Andrew Thompson u0879848

3.1

2.

For my features, I used the distance to the closest food, the amount of food remaining, the distance to the closest ghost, being in a win or lose state, and taking a stop action. These are all weighted accordingly.

3.2

2.

Pacman will rush to the closest ghost because it would be more costly to keep living, that is, the values of the states where the death is prolonged are smaller than the values of the states/actions leading to the quick death.

3.4

2.

As noted in the project, Expectimax does indeed win about half the time when tested on trappedClassic for example. In this map, Pacman is trapped between two ghosts. In Minimax, it expects the ghosts to close in on him, and thus, as discussed above, pacman runs into the ghosts as soon as possible to minimize cost. For Expectimax, Pacman takes a chance that that the ghosts will not act perfectly, or in this case, not close in on him, and thus continues to try to reach the dots beyond the ghost.

3.5

2.

I used a lot of the same features as I did before. I used the distance to the closest food, the amount of food remaining, the distance to the closest ghost, and being in a win or lose state, but this time I also used the distance to the closest capsule, and whether the ghosts are scared. Capsules and food increase value, and moving closer to ghosts or not collecting food lowers value. If ghosts are scared, moving towards them is increases value. Reaching the start or end state yields the highest value. Though, when there is little food left, pacman will sometimes just stop until the ghost gets close, and then move again, so it could probably be improved. But I'm meeting the autograder requirements, so I'll leave it.

4.

1.

The hardest part of the assignment was trying to apply the pseudocode within the constraints of Pacman. For example, it was kind of hard to figure out what exactly to use as the "successor of state" or the state's utility.

2.

The easiest part of the assignment was doing Alpha-Beta and Expectimax after having done Minimax. It was largely the same code for both of those implementations, with minor changes.

3.

Andrew Thompson u0879848

Problems 2, 3, and 4 helped my understanding of the material the most.

4.

Problems 1 and 5 weren't as helpful as the others. They were a bit too open-ended, a little bit of a guided exercise here might have helped me to see what makes a good evaluation function.

5.

The feedback for #4 is probably my biggest complaint, but my praise would be the autograder, it was informative and helped me see where I was going wrong for the most part.