Exercise 11.1  Path Planning in Labyrinths.

You are given a labyrinth as a drawing on squared paper, like in the example below. At the marked point there is a robot facing the direction indicated by the arrow. The question is how fast the robot can escape the labyrinth. The robot can travel “forward” one square in the direction it is facing within 3 seconds. To stop after a forward movement takes 2 seconds. While standing still, the robot can rotate 90 degrees, which costs 2 seconds. The robot does not need to stand still between two consecutive forward movements (although it could, but that would take more time).

In the examples below, the robot needs 113s and 79s to escape.

![Labyrinth Example 1]

![Labyrinth Example 2]

a) Model the above problem as a shortest path problem. Describe how to represent the labyrinth as a graph such that the length of the shortest path in the graph equals the time that the robot needs to escape.

b) What is an efficient algorithm to solve this problem?

c) Which running time in dependency of the number of squares of the labyrinth does this algorithm have when it is applied to the graph constructed in a)?

Please turn over.
**Exercise 11.2  Max-Flow by Hand.**

Apply one of the algorithms presented in the lecture for finding a maximum flow from \( s \) and \( t \) in the following network. The capacities are given next to the corresponding edges. Provide the resulting maximum flow, minimum cut and residual graph.

![Network Diagram](image)

**Exercise 11.3  Evacuation Problems.**

Consider an \((n \times n)\) grid, i.e. an undirected graph with \( n \) rows and \( n \) columns containing an edge between each pair of vertices that are neighbored in horizontal or neighbored in vertical direction. The *escape route problem* asks to decide, given \( m \leq n^2 \) pairwise distinct starting vertices \( s_1, \ldots, s_m \), whether there exist \( m \) edge-disjoint paths from the starting vertices \( s_i \) to some boundary vertices (in the interior of the grid you are in danger, on the boundary you are safe). Notice that two paths may share one or more vertices, but no edges. In the example on the right, the starting vertices are pictured black, and the escape routes are pictured bold.

a) Model the above problem as a flow problem. Notice that the grid is undirected while flow algorithms assume that the network is *directed*. Describe in detail, how an appropriate network \( N = (V, E, c) \) can be constructed, i.e., which vertices \( V \) and which edges \( E \) have to be defined, and which capacities have to be assigned to the edges. Describe how you can conclude from the value of a maximum flow whether or not there exist \( m \) edge-disjoint escape routes.

b) Provide the running time of the solution from a) in dependency of \( n \) and \( m \) assuming that the algorithm of Ford and Fulkerson is used.

c) Now we want to decide whether there exists a set of \( m \) *vertex-disjoint* escape routes, where no pair of two paths shares a common vertex. Describe in detail how your solution from a) has to be modified such that every vertex is used by at most one escape route. Does the running time of your solution change, and if yes, how?

**Hand-in:** until Wednesday, 13th May 2015.