MA/CSSE 473 – Design and Analysis of Algorithms

Homework 5 58 points total

When a problem is given by number, it is from the textbook. 1.1.2 means "problem 2 from section 1.1" .

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.

3.2.2 [3.2.2] (best/worst case for sequential search)
3.2.8 [3.2.9] (Substrings that begin/end with specific characters) Brute force is $\Theta(N^2)$
3.3.3 [3.3.2] (closest straight-line post office)
3.4.2 [3.4.2[(Hamiltonian Circuit)
3.4.10 [3.4.9] (Magic Squares)
3.4.11 [XXX] (Famous alphametic)

Problems to write up and turn in:

1. (14) 3.2.3 [3.2.3] (Gadget drop) If a gadget breaks, it cannot be repaired and used for another test.

   Let N be the total number of floors, and F the number of the lowest floor on which a gadget fails when dropped from there.

   Assume that due to weight, volatility, or some other factor, there is a high cost (C) for each floor that a gadget must be carried up. Also a cost (T) for each test to determine whether the drop caused the gadget to fail after each drop.

   First, give big-Theta worst-case running times in terms of N, C, and T for the obvious algorithm that tries every floor in succession (that algorithm only requires one gadget). Then design and analyze the most efficient algorithm you can.

   Can you think of any flaws in this general approach to testing?

   Points: (Algorithm/analysis: 2, efficient algorithm/analysis: 10, flaw: 2)
2. (2) 3.2.4 [3.2.4] (String matching count)
3. (5) 3.2.6 [3.2.6] (String matching worst case). Searching for a pattern of length n in a string of length m.
4. (10) 3.3.4 [3.3.3] (Manhattan distance). Clarifications: (b) You can sketch or simply list them.
   (c) By "a solution", they mean "an algorithm to solve the problem".
   Points: (a:5 , b:3, c:2).
5. (5) 3.3.7 [3.3.5] (brute-force k-dimensional closest-point algorithm)
6. (4) 3.3.10 [3.3.8] (dealing with collinear points in brute-force complex hull algorithm)
7. (7) 3.4.1 [3.4.1] (traveling salesman analysis) Points: (a:3, b:4)
8. (3) 3.4.5 [3.4.5] (simple cost matrix for an assignment problem whose optimal solution does not include smallest element)
9. (8) 3.4.6 [3.4.6] (partition problem)