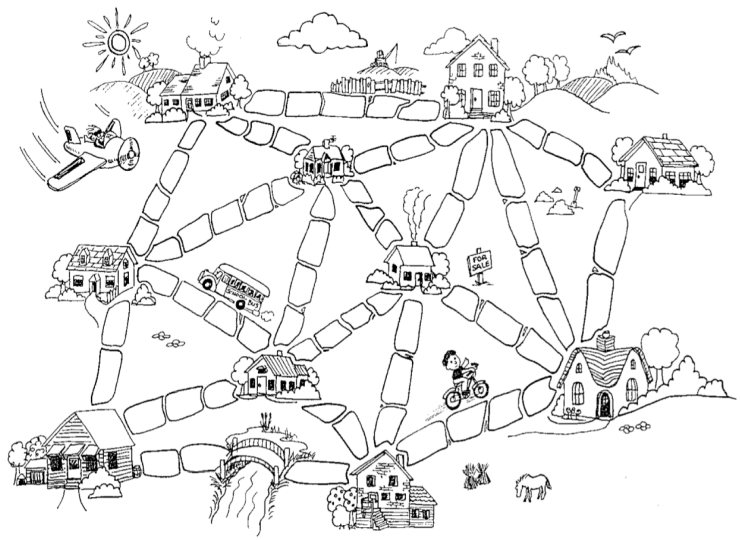


Muddy City

The Muddy City Activity from CS Unplugged

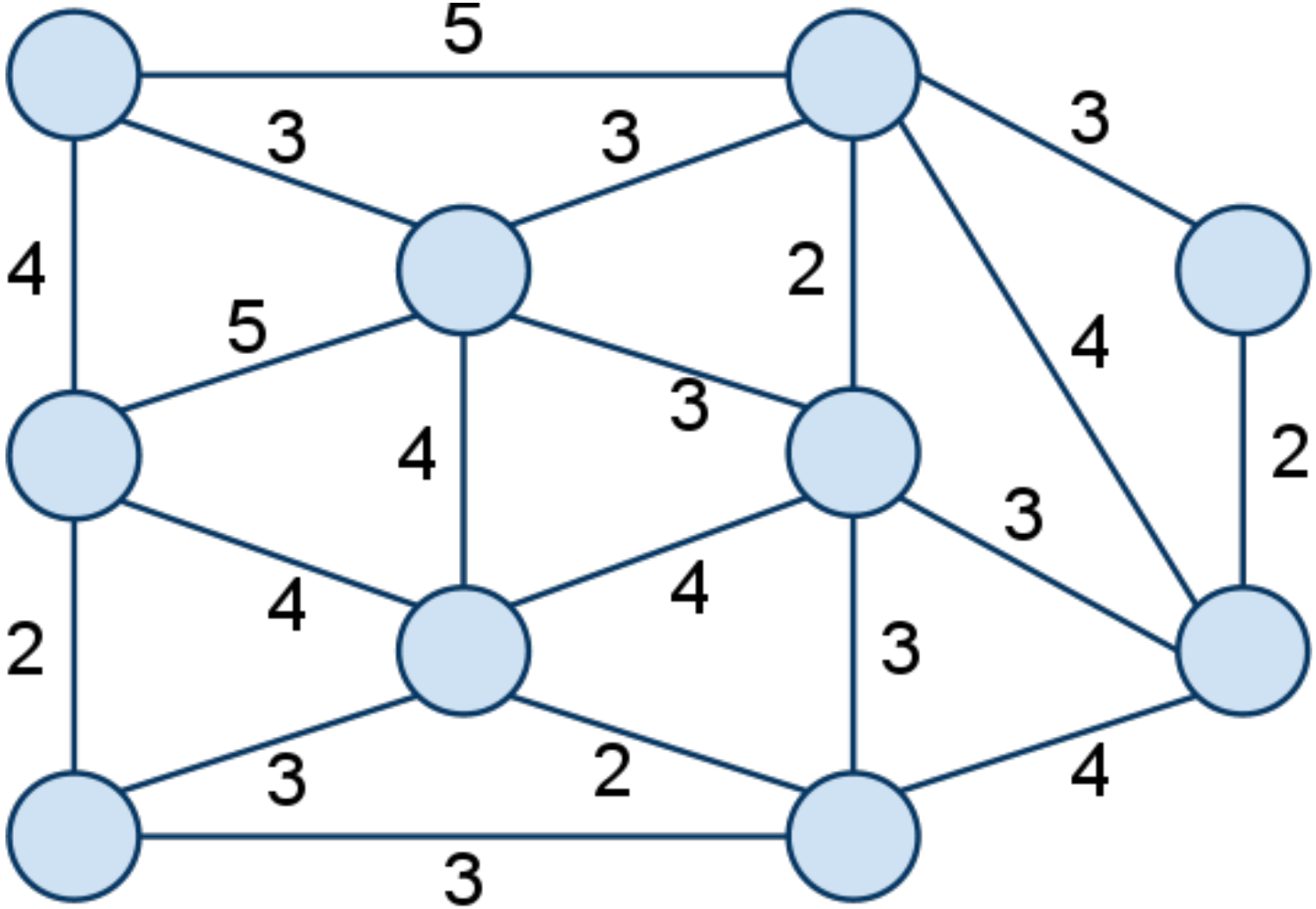
(1) Complete page one of the activity, then
(2) with your elbow partner, discuss the following points:

- What strategies did you use to solve the problem?
- Describe the algorithm (process) that you used.
- Would your strategy work for another city? (Why or why not?)
- For any city?



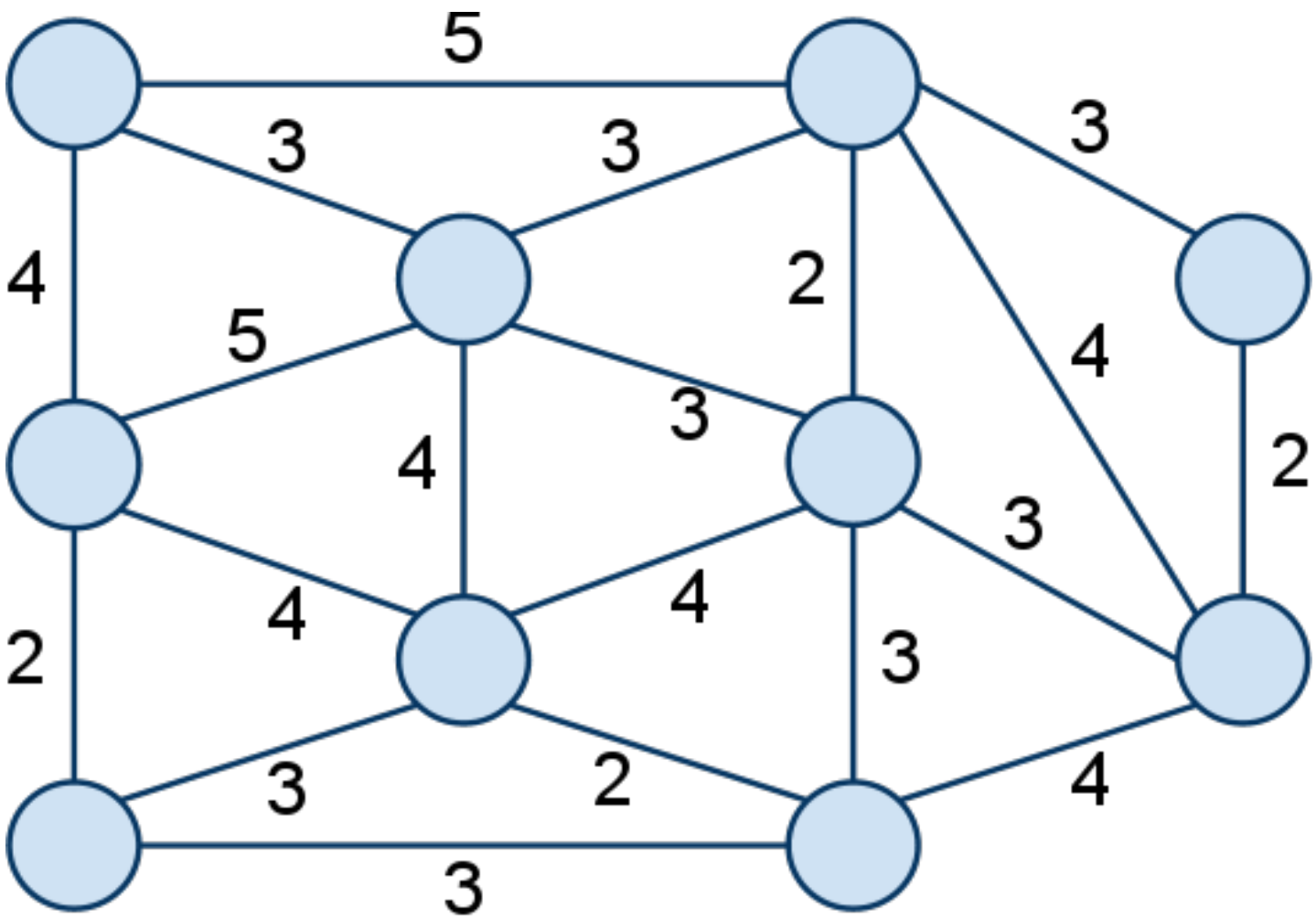
Graphs

As we saw, we can also represent the city as a **graph**, like this:



The dots are called **vertices** (singular- vertex) and the lines are called **edges**.

What is your solution?



on the board...

Graphs

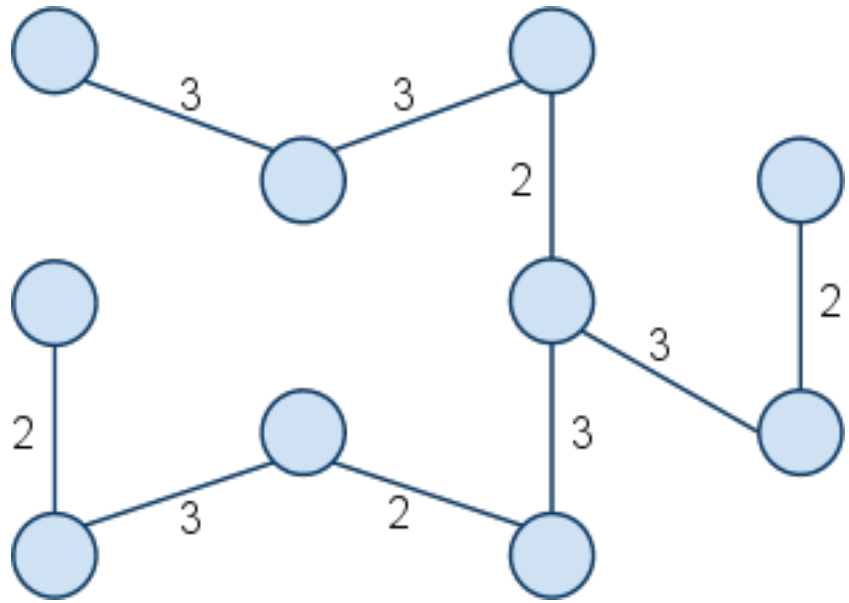
Since the edges have numbers associated with them, this is called a weighted graph.

The solution to the muddy city problem is called a minimum spanning tree.

It is **spanning** because it includes all of the vertices.

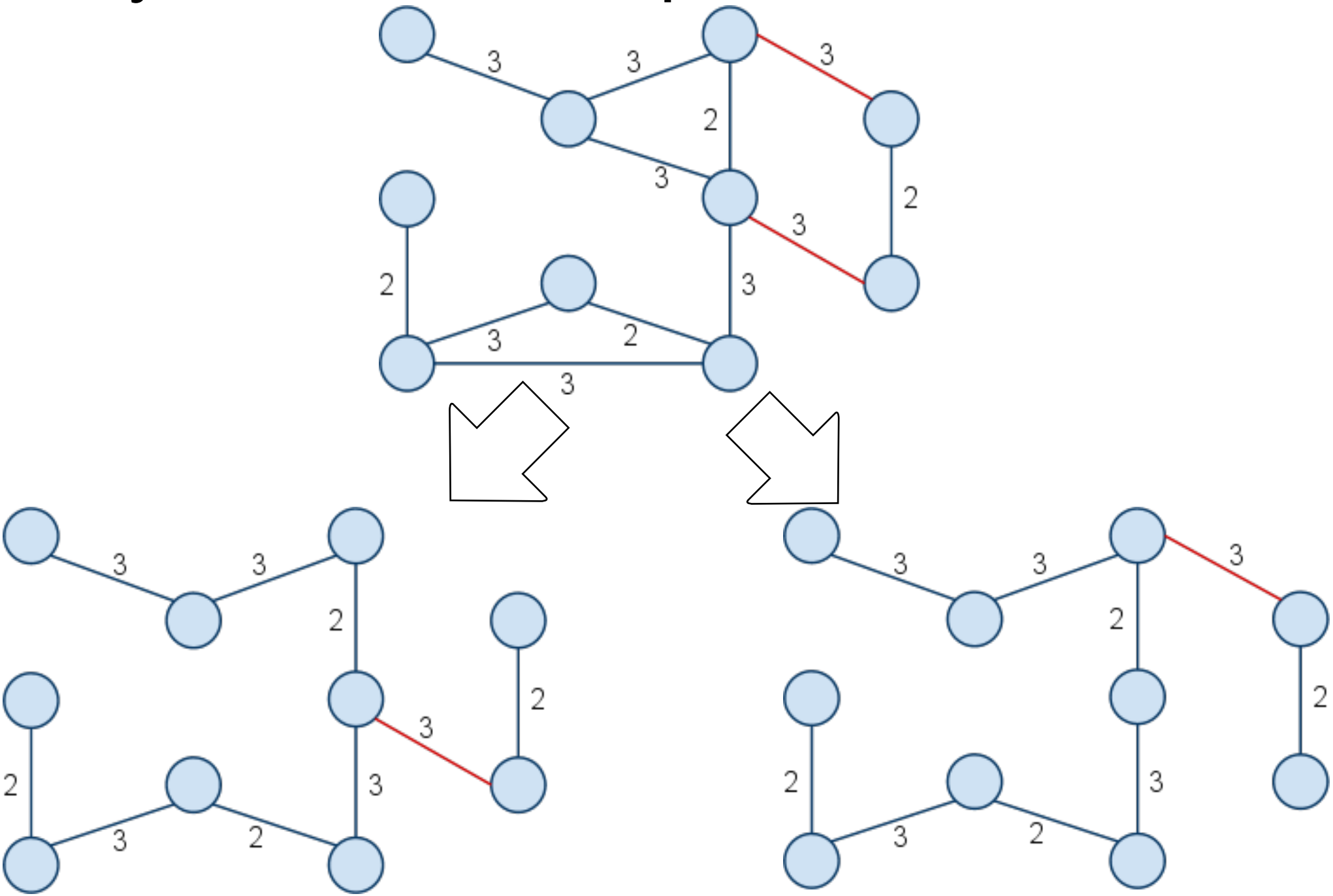
It is a **tree** because there is a unique path connecting every pair of vertices.

It is **minimum** because no other spanning tree "costs" less.



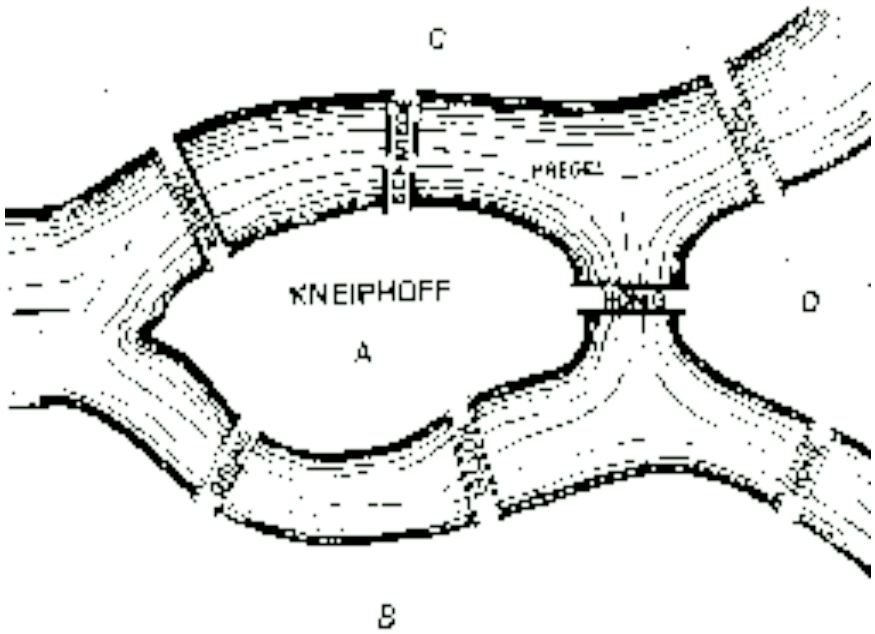
Is this minimum spanning tree unique?

Why are there multiple solutions?

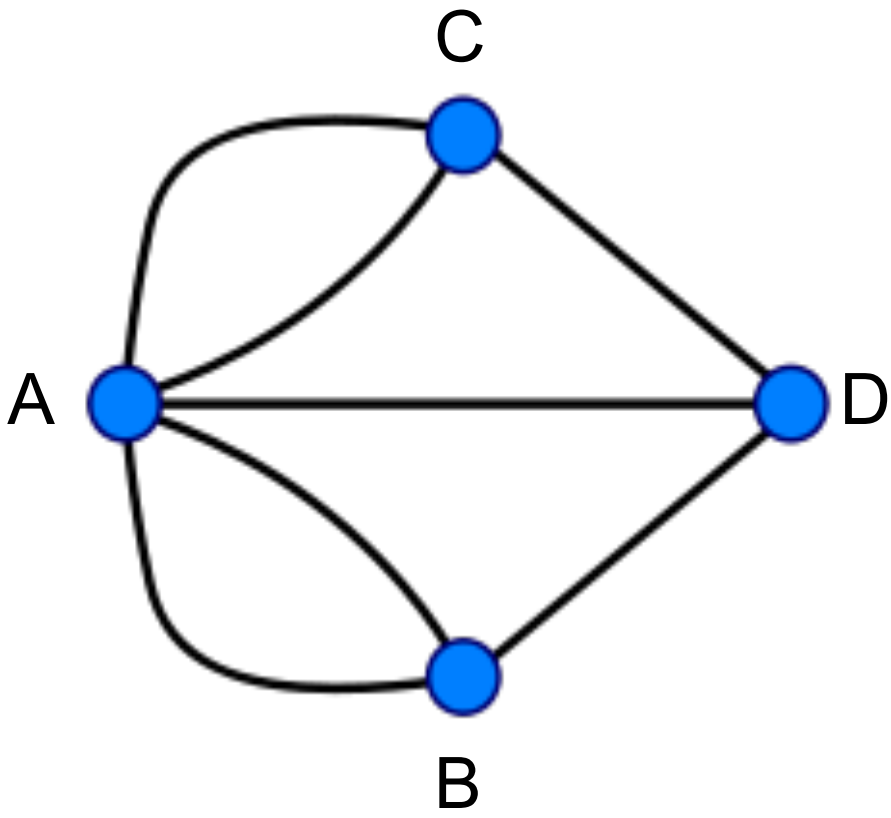


Other problems in graph theory

Euler's Bridges of Königsberg: Can you go for a walk and cross every bridge only once and end up where you started?



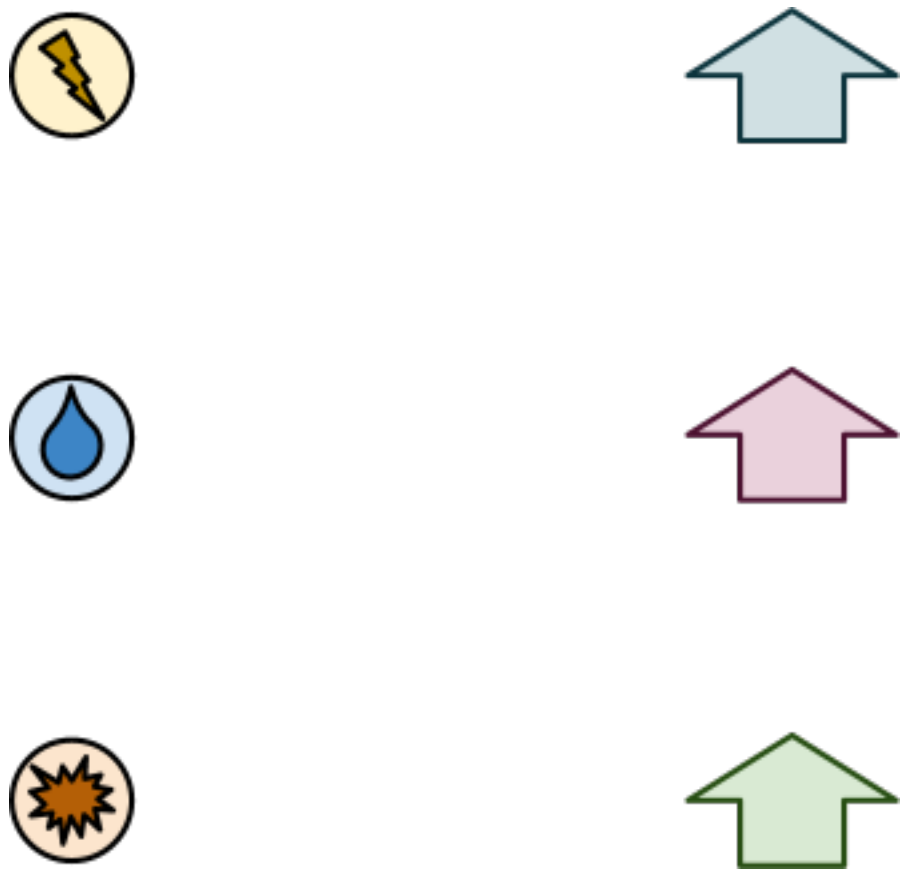
Map Representation



Graph Representation

Other problems in graph theory

The Utilities Problem: Can you connect all three utilities to each of the three houses without any lines crossing?



Final Unit 2 Project

- Students collect data about places they visit.
- Students find the best carpooling route: starting here (the school), visiting their locations, and returning.

See you tomorrow!

Unit 1: Human Computer Interaction
Unit 2: Problem Solving
Unit 3: Web Design
- Days 1-2: Web & Society
- Days 3-4: Basic HTML
- Day 5: HTML Formatting
- Days 6-7: Image editing for the web
- Days 8-10: Basic CSS
- Days 11-13: Style vs structure: HTML & CSS
- Day 14: Linking websites
- Days15-16: Page layout styles
- Days 17-19: Practice using design elements
- Days 20-21: Enhancements for web design
- Days 22-25: Final unit projects

Unit 4: Introduction to Programming
Unit 5: Computing Applications
Unit 6: Robotics