Assignment no. 1

due: November 1st, 2021

The letter **(p)** after the exercise number indicates that this exercise has a programming component. The number **(2)** after the exercise number indicates that students may work in pairs on the exercise and submit their solution jointly.

Exercise 1.1 (p) Write a program that solves Oskar's cube. The matrices that describe the faces of the cube appear in the course's website, together with specifications how to output a solution path from start to goal.

Exercise 1.2 We are given a convex polygonal robot P with m vertices that is free to translate inside a convex polygonal room Q with n vertices. The only obstacles to the motion of P are Q's walls. What is the maximum combinatorial complexity of the free space in this case? Describe an efficient algorithm to compute a representation of the free space.

Exercise 1.3 (2) Recall that you are expected to be familiar with the contents of Chapter 2, *Line Segment Intersection*, of the Computational Geometry book by de Berg *et al.* Specifically, for the current exercise you will need to employ the DCEL data structure and the sweep-line algorithm.

We are given a rectangular room with non-overlapping disc-like obstacles, and a Roomba robot together with free start and goal positions for it. We wish to plan a collision-free motion for the Roomba from start to goal. You may assume *general position* here, namely that you do not need to handle degenerate situations. In particular, you may assume that the boundary curves of three distinct expanded obstacles do not meet in a single point.

(a) Design an exact algorithm to solve this motion-planning problem, based on computing the union of the expanded obstacles. Describe it in detail, and analyze its running time.

(b) optional (bonus) One can mimic the proof of the combinatorial bound on the complexity of the free space as we saw in class, to derive an exact $O(n \log n)$-time algorithm for solving the motion-planning problem at hand. Describe such an algorithm in detail, and show that its running time is indeed $O(n \log n)$. You have at your disposal a procedure, which given n planes in 3-space returns all the edges (segments and rays) of their upper envelope, and does that in $O(n \log n)$ time—no need to describe the details of this procedure.