

COMPUTER PROGRAMMING IN JAVA

COLUMBIA UNIVERSITY HIGH SCHOOL SCIENCE HONORS PROGRAM

2007 Oct 27 Sat, Week 05

1. Review

Asymptotic Analysis and Big-Oh notation.
Sequential and Binary Search.
Insertion Sort and Selection Sort.

2. More Insertion Sort and Selection Sort.

Best-case performance of IS: $O(n)$. Why?
Worst-case performance of IS: $O(n^2)$. Why?

Best-case performance of SS: $O(n)$. Why?
Worst-case performance of SS: $O(n^2)$. Why?

3. Bubblesort

The simplest sort:

```
public static int[] sort(int[] array)
{
    for (int i = 1 ; i < array.length; i ++)
    {
        for (int j = 1; j < array.length; j ++)
        {
            if (array[j - 1] > array[j])
            {
                // swap
                int temp = array[j - 1];
                array[j - 1] = array[j];
                array[j] = temp;
            }
        }
    }
    return array;
}
```

4. MergeSort

So we learned some $O(n^2)$ time sorting algorithms, a natural question arises: can we do better? In fact, we can: MergeSort has a guaranteed $O(n \lg n)$ worse-case runtime. MergeSort uses recursion.

I've left some space so you can draw the diagrams if you want.

5. Assignments

Notes:

- I've provided an entire framework for you to work within. The main class is “Sorting”. If you look inside, you will see a whole bunch of code. What this class does is it uses other classes (InsertionSort, SelectionSort, etc) to sort a dataset which the Sorting class loads for you.
- Each of the sort classes (InsertionSort, SelectionSort, BubbleSort, and MergeSort) each has one method that is used by the Sorting class. The name of this method is “sort”. All of the “sort” methods in each of the classes are the same: they take in an array, and they return an array that's sorted. It doesn't matter if you modify the original array or if you return a copy of the array.
- What you should do is implement each of the different kinds of sorts we talked about in its respective class.
- A good way to begin would be to copy and paste the bubblesort algorithm from above into the BubbleSort class, and then you can see the results of one of the sorts.
- There is a variable in Sorting.java class “fileName”. You should change this to match the name of the dataset file (e.g. “small1.txt” or small2.txt”, etc).
- MergeSort is hard. A class has been provided for you, which implements most of the algorithm, except for the merge, which currently does nothing. Your job, should you choose to accept, is to complete the merge algorithm.
- The times printed are very misleading: the first sort will always take long, because it takes time for the computer to “warm up” (like a car) each time the program is loaded. See if you can figure out how to modify Sorting.java so that it runs each sort a bunch of times (say 10 times), and calculates the average time.
- Important note about datasets: The first line does not mean anything! It is simply the number of values in the list.