Lesson 7.02 — Sorting Algorithms

Overview

Objectives — *Students will be able to...*

- **Compare and contrast** different sorting methods and **evaluate** their relative speed and efficiency.

Assessments — *Students will...*

- **Complete** some short answer questions on worksheets

Homework — *Students will...*

- **Read** BJP 13.1 "Shuffling." (**Check Differentiation for advanced homework assignment and alternate classroom activities**)

Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **CS Unplugged Activity:** Sorting Algorithms (http://csunplugged.org/sorting-algorithms)
- **Balance scales** (one for each group of students)
- **Classroom copies** of worksheet activities (included in CS Unplugged Activity)
- **Sets of 8 containers** of equal size but different weights (one for each group)
- **Student pair/group assignments**

Your physics, mathematics, or chemistry teacher may have weight sets and scales that you can use for this lesson. If you do not have weight sets, using prescription bottles, M&M Mini containers, or old film canisters filled with different amounts of sand will also work. If you do not have time to collect entire sets of these items, placing pennies, beans, or sand in sealed paper lunch bags will work in a pinch (warning: this can get messy if your students aren't careful!). It's important to choose opaque containers so students don't take the shortcut of eyeballing the weights instead of using the scale.

If you cannot find balance scales, directions on how to make a set can be found on these websites:

- https://kriegerscience.wordpress.com/2011/09/28/how-to-make-a-set-of-weighing-scales/
- http://www.wikihow.com/Make-a-Balance-Scale-for-Kids

Pacing Guide: Day 1

Section	Total Time
Bell-work and attendance	5min
Introduction	10min
Student activity 1: Sorting Weights	35min

Pacing Guide: Day 2

Section	Total Time
Introduction & Setup	5min
Student activity 2: Divide and Conquer	30min
Whole group discussion & code demo	10min

Procedure

Hook your students by placing the materials for the lesson out around the room.

Bell-work and Attendance [5 minutes]

Introduction [10 minutes]

1. Lead a class discussion about why it would be important to sort data:
    - Ask students which searching method went faster yesterday (binary search), and what a prerequisite for this type of search was (array needs to be ordered/sorted).
    - Ask for additional examples of when it might be important to sort data. (Sorting songs in iTunes, sorting homework assignment files on your home computer, sorting goods in a store by location, price, or type, etc.)

Student Activity 1: Sorting Weights [35 minutes]

1. Distribute worksheets for "Sorting Weights" to student pairs or groups.
2. Review the directions with the class, and distribute materials.
    - If your class is not familiar with how to use the scales, take a few minutes to demonstrate the proper way to read the scales and record the weight data.
    - Be sure to remind students to count how many comparisons they make during the activity!
    - Make the "extra for experts" question mandatory.
3. Give students 35 minutes to complete this activity—they will need to make 28 comparisons for this exercise.

DAY 2

Student Activity 2: Divide and Conquer [30 minutes]

1. Briefly review the next activity for the "Divide and Conquer" activity.
    - Point out to students that they must try out the additional sorting methods on the last page (insertion sort or bubble sort).
    - Remind students to keep track of how many comparisons they make for each type of sort.
2. Give students ~30 minutes to complete this activity, then call the class together for a whole group discussion of their answers.

Whole Group Discussion & Code Demonstration [10 minutes]

1. Discuss the worksheet questions as a class; ask students:
    - what sorting algorithms were the quickest
    - which sorting algorithms should use more memory
    - which sorting algorithms they've used in their lives, and what those situations were. (Some examples to get them started include sorting notes or homework into date order, organizing shuffled cards, etc.)
2. The AP subset does not include specific sorting code, but students are required to conceptually understand the mechanisms used in each of the sorting algorithms. Students should be familiar with the best- and worst-case scenarios for each of the sorting algorithms. Give the students a summary of this information by having students copy the notes below, or by offering this information as a handout.

SORTING ALGORITHMS

Selection Sort For an array of n elements, the array is sorted after n-1 passes. After the ith pass, the first i elements are in their sorted position.

The steps are:

1. The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list.
2. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list.
3. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

A useful animation of selection sort can be found on Wikipedia here: (http://tinyurl.com/muo8ycd).

Insertion Sort For an array of i elements, the array is sorted after n-1 passes. After the ith pass, the first i elements are sorted with respect to each other, but not in their final sorted positions. The worst-case for insertion sort is when the array is initially sorted in reverse order (because sorting takes the most number of comparisons/moves). The best-case scenario is if the array is initially sorted in increasing order (because no elements will have to be moved).

The steps are:

1. Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list.
2. Each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there.
3. It repeats until no input elements remain.

A useful animation of insertion sort can be found on Wikipedia here: (http://tinyurl.com/ldw8bj6).

Comparison of Sorting Algorithms Going indepth on every sorting algorithm is beyond the scope of this course. This is an introduction on comparing sorting algorithms.

1. Start a discussion on the insertion sort algorithm above on how would you measure the amount of resources used. Lead the students to: execution time, memory, number of swaps.
2. What would the resources used be if the array was in reverse order for the selection sort vs insertion sort? What if the array was already sorted? What is the average execution time? Algorithms are classified by the worse case execution time. For both selection and insertion sort, they are n-squared algoritms.
3. Introduce the concept of non-n squared algorithms in the following video: https://www.youtube.com/watch?v=ZZuD6iUe3I
   A faster sorting algorithm, Merge Sort, will be introduced in the recursion Unit.

Accommodation and Differentiation

If your students need more time to complete the exercise, have them skip the bubblesearch exercise.

Have several shuffled decks of cards available for students to sort if they finish the exercise early. You can time student sorting and turn it into a competition if you need to offer extra engagement/motivation.

Encourage students to write the algorithms for different sorting methods in their own words. If a student comes up with a particularly well-worded example, encourage them to turn it into a poster or handout for the entire class to use.

If you have enough time, and your students are following along well, have your students practice implementing sorting code. Use AP questions (like #17 in the 2009 exam) to give students practice at recognizing a right-to-left selection sort and choosing the correct inner for loop.

Optional Lesson

An **alternative lesson plan** is outlined here. This lesson examines the sorting methods in greater detail, and takes at least 2 class periods to conduct. If you are pressed for time, you should not use these plans since they take more time to cover, and address the material in a more challenging way (though the activities are cleverly disguised as fun)!

1. Introduce these activities in lieu of the ones mentioned in the lesson plan above.
    - Bring 8 students to front of classroom, give each a piece of paper with a unique simple number on it, which they hold so class can see.
    - Pick "SortMasters" from the remainder of the class to give the 8 students instructions about how to get into smallest-to-biggest order.
    - Allow the first two SortMasters to give instructions without commentary, unless it looks like they are explicitly using selection sort (move lowest/biggest number to one end) or insertion sort (move a number to correct position) or bubble sort (A and B are in wrong order, so swap), in which case name what they are doing and encourage them to be explicit in describing their instructions.
    - In this first activity, do not worry about explicit comparisons of two numbers at a time, SortMasters can pick smallest/largest directly.
    - If the first two SortMasters have not shown insertion or selection sort, tell the class that they're going to look at this more systematically/algorithmically now.
        - If selection sort not covered– ask a new SortMaster who should go first? Who should go second (selection sort)?
        - If insertion sort not covered – ask a new SortMaster to start with asking whether second person should go before or after first? Where third person should go relative to first and second?
    - Students back to seats; ask for a comparison of selection vs. insertion sort:
        - Selection: Who goes in the next position?
        - Insertion: Into which position does the next person go?
2. **Activity 1A:** Introduce idea that while SortMasters could eyeball everyone to see who goes next, computers can only compare two things at a time.
    - Distribute balances and weights. Instruct each team to start with weights 1-8 in order. Activity 1A is essentially same as Activity 1, except you explicitly tell students to use the idea of Selection Sort when ordering their weights.
    - Students discuss questions in their groups, answering questions including number of comparisons.
3. **Activity 1B:** Students then move on to Activity 1B, same as 1A, except examining insertion sort. Students should be explicitly told to reset their weights in initial order.
    - As a trick, set up initial weights for most groups in random order, but at least one group should have initial positions in ascending weight order, and another with weights in descending weight order.
    - Come back to class discussion. Expect groups to have similar numbers of compares for selection sort, but widely different numbers for insertion sort.
    - As a "grand reveal," announce the different start conditions, so students will discover this significant behavior difference between selection and insertion.

Teacher Prior CS Knowledge

- Many students learn bubble sort as one of the first sorting algorithms. Given bubble sorts large amount of swapping in the worst case, it is a terrible sorting algorithm. However, if you ask students to form a line creating a random list and then to get in size place order, the algorithm they usually use is some sort of bubble sort where each student is compared to the student next to them.
- When given a deck of random cards and asked to sort them, students usually come up with either selection sort or insertion sort. However, the selection sort used to sort the random cards is different from the actual algorithm used. Selection sort swaps the next smallest (or largest) item in the list. However, when students select the next smallest item, they insert it in the cards already sorted and shift the remaining cards down. You can implement a selection sort in this manner, it takes a much larger number of swaps.

Misconceptions

Selection sort requires the use of indexes in an array in order to keep track of the next lowest (or highest) item. This requires students to think in one more level of abstraction. Often times students will start with saving the number and then realize they need the index. What they end up with is two variables, one for the saved number and one for the index.

Video

- BJP 13-2, *Sorting* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c13-2
- CSE 143, *Selection Sort* (4:43:14–8:50) https://www.youtube.com/watch?v=CTDtbbCKmSY&start=283
- CSE 143, *Insertion Sort* (9:26–12:00) https://www.youtube.com/watch?v=CTDtbbCKmSY&start=566
- CSE 143, *Bubble Sort* (optional) (14:04–16:18) https://www.youtube.com/watch?v=CTDtbbCKmSY&start=844

Forum discussion

Lesson 7.02 Sorting Algorithms (TEALS Discourse account required)