CS 577: Introduction to Algorithms — Homework 1

Out: 02/01/18 — **Due: 02/08/18**

Ground Rules

- The homework is worth 5 points (out of a total of 100 points you can accumulate in this course).
- The homework is to be done and submitted in pairs. You can partner with someone from either section.
- The homework is due at the beginning of either lecture on the due date.
- No extensions to the due date will be given under any circumstances.
- Turn in your solution to each problem on a **separate sheet** of paper (or sheets stapled together), with your names clearly written on top.

Problems

1. **(Worth: 2 points. Page limit: 1 sheet; 2 sides)** You are given $n = 2^k$ coins, all of which look identical. However, one of the coins is defective – it weighs either slightly more or slightly less than the rest (you don't know which). You also have at your disposal an electronic equivalence tester, which has two compartments. You may place any set of objects in each compartment; the tester tells you whether or not the two sets weigh the same. Note that the tester does not tell you which side is heavier and which one lighter – only whether they weigh the same or not. Your goal is to determine which of the coins is defective using the tester at most k times. You may assume that $k > 1$, that is, $n > 2$. (Is it possible to find the defective coin when $n = 2$?) Note that you are allowed to use the tester only k times, not $O(k)$ or k plus a few more times. You will receive partial credit for slightly worse solutions.

   (a) Describe a divide and conquer algorithm for this problem.

   (b) Argue that your algorithm uses the tester at most k times.

   (c) Prove the correctness of your algorithm, in other words that it always correctly identifies the defective coin.

(*See next page for problem 2*)

2. **(Worth: 3 points. Page limit: 2 sheets; 3 sides)** Assume that n is a power of two. Consider the following sorting algorithm:

```
SLOWSORT(A[1..n])
 1: if n > 1 then
 2:     SLOWSORT(A[1 .. n/2])
 3:     SLOWSORT(A[n/2+1 .. n])
 4:     GATHER(A[1 .. n])
 5: end if
```

```
GATHER(A[1 .. n])
 1: if n = 2 then
 2:     if A[1] > A[2] then
 3:         swap A[1] ↔ A[2]
 4:     end if
 5: else
 6:     for i ← 1 to n/4 do
 7:         swap A[i+n/4] ↔ A[i+n/2]
 8:     end for
 9:     GATHER(A[1 .. n/2])
10:     GATHER(A[n/2+1 .. n])
11:     GATHER(A[n/4+1 .. 3n/4])
12: end if
```

(a) Give an example proving that SLOWSORT would not correctly sort if we removed lines 6-8 in GATHER.

(b) Give an example proving that SLOWSORT would not correctly sort if we swapped lines 10 & 11 in GATHER.

(c) Prove by induction that SLOWSORT correctly sorts any input array (of size a power of two).
(*Hint*: Formulate a claim for what GATHER is supposed to accomplish, and prove it.)

(d) Write a recurrence to count the number of comparisons made by GATHER on an array of size n. Write a recurrence to count the number of comparisons made by SLOWSORT on an array of size n. Solve these two recurrences to obtain an asymptotic expression for the number of comparisons done by GATHER and by SLOWSORT. No need to show your calculations.