# MA/CSSE 473 – Design and Analysis of Algorithms

## Homework 6B (64 points total) Updated for Winter, 2017

### Problems for enlightenment/practice/review (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

| | |
|---|---|
| 4.1.8 [5.1.5] | (insertion sort sentinel) |
| 5.1.12[5.1.10] | (Shell's sort) This should be review from 230 |
| 4.2.1 [5.3.1] | (Topological sort examples) |
| 4.2.2 [5.3.2] | (Theoretical properties of topological sort) |
| 5.2.1 [4.2.1] | (quicksort example) |
| 5.2.4 [4.2.4] | (quicksort sentinel) |
| 5.2.6 [4.2.6] | (increasing arrays in quicksort) |

### Problems to write up and turn in:

1. ( 6) 3.5.3 [5.2.3] (independence of properties from specific DFS traversals) Explain your answers.

2. (10) 3.5.8a [5.2.8a] (Bipartite graph checking using DFS)

3. ( 5) 4.1.1 [5.1.1] (Ferrying Soldiers)

4. ( 5) (not in 3rd) [5.1.9] (binary insertion sort efficiency).

   Binary insertion sort uses binary search to find the appropriate position to insert A[i] among the previously sorted A[0] ≤ … ≤ A[I - 1]. Determine the worst-case efficiency class of this algorithm. I.e. get big-Θ time for number of comparisons and number of moves.

5. ( 9) 4.2.6 [5.3.6 ] (finding dag sources) Be sure to do all three parts.

   In the 3rd edition, it says "Prove that a **nonempty** dag must have at least one source." That additional word is necessary!

6. ( 9) 4.2.9 [5.3.9] (Strongly connected components of a digraph)

**Problem 7 is on the next page.**

7. (20) (Miller-Rabin test) For this problem you will need the excerpt from the Dasgupta book that is posted on Moodle, and/or Weiss section 9.6.
Let N = 1729 (happens to be a Carmichael number, but you should not assume that as you discover the answers) for all parts of this problem.

(a) How many values of **a** in the range 1..1728 pass the Fermat test [i.e. $a^{1728} \equiv 1 \pmod{1729}$]?

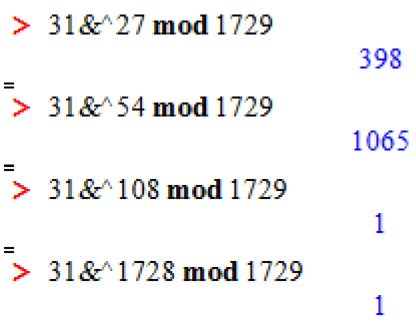
(b) For how many of these "Fermat liar" values of **a** from part (a) does the Miller-Rabin test provide a witness that N is actually composite?

(c) If we pick **a** at random from among 1, 2,..,1728, what is the probability that running the Miller-Rabin test on **a** will show that N is composite? I.e., Rabin showed that for any N, the probability is at least 75% for every N; what is that probability for the N=1729 case?

[**Hint**: writing some code is likely (probably necessary) to help you solve this problem. If you do that, include the code in your submission. If not, explain how you got your answers].

For part (b), I want to demonstrate a number **a** that fits the description of the numbers that you are supposed to count. It is **a**=31. I discovered how to do fast modular exponentiation in Maple, so I illustrate that here as well. You can also use the **modexp** Python code that I gave you, or use Python's **pow** function:

```
>>> pow(31,54,1729)
1065
```



```
> 31&^27 mod 1729
                                398
> 31&^54 mod 1729
                                1065
> 31&^108 mod 1729
                                1
> 31&^1728 mod 1729
                                1
```

Notice from the last line that 31 is a Fermat liar for 1729, since 1729 is composite but the 1728th power of 31 is congruent to 1 mod 1729.

In the Miller-Rabin test, we start with the 27th power, and keep squaring until we either get to the 1728th power or we find a non-trivial square root of 1 (mod 1729). In this case the latter happens; we see that 1065 is a non-trivial square root of 1, and so 1729 is composite.