Exercises 3.2

1. Find the number of comparisons made by the sentinel version of sequential search

   a. in the worst case.

   b. in the average case if the probability of a successful search is p ($0 \leq p \leq 1$).

2. As shown in Section 2.1, the average number of key comparisons made by sequential search (without a sentinel, under standard assumptions about its inputs) is given by the formula

$$C_{avg}(n) = \frac{p(n+1)}{2} + n(1-p),$$

   where p is the probability of a successful search. Determine, for a fixed n, the values of p ($0 \leq p \leq 1$) for which this formula yields the largest value of $C_{avg}(n)$ and the smallest value of $C_{avg}(n)$.

3. *Gadget testing* A firm wants to determine the highest floor of its n-story headquarters from which a gadget can fall without breaking. The firm has two identical gadgets to experiment with. If one of them gets broken, it can not be repaired, and the experiment will have to be completed with the remaining gadget. Design an algorithm in the best efficiency class you can to solve this problem.

4. Determine the number of character comparisons 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.)

5. How many comparisons (both successful and unsuccessful) are made by the brute-force string-matching algorithm in searching for each of the following patterns in the binary text of 1000 zeros?

   a. 00001 b. 10000 c. 01010

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

7. In solving the string-matching problem, would there be any advantage in comparing pattern and text characters right-to-left instead of left-to-right?

8. Consider the problem of counting, in a given text, the number of substrings that start with an A and end with a B. (For example, there are four such substrings in CABAAXBYA.)

   (a) Design a brute-force algorithm for this problem and determine its efficiency class.

   (b) Design a more efficient algorithm for this problem [Gin04].

9. Write a visualization program for the brute-force string-matching algorithm.

10. *Word Find* A popular diversion in the United States, "word find" (or "word search") puzzles ask the player to find each of a given set of words in a square table filled with single letters. A word can read horizontally (left or right), vertically (up or down), or along a 45 degree diagonal (in any of the four directions), formed by consecutively adjacent cells of the table; it may wrap around the table's boundaries but it must read in the same direction with no zigzagging. The same cell of the table may be used in different words, but, in a given word, the same cell may be used no more than once. Write a computer program for solving this puzzle.

11. *Battleship game* Write a program for playing Battleship (a classic strategy game) on the computer which is based on a version of brute-force pattern matching. The rules of the game are as follows. There are two opponents in the game (in this case, a human player and the computer). The game is played on two identical boards (10-by-10 tables of squares) on which each opponent places his or her ships, not seen by the opponent. Each player has five ships, each of which occupies a certain number of squares on the board: a destroyer (2 squares), a submarine (3 squares), a cruiser (3 squares), a battleship (4 squares), and an aircraft carrier (5 squares). Each ship is placed either horizontally or vertically, with no two ships touching each other. The game is played by the opponents taking turns "shooting" at each other's ships. A result of every shot is displayed as either a hit or a miss. In case of a hit, the player gets to go again and keeps playing until this player misses. The goal is to sink all the opponent's ships before the opponent succeeds in doing it first. (To sink a ship, all squares occupied by the ship must be hit.)

Hints to Exercises 3.2

1. Modify the analysis of the algorithm's version in Section 2.1.

2. As a function of p, what kind of function is C_{avg}?

3. Solve a simpler problem with a single gadget first. Then design a better than linear algorithm for the problem with two gadgets.

4. The content of this quote from Mahatma Gandhi is more thought provoking than this drill.

5. For each input, one iteration of the algorithm yields all the information you need to answer the question.

6. It will suffice to limit your search for an example to binary texts and patterns.

7. The answer, surprisingly, is yes.

8. a. For a given occurrence of A in the text, what are the substrings you need to count?

   b. For a given occurrence of B in the text, what are the substrings you need to count?

9. You may use either bit strings or a natural language text for the visualization program. It would be a good idea to implement, as an option, a search for all occurrences of a given pattern in a given text.

10. Test your program thoroughly. Be especially careful about the possibility of words read diagonally with wrapping around the table's border.

11. A (very) brute-force algorithm can simply shoot at adjacent feasible cells starting at, say, one of the corners of the board. Can you suggest a better strategy? (You can investigate relative efficiencies of different strategies by making two programs implementing them play each other.) Is your strategy better than the one that shoots at randomly generated cells of the opponent's board?