CSE 573 Problem Set 1.

Answers on 10/17/08

Please work on this problem set individually. (Subsequent problem sets may allow group discussion). If any problem doesn't contain enough information for you to answer it, feel free to make any assumptions necessary to find the answer, but state the assumptions clearly. You will be graded on your choice of assumptions as well as the clarity of you written answers.

1. Let T be an acyclic, connected, undirected graph (i.e, a ***tree***) whose edges are uniformly of unit length. The ***diameter*** of T is the maximum distance between any two nodes in T.

   a) How might you use breadth-first search (BFS) to find the diameter of T? Do you need to run BFS from different initial, starting nodes?

      Perform BFS from each state, and pick the greatest depth.

   b) What is the time complexity of your algorithm (in ***n***, the number of nodes in T and **d**, the diameter and any other variables you deem relevant).

      $O(n^2)$

   c) Can you devise an algorithm which is linear in n (or prove that such an algorithm is impossible?

      We denote the shortest path length between two states s_1 and s_2 as $l(s_1,s_2)$.

      Algorithm: Pick any node *s*, and perform a BFS from *s*. Then pick the last state (*t*) visited, and do a BFS from *t*. *t'* is the last state visited. Then *l(t,t')* is the diameter. It's linear since it takes only two BFS'.

      Proof that the algorithm finds the diameter: Suppose otherwise, so there is another path *u* to *v* that has a longer shortest path (or *l(u,v)>l(t,t'))*.

      Case 1: either *u* or *v* coincides with *s*. Suppose *u*=*s*, then $l(u,v) = l(s,v) \le l(s,t) \le l(t,t')$

      Case 2: shortest path of (*u,v*) contains *s*. So both *u* and *v* cannot be ancestors of *t* in the BFS rooted at *s*. Suppose *u* is not. Then:

      $$l(u,v) = l(u,s) + l(s,v) \le l(t,s) + l(s,u) = l(s,t) \le l(t,t')$$

      Case 3: shortest path of (*u,v*) does not contain *s*. Then pick the "deepest" common predecessor *p* of *u* and *v* in the BFS rooted at *s.* Follow the same steps of case 2, we have:

      $$l(u,v) = l(u,p) + l(p,v) \le l(u,s) + l(s,v) \le l(t,s) + l(s,u) = l(s,t) \le l(t,t')$$

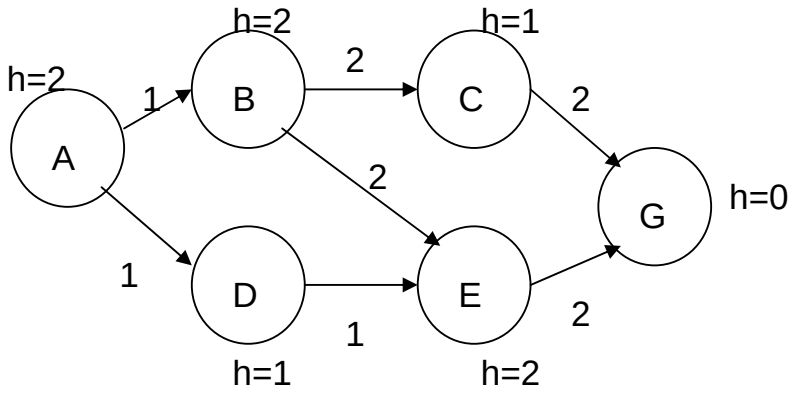
2. The figure below shows a problem-space graph, where A is the initial state and G denotes the goal. Edges are labeled with their true cost. We have a heuristic function, f(), written in the standard form: f(n)=g(n)+h(n) where g(n) is the cost to get from A to n and h(n) is an estimate of the remaining distance to G.

a) In the graph below, is f() admissible? Why or why not?

Yes. Verify that for every state s, *h(s)<c(s)*

b) Is f() monotonic? Why or why not?

Yes. Verify that for every state *s*, each of its successor s', $h(s) \le g(s') + h(s')$



c) Suppose we use IDA* to search the graph and that states having the same *f* values are visited in alphabetical order. In what order does the algorithm consider f-limits and visit states? Fill out the table below:

f-limit	**States visited (in order)**
2	A
3	A D B
4	A D B C E G

3. Prove that if a heuristic is monotonic, then it must be admissible (or provide a counter example).

True. This can be proved by induction on the distance to the goal. A heuristic is monotonic if for every state *s*, $h(s) \le g(s') + h(s')$

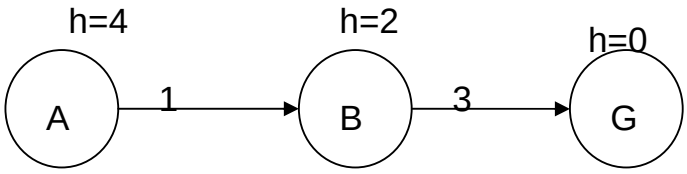
Base case: *h(g)*=0 (*g* is any goal state)
Inductive step: Suppose the *h* values of all states that are less than *k* steps away from *g* are admissible. Consider state *s* the first state that is no less than *k* steps away from *g*. Since *s* is the

first of such state, all its successors must all be less than *k* steps away from *g*, so they are all admissible.

$$h(s) = \min_a c(s,a,s') + h(s') \le \min_a c(s,a,s') + f(s') = f(s)$$

4. Prove that if a heuristic is admissible, then it must be monotonic (or provide a counter example).

False. In the following example, $h(A) > c(A,B) + h(B)$, but the heuristic is admissible



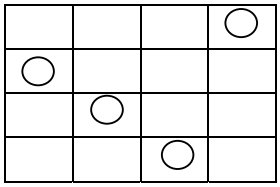
5. Recall that with the n-queens problem, we seek to put n queens on an n x n chess board.

a) Let n=4 can consider the state space of possible board positions. Suppose that the initial state has no queens on it and a goal state has 4 peaceful queens (i.e. none are attacking each other). Operators add one queen to the board. The rules say that one can never put more than four queens on the board. What is the total size of the complete state space (i.e., including states where some queens are attacking each other)?

First consider the number of queens *k* that are already on the board. *k* can be 0,1,2,3,4. The board has 16 cells. Each state corresponds to a way of putting *k* queens on the board. So the total number of states is $\binom{16}{0}+\binom{16}{1}+\binom{16}{2}+\binom{16}{3}+\binom{16}{4}$.

b) By observing the fact that one can never put two queens on the same *column* (without violating peacefulness) one can come up with a simpler representation of state (e.g. encoding one which takes less memory if encoded as a data structure). Describe such a representation. How many 4-queens states are possible in this representation (again, some states may not be peaceful)?

One data structure could be to use a size 4 integer array to store the row number of the queen that occupies the corresponding column. For example, A(1)=2, A(2)=3, A(3)=4, A(4)=1 correspond to the following situation. State size = 4^4.

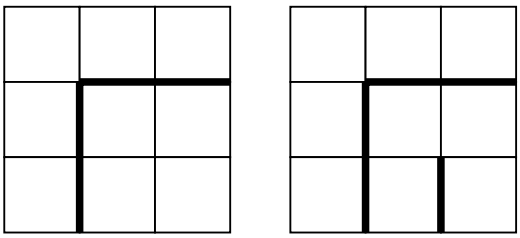


c) Can you shrink the state space even more by eliminating symmetry? Explain?

Various ways. You can use the symmetry by row, column, diagonal, rotations, etc. Suppose that we want to use symmetry by column. Consider one simple case where there is only one queen on the board. The symmetry means that the state where the queen is at cell (i,j) is symmetric to state that the queen is at cell (i,n-j). By using this symmetry and extend to states where there can be multiple queens, we can merge any two states whose column numbers of all the queens that are on the same row add up to n into one state. Doing this, we can halve the state space.

6. Do problem 4.14 from R&N. And be sure that your answer for part (a) describes the offline problem they are presenting formally as a search problem. (What is a belief state in this case?)

a) This online search problem can be viewed as an offline search in belief state space. Where a belief state is a set of all possible wall configurations given our known information so far. There are 12 possible locations for internal walls (6 vertical and 6 horizontal, so there are 2^{12} = 4096 configurations of the world. Because the agent knows which actions are legal, she always knows where she is, hence the environment is the only uncertainty. But since she can be in one of 9 different squares, that expands the space as well. A belief state denotes a subset of the possible world configurations. So there are 2^{4096} belief states. (though some of these are unreachable, so the agent could never distinguish between the following mazes because she can't get to 1-2, 1-3, 2-2, or 2-3 – the only places where one could sense the missing wall.

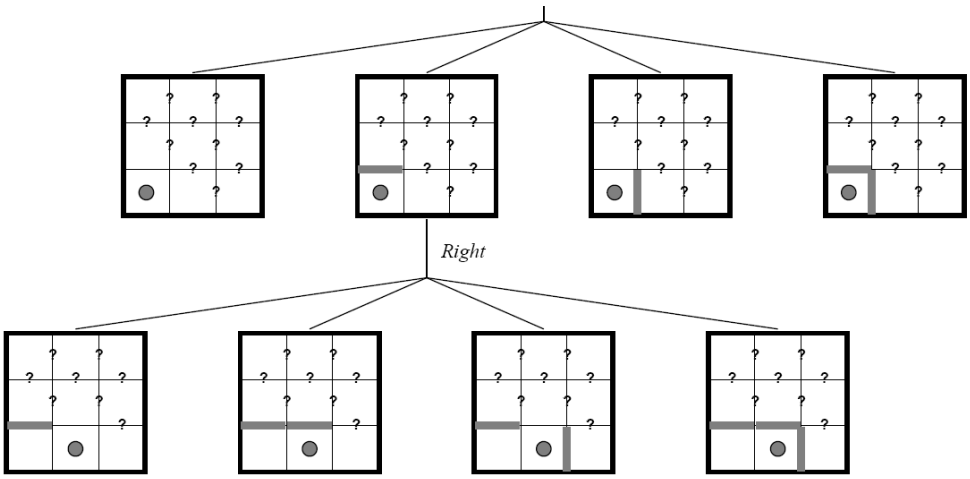


After each action and percept, the agent learns about new wall locations. Since these walls are independent, and the walls don't change their position after we sense them, there is some regularity over the state space which reduces its size quite a bit. We can represent the (huge) space in a compact form – the state of each wall (yes, no, don't-yet-know) for a space of 3^{12} configurations. Factoring in locations, there are $9*3^{12}$ belief states.

b) In the initial state, we know both LEFT and DOWN are illegal. However, UP could be legal or not and RIGHT could be legal or not for a cross product of 4.

c) The first few branches of a contingency plan are shown in the graph below. We assume that when an agent enters a room, it perceives all the adjacent walls. As we have

discussed, the agent initially doesn't know which of 4 possible worlds it will start in, but upon being deposited in the maze it receives percepts and knows the state of the upper and right-hand walls. Thus our diagram starts with four possible states. (As an encoding trick, this is sometimes encoded by having the agent execute a dummy NoOp action which places it in the maze and gives it the percepts, but I haven't drawn such a thing.) The size of a plan is bounded by both the branching factor of the plan and the depth of the plan. Since the agent knows if an action is legal, there is only one outcome, but then there are up to 3 binary percepts so there are 8 possible states she could be in if she just moved to cell (2,2) with fewer possible wall-states if she moves along a wall and even less when moving into a corner.



The longest plan where the goal state can be reached is 13, as shown in the following graph (suppose that the agent foolishly moves right until it gets caught in a deadend and has to backtrack. But to explore the entire maze, we need at most 18 steps. So the size of the plan is bounded by 8^{13} (or 8^{18} if there can be no solutions).