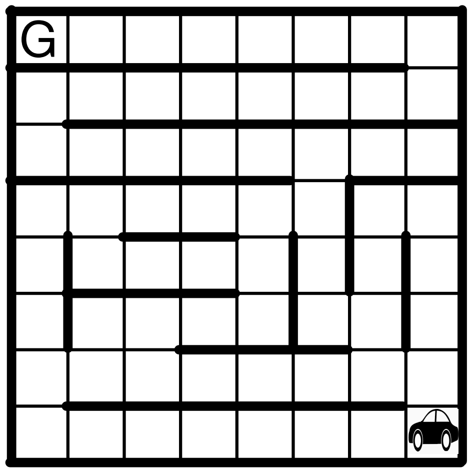
Artificial Intelligence Homework 1
Due at the start of class, in hard copy, on 9/30/2014

1. Imagine a car-like agent wishes to navigate a maze such as this one:



The agent is directional and at all times faces some direction $d \in \{N, S, E, W\}$. With a single action, the agent can either move forward at an adjustable velocity *v,* or turn.

The turning actions are LEFT and RIGHT, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero).

The moving actions are MOVE, MOVE-FASTER and MOVE-SLOWER. MOVE moves the car in its current direction forward by the number of squares equal to its velocity. MOVE-FASTER increments the velocity by 1 and then moves forward the number of squares equal to its new velocity. MOVE-SLOWER decrements the velocity by 1 and then moves forward the number of squares equal to its new velocity.

Any action that would result in crashing into a wall is illegal (meaning it is not available as an action from the state in question). Any action that would reduce v below 0 or above some maximum speed V_{max} is also illegal.

The agent's goal is to find a driving strategy that parks it (stationary, with $v = 0$, direction facing irrelevant) on the goal square G in as few actions (time steps) as possible. Every action (turning 90 degrees, or driving forward at the current velocity) takes 1 unit of time.

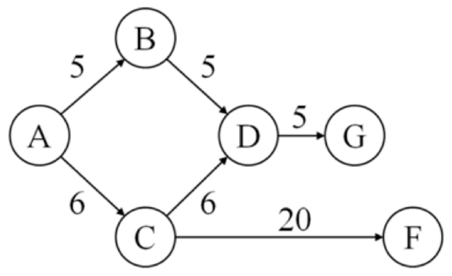
Note that this is problem formulation is different from the regular navigation problem!

All of these questions pertain to a generic maze of size m by n, not specifically the maze above, which is only shown to illustrate how the problem is set up.

(a) If the grid is m by n, what is the (maximum) size of the state space? Justify your answer. You should assume that all possible states are reachable from the start state.
(b) What is the maximum branching factor of this problem? You may assume that illegal actions from a state are never considered. Briefly justify your answer.
(c) Is the Manhattan distance from the agent's location to the goal location an admissible heuristic? Why or why not?
(d) Describe and justify a non-trivial admissible heuristic for this problem that is not the Manhattan distance to the goal.
(e) If we used an inadmissible heuristic to solve this problem, could it change the completeness of the search? Why or why not?

(f) If we used an inadmissible heuristic to solve this problem, could it change the optimality of the search? Why or why not?

2. Consider the following graph, where we are trying to find the shortest path from vertex A to either F or G (both F and G are goal states).



Here are four possible heuristic functions h1(n) through h4(n):

	A	B	C	D	F	G
h1	0	0	0	0	0	0
h2	11	7	7	3	0	0
h3	13	9	7	1	0	0
h4	15	10	11	5	0	0

(a) Which, if any, of the heuristics are admissible?
(b) Which, if any, of the heuristics are consistent?
(c) Run A* (by hand) using h3, showing the order that nodes are considered, along with keeping track of the frontier and explored lists. This is probably easiest by constructing a table with each row corresponding to a step of the algorithm, keeping columns for which node is being expanded, which nodes are on the frontier, and which states are on the explored list. For the nodes on the frontier and the one currently being expanded by A*, make sure to list out the entire path so it's clear that, for instance, you're examining the path ABD rather than ACD. What is the best path returned by A*?
(d) Recall that the term best-first search refers to any search algorithm that uses a heuristic function to prioritize the frontier by which nodes would be best to examine first. A* is an example of an algorithm that falls into this category. Consider a different algorithm that also falls into this category, called greedy best-first search. This algorithm sorts the frontier by h(n), rather than g(n) + h(n). Run greedy best first search by hand using h3, following the same procedure as part (c). What is the best path returned by greedy best-first search?

3. Rhodes College has just built a brand new dormitory with *n* single rooms. Coincidentally, exactly *n* students have applied to live in this dorm for the upcoming year. Naturally, students prefer certain rooms over others, but each student may have different preferences as to what rooms they like or don't like. The college hires you to write a computer program to assign students to rooms. You decide to write a local search hill-climbing style algorithm for this task.

(a) Describe what a state looks like in your algorithm. Be precise (e.g., for the n-queens problem, we said a state was an integer array A of length 8 where integer A[i] specifies the position of the queen in the i'th column was in row A[i]).

(b) Describe the possible actions that move the search from state to state.

(c) Describe a heuristic function, h, which takes a state and returns some number specifying how good or bad that state is. You may assume that you have easy access to all *n* students that will be living in the dorm and may obtain any information you need from them as to their preferences. Be explicit about what information you need from the students and how your heuristic uses this information.

(d) Illustrate a sample state for n=3 students and show the value of your heuristic for this state. You can invent the information needed for the heuristic, just make it plausible.

4. Sam is an ambitious undergraduate CS student entering spring of junior year. Sam needs to take four classes in the remaining three semesters: Advanced Algorithms (A), Beginning Operating Systems (B), Computer Organization (C), and Databases (D).

   There are some restrictions on these courses. Specifically:
   • Advanced Algorithms and Databases are co-requisites (meaning they must be taken during the same semester);
   • Advanced Algorithms and Basic Operating Systems are always offered during the same time of day, and therefore cannot be taken simultaneously;
   • Computer Organization is a pre-requisite for both Databases and Basic Operating Systems, and therefore must be taken in a preceding semester.

   We will now formulate Sam's scheduling problem as a constraint satisfaction problem in which each variable corresponds to a class Sam must take to graduate, and the values correspond to a numbering of the remaining semesters: 1 for the spring of junior year, 2 the fall of senior year, and 3 the final spring.

   (a) State the domains of the variables and the constraints in mathematical notation using the following symbols: =, ≠, <, >, ≤, ≥ (Assume that each class is offered every semester.)

   We will now solve the constraint satisfaction problem formulated in (a) using backtracking depth-first search and arc consistency. Run arc consistency (AC-3) as a preprocessor *and* after each variable assignment. When there is ambiguity as to which variable to assign next, choose the variable with the letter earliest in the alphabet. Furthermore, when the next value to assign to a variable is ambiguous, break ties by assigning the earliest possible semester first.

   (b) What are the remaining domains after enforcing arc consistency on the initial CSP with no assignments? Show the steps of AC-3 as clearly as possible.

   (c) What are the remaining domains after assigning A = 2 and re-enforcing arc consistency?

   (d) State a solution to the CSP.

   (e) True / False: With arc consistency, we will never have to backtrack while solving a CSP. Justify your answer.