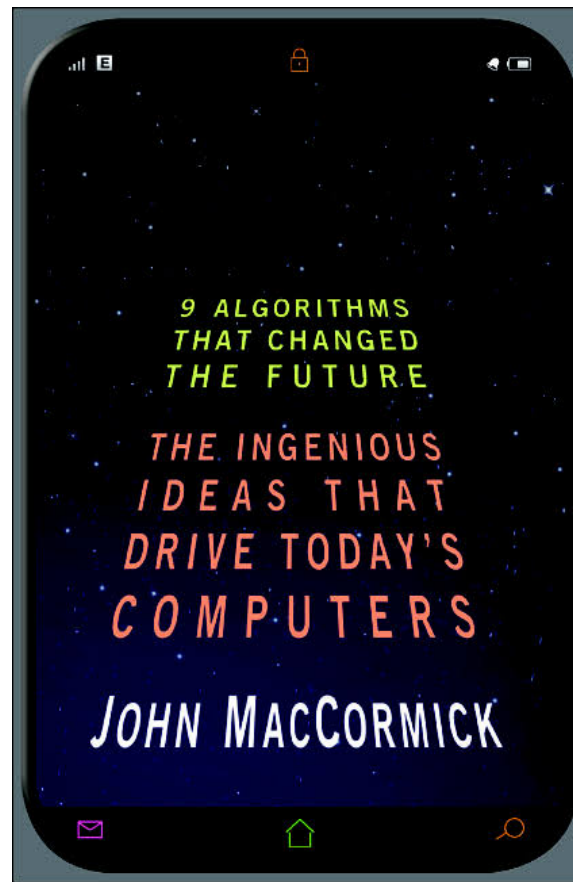

Introduction to Computational Thinking – ISC1057

Goals for this lecture:

- Go over the syllabus for the course
- Look at some characteristics of computational thinking
- See the difference between computer hardware and computer software
- Introduce the concept of algorithm
- Watch a video which demonstrates that it's not so easy to describe even a simple algorithm precisely
- Compare the work required for a “brute-force” algorithm and a “smarter” algorithm
- Get to know each other

ISC1057 - Computational Thinking Syllabus

Text: 9 Algorithms That Changed The Future: The Ingenious Ideas That Drive Today's Computers



Final Grade Determination. Your grade for the course is determined by class participation, homework and group projects. The distribution of grading for the course is:

- Class Participation - 5%
- Quizzes - 15%
- Homework - 50%
- Midterm Project - 15%
- Final Exam - 15%

The in-class quizzes will be taken using the app **Socrative** which you will access through your smart phone. Quizzes will be given almost daily and are typically given at the end of each lecture. No make-up quizzes are given. For the in-class quiz grade we average your top 20 quizzes so you are allowed to drop some scores. The class participation portion of your final grade is determined by the percent of ALL daily quizzes that you attempt. Your score on the quiz does not affect the class participation grade; essentially we are using the quiz for attendance.

Assigned homework must be done individually and submitted by the due date unless you are using your late homework days. **You will have a total of 7 late homework days throughout the semester** unless extenuating circumstances arise. Weekly homework assignments are typically available on Tuesdays and due on Thursday of the next week. Homework assignments consist of problems similar to the examples discussed in the lecture and assist you in understanding the material. You will be able to drop one homework assignment during the semester.

The midterm project is much more involved than homework assignments and serves to enhance or expand on the material covered in class. Consequently, students will be expected to work on the project over a longer span of time. For each project, several choices of topics will be suggested to accommodate the varied interests of students. Students may work alone or in pairs on the project.

This course is approved to satisfy FSU's Liberal Studies Quantitative/Logical Thinking requirement.

Liberal Studies Rule: In order to fulfill this requirement the student must earn a "C" or better in the course.

Some quotes about Computational Thinking

“Computational Thinking is the new literacy of the 21st century. The goal is for it to be a fundamental skill used by everyone in the world by the middle of the 21st century. Just like reading, writing and arithmetic.”

“Even people who aren’t going into computer science or engineering programs, should be exposed to computer science. Programming teaches you how to take problems, break them down into smaller problems, and solve them. Any kind of job that involves problem solving can benefit from computing. ”


“Computers are extremely talented at performing menial tasks quickly; Computational Thinkers can leverage that power to find solutions to previously impossible problems.”

“I think everyone should get a little exposure to computer science because it really forces you to think in a slightly different way, and it’s a skill that you can apply in life in general.”

What is computational thinking? Who needs it? Why?

☰ YouTube 🔍

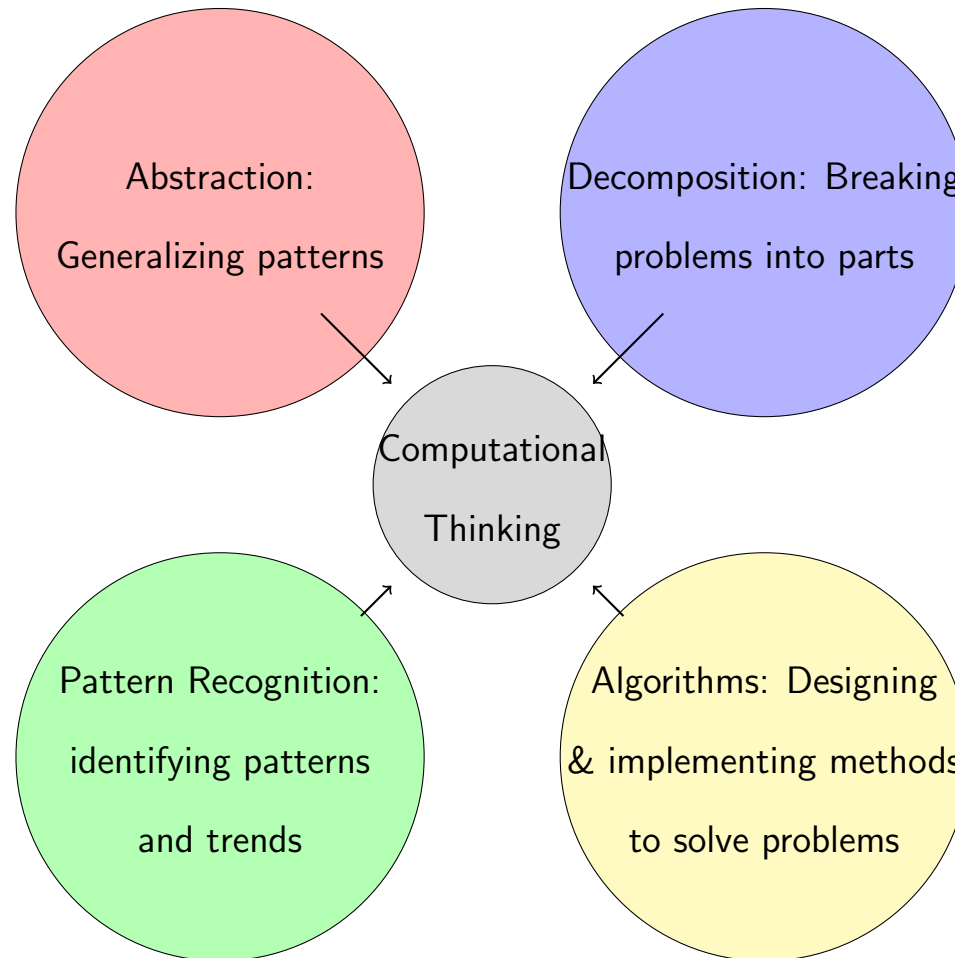
Computational Thinking
for Educators



What is
Computational Thinking?

▶ ⏪ 🔊 0:02 / 5:37 CC ⚙️ 📺 🗉

Computational Thinking is a problem-solving process that includes the following characteristics.

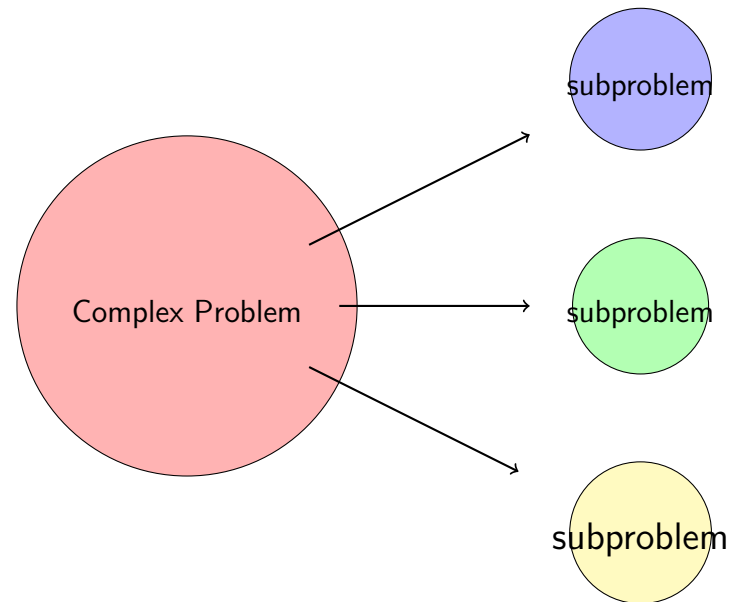


Decomposition: Breaking problems into simple parts



“If you can't solve a problem, then there is an easier problem you can solve: find it”

George Pólya





1. Make chocolate crust
2. Make mousse and chill
3. Wash & slice fresh strawberries
4. Whip cream
5. Make chocolate hearts
6. Assemble

Abstraction

- Abstraction is the process of filtering out unnecessary information.
- Abstraction allows us to create a general idea of what the problem is and how to solve it.

Example: Abstracting a general procedure for baking cookies



1. Bring butter and eggs to room temperature
2. Sift dry ingredients
3. Cream sugars and butter
4. Add eggs and vanilla
5. Mix in dry ingredients
6. Mix in add-ins like chips, nuts, etc.
7. Bake

Pattern Recognition

- The goal of pattern recognition is to find common similarities and differences among objects.
- This allows us to solve seemingly diverse problems by a single algorithm.

Example of pattern recognition - Putting together Ikea furniture



Designing algorithms

- Determine the right steps
- Determine the correct order of the steps
- Follow the steps completely
- Verify that the algorithm is working correctly
- Determine if algorithm is the “best” approach for solving problem

Example. Computer algorithm for multiplying a number by an integer using only addition

Assume that the computer only has the ability to do addition and we want to multiply something like

$$351.6 \times 14$$

Let a be a given number which we want to multiply by an integer b

product = 0

for $i = 1, b$

 product = product + a

end

This is called **pseudo code** for our algorithm because it does not use language specific terms.

Computer Hardware vs Computer Software

Computer **hardware** is any **physical device** used in or with your computer. This includes your monitor, mouse, keyboard, hard drive, sound card, video card, etc.

Computer **software** is a general term used to describe a collection of computer programs that perform some task on a computer system. Computer software is generally divided into two classes – system software and application software. Operating systems, utilities, device drivers, programming languages, interpreters are examples of system software. Application software are a group of computer software that are created to help the user complete tasks such as creating documents, performing calculations, storing and managing data, playing videos, etc. This includes programs like Microsoft Word, Adobe Acrobat, etc.

Computer hardware operates under the control of software.

We are interested in the [algorithms](#) that are contained within the software.



"WHICH CAME FIRST:
THE HARDWARE OR THE SOFTWARE?"

What is an algorithm?

“An algorithm is like a recipe, it must be followed exactly, it must be unambiguous, and it must have an end.”

An algorithm is a precise sequence of steps to solve a problem.

An algorithm should generate a solution. The solution may NOT be the “best” solution to the problem. The algorithm may NOT be the most efficient way to solve the problem.

An issue that arises in designing an algorithm is that we have to describe the algorithm in an orderly and unambiguous way. This is not as easy as it may seem.

As an example, consider the following video from an introductory computer science course at Harvard where the class is attempting to give

three volunteers an algorithm for making a peanut butter and jelly sandwich. Note that the center participant has a different type of jelly container than the other two. This is to illustrate that we often have to take care of special cases when designing an algorithm.



No Man's Sky: A Game Crafted by Algorithms

The virtual universe in this program is generated by an algorithm, not by an artist's pen. The generated universe is, of course, not infinite but "if you were to visit one virtual planet every second then our own sun will have died before you have seen them all."

The most common example of this type of game is Minecraft which creates a unique world for each of its players, randomly arranging rocks and lakes from a limited palette of bricks whenever someone begins a new game. But No Man's Sky is far more complex and sophisticated because the algorithm contains rules for the physics, biology, etc. of each virtually generated planet. The tens of millions of planets that comprise the universe are all unique. Each is generated when a player discovers it, and is subject to the laws of its respective solar systems

and vulnerable to natural erosion. The multitude of creatures that inhabit the universe dynamically breed and genetically mutate as time progresses.



A simple, but outdated, example to compare Brute-Force and Divide and Conquer algorithms

Consider the problem of looking up a name in the phone book. (I know that you have probably never had to do this in your life!)



If you are looking up, e.g., [Mike Shephard](#) in the phone book then the following would be an algorithm for finding the phone number.

Step 1 Open the phone book to the first page of names, i.e., the ones beginning with “A”;

Step 2 Check to see if the name **Mike Shephard** is on the page;

Step 3 **If** the desired name is not on the page **then** go to the next page and go back to Step 2;

If the desired name is on the page **then** write down the phone number and stop because you are done.

If you implement this algorithm will you always find the desired phone number? (Assuming Mike Shephard is listed in the book.)

The way we have written this algorithm is called **pseudo-code** because it describes the necessary steps in an ordered and unambiguous way without the use of any programming language.

Notice that we have a **conditional**, i.e., an **if-then statement** which

allows us to determine if we are done or if we have to go to the next page of the phone book.

Note that we also have a **repeated loop**, i.e., we keep doing steps 2 & 3 until we are done.

Now let's look at how much work it takes to find the name **Mike Shepard** using this algorithm.

- Assume that names beginning with "Sh" appear 81% into the book. This means that if the phone book has 100 pages then "Shepard" will appear on page 81.
- If the phone book has 200 pages then "Shepard" will appear on page $162 = .81 \times 200$ so we have to look at twice as many pages as when the phone book had 100 pages.
- If the phone book has 500 pages then "Shepard" will appear on page $405 = .81 \times 500$ so we have to look at five times as many pages as when the phone book had 100 pages.
- For obvious reasons, this is called a **Brute-Force algorithm** or the **Exhaustive search algorithm**.
- If the phone book doubles in size the next year to 1000 pages and

“Sh” still appears 81% into the book, we have to look at 810 pages before we find the phone number. If the book doubles again to 2000 pages then we have to look at 1,620 pages.

- This means that our work (i.e., the number of pages we must scrutinize) doubles as the size of the book doubles. This means that if we represent the work in a graph, we will have a straight line. For example, if x is the number of pages in the phone book and y represents the number of pages we have to look at then we write the line which passes through any of the two points, say $(500, 405)$ and $(1000, 810)$ so we have

$$y - 810 = \frac{810 - 405}{1000 - 500}(x - 1000) \implies y = 810 + \frac{405}{500}(x - 1000)$$

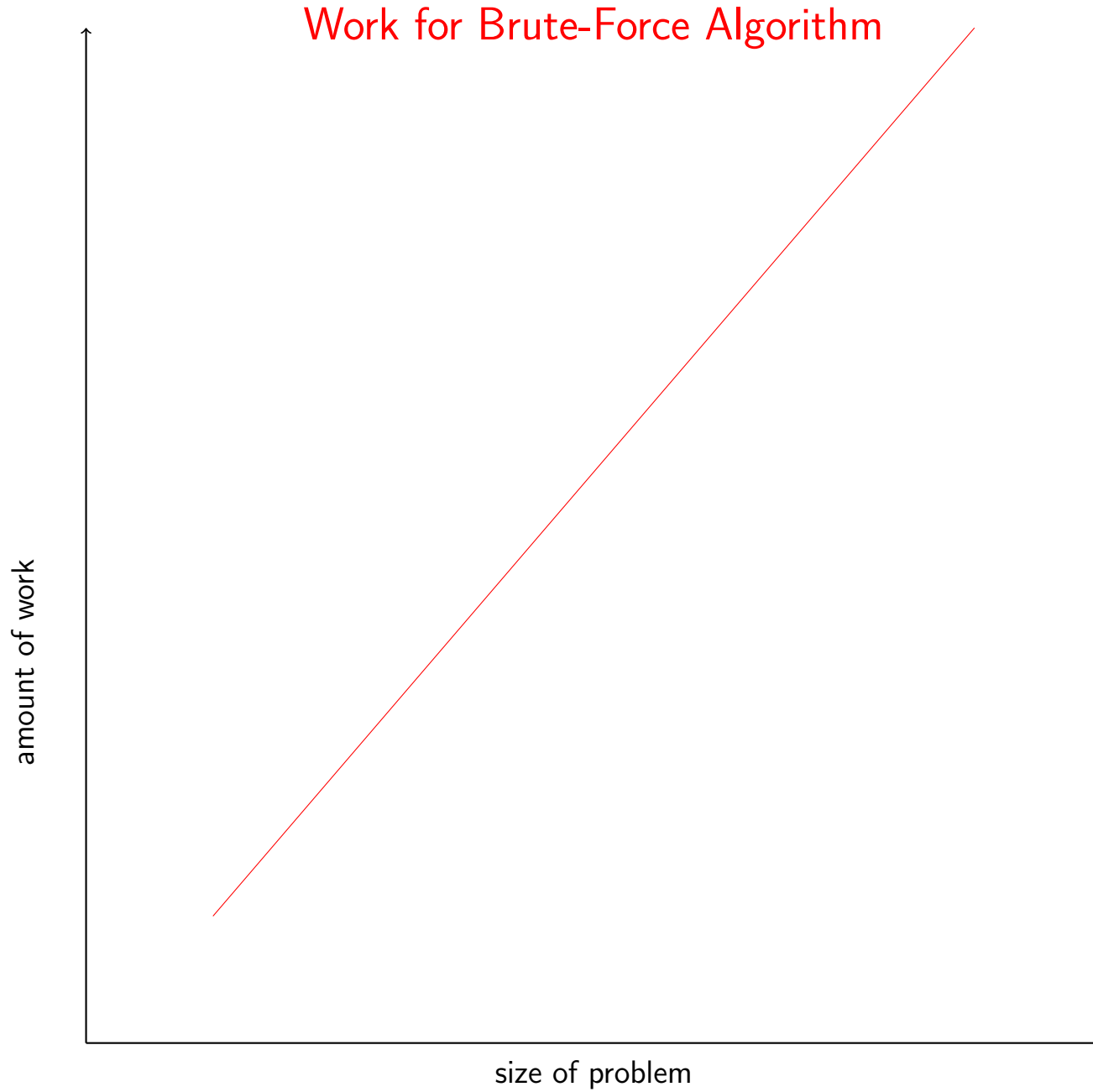
or

$$y = 810 + .81(x - 1000)$$

If there are only 100 pages in the book then $y = 810 + .81(100 - 1000) = 810 - .81(900) = 810 - 729 = 81$ which agrees with what we said the amount of work was for a 100 page phone book .

If we are looking for the phone number for [Karen Allen](#) then it will take less work using this algorithm but if we are looking for [Eric Yeager](#) then it will take more work.

In general, the amount of work to find a phone number is proportional to the number of pages N in the phone book which means it is a constant times N .



What can we do to reduce the amount of work?

Consider the following algorithm which intuitively seems to cut the work in half.

Step 1 Open the phone book to the first page of names, i.e., the ones beginning with “A”;

Step 2 Check to see if the name **Mike Shephard** is on the page;

Step 3 If the desired name is not on the page then skip next page, go to next page and then go back to Step 2;

If the desired name is on the page, write down the phone number and stop because you are done.

Does it always work?

Divide and Conquer Approach



Step 1 Open the phone book to the middle;

Step 2 Check to see if the name **Mike Shephard** is on the page;

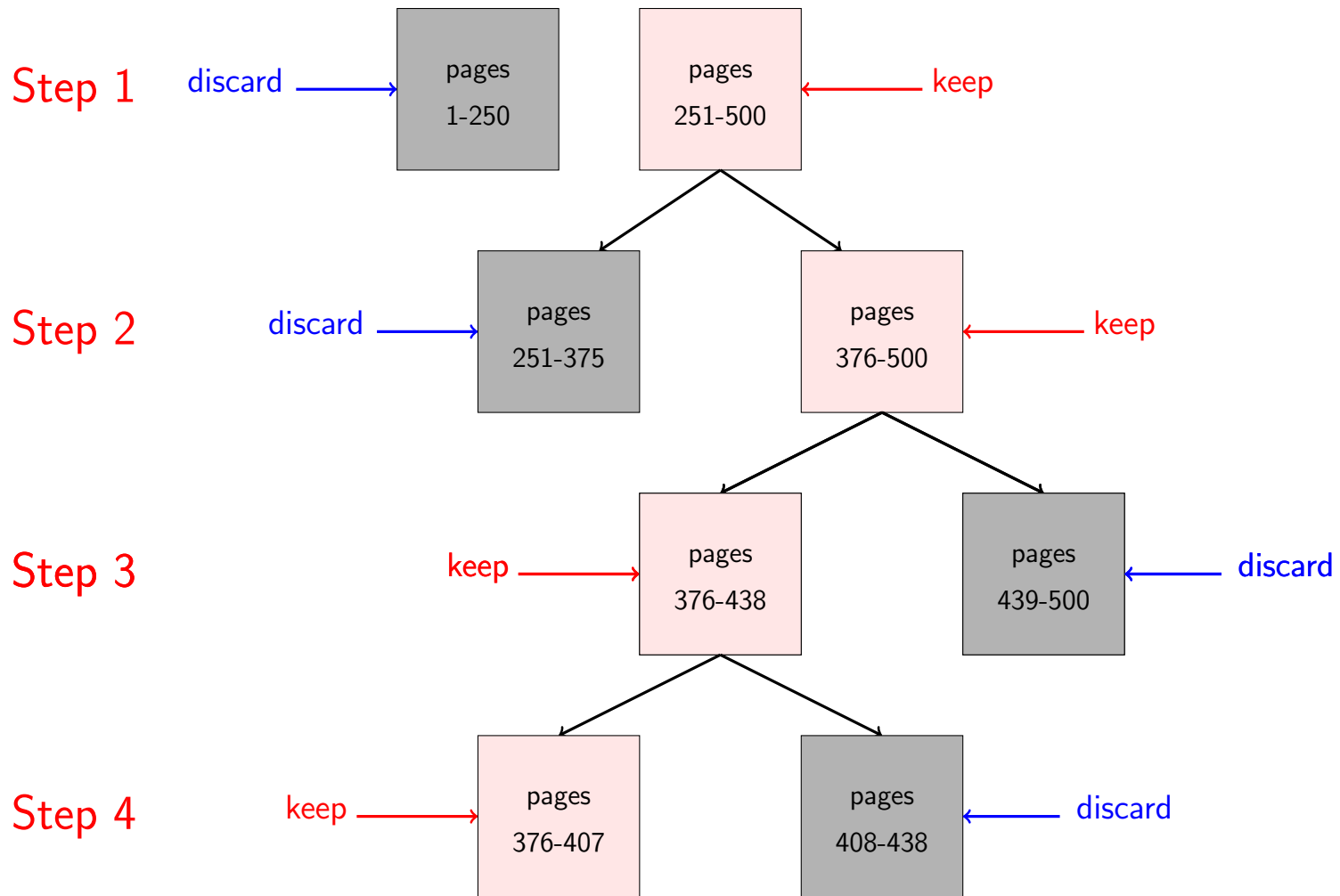
Step 3 If the desired name is not on the page then

- if the names on the page begin with a letter combination before “Sh” then throw the first half of the book away and go back to Step 1;

- if the names on the page begin with a letter combination after “Sh” then throw the second half of the book away and go back to Step 1;
- if the desired name is on the page, write down the phone number and stop because you are done.

How much work does this algorithm take?

Assume again that **Shepard** appears 81% of the way into a 500 page book, i.e., **Shepard** appears on page 405.



Continuing in this manner, we narrow the search to the following number of pages

Step 5 376-391 392-407

Step 6 392-399 400-407

Step 7 400-403 404-407

Step 8 404-405 406-407

Step 9 404 405

Now what happens if the phone book doubles in size to 1000 pages and the phone number is on page 810? Remember before that the work

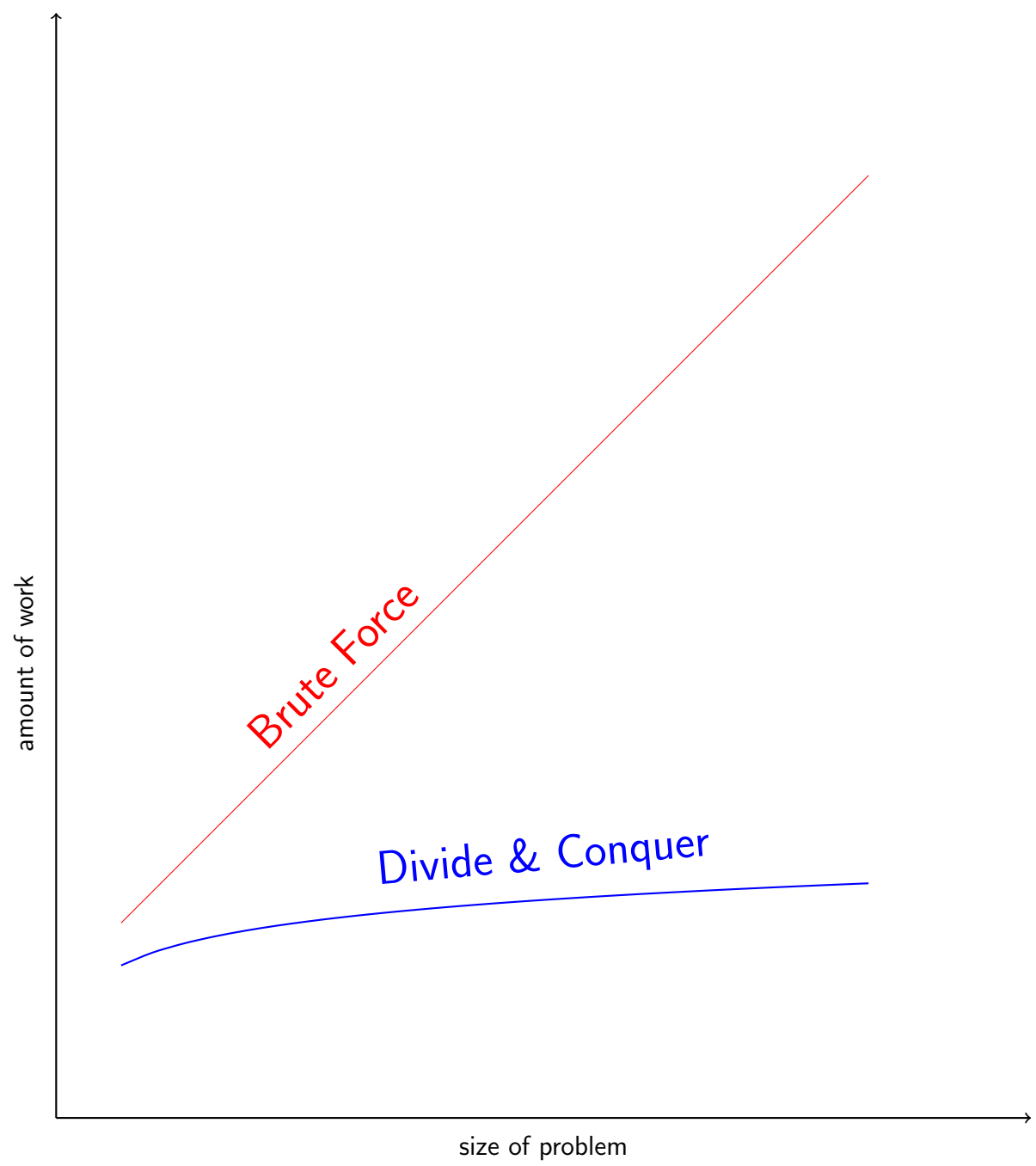
doubled with the brute-force approach.

Step 1	1-500	501-1000
Step 2	501-750	751-1000
Step 3	751-876	877-1000
Step 4	751-813	814-876
Step 5	751- 782	783-813
Step 6	783- 798	799-813
Step 7	799-806	807-813
Step 8	807-810	811-813
Step 9	807- 808	809-810

Step #	# number of pages to search		
	500 pages	1000 pages	2000 pages
1	250	500	1000
2	125	250	500
3	62	125	250
4	31	62	125
5	15	31	62
6	7	15	31
7	3	7	15
8	1	3	7
9		1	3
10			1

It turns out that the work required depends upon $\log N$ where N is the size of the phone book. Now it's not important whether you remember what a logarithm is, but it is important that you understand the

following plot where we plot the amount of work as the phone book size increases for both the **Brute-Force** approach and the **Divide and Conquer** approach. For any $N > 1$ $\log N$ is always smaller than N as can be seen from the plot.



Computer Algorithms are Changing the World

- Information at our fingertips through online searches.
- Buying products securely online.
- Finding “significant others” through online matching services.
- Immediate news coverage around the world
- On demand entertainment.
- Virtual currency

How do you think algorithms & computers will change the the way we live our lives in the next decades?

Reading Assignment:

1. *Computational Thinking*, Jeannette Wing¹, 2006
2. *The Algorithm: Idiom of Modern Science*, Bernard Chazelle, introduction and section “Thinking algorithmically”

Both articles are posted on our Blackboard site.

Next time we will have a Socrative quiz so be sure and **bring your phones!**

Let's adjourn for some cookies and get to know each other!

¹Former Chairman of Computer Science at Carnegie Mellon University and currently Corporate Vice President of Microsoft Research

Introduction to Computational Thinking - Continued

Goals for this lecture:

1. To look at different ways to design algorithms through specific examples
2. To see how to use the free online plotting program PLOTLY.
3. To understand that two key components of many algorithms are (i) repeated loops (recursion) and (ii) conditionals (if - then statements)
4. Introduction to Socratic through quizzes
5. Overview of topics we plan to cover this semester

Last time we said that different algorithms for the same problem may give different answers. The following example illustrates this.

Example. Suppose you are packing your backpack for a hike but you have 6 items you are interested in taking but their total weight is 96 lbs. The maximum weight you are willing to carry is 50 lbs. You assign each item an importance value between 0 and 25; the higher the number the more valuable. What items should you take?



Item	A	B	C	D	E	F
weight	30	28	15	10	9	5
value	16	25	24	15	12	2

For the first algorithm we simply pick an item at random and put in the backpack. We then weigh the backpack and if it weighs < 50 lbs then we pick another item. If it weighs more than 50 lbs then we remove the last item and try another item at random.

Algorithm 1 - Random

Step 0 Put all items in a selection pile.

Step 1 If there are any items in the selection pile, pick an item at random from the selection pile and put into the backpack. If there are no items left in selection pile you are done.

Step 2 Weigh the backpack.

If it weighs < 50 lbs, go to Step 1.

If it weights 50 lbs, you are done.

If it weighs > 50 lbs, remove item from backpack and from selection pile. Return to Step 1.

The strategy behind this algorithm is **randomness**. It can generate any possible solution if we perform it many times.

Items	Weight	Value
A, C, F	$30 + 15 + 5 = 50$ lbs	$16 + 24 + 2 = 42$
A, D, E	$30 + 10 + 9 = 49$ lbs	$16 + 15 + 12 = 43$
A, D, F	$30 + 10 + 5 = 45$ lbs	$16 + 15 + 2 = 33$
B, D, E	$28 + 10 + 9 = 47$ lbs	$25 + 15 + 12 = 52$
B, D, F	$28 + 10 + 5 = 43$ lbs	$25 + 15 + 2 = 42$
B, C, F	$28 + 15 + 5 = 48$ lbs	$25 + 24 + 2 = 51$
C, D, E, F	$15 + 10 + 9 + 5 = 39$ lbs	$24 + 15 + 12 + 2 = 53$

The strategy behind the next algorithm is to pick the items in their order from most valuable to least valuable.

Algorithm II - Arrange items from most valuable to least valuable and add items in order if they fit

Step 0 Arrange the items in a selection pile in order of value from largest to smallest

Step 1 Pick the next item in order from the selection pile and put into the backpack. If there are no items left in selection pile you are done.

Step 2 Weigh the backpack.

If it weighs < 50 lbs, go to Step 1.

If it weights 50 lbs, you are done.

If it weighs > 50 lbs, remove item from backpack and the selection pile; return to Step 1.

What result does this strategy give us?

Step 0 We order our objects as B, C, A, D, E, F

Step 1 Put item B in backpack

Step 2 Backpack weighs 28 lbs < 50 lbs so go to Step 1

Step 1 Pick item C from pile and put in backpack

Step 2 Backpack weighs 43 lbs < 50 lbs so go to Step 1

Step 1 Pick item A from pile and put in backpack

Step 2 Backpack weighs 73 lbs > 50 so remove item A from backpack and from selection pile. Go to Step 1

Step 1 Pick item D from pile and put in backpack

Step 2 Backpack weighs 53 lbs > 50 so remove item D from backpack and from selection pile. Go to Step 1

Step 1 Pick item E from pile and put in backpack

Step 2 Backpack weights 55 lbs $>$ 50 so remove item E from backpack and from selection pile. Go to Step 1

Step 1 Pick item F from pile and put in backpack

Step 2 Backpack weights 48 lbs $<$ 50 so go to Step 1

Step 1 No items left in selection pile so we are done.

Answer: Items B, C, and F which weigh a total of 48 lbs (28+15+5) and have a value of $25+24+2=51$

The strategy behind this algorithm is to order the items based upon their value per weight so that if you have two items that weigh 10 lbs and one has a value of 15 and the other of 5, then clearly the item of value 15 is a better choice for the given weight.

Algorithm III- Compute the ratio of **value per unit size** and add items in order if they fit

Step 0 Compute the ratio of value/weight for each item. Arrange the items in a selection pile in order of value from largest to smallest ratios.

Step 1 Pick the next item in order from the selection pile and put into the backpack. If there are no items left in selection pile you are done.

Step 2 Weigh the backpack.

If it weighs < 50 lbs, go to Step 1.

If it weights 50 lbs, you are done.

If it weighs > 50 lbs, remove item from backpack and selection pile and return to Step 1.

What result does this strategy give us?

Step 0 We compute the **value per unit size of each item** as

$$\begin{array}{lll} \text{A: } \frac{16}{30} = \frac{8}{15} & \text{B: } \frac{25}{28} & \text{C: } \frac{24}{15} = 1\frac{9}{15} \\ \text{D: } \frac{15}{10} = 1\frac{1}{2} & \text{E: } \frac{12}{9} = 1\frac{1}{3} & \text{F: } \frac{2}{5} \end{array}$$

We order our objects as **C, D, E, B, A, F**

Step 1 Put item C in backpack

Step 2 Backpack weighs 15 lbs so go to Step 1

Step 1 Pick item D from pile and put in backpack

Step 2 Backpack weights 25 lbs so go to Step 1

Step 1 Pick item E from pile and put in backpack

Step 2 Backpack weights 34 lbs so go to Step 1

Step 1 Pick item B from pile and put in backpack

Step 2 Backpack weights 52 lbs so remove item B from pile and go to Step 1

Step 1 Pick item A from pile and put in backpack

Step 2 Backpack weights 64 lbs so remove item A from pile and go to Step 1

Step 1 Pick item F from pile and put in backpack

Step 2 Backpack weights 39 lbs

Answer: Items C, D, E and F which weigh a total of 39 lbs and have a value of $24+15+12+2 = 53$

Which algorithm gives the best answer?

Which algorithm may give the best answer or it may give the worst answer?

Last time we saw an example of a brute-force algorithm and a Divide & Conquer algorithm for finding a phone number in a telephone book.

As another example, consider an algorithm for counting; specifically we will determine the number of people in a room.



The brute-force approach is for me to point at each person and count 1,2,3,4,... This approach requires me to go through all the numbers from 1 to N , where N is the number of people in the room.

A second, but slightly more efficient, brute-force approach is to count individuals by two, i.e., 2, 4, 6, 8, This approach takes half of the work of counting each person individually.

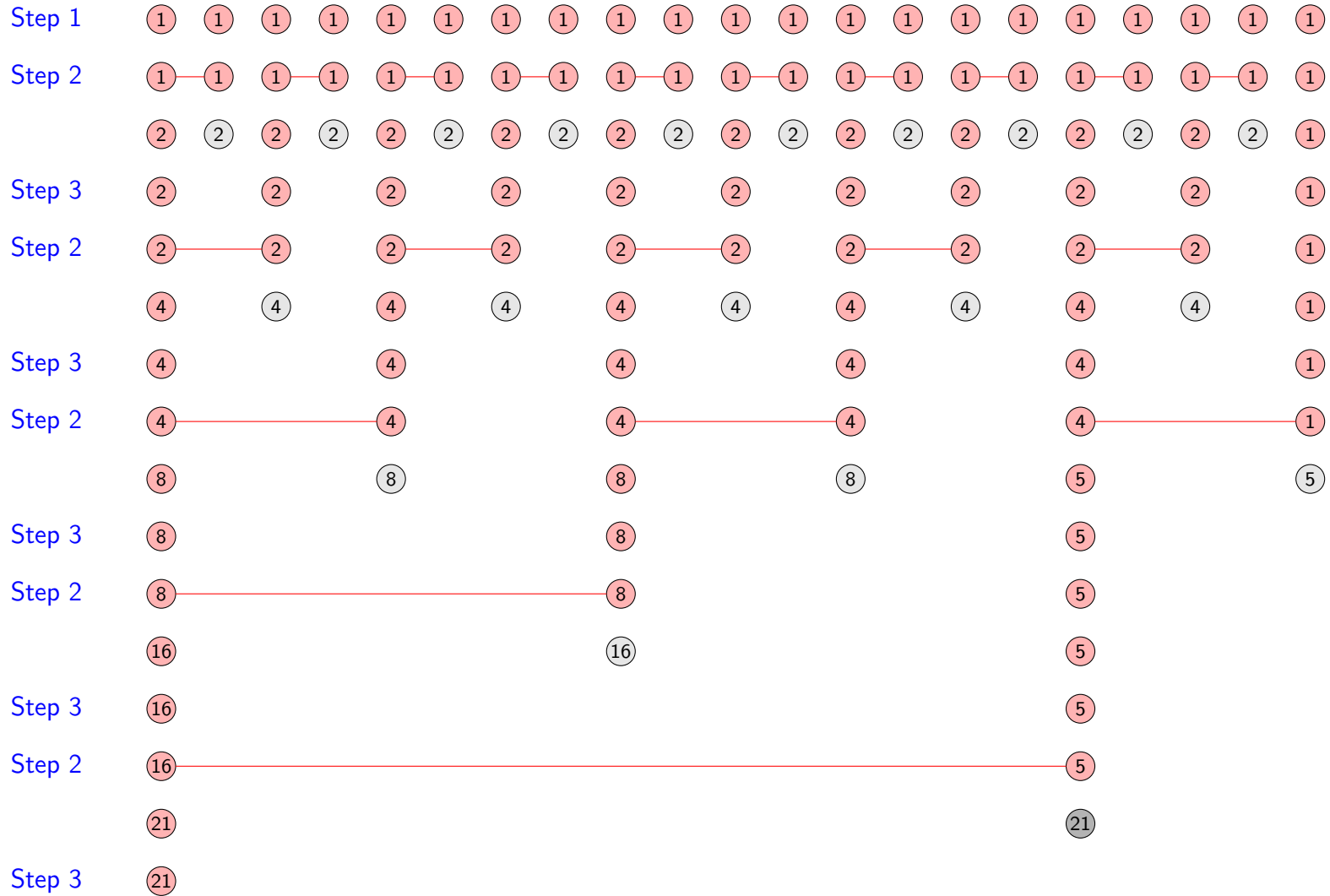
A smarter approach is the following algorithm.

- Step 1** Each person in the room stands up and is assigned the number 1.
- Step 2** Each standing person must pair up with another person standing in the room and add your respective numbers together.
- Step 3** One of the two of you must sit down.
- Step 4** If more than one person is standing, go to **Step 2** and repeat. If only one person is standing then their number is the total number of people in the room.

Again, our algorithm has a conditional (an **IF-THEN statement**) and a repeated loop.

Let's try the algorithm to count the number of people (excluding instructor) in the class.

This algorithm does not require the room to have an even number of people in it as the following schematic illustrates with 21 people. Here it takes 5 loops (or recursive steps) to compute the algorithm.

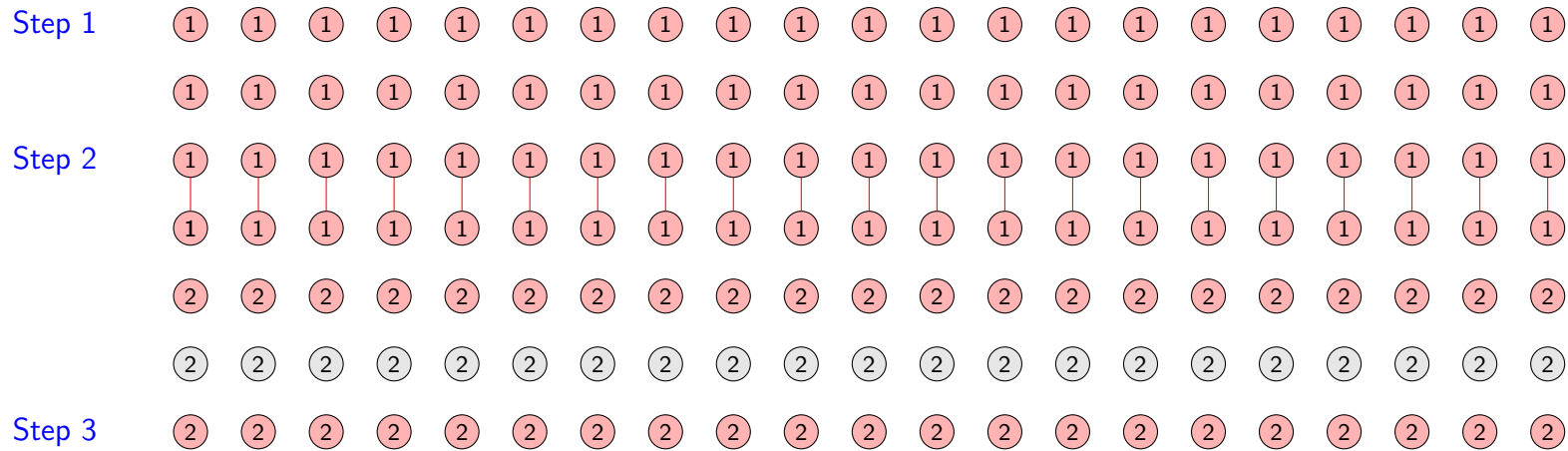


- At the end of the first recursive step (Step 3) we have 11 people

standing.

- At the end of the second recursive step (Step 3) we have 6 people standing.
- At the end of the third recursive step (Step 3) we have 3 people standing.
- At the end of the fourth recursive step (Step 3) we have 2 people standing.
- At the end of the fifth recursive step (Step 3) we have 1 person standing so we are done.

If we double the number of people in the room how many more steps does it take?



At the end of the first recursive loop there are 21 people standing which is the same number as in the previous example which took 5 recursive steps so when we start with 42 people it takes one more step or 6 total.

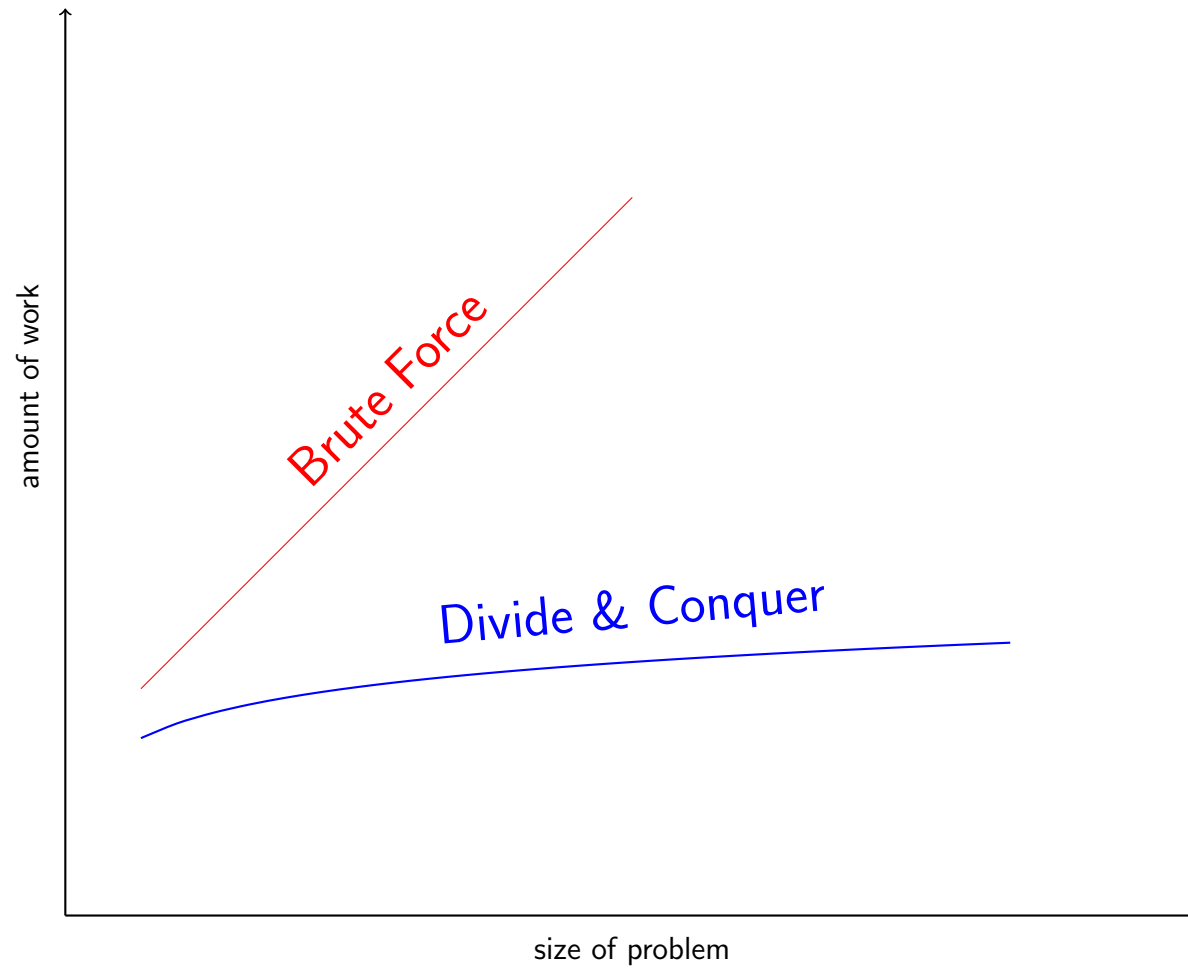
The table below compares the number of people remaining after Step 3 is completed (when one member of each pair sits down) for different

size classrooms. Note that as in the Divide & Conquer algorithm the work increases by one as the number of people is doubled.

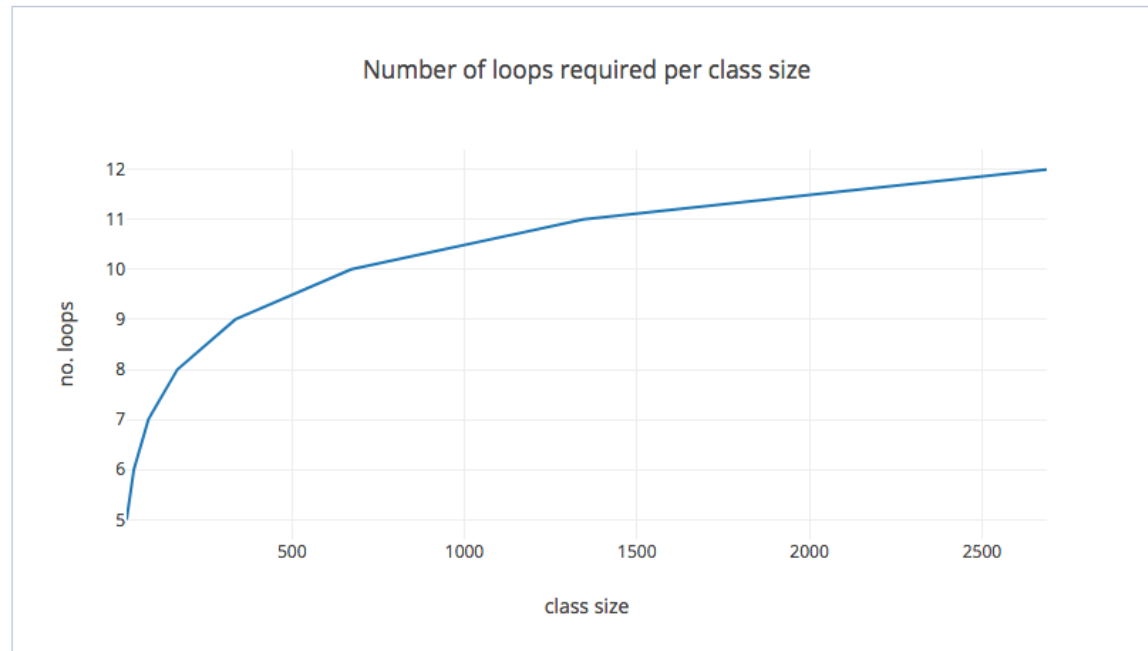
Loop #	# people standing after each loop			
	class: 21	class: 42	class: 84	class: 168
1	11	21	42	84
2	6	11	21	42
3	3	6	11	21
4	2	3	6	11
5	1	2	3	6
6		1	2	3
7			1	2
8				1

Based upon the trend we see from this table we can predict how many steps it will take for a classroom of size 336, 672, etc. and when we

plot the points we get a curve that looks like the log plot we had before.



Class Size	Number of loops required
21	5
42	6
84	7
168	8
336	9
672	10
1344	11
2688	12



Many people believe that computers are so powerful that they can solve any problem by a Brute-Force approach. This is not true! We need to think in different ways to design more efficient algorithms so results can be obtained in a reasonable amount of time.

In your homework, you are asked to make a similar plot using the computer (not by hand).

How can you do this? You can use software such as Microsoft or a free online program called **PLOTLY**.

I will show you how to create such a plot now using PLOTLY. Simply do a search for PLOTLY. You must create your own account but it's FREE. To turn in your plot you can either save it in the program or perform a ScreenShot of the plot.

Collapse All + Trace

Chart Type

Line plot

Try An Example

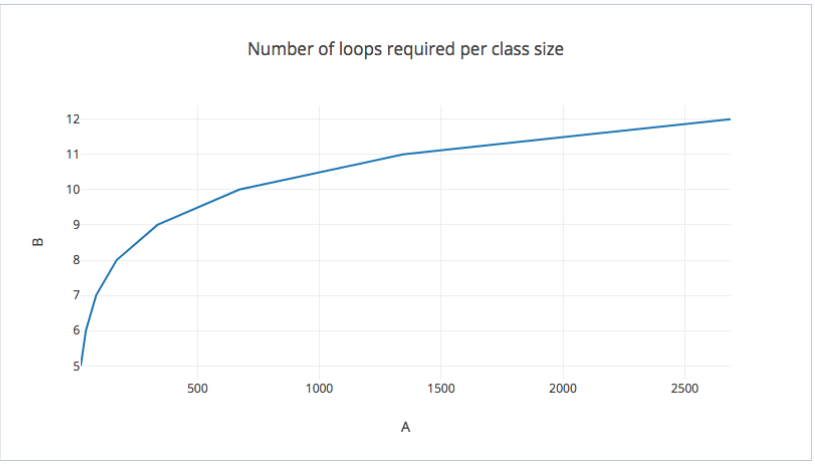
X A

Y B

Hover Text

Grid 1 + Grid

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
2	42	6													
3	84	7													
4	168	8													
5	336	9													
6	672	10													
7	1344	11													
8	2688	12													
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															



In these algorithms, as well as the ones from last time, we had a conditional statement, i.e., an **if-then statement**. Here is Bill Gates (Microsoft) explaining these.



YouTube video

www.youtube.com/watch?v=m2Ux2Pnje6E

In these algorithms, as well as the ones from last time, we had a **repeat loop**. Here is Mark Zuckerberg (Facebook) explaining the importance of these.



YouTube video

www.youtube.com/watch?v=mgooqyWMTxk

Socrative Practice Quiz

CTISC1057

Go to b.socrative.com and you will see a login page.

Under [Student Login](#) enter the Room Name **CTISC1057**.

The next page has you enter a name to identify yourself. Use **YOUR FSU LOGIN** (such as mc12s; do not include “@ myfsu.edu”)

The first question on the correct quiz should appear. Because the question is True/False you simply click the correct box.

1. The term “algorithm” is only used to refer to computer programs.
2. Your computer’s trackpad is an example of software.

3. An example of a brute-force algorithm for counting the number of people in a room is counting the individuals by twos.
4. If a phone book has 1000 pages then the Divide & Conquer algorithm for finding a number in this phone book takes twice as much work as for a phone book with 500 pages.
5. Jeannette Wing believes that computational thinking should only be taught in college.
6. All algorithms for a specific problem give the same answer.

If we want to implement an algorithm on a computer, how is this accomplished?

Suppose you have a recipe for hush puppies and you want to tell a person in China how to make this dish but they don't speak English. The only way to achieve this is to write



the recipe in English and have an interpreter translate it into Chinese. In this analogy the recipe is the algorithm, you are the programmer and the person in China plays the role of the computer. You might be able to write the recipe but the “computer” can't understand it!

This analogy is like the programmer/computer situation. The programmer uses a [programming language](#) such as C++, Java, Python, Fortran, etc.. to write the software. Unfortunately, the computer doesn't

understand these languages because all it understands is **machine language**. To complicate matters, programmers don't understand machine language.

To solve this problem we need the equivalent of an interpreter which is what the **compiler** does. It converts our computer program written in a programming language that we can understand into one that a computer understands, basically into binary which is a language that only consists of ones and zeros (we will talk about this later).

Why can't we write our computer programs in English? Our language is too imprecise for computers!

We will NOT learn a programming language in this course but rather concentrate on the ideas behind the algorithms which a programmer implements.

Software Bugs

Oftentimes one hears the term **bug** when referring to software.

A software bug is an error or flaw in a computer program that causes the result to be incorrect or to behave in unintended ways. These errors typically occur due to mistakes made by humans either in the actual program or in the algorithm design.

A study commissioned by the US Department of Commerce's National Institute of Standards and Technology in 2002 concluded that "software bugs, or errors, are so prevalent and so detrimental that they cost the US economy an estimated \$59 billion annually, or about 0.6 percent of the gross domestic product".

Where did the term "bug" come from?

Grace Hopper, a computer scientist and Navy Rear Admiral, joined the Harvard Faculty at the Computation Laboratory after WWII to continue her work on the Mark II computer. Operators traced an error in the Mark II to a moth trapped in a relay, coining the term bug. This bug was carefully removed and taped to the log book. Stemming from the first bug, today we call errors or glitches in a program a bug.



9/9

0800 Antan started
1000 " stopped - antan ✓
13⁰⁰ (033) MP-MC $\left. \begin{matrix} 1.2700 & 9.037847025 \\ 2.130476415 & 9.037846995 \end{matrix} \right\}$ cond
 (033) PRO 2 2.130476415
 cond 2.130676415

Relays 6-2 in 033 failed special speed test
in Relay
Relays changed

Relay
2145
Relay 3376

1100 Started Cosine Tape (Sine check)
1525 Started Multi-Adder Test.

1545



Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.
1630 antan started.
1700 closed down.

Typos in a program are usually caught by a **compiler**. Computer bugs often occur when the programmer makes a logic mistake or fails to take into account special cases (as in the PB&J video with the differences in the jam containers).

Computers Are Reshaping Our World

This class looks at ways in which computers have begun to change the way we live.

Many recent discoveries and inventions can be attributed to the power of computer hardware (the machines), the development of computer programs (the instructions we give the machines), and more importantly to the ability to think about problems in new ways that make it possible to solve them with computers, something we call **computational thinking**.

Issues in Computational Thinking

What problems can computers help us with? What is different about how a computer seeks a solution? Why are computer solutions often not quite the same as what a human would have found?

How do computer programmers write out instructions that tell a computer how to solve a problem, what problem to solve, and how to report the answer?

How does a computer “think”? How does it receive instructions, store information in a memory, represent and manipulate the numbers and words and pictures and ideas that we refer to as we think about a problem?

We will see that making a computer carry out a task in a few seconds actually can take years of thinking, research, and experiment. So what seem like simple tasks actually can hide a great deal of technical construction.

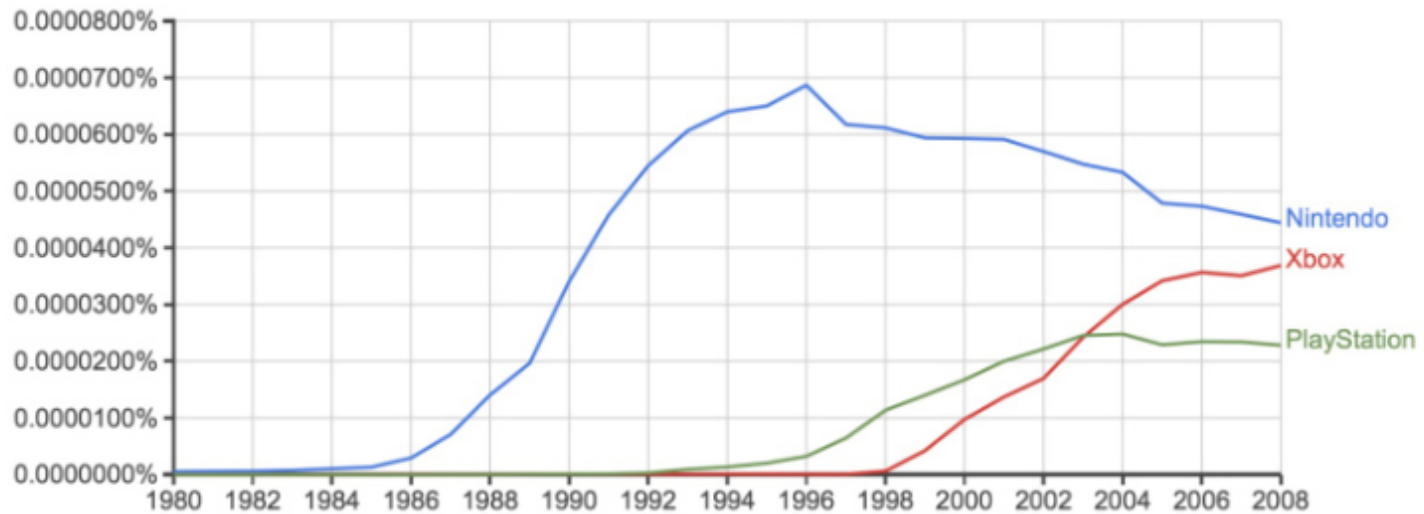
But the rewards can be enormous; depending on what you want, you can hope that a computational approach will give you an unbeatable chess player, a drug that protects against the Zika virus, or a car that safely drives itself to any destination you want.

Topics to be covered this semester

Google Books & Google Viewer

- One week
- In 2004, Google announced a plan to systematically scan a copy of every book ever published, estimated to number 130 million. The result will be a huge database called [Google Books](#).
- The idea was not to just copy all the publications but to make a searchable data base. To allow everyone easy access to this data base, Google developed a viewer which allows the user to compare the number of occurrences of words or phrases in print through the years.

- We will be looking at some of the history of digitizing text, the problems Google encountered with this project, the type of computer algorithms needed to convert text into a searchable document and how to glean information from the viewer.



Cryptography, Public Key Encryption & Digital Signatures

- 3.5 weeks
- Algorithm # 4 (Public Key Encryption) and Algorithm # 9 (Digital Signatures) in text
- We will begin by looking at cryptography from a historical perspective. We will see how it has been used to send secret messages as early as Julius Caesar (who has a cipher named for him) up through World War II. You will learn how to encrypt/decrypt a message given a key and how to decrypt certain types of ciphers without a key by looking at letter and word frequency plots.
- Cryptography became critical for everyone, not just military leaders, when computers became prevalent. Now we enter our credit card information in a website to make purchases with confidence that the information is secure. How does this work? Public Key Encryption is

the algorithm that makes this possible. We will strive to understand this algorithm by looking at simple examples.

- Another application of encryption is when you sign a document electronically. How can the individual receiving your signed document know that it was really you who signed it?

GHIHQG WJH HDVW ZDOO RI WKH FDVWOH

G	H	I	H	Q	G	W	J	H	H	D	V	W
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
D	E	F	E	N	D	T	H	E	E	A	S	T

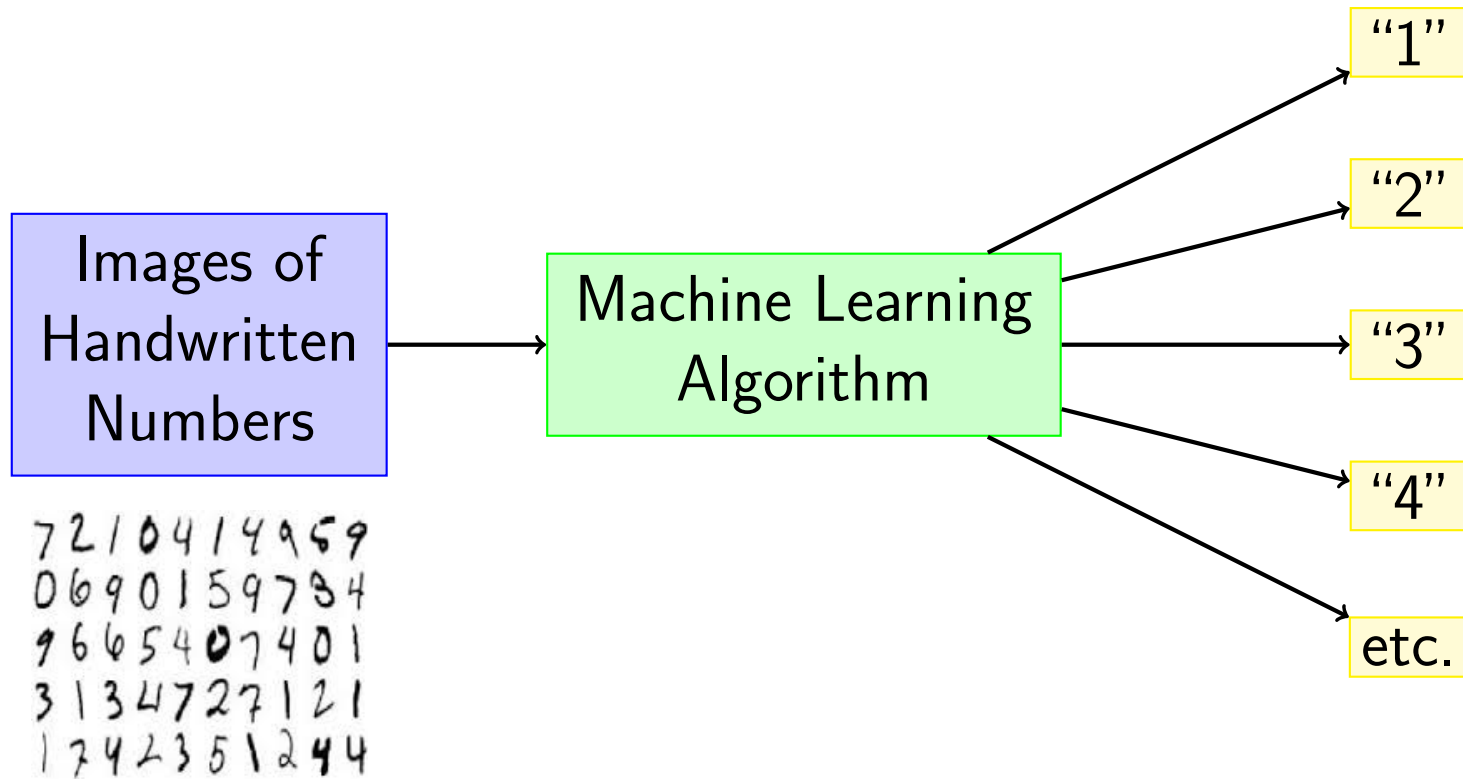
DEFEND THE EAST WALL OF THE CASTLE

Machine Learning: Linear Regression and Pattern Recognition

- 3 weeks
- Algorithm # 6 in text
- Since the advent of computers, people have considered whether computers can become the dominant form of intelligence on Earth; this has been a theme in many science fiction stories/movies through the decades. There is actually a test called the Turing test (1950) which is used to measure a machine's ability to exhibit intelligent behavior.
- Machine learning algorithms are used in many applications from reading postal codes to autonomous land vehicles. Unlike other algorithms we have considered, in machine learning we typically “train” the algorithm with a set of data and then, after it has “learned”, we input new data for it to identify. For example, when reading hand written postal codes we could train the algorithm with a set of

handwritten numbers and then give it a new code to identify.

- Recognizing patterns is something that humans do better than computers. Typically a human can identify an image of a person after having seen another picture of the person but this ability is difficult to incorporate into an algorithm.
- We will look at various types of Machine learning algorithms and see the types of applications they are best suited for. An underlying theme is there is not one algorithm which works for all problems.

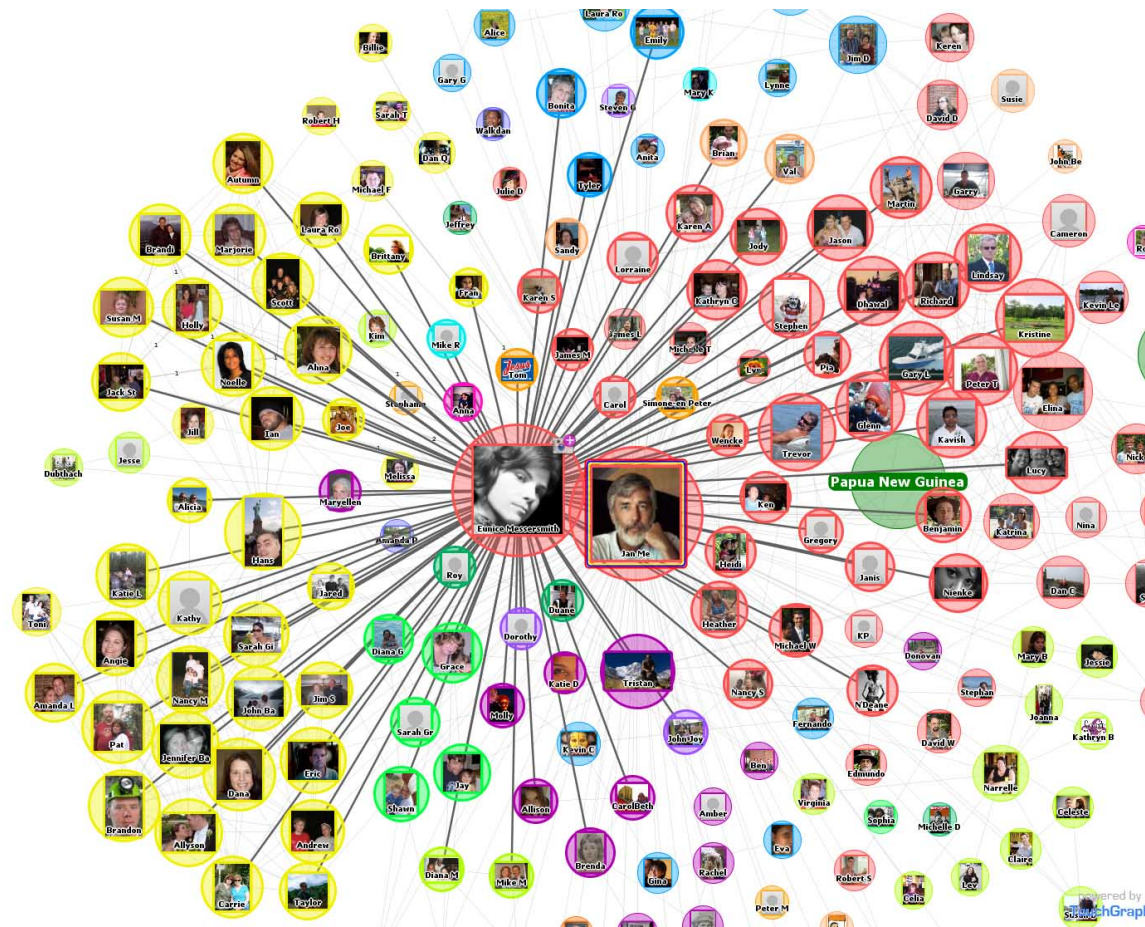


Training an algorithm to read a zip code

Machine Learning - Clustering

- 1.5 weeks
- Clustering is the process of organizing objects or people into groups. You could say that your friends on Facebook is a cluster and my friends are another cluster. Your neighbors form one cluster and my neighbors form another. Clustering is used in many applications without us even realizing it. For example, if we compare a photo on our computer with one that we print, we see that they are not exactly the same because the monitor has many more colors available than most printers. How did the printer assign a color to a pixel?
- Of course to develop an algorithm to cluster objects requires having criteria for forming the clusters such as distance, color, etc. We will look at various types of clustering and applications.
- We will see how clustering was used to discover that the disease

cholera was transmitted through the water.



How can computers solve word games?

- 1 week
- Lewis Carroll (Alice in Wonderland) developed a word game called Doublets (also called Word Ladder). For example, can you take the word **MAN** and turn it into **APE** by changing only one letter at a time. Moreover, we would like to do this in the shortest number of moves possible.
- We will look at strategies for solving this puzzle and variants and see how they can be implemented in an algorithm for a computer to solve the puzzle.
- This will give you an insight into algorithms for game solving such as chess. In 1997 IBM's chess computer beat the top-rated chess player in the world and over the next few years humans and computers traded blows until eventually by 2005-6, computer chess programs

were in the lead.

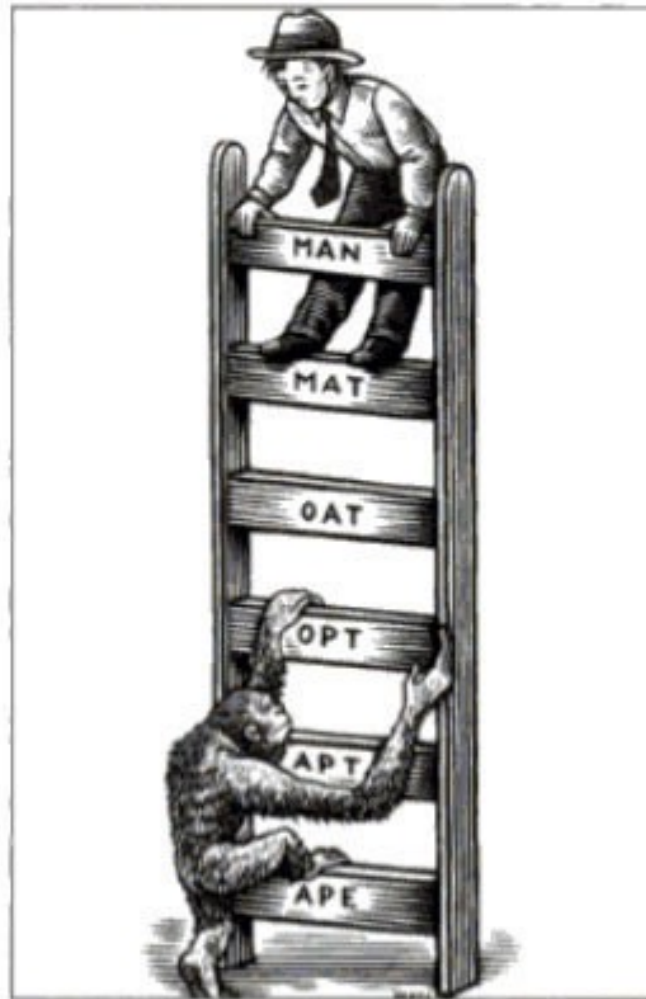


Illustration by Gregory Nemeec

Sorting items and Searching the Web

- 2.5 weeks
- Algorithm # 2 (Page Match) and Algorithm # 3 (Page Rank) in text
- Whenever you have items which you want to search repeatedly (such as web pages), then it is important to first sort the items. We will look at several approaches to sorting.
- When you do a search on the Web then not only does the search engine find information that matches your search words but it also ranks them, i.e., it puts the most relevant matches on the first page. Moreover, it does this in seconds not hours.
- Just finding matches seems like an impossible task because it is estimated that there are over a [billion](#) websites. What we will see is that when you do a search, the search engine doesn't really search

the Web but rather a sorted index of the Web which it has created earlier.

- Ranking the sites which match the search words seems like another impossible task because a computer doesn't really understand what is written on the website. How can it distinguish between spam websites and ones which have meaningful information? How can it decide which are the most relevant sites? What can webmasters do to help their sites appear on the first page of results? These are some of the questions that we will look at.

Newton's - Kepler's E-mail to Alfred Nobel Prize winner Physicists

There are no physicists after us only "University" Idiots like you

Space time - Relativistic - Quantum - Strings mechanics is not physics

Real time Universal Mechanics

Newton's - Kepler's equations solved wrong for 350 years

1001 new real time physics formulas

Changing physics and the History of physics

Annexing quantum mechanics to classical mechanics and deleting relativity and strings

It is the math formulas that matches a physics experiment results

Real time Newton's - Kepler's mechanics

Professor Joe Nahhas July 4th 1973

joenahhas1958@yahoo.com



Read my Lips: I Joe Nahhas (lucid) will end Alfred Nobel Mafia of Physics and physicists' stupidity.

There is one and only one Mechanics

Real time universal mechanics

Real time mechanics is the natural law of past present and future of mechanics.

For 400 years Newton's - Kepler's were formulated and solved wrong.

The new **real time solution** of Newton's Kepler's equations deletes modern physics which is based on relativistic quantum string time travel mechanics and matches experiments with unprecedented accuracy to change physics and the history of physics in its entirety.

Modern Physics is based on relativistic quantum string time travel mechanics. Time is caveman and modern man scale of convenience and is not Alfred Nobel dimension for time travel regardless of what all Alfred Nobel time travel "Physicists" have to say about it. Time travel is not physics and accepting time travel as physics in classrooms and using it in scientific

Data Compression

- 1 week
- Algorithm # 7 in text
- **Big data** is the new buzz word for this century. Jobs requiring big data expertise are predicted to be numerous and carry high salaries.
- Amazon gets 35 orders per second or 1.1 billion per year. That is a lot of data! In order to handle this much data it must be compressed in such a way that it can be decompressed at a later date. We want to look at some of the ideas behind data compression.



Your startup disk is almost full.

To make more space available on your startup disk, delete some files.

Don't warn me again about this disk.



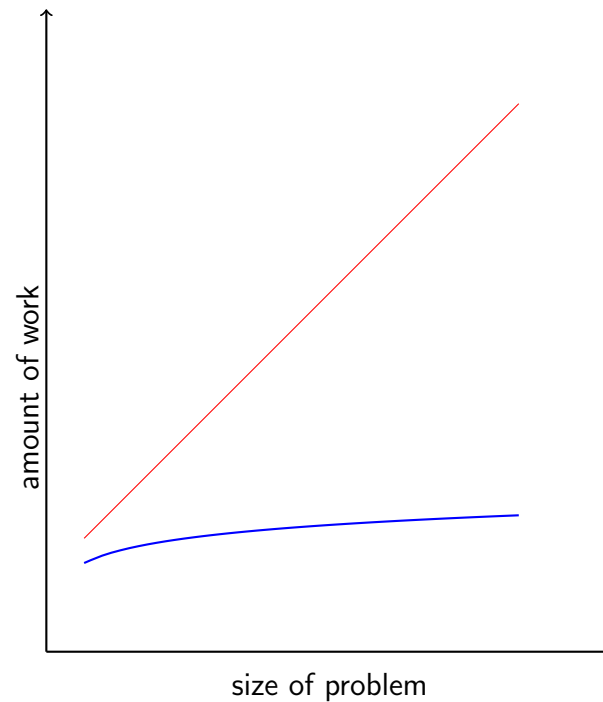
OK

Reading Assignment

Computational thinking, 10 years later, Jeannette Wing, Microsoft Research Blog, 2016

Socratic Quiz Introduction_Quiz1

CTISC1057



1. In the graph of the work required to find a name in a phone book,

the algorithm described by the blue curve requires less work for a given phone book than the one described by the red curve.

2. In the graph of the work required to find a name in a phone book using the Divide and Conquer and Brute-Force algorithms, the red line is for the Divide and Conquer algorithm and the blue curve is for the Brute-Force algorithm.
3. Different algorithms for a specific problem may give different answers.
4. At each step of a Divide & Conquer algorithm you break the problem into smaller pieces.
5. In Jeannette Wing's article on Computational Thinking she advocates teaching computational thinking in K-12 grades.
6. In programming lingo a **conditional statement** means that you have an **if - then** statement.
7. The two most common components of an algorithm are repeated

loops and conditionals.

8. Computer algorithms are better at repeated tasks than humans.
9. Microsoft Word is an example of computer hardware.
10. Computational thinking involves abstraction and breaking complex problems into smaller parts.