Complexity – It's Simple

Provided by TryEngineering.org - www.tryengineering.org

Lesson Focus
This lesson allows students to playfully understand algorithms and complexity.

Lesson Synopsis
The Complexity lesson allows students learn about complexity through illustrative games, teamwork activities and design tasks. Students will gain an intuitive understanding of different growth rates and how they determine the performance of algorithms such as sorting. Advanced students can also develop skills in analyzing the complexity of algorithms.

Age Levels
14 - 18

Objectives
- Learn about the growth of sequences
- Learn about the difference between complexity and runtime
- Learn fundamental algorithms in computer science
- Learn how good algorithm design can drastically improve performance

Anticipated Learner Outcomes
As a result of this activity, students should develop an understanding of:

- the growth of sequences
- the importance of algorithm design
- sorting algorithms
- teamwork

Lesson Activities
Students gain an intuitive understanding of complexity and runtime of algorithms through illustrations with small rewards, teamwork activities and design tasks. More advanced students are introduced to the mathematical concepts underlying the topic of complexity.

Resources / Materials
- Teacher Resource Documents (attached)
- Student Worksheets (attached)
- Student Resource Sheets (attached)

Internet Connections

- TryEngineering: (www.tryengineering.org)
- Complexity Zoo: (https://complexityzoo.uwaterloo.ca/Complexity_Zoo#sthash.1Gnn1E1Z.dpuf)
- P vs NP Problem: (http://www.claymath.org/millenium-problems/p-vs-np-problem#sthash.1Gnn1E1Z.dpuf/)
- Moore's Law: (http://www.intel.com/content/www/us/en/silicon-innovations/moores-law-embedded-technology.html)
- Algorithm Analysis: (http://www.londoninternational.ac.uk/sites/default/files/computing-samples/co2226_vol2_ch1.pdf#sthash.1Gnn1E1Z.dpuf)
- ITEA Standards for Technological Literacy: Content for the Study of Technology (http://www.iteaconnect.org/TAA/)
- National Science Education Standards: (http://www.nsta.org/publications/nses.aspx)
- Scaling (http://galileoandeinstein.physics.virginia.edu/lectures/scaling.html)

Recommended Reading

- Growth of Sequences and Sorting Algorithms: Introduction to Algorithms by T. H. Cormen et al. (ISBN: 0070131511)
- Dimensional Analysis: The Pleasures of Counting by T. W. Körner (ISBN: 0521568234)

Optional Writing Activity

- Compare different sorting methods and explain why you would prefer one over the other.
- Write a short text on the importance of computational complexity in cryptography.

Credits

- This lesson plan was developed by Clemens Wiltsche, an IEEE Graduate Student Member; Freilassing, Germany (Region 8), and Kevin Zemmer as part of the IEEE TryComputing.org Lesson Plan Competition.

For Teachers: Teacher Resources

Lesson Goal

The complexity lesson gives students the opportunity to learn about the growth of sequences and its fundamental importance in algorithm design. Students will analyze, develop and execute algorithms in a playful way. Advanced students can explore state-of-the-art ideas to improve the complexity of sorting algorithms.

Lesson Objectives

- Be able to describe how sequences differ in terms of how quickly they grow;
- Be able to use an algorithm to generate a sequence;
- Be able to predict the relative runtime speed of two algorithms given their complexity;
- Use complexity and data set size to explain why one algorithm would be take less time execute than another

Materials

- Student Resource Sheets and Worksheets
- Cardboard to make numbered cards, or several decks of playing cards
- Reward items: Rice, small candy or other low-cost items. Taking rice as an example, about 25 tablespoons are needed (this is a little less than 400 g)

Preparation

- Make cards from cardboard or paper and number them sequentially starting from 1. Three cards per student are required. Alternatively two or three decks of playing cards can be used but the order of the cards has to be explained to the students.

Procedure

1. Introduce the subject to the students by playing the reward game with them, which is described in the teacher resource “Group Games”.
2. Hand out and discuss the student resource “The Growth of Sequences”. Its understanding is vital for the rest of the lesson.
3. Test the understanding of the material with the student worksheet “The Growth of Sequences”. Solutions can be found in the teacher resources.
4. Optional: Have students read the student resource “Algorithms” to get to know the terminology used in Computer Science.
5. Play the number guessing game explained in the teacher resource “Group Games”. Then hand out the student worksheet “A Number Guessing Game”. The solutions for this are given in the teacher resources.
6. Finally, introduce the students to sorting algorithms by following the instructions on the student worksheet “Sorting Algorithms”, i.e. deal out the cards and put the students into the suggested group sizes. Answers to the questions are provided in the teacher resources.

Time Needed

- Two 45 minute sessions

Advanced Options

- Advanced students can write a comparative essay on various sorting methods and the explain criteria for preferring one method over the other.
- Advanced students can be asked to give arguments of the computational complexity of the considered algorithms and why the performance of comparison sort algorithms is fundamentally limited.

For Teachers:
Teacher Resources – Group Games

◆ Reward Game

The game consists of providing rewards, in form of small items, to the students. The game is explained here with rice as an example, but it works equally well for other small rewards.

Rice is given out in rounds. The goal of the game is to get as much rice as possible. The amount of rice given out in each round is determined by one of two reward strategies:

Option 1: Every round, a teaspoon of rice will be dealt out.

Option 2: In the first round the reward is only a single grain of rice. Then, in each successive round, the amount of rice given out is doubled.

Let each student choose one of the reward strategies and group the students according to the choice they made.

Deal out the rewards round by round to each group. You could appoint a student in each group to help count out the rice. Note that towards the end an exact count of grains is not important.

After the first four rounds, ask the students to compare the cumulative reward. Do the same after eight and twelve rounds. Then discuss the outcome of the game with the students.

Students should get an intuitive understanding of the different growth rates. This can also be related to the increase in computing power of modern microprocessors. According to Moore's Law, the number of transistors that can be put on a chip doubles approximately every two years. The students can be asked to think about the consequences of this observation and whether it is possible that this growth continues much longer.

◆ Number Guessing Game

The aim of the game is to guess a number with as few tries as possible. The game is first played with the whole class, and then the students will be put into pairs to investigate their own strategies to solve the game.

Tell the students to stand up. Think of a number between 1 and 50 and don't tell it to the students. They are now allowed to only ask questions such as “Is it 37?” or “Is it 20?” and you may only answer with “yes” or “no” until they find the correct number. A student that asks a question leading to a “no” answer will have to sit down and may not ask any more questions until the game is finished. Note that all students might be sitting before the game is finished. In this case you might decide to play the game again and discuss why it did not work and that with this type of questions in the worst case 50 tries are required.

Now let the students have a try at developing their own strategies when they have more useful questions available by letting them work on the student worksheet “A Number Guessing Game”.

For Teachers:
Teacher Resources (continued)
Student Worksheet: The growth of sequences - SOLUTION

◆ Water Lilies

9 weeks. Since the number of lilies doubles every week, if at 9 weeks the pond is halfway covered, it is completely covered after 10 weeks. An example for six weeks is given in the teacher resource "Water Lilies".

The growth of the sequence describing the area covered is exponential.

◆ How much can they carry?

2D surface areas grow quadratically with the length of the sides, while 3D items can grow cubically (i.e. to the power of three). Therefore, the weight that a skeleton can carry is proportional assuming that all sides are growing linearly a cubic sequence grows faster than a quadratic one.

An increase in an animal's height can only be achieved by making that animal's skeleton of a material which is harder and stronger than for smaller animals, or by enlarging the size of the animal's bones. Because of the small weight of an ant, its skeleton (and muscles) are strong enough to carry ten to fifty times its own weight, while the same is not true for humans. Think about a building's weight supported by a pillar. If the upper floors are too heavy for the stone pillar, it begins to crack and crumble. For a uniform material, the weight it can carry is proportional to the cross-sectional area. So if you double all the dimensions of the building its weight increases by 8 times, the pillars capacity will only go up fourfold.

This can also be illustrated by considering a mouse falling down an elevator shaft. Due to its small weight it can survive, while larger animals would not.

(Example adapted from T.W. Körner, The Pleasures of Counting, Cambridge University Press, 1996)

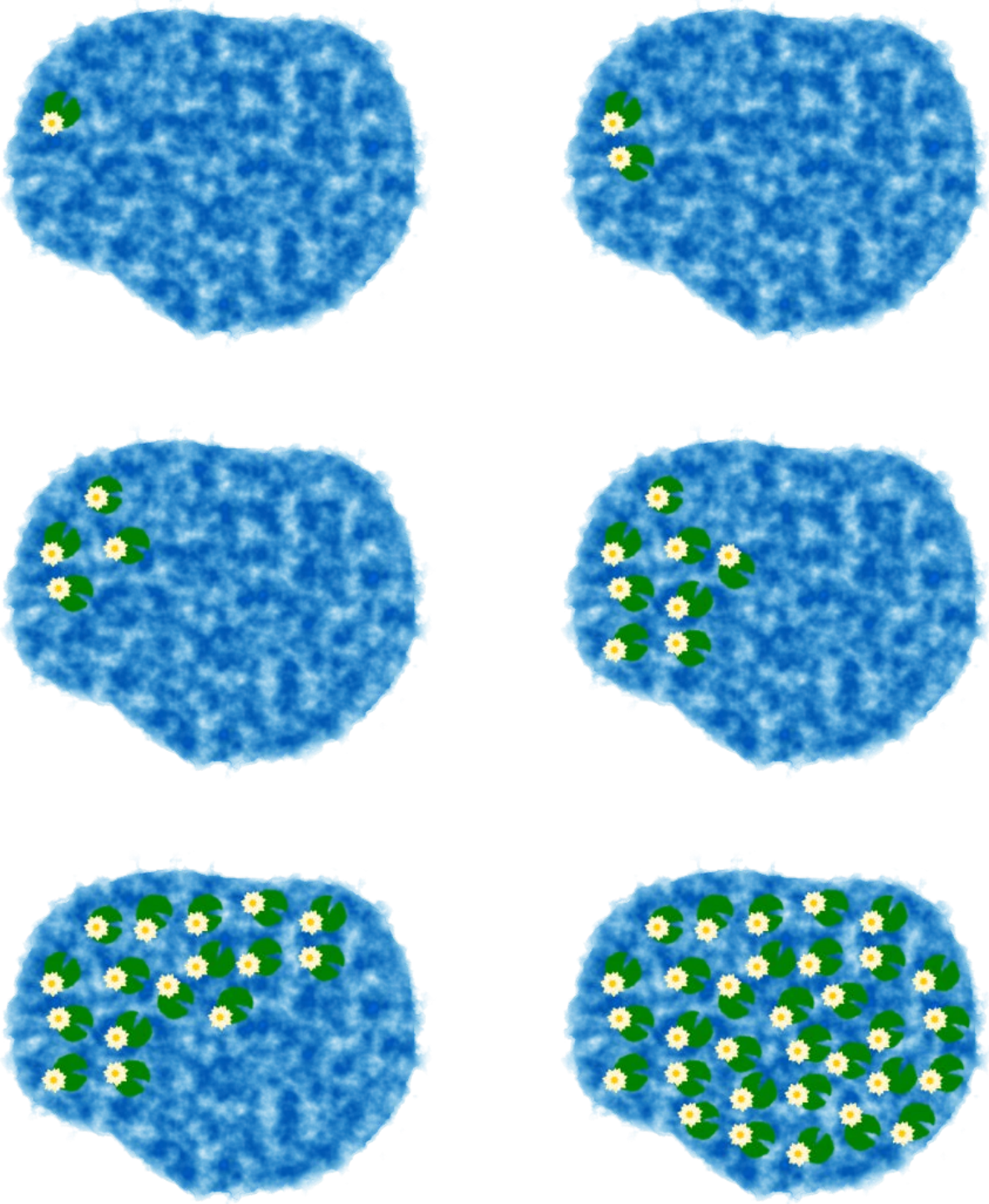
◆ Complete the Sequences

The sequences are continued as follows, with the respective growth functions given in parentheses:

- 2, 2, 2, 2, **2**, **2** (2)
- 3, 6, 9, 12, 15, **18**, **21** (3 x n)
- 4, 9, 16, 25, **36**, **49** ($(n + 1)^2$)
- 2, 5, 10, 17, 26, **37**, **50**
- 64, 96, 112, 120, 124, **126**, **127**

The first sequence grows slowest (its growth function is just a constant) while the third sequence grows fastest (its growth function is quadratic). The fourth sequence also has a quadratic growth function and thus grows at the same rate, but its individual items will always be lower than those of the third sequence.

For Teachers:
Teacher Resources (continued)
Student Worksheet: The growth of sequences – Water Lilies



For Teachers:
Teacher Resources (continued)
Student Worksheet: A Number Guessing Game - SOLUTION

◆ The Bisection Method

An optimal strategy to approach the number guessing game is the **bisection method**: Start by asking "Is it (strictly) greater than 50?" If the answer is "yes" proceed with "Is it greater than 75?", if not, ask "Is it greater than 25?" Continue in this manner of always asking for the middle number until there is only one option left. For example:

(The number is 54)
Is it greater than 50? Yes
Is it greater than 75? No
Is it greater than 63? No
Is it greater than 57? No
Is it greater than 54? No
Is it greater than 52? Yes
Is it greater than 53? Yes
So after 7 questions it is clear that the number must be 54. (There are cases in which only 6 questions are necessary. In fact, the average number of questions is log 100 = 6.64.)

This is an optimal method because with every question the set of possible numbers is split in two approximately equal parts, with one containing the required number. In fact, for this problem, every method that halves the possible numbers per question is optimal. Another solution would be to ask questions such as "Is the number divisible by 2?"

◆ Optional: Find the Complexity

The complexity of the bisection method is logarithmic, i.e. of order O(log N). This is a consequence of halving the possible numbers at every question.

If all integers are allowed, then no method can solve the problem in finite time.

For Teachers:
Teacher Resources (continued)
Student Worksheet: Sorting Algorithms - SOLUTION

◆ Bubble Sort

If the deck is not sorted, there will be at least two consecutive cards that are not in the correct order. The algorithm will eventually swap these two cards. Therefore, if there are no more cards to be swapped, the algorithm terminates and the deck is sorted.

Bubble Sort has the advantage of being very simple to implement, but it is not as efficient as more advanced sorting algorithms such as merge sort (see below). The run time complexity of bubble sort for n cards is quadratic, or $O(n^2)$.

◆ Faster Sorting Algorithms

While the run time complexity of bubble sort is quadratic, better methods exist. For example merge sort has a run time complexity of O(n log n) for n cards. An interesting feature of merge sort is that its structure allows it to be parallelized. Another algorithm with run time complexity O(n log n) that is often used in practice quick sort. It generally performs well even though its run time complexity is quadratic and thus worse than that of merge sort. It is also common to use hybrid versions of different sorting algorithms to adapt to specific problems. For a derivation of these results see T. H. Cormen et al., Introduction to Algorithms.

For sorting based on comparing individual items, there is no method with a worst case run time complexity better than O(n log n) that does not exploit parallel operations such as merging different stacks of cards at the same time. This is because n cards can be ordered in n^n different ways, and each comparison can half the number of possible orderings, leading to a complexity of $O(\log (n^n)) = O(n \log n)$.

Student Resource: Growth of Sequences

◆ The Growth of a Sequence

Why is it better to choose a small reward that doubles every time than a large reward that stays the same? Such considerations can be very important in for example in economics or when trying to find the most efficient way to solve a problem. Mathematics can be very useful in understanding the principles governing processes evolving over time.

Let's look at some sequences:

- 5, 10, 15, 20, 25, 30, ...
- 1, 4, 9, 16, 25, 36, ...
- 2, 4, 8, 16, 32, 64, ...

The pattern in a sequence can be described by **growth functions**. These patterns are what computer scientists commonly call algorithms. Such a function answers the question: "What's the number in the sequence at a given position n?" In the examples above these growth functions are:

- **Linear**: 5 x n – in each step the number grows by 5
- **Quadratic**: n x n = n^2 – in each step the number is multiplied by itself
- **Exponential**: 2 x 2 x ... x 2 = 2^n – in each step the previous number is doubled

Such sequences can indicate how great a (positive) reward is at each step or how high a (negative) resource requirement such as energy or time is at each step. Engineers are often concerned with maximizing a reward or minimizing a resource requirement, so they compare these sequences with each other. In the three examples above, you can see that the first sequence overtakes the second sequence and third second sequence overtakes the other two sequences.

◆ Optional: Big-O Notation

Engineers like to write in precise and clear ways, so when comparing the above sequences this is rather written as

$$O(5 \times n) < O(n^2) < O(2^n)$$

Such an inequality immediately allows comparing the growth of sequences.

The mathematical notation above is called **big-O** notation and describes an upper bound on the sequence for all indexes n greater than a (fixed) index m. Constant factors are usually omitted and so O(5 x n) = O(n). Can you guess why O(5 x n + 1) = O(n)? (Hint: Choose m and omit the constant factor.)

Therefore, in big-O notation only the fastest growing function is considered.

Student Resource: Algorithms

◆ What is an Algorithm?

A computer can be used to perform a plethora of tasks that can help us with our work. Sometimes these tasks are easy to describe in words, such as “sort this deck of cards”, or “find the fastest way from Paris to New York”. However, for a computer these tasks might be very hard to solve. Internally a computer can only perform rather **primitive operations** and only their correct arrangement solves a problem. Such an arrangement is called an **algorithm**. Since each primitive operation takes some time (a few nanoseconds in a modern computer), an algorithm requires a certain **run time** in order to finish its task.

◆ Why are Some Algorithms Better than Others?

Each task requires a specially designed algorithm that can solve it. Sorting a set of cards cannot be done with an algorithm that is designed to find the shortest path between two cities. Also, quite intuitively, sorting a deck of 200 cards takes longer than sorting just 52 cards (or in the trivial case, 2 cards). Therefore the run time depends on the problem and its size (computer scientists speak of a **problem instance**).

Algorithms should be compared independently of the problem instance (otherwise the best algorithm would always be to just immediately give the answer calculated before.) This is where the concept of the growth of a sequence comes in useful: The size of the problem can be regarded as the parameter n in a sequence that is specific to the algorithm. There are therefore algorithms of various **complexity**, corresponding to the sequence describing them. Algorithms are then compared by comparing the growth of the sequences. Therefore an algorithm of quadratic complexity is better than an algorithm of exponential complexity.

◆ Complexity - it's Useful

The efficiency of algorithms is crucial for a plethora of current computing applications. Some problems are very hard to solve because the only known algorithms solving them have a very high complexity.

A widely known example of high complexity is the traveling salesman problem (TSP). In this problem, a trader wants to do business in a set of cities. He wants to find the shortest possible route between these cities without visiting any city more than once, and he wants to come back to his starting city at the end of his itinerary. Solving this efficiently would be very useful for the assignment of routes for airplanes, scheduling of tasks on machines and even the manufacturing of microprocessors.

While the complexity of the TSP prevents an efficient solution of these problems, for some applications, high complexity can even be desirable. When an encrypted message is sent, it should be very hard to get at the transmitted information without knowing some shared secret. In this case, the encryption is designed in such a way that the decryption has a very high complexity and thus eavesdropping is prevented. Secure communication would not be possible without the high complexity of decryption. These examples and numerous others show that knowing the complexity of a problem is very useful for computing and beyond.

Student Worksheet: The Growth of Sequences

◆ Water Lilies

On a pond there is a single water lily. Every week the number of water lilies on the pond doubles. The pond is completely covered by water lilies after 10 weeks. After how many weeks is only half the pond covered by water lilies? Explain your reasoning.

◆ How much can they carry?

The weight that a skeleton can carry is proportional to its cross-sectional area. The weight of an animal is proportional to its volume. Why do you think an ant can carry ten to fifty times its own weight while a human can hardly carry its own weight? (Hint: Compare the growth of an area and a volume.)

◆ Complete the Sequences

Find the next two elements in each of the following sequences.

- 2, 2, 2, 2, ___, ___
- 3, 6, 9, 12, 15, ___, ___
- 4, 9, 16, 25, ___, ___
- 2, 5, 10, 17, 26, ___, ___
- 64, 96, 112, 120, 124, ___, ___

For the first three sequences, also find their respective growth function (in terms of n).

Which sequence grows slowest?

Which sequences grows fastest?

Student Worksheet: A Number Guessing Game

Work together with your neighbor. One of you has to think of a number between 1 and 100. Don't tell it to your partner yet. The aim of your partner is to guess that number in as few tries as possible. You are only allowed to answer questions with “yes” or “no”. Once your partner has found the correct number, switch roles.

◆ The Bisection Method

You are only allowed to ask questions such as “Is the number greater than 20?” or “Is the number greater than 54?” What is the best strategy you can find in this case? How many tries do you need in the worst case?

Explain in a few words why your method is optimal.

◆ Optional: Find the Complexity

Can you find a sequence describing how many questions are required in the worst case if numbers from 1 to N are allowed? The answer is best expressed in big-O notation. (Hint: Try the logarithm.) What happens if all integers are allowed?

If you want to know more about finding the complexity, you can look at the book by T. H. Cormen et al., Introduction to Algorithms or at the website http://www.londoninternational.ac.uk/current_students/programme_resources/cis/pdfs/subject_guides/level_2/cis226_vol2/cis226_chap1.pdf

Student Worksheet: Sorting Algorithms

You will now look at a few sorting methods. Your teacher will give you cards and you will learn two different sorting methods.

To mimic the behavior of a computer, you can only to perform certain simple operations: You are allowed to look at two cards at a time, compare them and move them wherever you want, depending only on the values of the two cards. Such operations could include swapping the cards, inserting one card before the other, etc. Of course, you may also open new auxiliary stacks of cards.

However, you are not allowed to just look at all the cards, remember their value and immediately place them in the right position.

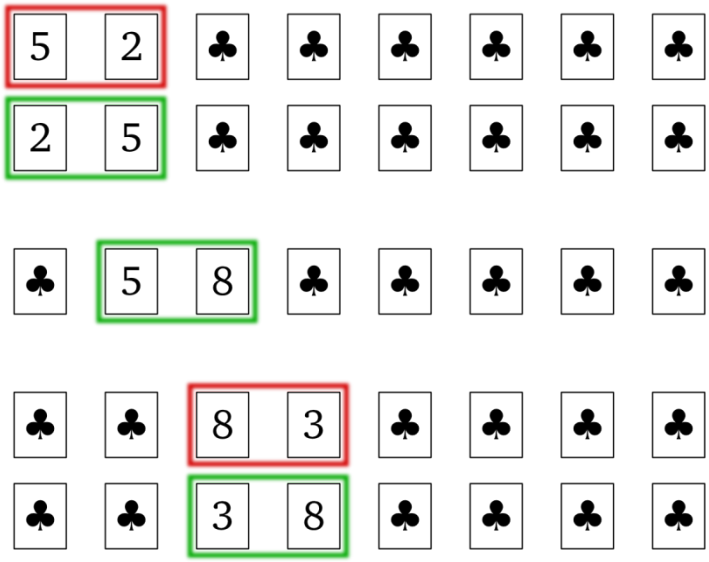
◆ Bubble Sort

Get in groups of three. Each group will get eight cards. Shuffle the cards and place the cards face down on the table.

A simple method for sorting is bubble sort. Apply the following steps to your 8 cards:

1. Look at the first two cards and compare them
2. Swap the cards (if necessary) so that they are in increasing order
3. Now look at the second and third card and swap them if they are not in increasing order
4. Continue doing so until you reach the seventh and eighth card. Swap them if necessary
5. Now repeat steps 1-4 until you don't need to do any more swaps.

The first few swaps are illustrated in the figure to the right. The first two cards need to be swapped, in the center the cards are already in the correct order and the "8" and the "3" need to be swapped again.



After completion, turn over all the cards and check whether they are sorted. Explain in a few words why this method works.

Optional: Get in groups of five to ten and find your own sorting method using only the allowed operations as explained above. Can you get any better than bubble sort? Why / why not?

Teacher Resource: Alignment to Curriculum Frameworks

Note: Lesson plans in this series are aligned to one or more of the following sets of standards:

- U.S. Science Education Standards (http://www.nap.edu/catalog.php?record_id=4962)
- U.S. Next Generation Science Standards (http://www.nextgenscience.org/)
- International Technology Education Association's Standards for Technological Literacy (http://www.iteea.org/TAA/PDFs/xstnd.pdf)
- U.S. National Council of Teachers of Mathematics' Principles and Standards for School Mathematics (http://www.nctm.org/standards/content.aspx?id=16909)
- U.S. Common Core State Standards for Mathematics (http://www.corestandards.org/Math)
- Computer Science Teachers Association K-12 Computer Science Standards (http://csta.acm.org/Curriculum/sub/K12Standards.html)

◆ Principles and Standards for School Mathematics

Algebra Standard

As a result of activities, all students should develop

- ✦ Understand patterns, relations, and functions.
- ✦ Use mathematical models to represent and understand quantitative relationships.

Problem Solving Standard

As a result of activities, all students should develop

- ✦ Apply and adapt a variety of appropriate strategies to solve problems.
- ✦ Monitor and reflect on the process of mathematical problem solving.

◆ Common Core State Standards for Mathematics Grades 9-12 (ages 14 – 18)

Algebra Standard

- Reasoning with Equations & Inequalities
  - o Solve equations and inequalities in one variable
    - ✦ CCSS.Math.Content.HSA-REI.B.3 Solve linear equations and inequalities in one variable, including equations with coefficients represented by letters.

Functions Standard

- Building functions
  - o Build a function that models a relationship between two quantities
    - ✦ CCSS.Math.Content.HSF-BF.A.2 Write arithmetic and geometric sequences both recursively and with an explicit formula, use them to model situations, and translate between the two forms.
- Linear, Quadratic, and exponential models
  - o 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.1c Recognize situations in which a quantity grows or decays by a constant percent rate per unit interval relative to another.

Teacher Resource: Alignment to Curriculum Frameworks

◆ Standards for Technological Literacy – All Ages

The Nature of Technology

- ✦ Standard 1: Students will develop an understanding of the characteristics and scope of technology.
- ✦ Standard 2: Students will develop an understanding of the core concepts of technology.
- ✦ Standard 3: Students will develop an understanding of the relationships among technologies and the connections between technology and other fields of study.

The Designed World

- ✦ Standard 17: Students will develop an understanding of and be able to select and use information and communication technologies.

◆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.
  8. Use visual representations of problem states, structures, and data (e.g., graphs, charts, network diagrams, flowcharts).
  9. Interact with content-specific models and simulations (e.g., ecosystems, epidemics, molecular dynamics) to support learning and research.
- ✦ 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.
  4. Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.
- ✦ Computing Practice & Programming (CPP)
  4. Demonstrate an understanding of algorithms and their practical application.

Teacher Resource:
Alignment to Curriculum Frameworks

◆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)

  3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.

  8. Use modeling and simulation to represent and understand natural phenomena.

5.3.B *Computer Science Concepts and Practices (CP)*

- Computational Thinking (CT)

  5. Use data analysis to enhance understanding of complex natural and human systems.

  9. Analyze data and identify patterns through modeling and simulation.