COMP341 INTRODUCTION TO ARTIFICIAL INTELLIGENCE ASSIGNMENT-2 REPORT

Name-Surname: Barış Kaplan
KU ID Number: 0069054
KU Email Address: bkaplan18@ku.edu.tr

1-) In my evaluation function, I have used the features named the negative reciprocal of the closeness to the ghosts, the positive reciprocal to the closeness to the foods, the score of game, and the length of list keeping the places of the existing capsules. I have used the negative reciprocal of the closeness value to the ghosts. Because Pacman should be away from the ghosts to continue the game. Therefore, the distance value to the ghosts should be high. By using negative reciprocal of the closeness to ghosts, I obtained a more accurate evaluation function. I have used the reciprocal of the closeness to foods. The closeness to foods is a good metric for keeping Pacman move towards the foods. As the closeness to food decrease, we should obtain higher evaluation function value. Thus, I have used positive reciprocal of this closeness. As Pacman continues to eat foods, the game score increases and Pacman becomes closer to goal state. Thus, the game score is a supportive metric to use in the evaluation function. Finally, I utilized the length of the list keeping capsules. Because using capsules decreases the number of ghosts and increases the goodness of state. It is good to utilize the negatives and reciprocals of some metrics. Because some metrics are inversely proportional to the return value of the evaluation function and the goodness/promisingness of state.

```
subseq_value_of_score = successorGameState.getScore()
first_denominator = (adversary_closeness_metric + 11)
second_denominator = (14 + closness_to_grid)

initial_bool = (first_denominator > 0)
second_bool = (second_denominator > 0)
denominatorsNotZero = (initial_bool and second_bool)

if denominatorsNotZero:
    return 7 + 3 * (
            float(-2.86) / first_denominator + float(4.15) / second_denominator +
            35 * subseq_value_of_score - len(currentGameState.getCapsules())
    )
```

Figure 1: The evaluation function that I used for Reflex Agent.

2-) After 20 seconds, I observed that the score of Minimax agent is 18 and the score of AlphaBeta agent is 159. Moreover, I observed that the speed of the AlphaBeta agent is higher than the speed of the Minimax agent. Furthermore, for 20 seconds, I have observed that the total number of moves is approximately 7 for the Minimax agent and approximately 30 for the AlphaBeta agent. Thus, for the AlphaBeta agent, we can see that the game score after 20 seconds and the total number of moves during this 20 seconds are higher than the values of the same statistics for the Minimax agent. These situations/metrics indicate and bring a higher speed for the AlphaBeta agent.

3-) Yes, the pacman behave same in the AlphaBeta agent case and the Minimax agent case. Unlike the Minimax algorithm, the process of Alpha-Beta pruning just removes the states/nodes which the agent cannot choose and which do not influence the action decision of the agent. In other words, Alpha-Beta pruning just reduces the total amount of the nodes which the agent should traverse. Moreover, since the Alpha-Beta pruning gets rid of some redundant states/nodes, Alpha-Beta's action decisions are faster than the Minimax. Therefore, utilizing the process of Alpha-Beta pruning does not alter the moves/actions which are selected by the Minimax agent. We can conclude from this situation that the behaviors of the pacman are identical for AlphaBeta and Minimax.

4-) For the Expectimax case, the code runs slower than the AlphaBeta case. However, the code runs a bit faster than (closer to the Minimax case) the Minimax case. Therefore, I can say that the speed of runs for the Expectimax and Minimax are very similiar. I have expected these situations because both of the Expectimax and Minimax algorithms do not remove/prune the states/nodes which are unnecessary to check for agent (the states/nodes which the agents cannot choose). In other words, Minimax and Expectimax algorithms traverse all states. Nonetheless, the Alpha-Beta pruning performs this removal/pruning process for the states/nodes which are not required to be searched. Therefore, Alpha-Beta pruning is faster than the Expectimax and Minimax. Expectimax just adds the probabilities to the logic of Minimax. By using some chance nodes, Expectimax obtains the expectation from the children values. Minimax just obtains the minimum or maximum of children values depending on the nodes.

5-) I have changed the values of the weights of features that I have used in the evaluation function that I have implemented in the first question (Q1). While changing the weight values, I have considered whether the features I have used are directly proportional or inversely proportional to the return value of the evaluation function and thus to the promisingness/goodness of the state of Pacman. For the new evaluation function, I have taken just states as inputs and performed evaluations on these states. Nonetheless, for the initial evaluation function, in addition to the states, I have also utilized the actions as inputs and performed evaluations on these actions. As an another difference, instead of obtaining the score of the successor game state, I have used score of the current game state for the new evaluation function. ("Current Game State" is provided in the new evaluation function. "Successor Game State" is provided in the evaluation function in the first question).

```
subseq_value_of_score = currentGameState.getScore()
denominator1 = (adversary_closeness_metric + 15)
denominator2 = (8 + closness_to_grid)

first_denominator_condition = (denominator1 != 0)
second_denominator_condition = (denominator2 != 0)
denominatorsNotEqualToZero = first_denominator_condition and second_denominator_condition
if denominatorsNotEqualToZero:
    firstPart = (float(-2.97) / denominator1)
    secondPart = (float(5.07) / denominator2)
    thirdPart = 43 * subseq_value_of_score
    my_evaluation_variable_with_weights = (firstPart + secondPart + thirdPart) - len(currentGameState.getCapsules())
    return 17 + 5 * my_evaluation_variable_with_weights
```

Figure 2: My new evaluation function with the features with new weights

6-) While deciding whether to use negative, positive or reciprocal weights for the features, I considered whether the changes in the features decrease or increase the goodness of state (see Figure1 & Figure2 for observing weight assignments and changes). The main goal of the Pacman game is to be away from the ghosts, closer to the foods, eat more ghosts, and thus have more score. Thus, I have given negative reciprocal weight to closeness to a ghost, positive reciprocal weight to closeness to a food, and negative weight to capsule list length (see Figure 1&2). While deciding values of new weights of the features I used, I have performed an exploration (decreased/increased the weights and examined the effects of these changes on evaluation function). In this process, I have realized that the closeness features I used have more influence on evaluation function (few changes in closeness features resulted in high changes in evaluation function). Thus, instead of "eating ghosts" by consuming capsules, I have prioritized "being near to foods" and "being away from ghosts". Thus, to have better evaluation functions in Q1 and Q5, I have given more weights (&paid more attention) to the distances to foods and ghosts.

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_i^n w_i f_i(s)$$

The general evaluation function formula that I have followed while implementing Q1 and Q5 (screenshot taken from the lecture notes)

```
***     1158.6 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###
```

Figure 3: Average score of my new evaluation function in Figure2

```
*** PASS: test_cases/q5/grade-agent.test (6 of 6 points)
***     1158.6 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###
```

Figure 4: Average score of new evaluation function for weight values in Figure 3 but the weight of score is 103 instead of 43 in this case.

```
*** PASS: test_cases/q5/grade-agent.test (6 of 6 points)
***     1036.0 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###
```

Figure 5: Average score of new evaluation function where the numerator of the weight of closeness to ghost is 10 and all other weights are same with Figure3. (See the change in average score of new evaluation function ***(1158.6 vs 1036.0)).***

```
*** PASS: test_cases/q5/grade-agent.test (6 of 6 points)
***     1030.2 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###
```

Figure 6: The average score of new evaluation function if I only change the numerator of the weight of closeness to food from 5.07 to 1.07 (all other weights are same with Figure 3, see the change in average score by comparing Figure 3 and Figure 6 ***(1158.6 vs 1030.2)).***