



Sorting Socks is Algorithm Complexity

Provided by TryEngineering -www.tryengineering.org

Lesson Focus

How do you know how fast a computer can calculate an answer, or whether an answer can be calculated at all? The field of Computational Complexity is the study of whether problems can be solved, and how fast. This lesson introduces some simple ideas about algorithms and their complexity through a series of exercises involving a collection of socks. Of course, other objects can be used as well. This is an active learning lesson that does not require access to a computer. Linear, polynomial, and logarithmic algorithms are explored building an intuitive understanding of order of magnitude.

Age Levels

Intended for 11 - 13 (US Middle School grades 6 - 8)
Can be used in lower High School (e.g. 9th grade)

Objectives

Introduce students to classic algorithms:

- ✦ for finding something in a sequence.
 - ✦ for finding something in an ordered list.
 - ✦ for simple sorting.
 - ✦ to provide informal methods for determining algorithm complexity.
-

Anticipated Learner Outcomes

Students will be able to:

- ✦ describe why finding an item in a collection may require looking at each item.
 - ✦ discuss that finding the smallest or largest in an unordered collection requires looking at each item.
 - ✦ discuss that ordering objects significantly reduces the time needed to find a specified one.
 - ✦ discuss that there are many ways to sort objects.
-

Alignment to Curriculum Frameworks

See attached curriculum alignment sheet.

Internet Connections

- ✦ <http://www.trycomputing.org/lesson-plans/complexity-its-simple-lesson> (for high school)
- ✦ <https://youtu.be/18P38NisQLs> (Sorcerer's Apprentice spoken)
- ✦ <http://www.reelyredd.com/1006sorcerersapprentice.htm> (Sorcerer's Apprentice to read)
- ✦ https://youtu.be/INHF_5RIxTE (Many sorts)
- ✦ <https://youtu.be/3San3uKkHgg> (Quick sort)
- ✦ https://youtu.be/MtcrEhrt_K0 (Lego bubble sort)
- ✦ <https://youtu.be/D5SrAga1pno> (Binary Search - first two minutes)

Recommended Reading

For reference for teachers only, not appropriate for middle school!

- ✦ <http://bigocheatsheet.com/>
- ✦ <http://www.esi2.us.es/~mbilbao/complexi.htm>
- ✦ https://en.wikipedia.org/wiki/Sorting_algorithm
- ✦ https://en.wikipedia.org/wiki/The_Sorcerer%27s_Apprentice

Optional Writing Activity

- ✦ Pick your favorite algorithm among the ones you've studied. Describe how the algorithm would work on a collection of objects. What attribute would you use for the searching or sorting. Why is it your favorite?

Sorting Socks is Algorithm Complexity

For Teachers: Teacher Resources

◆ Lesson Objectives

Introduce students to classic algorithms:

- ✦ for finding something in a sequence.
- ✦ for finding something in an ordered list.
- ✦ for simple sorting.
- ✦ and provide informal methods for determining algorithm complexity.

◆ Materials

- ✦ (Optional) Access to the Internet to watch the videos listed under 'Internet Connections'.
- ✦ A collection of objects that can be sorted. Socks are suggested – each student should bring in two mismatched socks. Any type of culturally appropriate object(s) such as mittens, hats, crayons, pencils of different sizes and color, that can be placed into a sequence based on an attribute of the object.
- ✦ Paper and pencils for taking notes and keeping track of comparisons.
- ✦ Very small Sticky notes to put into students' socks to identify socks.

◆ Procedure

This lesson is intended to give your students an intuitive understanding of how computer scientists study algorithm efficiency. It is a subject that most non-computer scientists have never encountered, and good resources that don't involve complicated math are few and far between. However, the basic ideas are pretty simple. After reading through this procedure you might want to check out some of the reading recommendations to get a more formal perspective. But remember, this is for middle school (and perhaps 9th graders) to get an intuition about how this works. Please stay away from formal math. Instead, give your student a chance to invent some algorithms.

Teams of students are given the following challenges:

1. find something in an un-ordered list.
2. find something in an ordered list using the 'guess the number' technique.
3. find an extreme (typically the smallest or largest).
4. sort the list: using simple comparisons, or exploiting the 'guess the number' technique.
5. how many brooms come alive in the 'Sorcerer's Apprentice'.

The first two challenges focus on finding something specific in a sequence, which leads naturally to finding the extremes – the minimum and the maximum (the third challenge) which in turn sets up items for sorting. The Sorcerer's problem is included to contrast logarithmic, linear, and polynomial algorithms with exponential algorithms. They contrast finding something in an unordered list with finding something in an ordered list.

Study the student resource sheet yourself, and find your own solutions to each challenge. Make sure you are comfortable explaining the difference between linear solutions (those that take time corresponding to the number of things in the list), logarithmic solutions (those that halve the number of items considered at each step), and polynomial solutions (those that take time related to squaring the number of items). Become familiar with the

'Sorcerer's Apprentice' where the number of brooms doubles with each unit of time. These kinds of problems take exponential time, and unless there is some rule for stopping, they can go on forever.

Before the First Session:

Ask students to bring in a pair of old socks, or two mismatched socks that have lost their mates. Emphasize that the socks may not return home, and that new socks should not be purchased for this. Explain that part of doing laundry is sorting socks, and this session provides efficient techniques for doing this necessary task. Encourage them to bring in an interesting sock, perhaps mismatched socks from a younger sibling, colorful ones, or ones with interesting patterns. Socks with holes are also good, as are interesting textures. Sorting socks in order to match up pairs requires a field of computer science called 'algorithm complexity'. It might be a good idea to create your own collection of old socks, or ask for contributions from your colleagues. Have each student write her name on two Sticky notes, and place those notes inside her socks. Stress that this doesn't guarantee that the socks will be identified later.

Ask your students to put all of the socks in a pile, group your students into teams of three or four, and then give or have students select two socks from the pile. Don't let them purposely choose a pair, and be sure that a group has some variety in the socks.

If you live in a culture without a variety in hosiery, consider using some other objects, because you do not need to use socks for this exercise: any small objects such as stones, marbles, partially used pencils will do. Adapt the activity as needed. The key component is that you must identify an attribute like color, size, or decoration to first locate, and then sort by. For example you can look for the red sock, locate the smallest sock, and sort socks by color (useful for finding pairs).

Session 1:

Distribute the Student Resource Sheets and Worksheet 1. Review the definitions on the worksheet. Consider explaining the 'Guess the Number' game by telling them that you can guess their number between 1 and 1000 with 10 guesses. Divide them into teams of four (with a team of 3 or 5 if necessary) and set them loose with the three challenges. If teams quickly complete the problems, give them Worksheet 2. If they complete the sorting task, send them to the Internet to listen to, or read the Sorcerer's Apprentice poem. Leave enough time at the end to review the results obtained by the groups. Identify solutions that were linear and those that were logarithmic.

Session 2:

Give students the Student Resource Sheets and Worksheet 2. Review challenges 4 and 5 on the worksheet. You might want to show them one of the videos listed in the Internet Connections section. There isn't time to show all of the videos unless you assign them for homework beforehand. Divide them into teams of four and set them loose to come up with *one* algorithm for sorting and to answer the questions about the Sorcerer's Apprentice. Help them figure out the complexity. It won't be linear, but it may be ' $n \log n$ ' or ' n^2 ' or worse. Discuss with each group what their time complexity might be. Leave enough time for the whole group to work through challenge 6, or skip it entirely.

◆ Time Needed

- ★ 2 sessions, at most 1 hour each. If you have more time, spend a third hour really exploring the ideas of halving the size of a problem, or doubling the size of a problem.

Sorting Socks is Algorithm Complexity

Student Resource:

Searching for Something

Did you ever try to find a favorite item (a toy or piece of clothing) in a pile of stuff? How many things did you have to toss aside before you found what you were looking for? Does having things organized help you find a specific item? (Think about how a library is organized.) Because computers contain sequences of memory location, collections are naturally put into a list or row rather than scattered in two dimensions. One limitation of computers is that they can only compare two things at a time. Imagine looking for a red sock among a collection of different colored socks. It is pretty easy to compare a number of items at once and grab that red one. Imagine if the best you could do is look at just two things at a time.

Sorting Stuff

Has a parent or guardian ever asked you to put stuff away? Perhaps you had to sort the laundry by color so that the whites stay white, or gathered up the freshly washed glasses and place them on the shelf in some kind of order. Sorting is very important in computer science, because when things are sorted, it's faster to find them. (Ok, we gave away the question in the searching section.) Computer Scientists realized that in order to sort a bunch of objects, you have to find the things that are alike, so searching is needed for sorting, and sorting is needed for searching. This chicken and egg problem is at the heart of algorithm complexity.

I'm Thinking of a Number

A silly game that is critical to computer science is 'I'm thinking of a number between...'. There's a sweet trick to quickly guess it: Start at the middle of the range, and keep eliminating half the numbers based on the answer given. Let's do it with a number between 1 and 16. You can guess the number with four questions. For example, let's say the number is 5. Here are the questions:

- Q: Is the number less than or equal to 8. (Taking half of 16).
A: Yes. (If the answer had been 'no' then the number is > 8 and ≤ 16 .)
Q: Is the number less than or equal to 4? (Taking half of 8).
A: No. (If the answer had been 'yes' then the number is ≤ 4 .)
Q: Is the number less than or equal to 6?
A: Yes. (If the answer had been 'no', then the number is > 6 and ≤ 8 .)
Q: Is the number 5?

As you can see, it required 4 guesses for 16 numbers. It requires only one more guess for 32 numbers, and you can actually guess the number in 10 guesses for a number between 1 and 1024. This is because you are taking the Logarithm Base 2 each time:

$$\log_2(16) = 4, \text{ or alternatively } 2^4=16.$$

What is Algorithm Complexity?

Algorithms are often called recipes for doing something. Computer Scientists don't just invent algorithms. They study algorithms to see how they can do things more efficiently. *Algorithm Complexity* is the study of how fast an algorithm can solve a problem. Much of complexity theory is dependent on the number of items you have and a lot of the problems looking remarkably like either searching or sorting.

Time Complexity describes how long it takes to do something. Computer Scientists are very interested in the time complexity of algorithms because it gives them a way of talking about how efficient an algorithm is. This lesson explores four levels of efficiency that are defined here. This is all a bit abstract as definitions, but when you do the challenges you will see patterns that tell you what the time complexity is.

Logarithmic time complexity happens when a task can be split in half and then that task can be split in half until you've solved the problem. 'I'm thinking of a number between ...' is the classic example of this. Note that we are talking about Logarithms in Base 2 (binary), not Logarithms in Base 10 (decimal).

Linear time complexity is when the time is directly related to the number of items. For example, if you have 10 items it will take you approximately 10 steps, 100 items will take 100 steps, 10,000,000 will take you 10,000,000 steps. This is still pretty good. Trying to find the first person with a purple shirt is an example. You can't guarantee that anyone has a purple shirt so you have to check each one.

Polynomial time is when things slow down – this is when you multiply the number of steps by the number of steps. If you have 10 items it will take you 100 steps, if you have 50 items it will take $50 \times 50 = 2500$ steps. This is **not** multiplying by 2 (for example 10 items would take 20 steps). These examples require tasks that *square* the number of items. You could also triple the number of items: for example if you have 5 items, the number of steps required might be $5 \times 5 \times 5 = 125$ steps. AN example is the classic example where each thing is compared with every other thing.

Exponential time is when each step doubles the size of the task. One item requires two steps, two items require four, ten items require 1024 steps. Note that exponential time is the inverse (kind of an opposite) of Logarithmic time. An example is cutting a piece of paper in two, and then cutting those two in two, and cutting those four in two, and so on.

The Halting Problem: It is possible that some computer problems can't be solved, no matter how many steps your algorithm takes. You may have heard of an 'infinite loop' where a computer program just continues and continues because either a mechanism for halting was not built in, or somehow that step was skipped or you can't get there. Complexity theorists also study if, when, and how algorithms halt.

Sorting Socks is Algorithm Complexity

Student Worksheet 1:

Write your name on two stickies and put one note in each of your socks. Add your socks to the class collection.

You will be assigned to a team of four (or three, or five). Each of you will get two unique socks. Make sure there is some *attribute* (size, color, wear and tear, interesting designs) by which they can be categorized. In all of these challenges, line up your socks in a horizontal row. Remember that you can only compare two socks at a time.

For each challenge, review your answers with your teacher before moving on.

◆ **Challenge 1:** Find a particular sock in an unordered list.

1. Put the socks in a pile, then grab each sock without looking, to build a *random order* horizontal row of socks.
2. Create an algorithm (a set of instructions) for finding a sock with a particular attribute in your list of socks. Write down your algorithm, and test it by following the instructions you created in step 2 above.
3. How many socks do you have to compare, in order to guarantee you found the sock you were looking for?
4. How many socks do you have to compare, to guarantee that a sock with a particular attribute is *not* in the list?

◆ **Challenge 2:** Find a particular sock in a list that is organized by that attribute.

1. Pick a sock attribute by which you can organize your list of socks. For example, if your socks are many different colors, organize by colors (for example, black, purple, blue, green, yellow, orange, red, white, brown, gray). Or organize by size, or number of decorations.
2. Create an algorithm for finding a sock with a particular attribute in your list of socks. Write down your algorithm and test it.
3. How many socks do you need to compare in order to guarantee you found the sock you were looking for? Hint: it's like 'Guess the number between ...'.
4. How many socks do you have to compare to guarantee that a sock with a particular attribute is *not* in the list?

◆ **Challenge 3:** Find a particular sock at the extreme (e.g. the smallest or largest). First do this for an un-ordered list. Then do it for an ordered list.

1. Pick a sock attribute that describes something at the extreme, for example size.
2. Create an algorithm for finding the sock at the extreme. Write down your algorithm and test it. Can you find the sock at the other extreme (such as the largest?)
3. How many socks do you need to compare to find the sought after item?
4. Is there such a thing as *not* finding the item at the extreme? Why?

Sorting Socks is Algorithm Complexity

Student Worksheet 2:

◆ **Challenge 4:** Sort your list of socks.

1. Place your socks in an un-ordered list.
2. Create an algorithm to sort your list by some attribute. Remember that you can only compare two items at the same time. Write it down.
3. Place your socks in another un-ordered list. As you run your algorithm, count the number of comparisons you had to make to get the list sorted.
4. Hypothesize time complexity of your algorithm given the number of socks, is there a formula that explains its relationship to the number of comparisons? Try sorting a smaller set of socks, or pool resources with another group and sort a larger list.
5. Odds are your algorithm will be polynomial time. Is it? Sorting can be done much more efficiently if you think about the 'Guess the number between...' trick. This time, though, it's not about having the list in an order at the start, but thinking about how to work on half of the list at a time.

◆ **Challenge 5:** Sorcerer's Apprentice: How to halt a Runaway Algorithm

1. Ask to read or listen to the Sorcerer's Apprentice poem.
2. If you start with one broom, and every hour all of the current brooms are cut in half, how many brooms will there be in 10 hours? You might want to make a table of hours and number of brooms.
3. In the poem, the Wizard returns and stops all this nonsense. What would happen if the Wizard didn't return?
4. Is the Wizard a built-in mechanism to halt the algorithm? Why?

Whole group exercise. Answer these questions verbally.

◆ **Challenge 6:** The Returning of the Socks

1. Place all of the socks in a pile in the middle of the room, with all of the students in a circle around the pile.
2. Have a time keeper who starts a stopwatch say 'Go'.
3. Without pushing and shoving, but by politely asking each other for help, return all of the socks to their rightful owners. Stop the watch when all the socks are returned.
4. Did your algorithm halt? (Did all the socks get returned, did everyone who contributed socks get them back?)
5. Can you describe your algorithm in words?
6. Did you use more than one comparison at a time?
7. What abilities did you as humans have that a simple list-based, only-one-comparison algorithm doesn't have.
8. Any thoughts on the time complexity? Can you even calculate it with so many people involved?

Sorting Socks is Algorithm Complexity

Student Worksheet 2:

◆ Challenge 4: Sort your list of socks.

6. Place your socks in an un-ordered list.
7. Create an algorithm to sort your list by some attribute. Remember that you can only compare two items at the same time. Write it down.
8. Place your socks in another un-ordered list. As you run your algorithm, count the number of comparisons you had to make to get the list sorted.
9. Hypothesize time complexity of your algorithm given the number of socks, is there a formula that explains its relationship to the number of comparisons? Try sorting a smaller set of socks, or pool resources with another group and sort a larger list.
10. Odds are your algorithm will be polynomial time. Is it? Sorting can be done much more efficiently if you think about the 'Guess the number between...' trick. This time, though, it's not about having the list in an order at the start, but thinking about how to work on half of the list at a time.

◆ Challenge 5: Sorcerer's Apprentice: How to halt a Runaway Algorithm

5. Ask to read or listen to the Sorcerer's Apprentice poem.
6. If you start with one broom, and every hour all of the current brooms are cut in half, how many brooms will there be in 10 hours? You might want to make a table of hours and number of brooms.
7. In the poem, the Wizard returns and stops all this nonsense. What would happen if the Wizard didn't return?
8. Is the Wizard a built-in mechanism to halt the algorithm? Why?

Whole group exercise. Answer these questions verbally.

◆ Challenge 6: The Returning of the Socks

9. Place all of the socks in a pile in the middle of the room, with all of the students in a circle around the pile.
10. Have a time keeper who starts a stopwatch say 'Go'.
11. Without pushing and shoving, but by politely asking each other for help, return all of the socks to their rightful owners. Stop the watch when all the socks are returned.
12. Did your algorithm halt? (Did all the socks get returned, did everyone who contributed socks get them back?)
13. Can you describe your algorithm in words?
14. Did you use more than one comparison at a time?
15. What abilities did you as humans have that a simple list-based, only-one-comparison algorithm doesn't have.
16. Any thoughts on the time complexity? Can you even calculate it with so many people involved?

Sorting Socks is Algorithm Complexity

Teacher Resource:

Alignment to Curriculum Frameworks

Note: All lesson plans in this series are aligned to the Computer Science Teachers Association K-12 Computer Science Standards, the U.S. Common Core State Standards for Mathematics, and if applicable also to the National Council of Teachers of Mathematics' Principles and Standards for School Mathematics, the International Technology Education Association's Standards for Technological Literacy, and the U.S. National Science Education Standards which were produced by the National Research Council.

◆ National Science Education Standards Grades 5-8 (ages 10-14)

CONTENT STANDARD E: Science and Technology

As a result of activities, all students should develop

- ✦ Understandings about science and technology

◆ National Science Education Standards Grades 9-12 (ages 14-18)

CONTENT STANDARD E: Science and Technology

As a result of activities, all students should develop

- ✦ Understandings about science and technology

◆ Next Generation Science Standards & Practices Gr.9-12 (ages 14-18)

Practice 5: Using Mathematics and Computational Thinking

- ✦ Create and/or revise a computational model or simulation of a phenomenon, designed device, process, or system.
- ✦ Use mathematical, computational, and/or algorithmic representations of phenomena or design solutions to describe and/or support claims and/or explanations

◆ Principles and Standards for School Mathematics Gr. 6-8 (ages 11-13)

Algebra Standard

Understand patterns, relations, and functions

- ✦ represent, analyze, and generalize a variety of patterns with tables, graphs, words, and, when possible, symbolic rules;

◆ Principles and Standards for School Mathematics Gr. 9-12 (ages 14-18)

Algebra Standard

Understand patterns, relations, and functions

- ✦ analyze functions of one variable by investigating rates of change, intercepts, zeros, asymptotes, and local and global behavior

◆ Principles and Standards for School Mathematics (all ages)

Problem Solving Standard

- ✦ Solve problems that arise in mathematics and in other contexts

Connections

- ✦ Recognize and apply mathematics in contexts outside of mathematics

Representations

- ✦ Use representations to model and interpret physical, social, and mathematical phenomena

Sorting Socks is Algorithm Complexity

Teacher Resource:

Alignment to Curriculum Frameworks

◆ Common Core State Standards for Mathematics Gr. 9-12 (ages 14-18)

Functions Standard

Linear, Quadratic, and exponential models

- Construct and compare linear, quadratic, and exponential models and solve problems

- ✦ CCSS.Math.Content.HSF-LE.A.1 Distinguish between situations that can be modeled with linear functions and with exponential functions.
- ✦ CCSS.Math.Content.HSF-LE.A.1b Recognize situations in which one quantity changes at a constant rate per unit interval relative to another.

◆ Common Core State Practices & Standards for School Mathematics (all ages)

- ✦ CCSS.MATH.PRACTICE.MP1 Make sense of problems and persevere in solving them.
- ✦ CCSS.MATH.PRACTICE.MP4 Model with mathematics.

◆ Standards for Technological Literacy – All Ages

The Nature of Technology

- ✦ Standard 2: Students will develop an understanding of the core concepts of technology.

◆ CSTA K-12 Computer Science Standards Grades 6-9 (ages 11-14)

5. 2 Level 2: Computer Science and Community (L2)

- ✦ Computational Thinking (CT)
 3. Define an algorithm as a sequence of instructions that can be processed by a computer.
 4. Evaluate ways that different algorithms may be used to solve the same problem.
 5. Act out searching and sorting algorithms.
- ✦ Collaboration (CL)
 3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.

◆ CSTA K-12 Computer Science Standards Grades 9-12 (ages 14-18)

5.3 Level 3: Applying Concepts and Creating Real-World Solutions (L3)

5.3.A Computer Science in the Modern World (MW)

- ✦ Computational Thinking (CT)
 8. Use modeling and simulation to represent and understand natural phenomena.