(http://csmatters.org) 5 - 3

0b101 - 0b11

Sorting

Unit 5. Data Manipulation

Revision Date: Jul 22, 2019
Duration: 3 50-minute sessions

Lesson Summary

Summary

In this three-session lesson, students explore and confront the difficulties of the problem of sorting data and the difficulties involved in expressing a clear and efficient algorithm for sorting.

Outcomes

- Students will be able to relate a real-world task such as sorting cards to sorting/organizing information in a computer.
- Students will understand the problem of sorting and why it is nontrivial for large data sets.
- Students will be able to describe in pseudocode simple sorting algorithms (such as bubblesort).
- Students will be able to reason about the correctness and efficiency of different sorting algorithms, and will understand that the time required to sort a data set increases as the size of the data set grows.

Overview

Session 1:

1. Getting Started (5 min) - Journal
2. Activity Card Sorting (40 min)
    1. Explain the Problem [10 min]
    2. Paired Activity: Algorithm Creation [30 min]
3. Wrap Up (5 min)

Session 2:

1. Getting Started (5 min) - Journal
2. Group Activity: Algorithm Evaluation (20 min)
3. Group Activity: Algorithm Selection and Justification (20 min)
4. Wrap Up (5 min)

Session 3:

1. Getting Started (10 min) - Journal and short discussion
2. Guided Activity: Algorithm Analysis (10 min)
3. Paired Activity: Algorithm Analysis (20 min)
4. Wrap Up (10 min)

Learning Objectives

CSP Objectives

- *EU CRD-1 - Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.*
    - LO CRD-1.A - Explain how computing innovations are improved through collaboration.
    - LO CRD-1.C - Demonstrate effective interpersonal skills during collaboration.
- *EU CRD-2 - Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.*
    - LO CRD-2.E - Develop a program using a development process.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
    - LO AAP-2.L - Compare multiple algorithms to determine if they yield the same side effect or result.
- *EU AAP-4 - There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.*
    - LO AAP-4.A - For determining the efficiency of an algorithm: a. Explain the difference between algorithms that run in reasonable time and those that do not. b. Identify situations where a heuristic solution may be more appropriate.

Math Common Core Practice:

- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP8: Look for and express regularity in repeated reasoning.

Key Concepts

The algorithmic techniques and analysis involved in sorting data are seen in a wide variety of contexts and applications. Sorting numbers in a list is challenging but foundational to many algorithms in computer science.

Essential Questions

- How can computation be employed to facilitate exploration and discovery when working with data?

- What considerations and trade-offs arise in the computational manipulation of data?
- Why are some languages better than others when used to implement algorithms?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?
- Which mathematical and logical concepts are fundamental to computer programming?

What makes a "good" algorithm?

What should be taken into consideration when comparing algorithms that complete the same task?

Teacher Resources

Student computer usage for this lesson is: **optional**

- You will need enough playing cards for every pair of students to have eight different cards. Alternatively, you may make your own cards (e.g., using index cards) with different numbers on them in place of playing cards.
- Handout - Sorting Algorithm Evaluation Student Worksheet.docx (in lesson folder)
- Video collection - https://www.youtube.com/user/AlgoRythmics/videos (https://www.youtube.com/user/AlgoRythmics/videos)
  - **Note:** If students do not have access to computers to individually watch the sorting videos during the paired activity in Session 3, you could instead choose one of the algorithms and show it to the class, having all pairs complete the algorithm evaluation handout for that selected method.

Lesson Plan

Session 1

Getting Started (5 min)

Journal: Have students respond to the following questions:

- If you had 1 million books, and you had to be able to find any book by its title as fast as possible, how would you organize them?
- How many books would you need to look at in the worst case scenario to find the title *before* you have organized the books?
- How many books would you need to look at in the worst case scenario to find the title *after* you have organized the books?

Teacher note: Having just finished the lessons on searching, students should recall that searching an ordered list allows using the binary search which is faster than searching an unordered list with a linear or random search.

Activity: Card Sorting (40 min)

Teacher note: The focus should be directed more toward the problem-solving technique than nitpicking about the language used. Although students are writing instructions for a human to manipulate a set of playing cards, they still need to be precise, because the assumption is that the person doesn't know what they are doing. This problem is challenging and will require creativity.

Explain the Problem [10 min]

- Demonstrate the card sorting sorting task as you explain.
- Clarify the goal: Today, you and a partner are going to design an algorithm and list the instructions for a person to arrange a row of playing cards into order (from lowest to highest value).
- Explain the basic rules:
    - If a card is on the table, it must be face down.
    - You can only see the value of a card by picking it up and looking at its face.
    - You can only be holding and looking at two cards at a time (1 in each hand).
    - You can compare the values of any of the cards you are holding in your hands and determine if one is greater, less than, or equal to the other card.
    - When you put a card down, try to be clear about *where* it should be put back down. Cards should be put face down.
    - You cannot use your memory of face-down cards to make decisions about them. You should behave as though you have no recollection of cards that you aren't currently holding.
    - You will have eight cards to practice, but the procedure you follow should be general enough to work for *any number* of cards.
- Ask the students whether there are any questions about the rules.
- Suggest that it could be helpful to break the task down into parts. Using abstraction and collaboration can decrease the size and complexity of the task that each programmer has to solve. Abstraction allows you to build upon existing processes, collaboration allows you to break up the task once you agree on what the strategy is and what the separate tasks are that need to be solved. Example, cards need to be comared and swapped, one person could write the specific steps for how that should be done.

- Emphasize to students that there are *many* ways to achieve this task. Be creative. As a class, they should try to come up with as many different ways of sorting as possible.

Paired Activity: Algorithm Creation [30 min]

- If there is an odd number of students, there could be one group of three students.
- Distribute the cards to student pairs. The cards can be ordinary playing cards, but each pair should receive cards from the same suit that have been shuffled. Alternatively, you can use handmade cards with arbitrary numbers on them. Before starting, have students agree on the ordering of the cards (e.g., whether aces are high or low).
- Have students write their instruction list (algorithm) on a piece of paper.
- The format of the instruction lists is up to the students; they can create a numbered list, a flow chart, a diagram with text and arrows, or other means of communication.
- Students should be working productively for the rest of the class: designing and writing their card sorting algorithm.

- Circulate around the room to make sure that students are on task and that they understand the rules and goals of the activity.

Wrap Up (5 min)

Journal:

- If you were to give your algorithm a name that describes how it sorts, what would you name it?
- Identify the most difficult part of writing down the instructions for your algorithm.

Homework: Any pairs that did not finish the activity should complete it as a homework assignment before the next session.

Session 2

In this session, students review the sorting algorithms they wrote in session 1. Students will follow the algorithms created by their classmates and discover a variety of sorting strategies. By analyzing the various algorithms, students will attempt to find the "best" sorting strategy.

Teacher note: There are two main difficulties in algorithm design to highlight: (1) It is very difficult to be precise with language without some agreement about what terms mean. (2) Solving the problem by determining the strategies and steps required to sort objects correctly, as well as efficiently, presents a second level of difficulty.

Getting Started (5 min)

Journal: How do you think a sorting algorithm should be "measured" to determine if it is the "best"? Ask for ideas and discuss this during the group activity.

Group Activity: Algorithm Evaluation (20 min)

- Distribute the "Sorting Algorithm Evaluation Student Worksheet.doc" (in the lesson folder).
- Discuss: How will you determine which algorithm is better and why? How do you define “better”? What criteria will you use?Give students an opportunity to respond individually and collectively. Optionally, you may use a think-pair-share approach or small groups to develop ideas and then share with the class. Solicit student ideas and try to agree of 4 criteria to write down on page 2 of the student worksheet.
  1. Does it work correctly? (correctness)
  2. Is it well written and easy to follow?
  3. Is it efficient in terms of time? Is it efficient only under certain conditions?
  4. Is it efficient in use of space? Do you need a lot of extra table room as temporary places to arrange the cards in the process of sorting?
- Re-distribute playing cards to student pairs.
- Instruct students to follow directions on the Sorting Algorithm Evaluation worksheet and use it to record their experiences.
- At the end of the session be sure to point out:
  - An efficient algorithm for a problem can help solve larger instances of the problem.
  - What looks more complex in code may actually be a more efficient solution.

  - Different correct algorithms for the same problem can have different efficiencies.
  - An advantage of doing this collaboratively is that it can decrease the size and complexity of the task that each programmer has to solve.

Group Activity: Algorithm Selection and Justification (20 min)

Create groups of four by joining the pairs who previously exchanged algorithms.

Teacher note: This "swapping algorithm" activity works especially well when students exchange algorithms with a group that has a fundamentally different approach. However, as a practical matter, this can be hard to arrange. From your observations during Session 1, you might have a sense of groups with different approaches that you can assign to swap algorithms.

- Groups should give feedback to each other about the algorithm, making sure to discuss:
  - The algorithm's correctness - does it work; do you understand it; could you simulate it or act it out?
  - Explain confusing/ambiguous parts of the other group's algorithm in order to help them write it better.
  - Discuss which of the two algorithms is "better" and be able to explain why.
  - Each group of 4 will nominate one of the two algorithms as the better one of the group.

- Ask each group to volunteer the better algorithm at their table.
- Act out/simulate one group's instructions.

Teacher note: It is possible that the nominated algorithm won't work perfectly. If you encounter any problems with the directions, give them the benefit of the doubt and simulate it as best you can to enable the class to understand the intent.

- Ask if other groups solved the problem with a different strategy and demonstrate a few groups' algorithms.
- Engage students in a discussion about which algorithm was the best. Why is it best? How should algorithms be evaluated?
- Ask students to argue for one method or another and explain their reasons.

Teacher note: The students will perform an actual analysis in a later lesson, so it's okay at this point to simply guide the discussion to see how students are thinking. They will re-examine these ideas later.

- Point out to the students that the speed at which the actor can follow the algorithm is not a measure of the algorithm, but rather the speed of the person. As an analogy, you might compare a person's walking speed with the distance they have to travel. To compare algorithms, you need to measure some "units of work". What those units are is debatable, but must to be agreed upon. For card sorting, "work" could mean picking a card up, putting it down, comparing with another card, etc.

Wrap Up (5 min)

Remind students of the two main issues in writing effective algorithms:

- The need for a clear, unambiguous language for expressing algorithmic solutions.
- Defining criteria for determining whether an algorithm is "good."

Homework: Assign students to write a final version of their algorithm, working out any ambiguities or other problems revealed during the activities.

Session 3

In this session, we end the set of sorting activities by relating sorting to algorithms in the real world. A further exploration of algorithm analysis with some new algorithms will sharpen their intuition about what should and shouldn't be "counted" when analyzing algorithms, what is "hard" for a computer to do, or what takes a "long time."

Getting Started (10 min)

Journal: discuss ideas and elements from the previous lesson:

- What is "work" for a computer?
- Why would the efficiency of an algorithm be measured in terms of both speed and amount of "space" (memory) required?
- Why does it matter how an algorithm (or program) will be used to decide if it is an appropriate solution?

Teacher should clarify that in general, we want these things from an algorithm:

- To provide a correct solution for any given input.
- To use computational resources as efficiently as possible.
- The suitability of a solution depends on how it will be used. Some sorting algorithms work best in terms of time or memory used, others excel at particular situations like sorting a list that is already mostly sorted but work poorly on lists that are badly out of order.

Guided Activity: Algorithm Analysis (10 min)

- Go to the website that shows folk dancers simulating various sorting algorithms. https://www.youtube.com/user/AlgoRythmics/videos (https://www.youtube.com/user/AlgoRythmics/videos)
- With the students, click on the bubble sort video, and complete Sorting Algorithm Evaluation.docx for bubble sort, helping the students to identify when a "comparison" is occurring and when a "swap" is occurring.
- Play it again and help the students write the pseudocode for the bubble sort algorithm.
- Compare the students' algorithms to solutions in C++ (https://mathbits.com/MathBits/CompSci/Arrays/Bubble.htm (https://mathbits.com/MathBits/CompSci/Arrays/Bubble.htm)) and MatLab (https://www.mathworks.com/matlabcentral/fileexchange/45125-sorting-methods/content/Sorting%20Methods/bubblesort.m (https://www.mathworks.com/matlabcentral/fileexchange/45125-sorting-methods/content/Sorting%20Methods/bubblesort.m)). What features of the bubble sort are identifiable regardless of language? Point out that the language may change the readability of the solution, but that it can be done in any programming language because all provide sequence, conditionals and iteration.

Paired Activity: Algorithm Analysis (20 min)

- Assign each pair a different sorting algorithm.

- Have each pair watch their assigned sorting video and complete the Sorting Algorithm Evaluation for that method.
- Have each group attempt to write pseudocode for their assigned sorting algorithm. Some of the algorithms are quite complex, so emphasize to the students that the goal of this exercise should be to think about what's happening, rather than to get it completely "right."
- Compare this algorithm to the one you developed.
- What basic steps do the algorithms have in common? How can abstraction make it easier to solve similar problems by building on existing solutions?

Wrap Up (10 min)

- Instruct students to discuss the following prompts with an elbow partner and then, collaboratively write a response in their journals that incorporates the ideas of both partners.
  - What should and shouldn't be "counted" when analyzing algorithms?
  - What is "hard" or time consuming for a computer to do?
  - Why is the efficiency of algorithms important?
- Assign homework for next lesson on comparing algorithms (Unit 5, Lesson 4): Identify in your journal two places that you often travel between. Of the alternative routes available, what do you consider to be the best route? Why? Are there circumstances in which an alternate route is better? When is that the case?

Options for Differentiated Instruction

Suggestion: If you have a mix of new and advanced students, challenge the advanced students to sort twice as many cards with a parallel processing algorithm of their own design. Each student on the team can perform one action at the same time.

Evidence of Learning

Formative Assessment

Evaluation of algorithms

Convert actions into an algorithm

(http://www.umbc.edu/) (http://www.umd.edu/)

(http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)