CSE4705 - Introduction to Artificial Intelligence | September 18, 2019

Problem Set 1 - Search

Overview: Solve a few search problems using depth-first, breadth-first, and A^* (a version of best-first search).

1. For each of the following problems, you should solve it three ways: once using depth-first search, once with breadth-first search, and once using A^* search.

   (a) **Farmer-Cat-Duck-Grain**
   A farmer wants to move his cat, duck, and grain across a river. The farmer can cross the river alone or carrying one item. If the farmer leaves the cat and duck on the same side of the river without him, the cat will eat the duck; if he leaves the duck and the grain on the same side of the river without him, the duck will eat the grain. Find a sequence of operations that will get all four across the river without the duck or the grain getting eaten. For $A*$, you should minimize the number of steps.

   (b) **Water Problem**
   Given two containers that hold m and n units of water respectively, and a tub that holds i units of water, all initially empty, is it possible to get k units of water into the tub? What is the sequence of steps?
   Ground Rules: You can either fill either bucket, or empty the contents of a bucket into the other bucket, the tub, or onto the ground. For example, if one bucket has 5 units of water in it, and the other holds 3 units, then, by pouring the 5-unit bucket into the 3-unit, you end up with 3 units in the smaller bucket and 2 units in the larger bucket.
   The first configuration has all of the containers empty. Its neighbors are defined to be those configurations reachable by an allowable transition. Search through the configurations, and if you find a configuration with k units of water in the tub, report the path that got you there. (Note: Since goal testing is done using state equality, you should probably define the goal as k units in the tub with both containers empty). As with the Farmer-Duck-Grain problem, A^* should minimize the number of steps.

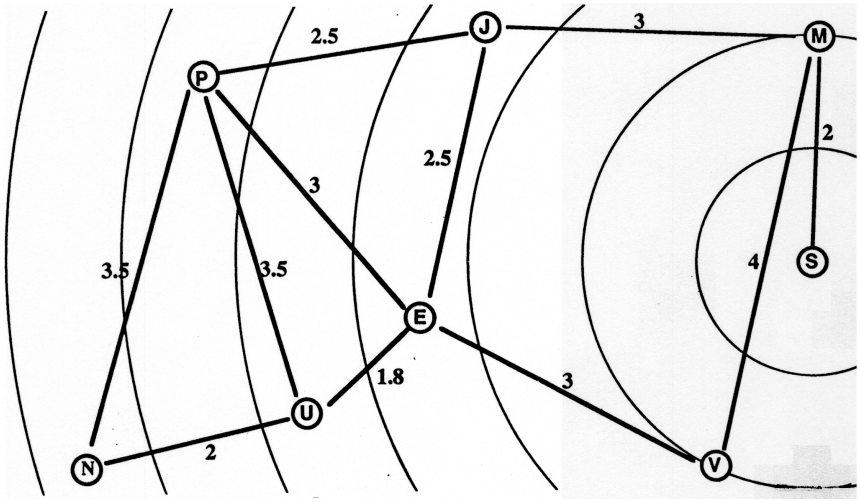

Figure 1: A sample minimum length path problem instance.

   (c) **Minimum-Length Path**
   The problem: given map information (places, direct connections, and distances), use search to find paths between any 2 locations. The goal is to find the minimum-length path.
   Specifically, you will use the different versions of search to return the path (and cost) between any two pairs of locations. You should demonstrate that your code works on the map in Figure 1, but feel free to use others

as well (such as the Romania example from the textbook which was discussed in lecture). This will generally require more than simply finding the first answer. You will need to represent the edge information so that you can find out 1) all edges leading from a location, and 2) the distance on each of those edges. You could use a set of tuples of the form `(place1, place2, distance)`, where these represent a link on the map such as `(v, m, 4)`. You can represent the states in different ways, but your search code should return a list containing the path from source to destination (which is a list) and the cost of that path.

For example, on the above map, the search from E to S should return the list `[[E, J, M, S], 7.5]`.

Again, you will write three versions of your search: two versions that search blindly (i.e. breadth-first or depth-first) until you have found all non-cyclic paths from the start to the destination, then return the best one, and another version that uses A^* search using path cost plus estimated distance to goal as an evaluation function. I used the following (x, y) locations for the places and used straight-line distance as an estimate of distance to goal.

N	$(0.2, 1)$
P	$(1.35, 4.25)$
U	$(2.15, 0.875)$
E	$(3.42, 2.125)$
J	$(3.8, 4.575)$
M	$(6.7, 3.875)$
S	$(6.7, 1.875)$
V	$(5.6, 0.1)$