MA/CSSE 473 – Design and Analysis of Algorithms

Homework 11 (72 points total) Updated for Summer, 2016

This is probably the longest assignment of the course, and it has some difficult problems. Start early!

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.

7.2.2 [7.2.2] (Horspool for patterns in DNA)
7.2.5 [7.2.5] (is there a case where Horspool does more comparisons than brute force?)
7.2.9 [7.2.9] (left-to-right checking OK after a single character match in Horspool, Boyer-Moore?)
7.3.1 [7.3.1] (insert specific keys into hash table with specific hash function and separate chaining)
8.1.1 [8.1.1] (Compare and contrast dynamic programming with divide-and-conquer)
8.1.4 [8.1.9] (Space efficiency of dynamic programming for Binomial coefficients)

Problems to write up and turn in:

1. (6) 7.2.3 [7.2.3] (Horspool for binary strings)

2. (9) 7.2.7 [7.2.7] (Boyer-Moore for binary strings)

3. (4) 7.2.8 [7.2.8] (does Boyer-Moore still work with just one table?)

4. (8) 7.2.11 [not in 2nd ed] (right cyclic shift) 3 points for part a, 5 for part b.

   You are given two strings S and T, each n characters long. You have to establish whether one of them is a right cyclic shift of the other. For example, PLEA is a right cyclic shift of LEAP, and vice versa. (Formally, T is a right cyclic shift of S if T can be obtained by concatenating the (n - i)-character suffix of S and the i-character prefix of s for some $1 \le i \le n$).
   a. Design a space-efficient algorithm for the task. Indicate the space and time efficiencies of your algorithm.
   b. Design a time-efficient algorithm for the task. Indicate the time and space efficiencies of your algorithm.

5. (5) 7.3.4 [7.3.4] (probability that n keys all hash to the same table location)

6. (6) 7.4.3[7.4.3] (minimum order of a B tree with no more than 3 disk accesses in a tree with 10^8 elements)

7. (12) 8.1.10 [not in 2nd ed] longest path in a DAG. Note that the material from Section 8.1 of the third edition of Levitin is not in the second edition. I have posted that section on Moodle in the Reading Materials section. See description on next page. Note that the hint refers to the coin-row problem, *not* the coin-collecting problem.

8. (12) 8.1.11 [not in 2nd ed] Maximum square submatrix. See description on next page.

10. *Longest path in a dag* a. Design an efficient algorithm for finding the length of a longest path in a dag. (This problem is important both as a prototype of many other dynamic programming applications and in its own right because it determines the minimal time needed for completing a project comprising precedence-constrained tasks.)

    ▷ b. Show how to reduce the coin-row problem discussed in this section to the problem of finding a longest path in a dag.

11. ▶ *Maximum square submatrix* Given an $m \times n$ boolean matrix B, find its largest square submatrix whose elements are all zeros. Design a dynamic programming algorithm and indicate its time efficiency. (The algorithm may be useful for, say, finding the largest free square area on a computer screen or for selecting a construction site.)

9. (10) 8.1.12 [8.1.10] (World Series odds) Note: In a 7-game series (such as the real American baseball World Series), the first team to win 4 games wins the series. 7 is the maximum number of games that can be played before one of the teams must win four games. But if one team wins 4 games sooner, the series ends immediately. Show your work.