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

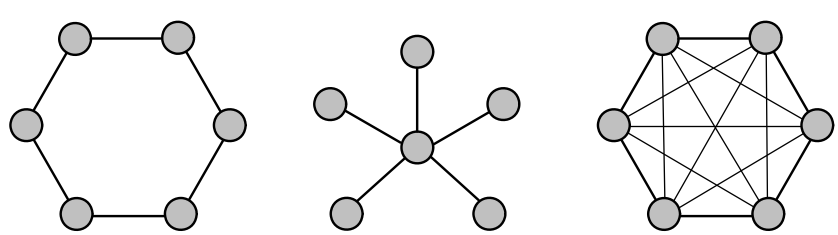
$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

   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?

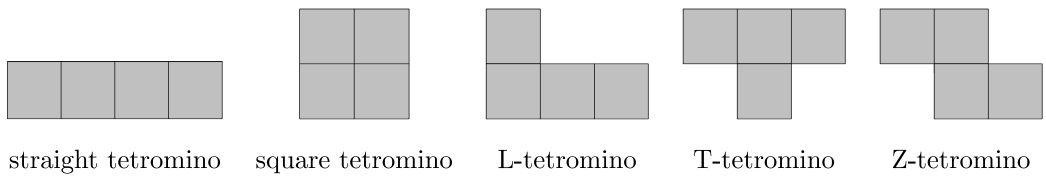
5. A network topology specifies how computers, printers, and other devices are connected over a network. The figure below illustrates three common topologies of networks: Ring, Star, and Fully Connected Mesh.



Ring   Star   Fully Connected Mesh

You are given a boolean matrix $A[0..n-1, 0..n-1]$, where $n > 3$, which is supposed to be the adjacency matrix of a graph modeling a network with one of these topologies. Your task is to determine which of these three topologies, if any, the matrix represents. Design a brute-force algorithm for this task and indicate its time efficiency class.

6. *Tetromino tilings* Tetrominoes are tiles made of four 1×1 squares There are five types of tetrominoes shown below:



   Is it possible to tile—i.e., cover exactly without overlaps—an 8×8 chessboard with

   a. straight tetrominoes? b. square tetrominoes? c. L-tetrominoes?

   d. T-tetrominoes? e. Z-tetrominoes?

7. *A stack of fake coins* There are n stacks of n identical-looking coins. All of the coins in one of these stacks are counterfeit, while all the coins in the other stacks are genuine. Every genuine coin weighs 10 grams; every fake weighs 11 grams. You have an analytical scale that can determine the exact weight of any number of coins.

   a. Devise a brute-force algorithm to identify the stack with the fake coins and determine its worst-case efficiency class.

   b. What is the minimum number of weighings needed to identify the stack with the fake coins?

8. Sort the list E, X, A, M, P, L, E in alphabetical order by selection sort.

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?

11. Sort the list E, X, A, M, P, L, E in alphabetical order by bubble sort.

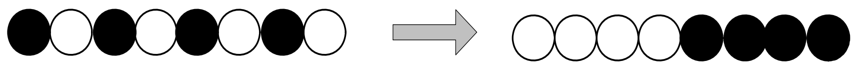
12. a. Prove that if bubble sort makes no exchanges on its pass through a list, the list is sorted and the algorithm can be stopped.

    b. Write pseudocode of the method that incorporates this improvement.

c. Prove that the worst-case efficiency of the improved version is quadratic.

13. Is bubble sort stable?

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 which interchange the positions of two neighboring disks.



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

Hints to Exercises 3.1

1. a. Think of algorithms that have impressed you with their efficiency and/or sophistication. Neither characteristic is indicative of a brute-force algorithm.

   b. Surprisingly, it is not a very easy question to answer. Mathematical problems (including those you have studied in your secondary school and college courses) are a good source of such examples.

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 a 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 bubble sort 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.