AI Project 2

Bhushan Sonawane (111511679) **Nishant Borude (111447198)**

1. Reflex Agent

- The reflex agent given earlier returned the score of the gameState.
- For the pacman to outperform ghosts and eat all the food, we need to consider all the food locations and ghost positions at that state.
- Our reflex agent calculates the distance to all food locations and distance to all ghosts in the grid. It then selects the closest food pellet and closest ghost from pacman.
- Then, the reflex agent returns the score of the current state + closest ghost distance – closest food distance. i.e. the score will be maximum if the ghost is far and food is close and alternatively, the score will be minimum if ghost is close and food is far.
- **Statistics for smallClassic maze:**
    1. Average Score: 1090.8

2. Minimax

- The objective of the minimax algorithm is to get the scores of all levels(min for ghosts and max for pacman) upto depth d, propagate them up to and then select an action that gives the maximum score at that current state.
- At the current state, pacman has to select an action, thus we call the max function to get the max value of the action. For each action in max, we call the min function of the next state. Considering there are multiple ghosts, we again call min function for all such ghosts and then at the end call the max function by decreasing the depth by 1.
- We repeatedly call the min and max functions until it reaches the depth 0 where it evaluates the score of the state.
- **Statistics for smallClassic maze:**
    1. Nodes expanded: 11280
    2. Running Time: 1216.55 ms
    3. Memory Used: 853.248 KB

3. Alpha Beta Pruning

- Alpha Beta pruning expands a fewer nodes that minimax algorithm. Alpha is initialized to negative infinity and beta is initialized to positive infinity.

- We pass alpha and beta to the max and min functions. Here, every max layer updates the value of alpha to the max of the scores in that layer and min layer updates the value of beta to the min of the scores in that layer.

- We use backtracking to keep a track of alpha and beta at each layer and compare alpha at max layer and beta at min layer with the value returned by their respective successors.

- We stop checking the values at that layer when alpha becomes greater than beta, thus expanding fewer nodes.

- **Statistics for smallClassic maze:**

    1. Nodes expanded: 9386

    2. Running Time: 1026.04 ms

    3. Memory Used: 767.82 KB

4. Expectimax

- Expectimax is very similar to the minimax algorithm except we consider a chance layer instead of a min layer.

- The max node calls the chance nodes in the next layer which selects the average score in that layer and passes it to the calling function. Then the topmost max layer selects an action closest to this average score.

- **Statistics for smallClassic maze:**

    1. Nodes expanded: 11242

    2. Running Time: 1209.00 ms

    3. Memory Used: 798.57 KB

Utility used

1. For Running time: Python utility time()

2. For Memory used: Python utility resource’s RUSAGE_SELF