Homework 2: Planning and Heuristic Search

15381: Artificial Intelligence (Fall 2013)

TAs: Subhodeep Moitra, Jesse Dunietz, Danny Zhu

OUT: Sep 11, 2013
DUE: Sep 24, 2013, 11:59 AM

Submission Procedure

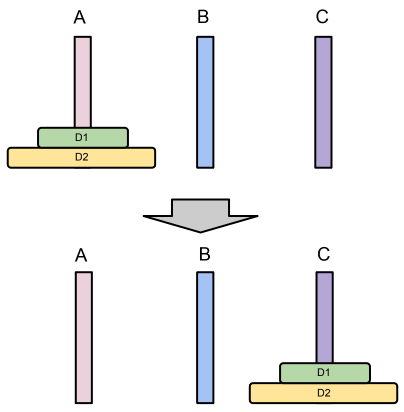
Submission will be via AFS. See instructions on Piazza.

- *Theoretical Problems:* Your answers to the written part should be submitted as a PDF file. You can either typeset your submission or scan in handwritten solutions.
- *Programming Problem:* You are encouraged to do the programming part in pairs. If you do, please only submit one copy of the required files. In your submission directory place a file named `README`. The first line of this file should contain the andrew ids of both partners separated by a single comma and nothing more (for example: jdunietz,dannyz).

1 Towers of Hanoi, 30 points [Subhodeep]

We will attempt to solve the famous problem of *"Towers of Hanoi"* using classical planning. It consists of three pegs, and a number of discs of different sizes which can slide onto any peg. The puzzle starts with the discs in a neat stack in ascending order of size on one peg, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another peg, obeying the following rules. (1) Only one disc may be moved at a time. (2) Each move consists of taking the upper disc from one of the pegs and sliding it onto another peg, on top of the other discs that may already be present on that peg. (3) No disc may be placed on top of a smaller disc.

Figure 1: Towers of Hanoi



Consider a problem instance in which the pegs are labeled: A, B, and C. There are two discs, d_1 and d_2. d_1 is smaller than d_2. d_1 and d_2 are initially stacked on peg A, and the goal is to move both of them, with correct ordering, to peg C.

- `disc`(x): States that an item x is a disc.
- `peg`(x): States that an item x is a peg.
- `clear`(x): States that the top of an item x is clear. x can be a disc or a peg.
- `on`(x, y): States that an item x is directly on top of an item y. x is a disc and y is a disc or a peg. If y is a peg, this asserts that x is the bottom disc on peg y.
- `smaller`(x, y): States that an item x is smaller than an item y. x is a disc and y is a disc or peg.
- The only action permitted is `MOVE`(d, s, t). This states that item d, which is on item s, should be moved to be on item t. d is a disc; s and t can each be a disc or a peg.

Also assume the following facts. You may exclude these facts from your state descriptions, since they are always constant.

- $\forall x \in \{d_1, d_2\}$: `disc`(x)
- $\forall x \in \{A, B, C\}$: `peg`(x)
- $\forall x \in \{d_1, d_2\}, y \in \{A, B, C\}$: `smaller`(x, y)
- `smaller`(d_1, d_2)

1. Give the initial state, the goal state, and the action schema (with its precondition and effect). **[8 points]**
2. Solve the planning problem using A* search with the bad heuristic function $h(x) = 0$ for all x. The cost of a plan is the number of actions that need to be taken. Draw out the search tree. Each node in the tree should be a state description, i.e., a conjunction of conditions. The edges from a node correspond to the actions possible from that state. You may omit nodes that are repetitions of previously reached states. *Note: You need not describe the states in complete detail. The conditions that you need to track are those that are changing/changed.* **[8 points]**
3. What is the relationship between this A* scheme and uniform cost search? **[2 points]**
4. Construct the levels 0-1 of the planning graph for the problem (i.e., your graph should contain 2 levels of states and 1 level of actions). Again, you can omit literals that always stay constant. Indicate mutex edges in a different edge style or color. Omit mutex edges between literals that are negations of each other and between the corresponding no-ops (though you should still consider them when constructing your graph). *Note: You may use a graph layout tool such as Graphviz if the planning graph gets too messy to draw by hand. If you drew out any additional layers of the planning graph correctly, you will receive an additional 4 bonus points.* **[8 points]**
5. Suppose the goal G is a conjunction of predicates $G_1, G_2, \cdots, G_n$. Consider plans $P_1, \cdots, P_n$, where each P_i achieves G_i. How might you use the number of levels in the planning graph you constructed to compute $Cost(P_i)$, a heuristic estimate of the cost to execute P_i? **[2 points]**
6. We want to combine $Cost(P_i)$ to get a heuristic estimate for the cost of P, a plan for achieving G. Which of the following is an admissible heuristic: $Cost(P) = \max_i Cost(P_i)$ or $Cost(P) = \sum_i Cost(P_i)$? Why is it admissible? **[2 points]**

2 Pacman, 70 points [Danny]

Instructions for this section can be found at `https://cs.cmu.edu/~15381/hw/hw2/pacman/`.