CSSE 473 HW #3 Levitin problems (3rd edition) from Section 3.1. The other problems are not from the textbook.

2. **a.** What is the time efficiency of the brute-force algorithm for computing a^n as a function of n? As a function of the number of bits in the binary representation of n?

b. If you are to compute $a^n \bmod m$ where $a > 1$ and n is a large positive integer, how would you circumvent the problem of a very large magnitude of a^n?

3. For each of the algorithms in Problems 4, 5, and 6 of Exercises 2.3, tell whether or not the algorithm is based on the brute-force approach.

4. **a.** Design a brute-force algorithm for computing the value of a polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

at a given point x_0 and determine its worst-case efficiency class.

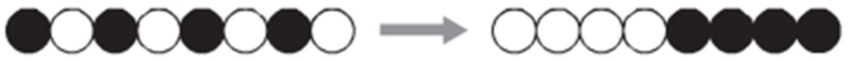
b. If the algorithm you designed is in $\Theta(n^2)$, design a linear algorithm for this problem.

c. Is it possible to design an algorithm with a better-than-linear efficiency for this problem?

9. Is selection sort stable? (The definition of a stable sorting algorithm was given in Section 1.3.)

10. Is it possible to implement selection sort for linked lists with the same $\Theta(n^2)$ efficiency as the array version?

14. *Alternating disks* You have a row of $2n$ disks of two colors, n dark and n light. They alternate: dark, light, dark, light, and so on. You want to get all the dark disks to the right-hand end, and all the light disks to the left-hand end. The only moves you are allowed to make are those that interchange the positions of two neighboring disks.



Design an algorithm for solving this puzzle and determine the number of moves it takes. [Gar99]

Author's hints on next page:

Author's hints: Section 3.1

2. a. The first question was all but answered in the section. Expressing the answer as a function of the number of bits can be done by using the formula relating the two metrics.

b. How can we compute $(ab) \bmod m$?

3. It helps to have done the exercises in question.

4. a. The most straightforward algorithm, which is based on substituting x_0 into the formula, is quadratic.

b. Analyzing what unnecessary computations the quadratic algorithm does should lead you to a better (linear) algorithm.

c. How many coefficients does a polynomial of degree n have? Can one compute its value at an arbitrary point without processing all of them?

5. For each of the three network topologies, what properties of the matrix should the algorithm check?

6. The answer to four of the questions is yes.

7. a. Just apply the brute-force thinking to the problem in question.

b. The problem can be solved in one weighing.

8. Just trace the algorithm on the input given. (It was done for another input in the section.)

9. Although the majority of elementary sorting algorithms are stable, do not rush with your answer. A general remark about stability made in Section 1.3, where the notion of stability is introduced, could be helpful, too.

10. Generally speaking, implementing an algorithm for a linked list poses problems if the algorithm requires accessing the list's elements not in sequential order.

11. Just trace the algorithm on the input given. (See an example in the section.)

12. a. A list is sorted if and only if all its adjacent elements are in a correct order. Why?

b. Add a boolean flag to register the presence or absence of switches.

c. Identify worst-case inputs first.

13. Can bubblesort change the order of two equal elements in its input?

14. Thinking about the puzzle as a sorting-like problem may or may not lead you to the most simple and efficient solution.