CS 182: Problem Set 2

Alan Turing

August 29, 2018

Introduction: Welcome to the second official homework for CS182! As you are hopefully already aware, this PDF comprises the written component of the second problem set. In addition to solving the problems found below, you will also need to complete the coding part of the assignment, found in the Github repo. Finally, we'd like to remind you that while you are allowed a partner for the coding part of the assignment, you are **NOT** allowed a partner for this and all future written components. All written work should be yours and yours alone. This being said, in addition to being able to ask questions at office hours, you are allowed to discuss questions with fellow classmates, provided 1) you note the people with whom you collaborated, and 2) you **DO NOT** copy any answers. Please write up the solutions to all problems independently.

Collaborators:

Problem 1: In addition to minimax/expectimax, another way to design an agent to play Pacman is to create an agent that uses an evaluation function to decide its actions. This means that at each step, the agent uses an evaluation function to rate each possible action (i.e. a direction that Pacman can move) and chooses the action with the best rating. Describe an evaluation function that could be used to play Pacman successfully.

Solution 1:

Problem 2: Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?

Solution 2:

Problem 3: AC-3 puts back on the queue every arc (X_k, X_i) whenever any value is deleted from the domain of X_i, even if each value of X_k is consistent with several remaining values of X_i. Suppose that, for every arc (X_k, X_i), we keep track of the number of remaining values of X_i that are consistent with each value of X_k. Explain how to update these numbers efficiently and hence show that arc consistency can be enforced in total time $O(n^2d^2)$.

Hint: For the sake of simplicity, assume that this new algorithm will require two new data structures not seen in AC-3. Namely, let $P(X_i, d_i)$ contain all consistent values $\{(X_j, d_j)\}$ in each of X_i's neighbors such that all pairs (d_i, d_j) are consistent. And, let $C(X_i, d_i, X_j)$ be the number of consistent domain values $d_j \in X_j$ that are consistent with d_i. Think about the prepossessing time it would take to fill these two data structures, and then think about how these two data structures allow us to replicate the behavior of AC-3.

Solution 3: