473 HW 5 Levitin Problems

Problem 1: 3.2.3 (also see details in assignment document]

3. *Gadget testing* A firm wants to determine the highest floor of its n-story headquarters from which a gadget can fall with no impact on the gadget's functionality. The firm has two identical gadgets to experiment with. Design an algorithm in the best efficiency class you can to solve this problem.

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)

Problem 2: 3.2.4

4. Determine the number of character comparisons that will be made by the brute-force algorithm in searching for the pattern GANDHI in the text

THERE_IS_MORE_TO_LIFE_THAN_INCREASING_ITS_SPEED

(Assume that the length of the text—it is 47 characters long—is known before the search starts.)

Problem 3: 3.2.6

6. Give an example of a text of length n and a pattern of length m that constitutes a worst-case input for the brute-force string-matching algorithm. Exactly how many character comparisons will be made for such input?

Problem 4: 3.3.4 [3.3.3]

3. **a.** There are several alternative ways to define a distance between two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in the Cartesian plane. In particular, the so-called ***Manhattan distance*** is defined as

$$d_M(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|.$$

Prove that d_M satisfies the following axioms which every distance function must satisfy:

i. $d_M(P_1, P_2) \geq 0$ for any two points P_1 and P_2, and $d_M(P_1, P_2) = 0$ if and only if $P_1 = P_2$;

ii. $d_M(P_1, P_2) = d_M(P_2, P_1)$;

iii. $d_M(P_1, P_2) \leq d_M(P_1, P_3) + d_M(P_3, P_2)$ for any P_1, P_2, and P_3.

b. Sketch all the points in the x, y coordinate plane whose Manhattan distance to the origin (0,0) is equal to 1. Do the same for the Euclidean distance.

c. True or false: A solution to the closest-pair problem does not depend on which of the two metrics—d_E (Euclidean) or d_M (Manhattan)—is used?

Clarifications: (b) You can sketch or simply list them.
(c) By "a solution", they mean "an algorithm to solve the problem". Obviously Manhattan distance may give different nearest points than Euclidean distance. But here is the question you must answer: is the closest point algorithm itself the same for both?

Points: (a:5 , b:3, c:2).

Problem 5: 3.3.7 [3.3.5]

5. The closest-pair problem can be posed on the k-dimensional space in which the Euclidean distance between two points $P' = (x'_1, \ldots, x'_k)$ and $P'' = (x''_1, \ldots, x''_k)$ is defined as

$$d(P', P'') = \sqrt{\sum_{s=1}^{k} (x'_s - x''_s)^2}.$$

What will be the efficiency class of the brute-force algorithm for the k-dimensional closest-pair problem?

Problem 6: 3.3.10 [3.3.8]

8. What modification needs to be made in the brute-force algorithm for the convex-hull problem to handle more than two points on the same straight line?

Problem 7: 3.4.1 (traveling salesman)

1. a. Assuming that each tour can be generated in constant time, what will be the efficiency class of the exhaustive-search algorithm outlined in the text for the traveling salesman problem?

b. If this algorithm is programmed on a computer that makes 1 billion additions per second, estimate the maximum number of cities for which the problem can be solved in

i. one hour.
ii. 24-hours.
iii. one year.
iv. one century.

I told students that you can use either the 2nd or 3rd edition of the textbook. One says "one billion" additions per second"; the other says "ten billion additions per second". Use "one billion additions per second" when you do this problem.

Problem 8: 3.4.5

5. Give an example of the assignment problem whose optimal solution does not include the smallest element of its cost matrix.

Problem 9: 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.

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.

Problem 10: 3.4.9 [not in 2nd edition]

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?