Sorting Presentation

Luke Burdette | Group 7

Results

- Score: group 7 took place 5. The sum of places is 10, the sum of medians is 21569.0
  - Dataset 1: 1252, 1369, 1246, Median: 1252.0
  - Dataset 2: 20012, 20216, 20088, Median: 20088.0
- There were no reported correctness issues

Algorithm

- Takes in array of strings from file
- Runs sort method calling sort(array, array start index, array end index)
- Splits array into chunks of equal small size (can be changed), and compares the items in each chunk and sorts them
  - I used size of 6
  - For comparison uses the default given comparator
- Sorts the chunks overall on the second pass
- Overwrites the sorted chunks into the original array
  - Performed in place so does not require more space
- This is a quicksort with a pivot

First Comparison Chunk

```
for (int chunk = fromIndex; chunk < toIndex; chunk += 6){
    int end = Math.min(chunk + 6, toIndex);
    for (int i = chunk + 1; i < end; i++){
        if (s.compare(a[i - 1], a[i]) > 0){
            int j = i;
            String elem = a[j];
            do{
                a[j] = a[j - 1];
                j--;
            }while (j > chunk && s.compare(a[j - 1], elem) > 0);
            a[j] = elem;
        }
    }
}
```

Efficiency

- Uses 2 nested for loops for chunk comparison
- Uses 2 nested for loops for second comparison
  - Both of these have an efficiency of $O(n^2)$
- Uses recursion and checks each item
  - Could be improved by changing sort type if recursion goes too deep
- Uses comparator
- Worst case/Expected efficiency is $O(n^2)$
- Best case is O(n) for already sorted array

Features

- Most Interesting part
  - The sorting algorithm uses recursion with multiple constructors
- What Worked
  - Algorithm correctly sorted data in all cases
  - Slightly more efficient than base sorting algorithm
- What Didn’t / Would Change
  - Did not have much prior knowledge of BigInteger
  - Algorithm is general, could be more specific to increase efficiency in this case
  - Spend more time learning about optimizing instances of objects and more efficient sorting methods (more time in general to work on it)
  - More refactoring of initial methods

Questions