Homework 1

Due: 4 February 2016

Problem 1.1

Consider the problem of building a computer from parts. There are a number of functionalities that have to be filled by the various parts: providing RAM, providing video, providing a CPU, and so on; and some parts provide only one of those while others do more than one thing. (For instance, some motherboards provide both the basic motherboard functionality and also on-board video.) You'd like to build a computer as cheaply as possible that still fills all the required functionalities.

Formalise this planning process as a problem space search. Include the data types and functions as discussed in class, and for the functions, describe (briefly) what they do or how they work.

Problem 1.2

Consider a 2D-grid-based pathfinding problem where locations in the grid are the states in the problem space, and one of them is a "start" location (the initial state) and another is a "finish" (the goal state). Define the available actions as moving in a cardinal direction for a cost of 1 or diagonally for a cost of 1.5 (unless this would end up inside a wall or off the edge of the grid), and the transition function as returning the location (state) that is in that direction from the current location.

Now consider the following three heuristic functions (where the subscripts stand for "zero", "Manhattan", and "Euclidean" respectively). Each function h evaluates a state X, where F represents the location of the goal:

- $h_Z(X) = 0$
- $h_M(X) = |F_x - X_x| + |F_y - X_y|$
- $h_E(X) = \sqrt{(F_x - X_x)^2 + (F_y - X_y)^2}$

Each can be used as the h component of a ranking function f:

$$f(X) = g(X) + h(X)$$

The resulting f functions can be labelled f_Z, f_M, and f_E and used as the priority function in an A* algorithm searching for a path from start to finish.

For each of the three f functions, decide whether its use gives rise to an admissible search in this system. If it *is* admissible, prove it (i.e. explain why—informal proofs are fine). If it *isn't* admissible, devise a test case that would lead the algorithm to find a non-optimal path (and explain/show how the test case fails).

Problem 1.3

Consider again the word ladder problem from Project 0. You implemented it as a search process, probably a brute-force breadth-first search; consider now what would be required to implement it as an A* search. What would be the definition of g(X) (distance-to-start)? What would be an appropriate h(X) heuristic function? For your heuristic function, explain why it is admissible, and discuss whether there could be an even more informed (dominating) heuristic function.