HW 1

Due: Thursday 11/02/2023 @ 11:59pm EST

Disclaimer

I encourage you to work together, I am a firm believer that we are at our best (and learn better) when we communicate with our peers. Perspective is incredibly important when it comes to solving problems, and sometimes it takes talking to other humans (or rubber ducks in the case of programmers) to gain a perspective we normally would not be able to achieve on our own. The only thing I ask is that you report who you work with: this is **not** to punish anyone, but instead will help me figure out what topics I need to spend extra time on/who to help. When you turn in your solution (please use some form of typesetting: do **NOT** turn in handwritten solutions), please note who you worked with.

Question 1: World Engineering (20 points)

Consider playing a game where you have three jugs and a sink. One jug can contain 3 gallons of water, another can contain 8 gallons of water, and another can contain 12 gallons of water. In this game, when filling up a jug at the sink, you must fill it entirely (you cannot partially fill up a jug at the sink). You may also decide to pour out the contents of a jug, but if you do so, you must completely empty the jug. While you can certainly always dump out the entire contents of a jug into the sink, you cannot empty one jug into another jug if the water will overflow. You also cannot partially empty a jug, so you cannot for example choose to empty the 12 gallon jug containing 12 gallons into the 3 or 8 gallon jug because an overflow is certain to occur. The game ends when you have a jug that contains exactly 1 gallon (doesn't matter which jug contains the 1 gallon). Design the world (i.e. give me the things we need to define such as an initial state, the actions, etc.) so that an agent may solve it in the optimal number of moves.

Question 2: Reverse Engineering Edge Weights(20 points)

Consider running A^* on a discrete world where every vertex in the world is associated with 2d integer coordinates. Since we are choosing to run an A^* agent, we are responsible for designing the graph that models the world. Let us choose a distance function (such as manhattan distance) that calculates the distance from a vertex to the goal state by inspecting coordinates. In order for A^* to be optimal, our distance function must be *admissible* and *consistent*, which must be determined on a world-by-world basis. In order for our distance function to be admissible and consistent, what is the smallest edge weight that we can allow in our graph? I am asking you to work backwards: instead of using the properties of the graph to determine if a distance function is admissible and consistent, I am saying we are picking a distance function and need to engineer the edge costs in such a way that allows the distance function to be admissible and consistent. Prove that, if every edge weight in the graph (which we have to decide as part of the graph modeling process) satifies your bound, that our distance function will be consistent and admissible.

Question 3: CSP Reduction (20 points)

Prove that any n-ary constraint can be converted into a set of binary constraints. Therefore, show that all CSPs can be converted into binary CSPs (and therefore we only need to worry about designing algorithms to process binary CSPs).

Hint: When reducing a n-ary constraint: consider adding synthetic variable(s) (i.e. inventing new variables). Each synthetic variable should have a domain that comes from the cartesian product of the domains of the original variables involved in that constraint. We can then replace the original constraint with a set of binary constraints, where the original variables must match elements of the synthetic domain's value.

Question 4: Correctness of Alpha-Beta Pruning (20 points)

Let s be the state of the game, and assume that the game tree has a finite number of vertices. Let v be the value produced by the minimax algorithm:

$$v = \texttt{Minimax}(s)$$

Let v' be the result of running Alpha-Beta Pruning on s with some initial values of α and β (where $-\infty \leq \alpha \leq \beta \leq +\infty$):

$$v' = \texttt{Alpha-Beta-Pruning}(s, \alpha, \beta)$$

Prove that the following statements are true:

- If $\alpha \leq v \leq \beta$ then $v' = v$
- If $v \leq \alpha$ then $v' \leq \alpha$
- If $v \geq \beta$ then $v' \geq \beta$

This means that if the true minimax value is between α and β, then Alpha-Beta pruning returns the correct value. However, if the tru minimax value if outside of this range, then Alpha-Beta pruning may return a different value. However, the incorrect value that Alpha-Beta pruning returns is bounded in the same manner that the true minimax value is (i.e. if the true minimax value is $\leq \alpha$ then the value produced by Alpha-Beta pruning is also $\leq \alpha$ and vice versa). Note that this implies that Alpha-Beta pruning will be correct with initial values of $(-\infty, +\infty)$ for (α, β).

Hint: use induction. If s is not a terminal state, then you can correctly assume that the claim above holds for all children of s. Use this assumption to prove that it also holds for s (the base case is trivial: minimax and Alpha-Beta pruning produce the same value for terminal states)

Extra Credit: Gradient Descent and Lipschitz Continuity (20 points)

Gradient Descent is the continuous version of hill climbing where we use the gradient of the objective function to measure which direction leads uphill (we can then decide whether or not to follow the gradient uphill or downhill). Let us consider going downhill (i.e. we are choosing to *minimize*). The trouble is that gradient descent is a local search algorithm, meanining that it is not guaranteed to:

- Converge at all. There may not be any local minima to find.
- Arrive at the global minima if one exists.

A function f is *Lipschitz Continuous* with constant $k > 0$ if $\forall \vec{x}, \vec{y} \quad ||f(\vec{x}) - f(\vec{y})||_2 \leq k||\vec{x} - \vec{y}||_2$. Think of it this way: for every pair of points, f is bounded on how fast it can change by k. Here is a good gif that visualizes lipschitz continuity.

If we know that our objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, *and* we know that the gradient of f (i.e. ∇f) is Lipschitz continuous with (minimum) constant $c > 0$, prove that **not only** is gradient descent **guaranteed** to converge (for specific values of the step size η), but that it **will** converge if $\eta \leq 1 - \frac{2}{c}$.

Hint: you may find it useful to use a 2nd degree taylor polynomial of f centered around $f(\vec{x})$, which can be expressed as:

$$f(\vec{y}) \leq f(\vec{x}) + \nabla f(\vec{x})^T(\vec{y} - \vec{x}) + \frac{1}{2}\nabla^2 f(\vec{x})(\vec{y} - \vec{x})(\vec{y} - \vec{x})^T$$