Artificial Intelligence	**Homework 2**
CS430 Fall 2003	**Due Monday October 13**

1 Abstract state spaces (20 points)

Consider a simulated office robot that lives in a grid world. A sample world is shown in Figure 1. Note that this is only a sample and not the only world in which the robot may exist. Your answer to this question should be general enough to apply to any world which meets the specifications below.

The two-dimensional grid world in which our robot lives can be divided into cells which appear as individual squares in the figure. Squares that are black are walls or obstacles in the environment, and the robot cannot pass through them. The perimeter of the world is always enclosed by blocked cells. The robot is free to move to any non-blocked cell that it is currently adjacent to.

The grid world in the figure is small enough to consider searching it exhaustively. However, imagine that we scaled this up to be a map of the entire 2nd floor of Dearborn hall but with grid cells of size one square inch. Such a grid would be able to represent the details of obstacles to within an inch, but it would be quite expensive to search such a grid exhaustively.

One way to do robot path planning over such a large grid is to use a two-tiered approach in which an initial (hopefully easier) search is performed in a more abstract space (in general, one which includes less detail than the original space) and then the path obtained as a result of the abstract search is used in order to perform more limited search in the original search space.

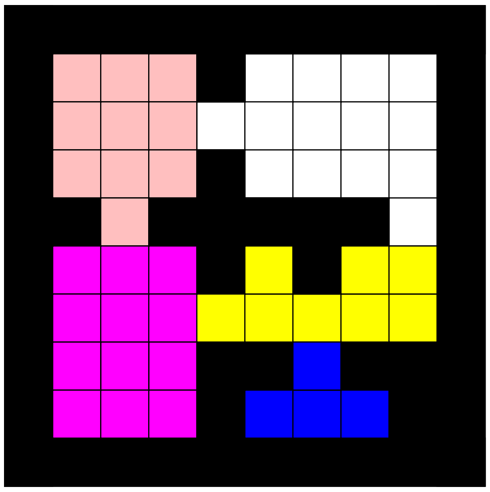

Figure 1: A sample grid world partitioned into areas

We will therefore partition the world into a number of *areas*. An area is simply a conglomeration of adjacent cells. A room or a hallway could be an area, for example. In the figure, each area is denoted by a different color. A change in color indicates a boundary between two areas. You can assume that every non-blocked cell has some area associated with it.

Consider the problem of using the 2-tiered approach to plan to get from the lower left corner (call it (0,0)) to the upper right corner (7,7). At the abstract level, this is just a problem of getting from the pink region to the white region (presumably by either going through the yellow region or through the salmon-colored region). At the detailed level, if the yellow region has been chosen, then the problem is to get from the lower left corner (call it (0,0)) to the nearest yellow region square (3,2). Then from (3,2) to the nearest white square (7,4). And finally from (7,4) to (7,7).

In this problem, you will formalize and analyze this idea.

a. (3 points) Define the abstract state space search problem precisely: i.e., describe the state space, the initial and goal states, the successor function, and provide a path cost function. You should define the path cost as an addition function of edge costs (cost of moving from one node to a neighboring node).

b. (2 points) Explain how a path obtained as a solution to the abstract search problem (at the level of areas) can be used to constrain the search for a path at the cell level from a start to a goal cell. [Hint: formulate search problems at the cell level using the solution at the area level.]

c. (3 points) Are the solutions (i.e., paths) at the cell level obtained by the two phase search method of (1) breadth-first search at the area level, followed by (2) a breadth-first search at the cell level guided by the abstract solution, optimal? Explain why or why not.

d. (2 points) For your abstract state space, can you construct an edge cost function such that for any search task (initial cell and goal cell), an optimal path in the abstract space will result in an optimal path in the grid or cell space? Either show the cost function or show why it is impossible to construct one by giving an example. [Hint: consider a world where there are multiple paths from one area to another. You can use grid level information to compute edge cost functions for the abstract level, as long as the computation is efficient.]

e. (3 points) Often, when an abstract edge cost function leading to an optimal solution at the cell level is difficult or impossible to construct, we can usually construct a cost function that yields a fairly good approximation to the optimal solution. Suggest a good edge cost function for the abstract state space.

f. (2 points) Suggest an efficiently computable heuristic function for the abstract space which is admissible relative to your edge cost function from the previous subpart. Recall that a heuristic function estimates the remaining distance to the goal. Explain why it is admissible.

g. (3 points) Now consider an alternative formulation of the abstract state space: one in which the states correspond to doorways between areas. A doorway is a pair of cells of two different colors. Describe an exact edge cost function and an admissible heuristic function appropriate for this formulation.

h. (2 points) Discuss the tradeoff between these two formulations. In which domains is one more appropriate than the other?

2 Agents with and without memory (20 points)

For each of the following vacuum-cleaner environments, explain whether a memoryless agent can perform just as well as an agent with memory given the same sensors. In other words, to perform optimally, is memory required?

a. (4 points) Environment: a 2 by 10 grid world. Each grid cell can contain dirt. In the starting state, each cell contains dirt with probability 0.2 and the agent is in location [2,1] (i.e., one step to the east of the home square). The performance measure is the same as in the book (-1 point for each move, $+100$ points for each piece of dirt collected, -1000 points if the agent does not return home before shutting off). The agent is the vacuum-cleaning robot from the textbook with the home, bump, and dirt sensors. The agent knows the size of the world, and it knows its initial position and orientation.

b. (4 points) Environment: Same as in part (a) with the following changes. The agent does not have a bump or a home sensor, but instead it has sensors that tell the current location and orientation of the agent.

c. (4 points) Write an agent program for the best memoryless agent in part (a) Write it in pseudo-code as a set of IF-THEN rules.

d. (4 points) Compute the size of the table required to implement a table-based agent for the environment in part (c). Explain your answer.

e. (4 points) Compute the number of possible states for the environment in part (b) Explain your answer.