Exercise 10.1  

a) Determine a minimum spanning tree of the following graph. What is its weight?

b) A shortest-path tree in a graph $G$ with edge weights is a spanning tree $B$ with root $r$, such that, for every node $v$ of the tree, the unique path from $r$ to $v$ in $B$ is also a shortest $r$-$v$-path in $G$ with respect to its edge weights. Prove or disprove: 1. every minimum spanning tree is a shortest path tree (for some root $r$). 2. every shortest-path tree is a minimum spanning tree.

c) Design an algorithm that finds a maximum spanning tree (a spanning tree with maximum sum of edge weights). Prove that your algorithm is correct; use modified versions of the selection rules from the lecture for selecting and discarding edges.

Exercise 10.2  

a) Can a Fibonacci heap degenerate to a structure that has exactly one node in the root list, with a linear list appended to it? Describe how this structure can occur, or prove that it is not possible.

b) Why cannot the heap operations (insert, extract_min, and decrease_key) have an amortized running time of $O(1)$ if we allow only comparisons with keys (but no computations)?

c) A Binomial tree $B_n$, $n \geq 0$ is recursively defined as follows: $B_0$ consists of exactly one node; $B_n$ is created by taking two Binomial trees $B_{n-1}$, and making the root the leftmost child of the root of the other tree. Provide a sequence of operations on a Fibonacci heap such that, for any $n \geq 0$, that produces a Binomial tree $B_n$. 

Exercise 10.3  Algorithm design: 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 maze. The robot can travel "forward" one square in the direction it is facing within 3 seconds. To stop after a forward movement takes 1 second. 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).

We want to use Dijkstra's algorithm to determine how long it takes the robot at least to escape from the labyrinth. Describe how to represent the labyrinth as a graph, such that the length of the shortest path in the graph equals the time it takes to the robot to escape. Which running time does Dijkstra's algorithm have on this graph depending on the number of squares of the labyrinth?

In the examples below, the robot needs 102s and 74s, respectively, in order to escape.

Hand-in: until Wednesday 9th May 2012.