Exercises II

Note that to *give an algorithm* means not only to describe the algorithm, but also to analyze its running time. Labelled algorithms, figures, and chapters refer to the course book.

II-1 Where in the matrix multiplication-based DP algorithm for the all-pairs shortest paths problem do we need the associativity of matrix multiplication?

II-2 Give an $O(n^2)$-time algorithm to find the maximum length of monotonically increasing subsequences of a given sequence of n numbers. For instance, if given $(3, 1, 2, 5, 2, 6, 8, 6, 7, 3, 5)$ as input, the algorithm should output 6 (the length of the subsequence $(1, 2, 5, 6, 6, 7)$). (*Side note:* Even an $O(n \log n)$-time algorithm exists.)

II-3 (**CLRS 15-3 Bitonic euclidean traveling-salesman problem**) The euclidean traveling-salesman problem is the problem of determining the shortest closed tour that connects a given set of n points in the plane. Figure 15.11(a) shows the solution to a 7-point problem. The general problem is NP-complete, and its solution is therefore believed to require more than polynomial time (see Chapter 34).

J. L. Bentley has suggested that we simplify the problem by restricting our attention to bitonic tours, that is, tours that start at the leftmost point, go strictly left to right to the rightmost point, and then go strictly right to left back to the starting point. Figure 15.11(b) shows the shortest bitonic tour of the same 7 points. In this case, a polynomial-time algorithm is possible.

Describe an $O(n^2)$-time algorithm for determining an optimal bitonic tour. You may assume that no two points have the same x-coordinate. (*Hint:* Scan left to right, maintaining optimal possibilities for the two parts of the tour.)

II-4 (**CLRS 15-4 Printing neatly**) Consider the problem of neatly printing a paragraph on a printer. The input text is a sequence of n words of lengths $l_1, l_2, \ldots, l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i < j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} l_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines. Give a dynamic-programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.

II-5 (**CLRS 25.1-9**) Modify Faster-All-Pairs-Shortest-Paths so that it can determine whether the graph contains a negative-weight cycle.