MA/CSSE 473 – Design and Analysis of Algorithms

Homework 6A (68 points total) Updated for Smmer, 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.

3.5.2 [5.2.2] (adjacency matrix *vs* adjacency list for DFS)
3.5.7 [5.2.7] (Use BFS/DFS to find a graph's connected components)
3.5.10 [5.2.10] (DFS and mazes)
5.1.7 [4.1.7] (Merge sort stability)
5.1.9 [4.1.9] (O(n log n) algorithm to count inversions in an array)

Problems to write up and turn in:

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

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 of RSA was the original message? How did you get your answer? [RSA details are found in Dasgupta, and in Weiss section 7.4.4]

2. (6) (RSA attacks) 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

3. (8) 3.4.6 [3.4.6] (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 these 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 above?

Previous questions and answers from Piazza:

Are all queens treated as equal? I wonder whether all queens are treated as equal because this affects what counts as repeated solutions and what doesn't. If two queens aren't treated as equal, then placing Queen A at (x1, y1) and Queen B at (x2, y2) and Queen A at (x2, y2) and Queen B at (x1, y1) are different solutions. But if they are treated as equal, then these two are the same solution.

Instructor Answer: All queens are created equal!

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. (3) 5.1.4 [4.1.4] (logarithm base in the Master Theorem)

6. (6) 5.1.5 [4.1.5] (Simple application of the Master Theorem)

7. (6) 5.2.2 [4.2.2] (Quicksort partition scan properties) Note that the old (2^{nd}) edition of the Levitin book has a part c, and I want you to do it, you can find it in the http://www.rose-hulman.edu/class/csse/csse473/201720/Homework/hw06A_levitin_probs.pdf document.

8. (10) 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.

9. (8) 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.