MA/CSSE 473 **HW 6A** Problem statements/hints

Problems 1-2 are related to Dasgupta pages 30-34 and Weiss section 7.4 (both on Moodle)

Problems 1 and 2 (copied here from the assignment document):

1. **(15)** (RSA decoding). If small primes are used, it is computationally easy to "crack" RSA codes.

   Suppose my public key is N=703, e= 53. You intercept an encrypted message intended for me, and the encrypted message is 361. What was the original message? How did you get the answer?

2. **(6)** Find and read about various ways of attacking the RSA cryptosystem.
   Write about two attacks that interest you. Write a paragraph about each one to explain in your own words how it works.

Problems 3 and 4 would have logically fit into HW 5, but were moved here to better balance out the length of the assignments. There are further clarifications on the HW6A assignment sheet

3. (8) 3.4.6 [3.4.6]

6. Consider the ***partition problem***: given n positive integers, partition them into two disjoint subsets with the same sum of their elements. (Of course, the problem does not always have a solution.) Design an exhaustive-search algorithm for this problem. Try to minimize the number of subsets the algorithm needs to generate.

(partition problem) There is only so much a brute force algorithm can do to make this efficient.
Mainly, try to avoid checking duplicate subsets or subsets that cannot possibly be a solution.

4. (6) 3.4.9 [not in 2nd edition] (non-attacking queens)

Eight-queens problem Consider the classic puzzle of placing eight queens on an 8×8 chessboard so that no two queens are in the same row or in the same column or on the same diagonal. How many ways are there so that

a. no two queens are on the same square?

b. no two queens are in the same row?

c. no two queens are in the same row or in the same column?

Also estimate how long it would take to find all the solutions to the problem by exhaustive search based on each of these approaches on a computer capable of checking 10 billion positions per second.

The real question in each case is "if we make only the given restrictions, then check each possibility to see if it is a solution, how many possibilities will we need to check?" If we made no restrictions at all (not even the restriction that multiple queens cannot occupy the same square), then there would be 64^8 placements to check. How many for each of the given cases?

Problems 5-9 relate to material that should be review from CSSE 230. In addition to the Levitin textbook, Weiss Chapter 8 should be good background for those.

5. **Problem 5:** **(3)** 5.1.4 [4.1.4] (logarithm base in Master theorem)

4. We mentioned in Chapter 2, that logarithm bases are irrelevant in most contexts arising in the analysis of an algorithm's efficiency class. Is this true for both assertions of the Master Theorem that include logarithms?

Problem 6: **(6)** 5.1.5 [4.1.5] (simple application of Master theorem)

5. Find the order of growth for solutions of the following recurrences.

a. $T(n) = 4T(n/2) + n,\ T(1) = 1$

b. $T(n) = 4T(n/2) + n^2,\ T(1) = 1$

c. $T(n) = 4T(n/2) + n^3,\ T(1) = 1$

Author's hints for problems 5 and 6:

4. Look at the notations used in the theorem's statement.

5. Apply the Master Theorem.

6. **Problem 7:** **(6)** 5.2.2 [4.2.2] (Quicksort partition scan properties)

2. For the partitioning procedure outlined in Section 4.2:

a. Prove that if the scanning indices stop while pointing to the same element, i.e., $i = j$, the value they are pointing to must be equal to p.

b. Prove that when the scanning indices stop, j cannot point to an element more than one position to the left of the one pointed to by i.

c. Why is it worth stopping the scans after encountering an element equal to the pivot?

Author's hint for problem 7: Use the rules for stopping the scans.

Problem 8: **(10) Not from the textbook.**

Show how to solve the average-case recurrence for quicksort. The recurrence is given on page 180 [133] of Levitin.
Feel free to look up a solution, understand it, and write it in your own words (and symbols). The Weiss Data Structures book (Section 8.6.2) is one place that has a solution.
You should write a reasonable amount of detail, enough to convince me that you understand it.

7. **Problem 7:** **(9)** 5.2.11 [4.2.11] (Nuts and bolts). In addition to writing the algorithm, write and solve a recurrence relation for average-case running time.

11. ▶ *Nuts and bolts* You are given a collection of n bolts of different widths and n corresponding nuts. You are allowed to try a nut and bolt together, from which you can determine whether the nut is larger than the bolt, smaller than the bolt, or matches the bolt exactly. However, there is no way to compare two nuts together or two bolts together. The problem is to match each bolt to its nut. Design an algorithm for this problem with average-case efficiency in $\Theta(n \log n)$. [Raw91]

Author's Hints:

Use the partition idea that is part of QuickSort..