Lesson 8.04 — MergeSort

Overview

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

- **Use** mergeSort to sort an ArrayList.
- **Use** recursion to traverse and Array.

Assessments — ***Students will...***

- **Build** a merge algorithm to be applied in mergeSort.

Homework — ***Students will...***

- **Summarize** notes for notebook check tomorrow

Materials & Prep

- **Projector and Computer**
- **Whiteboard and Markers**
- **Electronic survey** for student review requests

The homework tonight asks students to submit 2 questions for review; the number is reduced for this until since there is far less material than usual. Create an electronic survey for students to complete with 3 text fields, one for name, and 2 for questions they have about Ch. 12 content. Set a deadline by which time students must have submitted 2 questions from Ch. 12 If students do not have questions, stipulate that they still have to submit something to receive credit, even if it is only questions they think other students may have.

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and homework distribution	5min
Student work	35min
Students trade work, check, and submit	10min

Procedure

Today we will return to sorting and cover MergeSort. MergeSort acts recursively; it takes a list, splits it down into single elements, and compares the elements, merging them together in an orderly fashion. The first part of the lesson covers merge as it's conceptually easier, the second part of the lesson covers the splitting of the list.

Bell-work and Attendance [5 minutes]

Introduction to MergeSort [5 minutes]

1. Begin your lecture by bringing the class back to selection and insertion sort. Make the case that selection and insertion sort both work, but take more time for your computer to process (graph in slides).
2. Show your class an example diagram of mergeSort; they should realize (with your help) that the worst case performance for mergeSort is faster than either selection or insertion sort.

Merge [15 minutes]

1. Invite your class to try and merge two sorted lists of numbers. How would we compute a single sorted list containing all the numbers in list 1 and list 2? Have your class set up the pseudocode.
   - We must maintain an index for each list starting at 0.

    - We must create and empty list to hold the result.
    - When we haven't exhausted our two lists, insert the smallest element at the point in the new list and advance the index.

2. Let your class work for 10 minutes to try and create the merge function. Using the responses they come up with, lead them to the correct merge function. (Shown on slides, using `ArrayList<Integer>`)

MergeSort [30 minutes]

1. Challenge your class to try and implement merge into a sorting algorithm. Those who have done the reading should have an idea about where to get started, but may get stuck on the recursive portion.

   You can give them these instructions to get started:

    1. If the list's size is 0 or 1, just return the original list (as it is already sorted)
    2. Split the list parameter into two lists, of (roughly) equal size: list 1 and 2.
    3. Sort both list 1 and 2.
    4. Merge the two sorted lists, and return the result.

2. For lab today, your class will be tasked with making this mergeSort algorithm. At the end of class post the solution up on the board.

   Provide hints throughout the process, namely:

    1. When the list's size is 0 or 1 and you return the original, that is your base case.
    2. Split your lists based off ArrayList.size().
    3. The merge function does merging.

3. If the class is struggling, walk through the entire mergeSort algorithm with them, mergeSort is covered by the AP so a base level of understanding is important.

Accommodation and Differentiation

In ELL classrooms, pair students and allow them to work together to correct their work. If you noticed a particular problem was difficult for the majority of students, read the question aloud and help students work through it.

For those students who have nothing to correct (or finish very early), reward them with silent free time, or allow them to work on a free-choice programming project.

Video

- CSE 143, *Merge Sort* (12:09–14:04) https://www.youtube.com/watch?v=CTDtbbCKmSY&start=729
- CSE 143, *Merge Sort* (20:50–26:15) https://www.youtube.com/watch?v=CTDtbbCKmSY&start=1250

Forum discussion

Lesson 8.04 MergeSort (TEALS Discourse account required)