A first application of the brute-force approach often results in an algorithm that can be improved with a modest amount of effort.

Exercises 3.1

1. **a.** Give an example of an algorithm that should not be considered an application of the brute-force approach.
   b. Give an example of a problem that cannot be solved by a brute-force algorithm.
2. **a.** What is the 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?
5. Sort the list E, X, A, M, P, L, E in alphabetical order by selection sort.
6. Is selection sort stable? (The definition of a stable sorting algorithm was given in Section 1.3.)
7. Is it possible to implement selection sort for linked lists with the same $\Theta(n^2)$ efficiency as the array version?
8. Sort the list E, X, A, M, P, L, E in alphabetical order by bubble sort.
9. **a.** Prove that if bubble sort makes no exchanges on its pass through a list, the list is sorted and the algorithm can be stopped.
   b. Write a pseudocode of the method that incorporates this improvement.
   c. Prove that the worst-case efficiency of the improved version is quadratic.
10. Is bubble sort stable?
11. *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

The problem #s here are from Levitin 2nd edition (the numbers in brackets in the assignment document). Several of the problems in that document are not from the textbook.