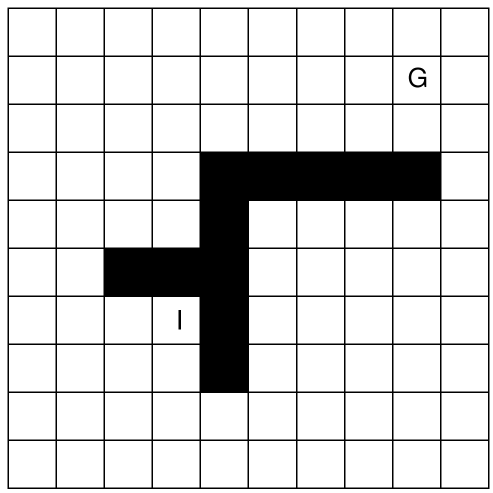
Assignment 6

Assigned: Tuesday, November 27, 2001
Due: Tuesday, December 11, 2001

In this assignment, you must program a simple simulated environment and construct a reinforcement learning agent that discovers the optimal (shortest) path through the environment to reach a goal. The agent's environment will look like:



Each cell in this grid is a state in the environment. The cell labeled "I" is the initial state of the agent. The cell labeled "G" is the goal state of the agent. The black cells are barriers—states that are inaccessible to the agent. At each time step, the agent may move one cell to the left, right, up, or down. The environment does *not* wrap around. Thus, if the agent is in the lower left corner and tries to move down, it will remain in the same cell. Likewise, if the agent is in the initial state and tries to move up (into a barrier), it will remain in the same cell.

You should implement a Q learning algorithm that selects moves for the agent. The algorithm should perform exploration by choosing the action with the maximum Q value 95% of the time, and choosing one of the remaining three actions at random the remaining 5% of the time.

The simulation runs until the agent reaches the goal state. The reward at the goal is 0, but at every other state is –1 (because it takes energy for the agent to move). Because this simulation has a finite running time, you do not need to use a discounting factor, i.e., you can set the parameter γ (described in Chapter 13 of the book) to one. Also, because the environment is deterministic, you can set the parameter α (which I will describe in class) to zero, which yields the exactly the algorithm inTable 13.1 of the book.

Write up

Your write up should include the following:

- a graph showing number of steps required to reach the goal as a function of learning trials (one "trial" is one run of the agent through the environment until it reaches the goal). When you make your graph, it will be very noisy if you plot each learning trial separately. Thus, you may want to plot average number of steps over 10-trials blocks.

- a figure showing the policy of your agent. The policy can be summarized by making an array of cells corresponding to the states of the environment, and indicating the direction (up, down, left, right) that the agent is most likely to move if it is in that state.
- a figure showing the value function. The value function can be summarized by making an array of cells corresponding to the states of the environment, and indicating the *minimum* Q value associated with a state in the corresponding cell. The value function thus indicates the number of steps to the goal assuming the optimal action according to the Q table is chosen.
- In addition to running the simulation with an exploration rate of 5%, try 1% and also 20%. Plot the learning curves for these different exploration rates. You should find that the lower exploration rates lead to slower learning but higher asymptotic performance.

Advice

Here is a suggestion about how to modularize your code:

- `read_environment`: read a layout of the environment, including initial and goal states and obstacles, from a file. (If you wish, you can make your code less general purpose and hardwire the environment into your code.)
- `update_state`: given the current state and an action, determine the agent's next state.
- `determine_reward`: given a state, determine the agent's reward as a result of moving into that state.
- `choose_action`: given a state, choose an action based on the current Q table.
- `initialize_q_table`: reset the Q values to zero
- `update_q_table`: given the previous state, an action, the current state, and an immediate reward obtained, update the Q table according to Equation 13.7 of the text