

COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Small Group Worksheet – 09.25.14
25 minutes

Instructions: You can work individually or as a small group (2-3) to solve these problems.

1. For an array of N items, in the best case, how many compares will Mergesort use in tilde notation? Give a very succinct explanation for what constitutes the “best case”.
2. In the worst case, how many compares will Mergesort use in tilde notation?
3. In the best case, how many array accesses does Mergesort use in tilde? (The answer is less than $\sim 6N$)
4. Why does the Mergesort code use `less(a[i], a[j])` instead of `a[i] < a[j]`?
5. Tough problem: Which of the following two loops is more likely to be faster? Why?

<code>for (int i = 0; i < N/4096; i++)</code>	<code>for (int i = 0; i < N; i += 4096)</code>
<code>sum += a[i]</code>	<code>sum += a[i];</code>
6. What was the point of teaching you bottom-up mergesort if it is always worse?
7. Is comb sort (http://en.wikipedia.org/wiki/Comb_sort) stable? If so, why? If not, give a counterexample.
8. Give two reasons why you might use a comparator instead of a built-in `compareTo` method?
9. Consider the following Comparator, which is an inner class of a class called student:

```
private STATIC? class ByName implements Comparator<Student> {  
    public STATIC? int compare(Student v, Student w)  
    { return v.name.compareTo(w.name); }  
}
```

Under what circumstance would you want to remove the first static keyword? You might find <http://algs4.cs.princeton.edu/25applications/Student.java> useful. Why can you NOT remove the second static keyword?
10. Let's consider the problem of lower bounding the number of exchanges for exchange-based sorting: Given a single exchange, what is the maximum number of inversions that we can fix at once?
11. Give an exchange-based sort algorithm that uses the minimum number of exchanges.

12. Is it possible for Merge Sort to behave as a quadratic algorithm for some bad set of inputs?
13. Given an array of size N , do you expect a partition operation or a merge operation on that array to be faster? Why?
14. Rewrite the insertion sort code below w/o using any exchanges

```
public static void sort(Comparable[] a)
{
    int N = a.length;
    for (int i = 0; i < N; i++)
        for (int j = i; j > 0; j--)
            if (!less(a[j], a[j-1]))
                exch(a, j, j-1);
            else break;
}
```