Algorithms, Spring 2012-13, Homework 6
due Wednesday 24 April, 10:00

For every coding problem, include the pseudocode of your algorithm and a short verbal description, briefly argue your algorithm's correctness and estimate its running time.

Problem 1

Given is an $m \times n$ board where every position corresponds to either an empty space or a wall. There are also two little robots, each occupying an empty space. In a single move, each robot can move along a direction parallel to the axes of the board until it hits a wall or the other robot (it stops on the empty space just before the wall or the other robot). The robots cannot stop midway and it is really bad if either of them falls off the board. Given the robots' starting positions and a pair of target locations that the robots need to get to, find the smallest number of moves the robots need to make to get to the target locations, or determine that it is not possible to get there. It does not matter which robot gets to which target location. A robot can get to a target location and then leave it to help the other robot get to their target location. At the end of the sequence of moves, both target locations need to be occupied by a robot. Your algorithm should run in time $O((mn)^2)$.

Problem 2

Given is a weighted undirected graph $G = (V, E)$ with positive weights and a subset of its edges $F \subseteq E$. An *F-containing spanning tree of G* is a spanning tree that contains all edges from F (there might be other edges as well). Give an $O(mn)$ algorithm that finds the cost of the minimum-cost F-containing spanning tree of G.

Problem 3

Consider the following BFS-inspired algorithm for the single source shortest path problem – see the next page. Let $G = (V, E, w)$ be a positively weighted undirected graph and let $s \in V$ be one of its vertices.

Does the algorithm work? If yes, argue why it works. If not, find a counterexample and compare the correct answer with the output of the algorithm.

BFSshortestPath($G = (V, E, w)$, s)

1. For $v \in V$ do
2. Let $found[v] = false$ and $dist[v] = \infty$.
3. Let $Q[0] = s$ and $found[s] = true$ and $dist[s] = 0$.
4. Let $beg = 0$ and $end = 1$.
5. While ($beg < end$) do
6. Let $v = Q[beg]$.
7. For every neighbor u of v do
8. If not $found[u]$ then
9. Let $found[u] = true$ and let $dist[u] = dist[v] + w(v, u)$.
10. Let $Q[end] = u$ and increment end by 1.
11. Else
12. If $dist[u] > dist[v] + w(v, u)$ then let $dist[u] = dist[v] + w(v, u)$.
13. Increment beg by 1.
14. Return $dist[]$.