

Practice exercises for the final exam

*do not return*

### Exercise 1

You have decided to make a CD of your favourite songs. You already picked the songs, but you still need to decide in what order they should appear on the CD. Your task is to compute the “best” possible order for them to appear. For each pair of songs  $i$  and  $j$  you have chosen a “compatibility rating”  $compat_{i,j}$ , which is a positive number indicating how good you think  $b$  sounds when it is played right after  $i$ .  $compat_{i,j}$  and  $compat_{j,i}$  are generally *not* equal. Use dynamic programming to find an ordering of the songs which maximizes the sum of the compatibilities of consecutive songs. You do not need to print out the actual ordering, just the maximum total compatibility. For  $n$  songs your algorithm should run in  $\mathcal{O}(n^2 2^n)$  time.

- List the table(s) your algorithm will use, and explain the meaning of each entry.
- Specify the recurrence and the base case(s) used by your algorithm.
- Implement your algorithm in pseudocode.
- Analyze the running time of your algorithm: justify your answer briefly.

### Exercise 2

120 students have pre-enrolled for “Algorithms and Data Structures” lecture. Each student must attend one of the five tutor sessions, and each tutor can handle a group with at most 24 students. Each student has specified three choices for tutor. A “happiness” rating is assigned to each choice: assignment of a student to her/his first choice is worth 10, while the second choice is worth 7, and the third choice 3. Assignment of a student to a tutor not among her/his top three is worth 0. The problem is to determine an assignment of students to tutors that maximizes the total happiness of the enrolled students, without exceeding the capacity of any lab section. Express this problem as:

- a linear programming problem,
- a min cost flow problem

### Exercise 3

Consider the Maximum Alternating Sum Subsequence (MASS) problem. Given a sequence  $S = [x_1, x_2, \dots, x_n]$  of positive integers, find the subsequence  $A = [x_{i_1}, x_{i_2}, \dots, x_{i_k}]$ , where  $i_1 < i_2 < \dots < i_k$ , that maximizes the alternating sum  $x_{i_1} - x_{i_2} + x_{i_3} - x_{i_4} + \dots \pm x_{i_k}$ .

For example, if  $S = [4, 9, 2, 4, 1, 3, 7]$ , the MASS is  $A = [9, 2, 4, 1, 7]$  which evaluates to  $9 - 2 + 4 - 1 + 7 = 17$ . If  $S = [7, 6, 5, 4, 3, 2, 1]$  the MASS is  $A = [7]$ . Clearly, the length of the MASS depends on the actual values in  $S$ . Assume that the length of  $S$  is at least one, and all its elements are integers greater than zero. The following questions ask you to develop a dynamic programming algorithm to find the value of the MASS (but not the actual subsequence) for a given sequence.

#### Exercise 4

The Euclidean TSP is to find a shortest cycle through  $n$  points in the plane. This problem differs from the general TSP problem in the sense that the interpoint distances are not arbitrary, but are inherited from the Euclidean metric. However, this is also NP-hard. We will design an  $\mathcal{O}(n^3)$  time 1.5-approximation algorithm to it now.

- find a Euclidean MST  $T$  of these points ( $\mathcal{O}(n \log n)$  time).
- find a minimum weight Euclidean matching  $M$  on the set  $X$  of vertices of odd degree in  $T$  (We can do this in  $\mathcal{O}(n^3)$  time).
- find an Eulerian cycle  $\phi$  on the graph  $T \cup M$  and shortcut it skipping already visited points.

Show that the obtained graph is 1.5-approximation of the Euclidean TSP. Also argue why the cycle  $\phi$  exists.

#### Exercise 5

Given a simple<sup>1</sup> polygon  $P$  and a point  $a$ , we want to know whether  $a$  is inside  $P$ .

- Let  $R$  be any ray originating at  $a$ . Show that  $a$  is inside  $P$  iff  $R$  intersects an odd number of  $P$ 's edges. (An intersection with a corner point is counted as two.)
- Describe a linear time algorithm that finds whether  $a$  is inside  $P$  or not.

#### Exercise 6

You are given a city with  $n$  post offices. Your goal is to find a place for a new post office as far as possible from the existing post offices, but it must be inside the convex hull of all the offices. Design an algorithm for solving this problem that runs in  $\mathcal{O}(n \log n)$  time.

#### Exercise 7

Do not forget to make the evaluation of this course: [www.st.cs.uni-sb.de/eva](http://www.st.cs.uni-sb.de/eva) using the password given to you at the lecture. If do not have a password, send a request to [dementiev@mpi-sb.mpg.de](mailto:dementiev@mpi-sb.mpg.de). Your feedback is highly appreciated.

---

<sup>1</sup>A polygon is simple if its boundary does not intersect with itself.