**Total possible marks: 100.** Homeworks must be solved individually. This set covers chapters 15–16 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

Note: for all numerical exercises, give only the final result. However, give a brief explanation of how you do the computation. For example, for exercise 2.2 about the matrix-chain multiplication, the caption of fig. 15.5 in the textbook illustrates how entry m[2, 5] is computed.

**Exercise 1: segmented least squares problem (35 points).** In the usual least squares problem, we are given *n* points in the plane,  $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \in \mathbb{R}^2$ , and we want to find a line y = ax + b that minimizes the sum of squared errors

$$E(a,b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

(see left panel in the figure). The optimal line is given by  $a = \frac{\overline{xy} - \overline{xy}}{\overline{x^2} - \overline{x^2}}$  and  $b = \overline{y} - a\overline{x}$ , where we define the following moments:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \quad \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \quad \overline{x^2} = \frac{1}{n} \sum_{i=1}^{n} x_i^2, \quad \overline{xy} = \frac{1}{n} \sum_{i=1}^{n} x_i y_i.$$

For the case n = 1, of the many lines with zero error we take a = 0 and  $b = y_1$  for definiteness.

1. (2 points) For the least-squares problem on n points, give the run time as a function of n (using order notation) of evaluating E(a, b) (for given values of a, b).

In the segmented least squares problem, the points are assumed to lie roughly on a sequence of several line segments (rather than a single segment). We are given n points in the plane,  $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ , with  $x_1 < x_2 < \cdots < x_n$ , and we want to find a sequence of lines that minimizes a tradeoff function  $\mathcal{E} + cL$ , where  $1 \leq L < n$  is the number of segments,  $\mathcal{E} = \sum_{l=1}^{L} E_l$  is the sum of the sums of squared errors within each segment, and c > 0 is a given constant. Each segment consists of a consecutive sequence of points, and each point belongs to one and only one segment. This allows us to achieve a good fit of the segments to the points ( $\mathcal{E}$ ), but also to have as few segments as possible (L). For example, for the right panel in the figure (for a certain c value), the optimal solution consists of L = 3 segments consisting of points 1–7, 8–15 and 16–25, respectively, and it has a cost  $\mathcal{E} + cL$  equal to



2. (3 points) For the segmented least-squares problem, give a lower bound on the run time of the brute-force approach that examines all possible solutions (i.e., all possible segmentations).

Solve the segmented least squares problem using dynamic programming:

- 3. (10 points) Write the value of an optimal solution as a recursive expression using the value of optimally solved subproblems. *Hint*: the optimal substructure is very similar to the rod-cutting problem.
- 4. (10 points) Write pseudocode that implements this solution top-down using memoization, and prints the solution. Give its run time.
- 5. (10 points) Write pseudocode that implements this solution bottom-up by filling in a table in order, and prints the solution. Give its run time.

Note: one of the lab assignments will ask you to implement the segmented least squares dynamic programming algorithm.

## Exercise 2: matrix-chain multiplication (25 points).

- 1. (7 points) (Exercise 15.2-2 in the textbook.) Give a recursive algorithm MATRIX-CHAIN-MULTIPLY(A, s, i, j) that actually performs the optimal matrix-chain multiplication, given the sequence of matrices  $\langle A_1, A_2, \ldots, A_n \rangle$ , the *s* table computed by procedure MATRIX-CHAIN-ORDER, and the indices *i* and *j*. (The initial call would be MATRIX-CHAIN-MULTIPLY(A, s, 1, n).) Assume you can use a function MATRIX-MULTIPLY(A, B) that returns a matrix C = AB.
- 2. (10 points) Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is  $\langle 15, 30, 20, 5, 4 \rangle$ . Fill in the *m* and *s* tables as in fig. 15.5 in the textbook.
- 3. (8 points) Give the result obtained using a greedy algorithm that picks the smallest local cost  $p_{i-1}p_kp_j$  over  $i \leq k < j$ . Comment on whether this greedy algorithm always solves correctly the matrix-chain multiplication problem or not.

## Exercise 3: longest common subsequence (15 points).

- 1. (12 points) (Exercise 15.4-3 in the book.) Give a memoized version of LCS-LENGTH (see p. 394) that runs in O(mn) time.
- 2. (3 points) Which version is faster, top-down using memoization or bottom-up by filling in the table? *Hint*: consider which subproblems are computed.

**Exercise 4: Huffman codes (25 points).** The following table gives the frequencies of each character in a file with 1 000 characters.

1. (2 points) How many bits do we need to store the file if using a fixed-length code?

- 2. (8 points) Show the code built by the Huffman algorithm, both as a tree and as a list (character, codeword).
- 3. (2 points) How many bits do we need to store the file with this Huffman code?

Assume now that all the characters have the same frequency (= 125).

- 4. (8 points) Show the code built by the Huffman algorithm, both as a tree and as a list (character,codeword).
- 5. (2 points) How many bits do we need to store the file with this Huffman code?
- 6. (3 points) Generalizing from this example, what can you say about the code built by the Huffman algorithm for an alphabet with  $n = 2^b$  characters each of which occurs with equal frequency f? Explain your answer.

Bonus exercise: longest common subsequence (15 points). Construct the matrices c and b as in fig. 15.8 in the textbook for the words SPANKING and AMPUTATION, and give a longest common subsequence for them.