Question Number 6

Success in the game

1. Theoretically, which of the algorithms is more suitable for the game 2048? Explain what each of the algorithms assumes about the opponent and which of the assumptions better suits the game.

> the expectimax assumes there is a probability for states between actions of the player, that means we dont know for sure what will be the next state. the minimax assumes there are 2 players (Usually) that compete against each other in a 0 sum game so one play gain is other player loss and therfore we assume each player does the best action for himself. . Theoretically expectimax better suits the game becuase the assumption it uses, in the game the tiles of the game are generated randomly and therfore the next state is unknown to the player.

2. Empirically, which of the algorithms is more successful in 2048? To answer the question, run each of the algorithms for 10 games and display the scores and the largest tiles that each of the algorithms achieves.

> **Running AlphaBeta Agent**:
>
> ```
> python3 2048.py --agent=AlphaBetaAgent --depth=2 --
> num_of_games=10 --display=SummaryDisplay
> ```
>
> | game number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
> |---|---|---|---|---|---|---|---|---|---|---|
> | score | 3260 | 11848 | 5900 | 6768 | 7160 | 3212 | 6948 | 6408 | 11880 | 7332 |
> | highest | 256 | 1024 | 512 | 512 | 512 | 256 | 512 | 512 | 1024 | 512 |
>
> **Running Expectiation Agent**
>
> ```
> python3 2048.py --agent=ExpectimaxAgent --depth=2 --
> num_of_games=10 --display=SummaryDisplay
> ```
>
> | game number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
> |---|---|---|---|---|---|---|---|---|---|---|
> | score | 14468 | 16156 | 14636 | 7392 | 11912 | 6332 | 5620 | 3220 | 6752 | 6680 |
> | highest | 1024 | 1024 | 1024 | 512 | 1024 | 512 | 512 | 256 | 512 | 512 |
>
> According to the data both algorithms recived the same highest tile 1024, But as is seems the `ExpectimaxAgent` did better and achive 4 times 1024 in compare to 2 times that

> `MinMaxAgent` achived. Also the the average score of `MinMaxAgent` is 7071.6 in compare to 9316.8 that `ExpectimaxAgent` achived. Those results indicate that in the empirical test the `ExpectimaxAgent` won.

3. Are the theoretical and empirical results consistent with each other? If so, explain. If not, explain how this is possible.

> The theoretical and empirical consists with each other , we can see that indeed the empirical results of the `ExpectimaxAgent` are better, with `ExpectimaxAgent` we get most of the time better results then the `MinMaxAgent` becuase the algorithem choose the path that will have the most likely the better results , while the `MinMaxAgent` allways takes the "the best worst solution" , and thats why we can see the results of it are more consistents around 6500 and a max tile of 512. in addition the `MinMaxAgent` get each action the worst tile placed he can get , while in `ExpectimaxAgent` we get the worst tile , best tile andothers that canbe placed with equal probability ,so thats better than getting the worst tile with probability of 100%.

4. Compare the standard deviations in the score and the largest tile of the two algorithms. Give an intuitive explanation for the difference in results.

> | | **MinMaxAgent** | **ExpectimaxAgent** |
> |---|---|---|
> | STD | 2781.287 | 4304.887 |
> | Highest Score | 1024 | 1024 |
>
> The `ExpectimaxAgent` algorithm makes decisions based on expectations, considering the best possible outcome rather than the worst-case scenario. As a result, when the opponent makes a move the `ExpectimaxAgent` does not always prioritize the worst-case scenario, and therefore the score spread more and it depends on the opponent move. In contrast to the `MinMaxAgent` is often able to anticipate and respond effectively, as it evaluates the best possible move among the opponent's suboptimal choices. Since it always thinks about the worst-case scenario. Consequently, we observe that the standard deviation of the `ExpectimaxAgent` is greater than that of the `MinMaxAgent`.

Alternative games

1. Imagine an alternative game "2048 Boom" where after every move of the player there is a one in a billion chance of a "boom" where the game stops and the player finishes with a score of 0. How will the change affect the performance of the algorithms? In this section only, assume an exact expectimax algorithm that takes into account the probabilities of the moves.

> As we said earlier since the `MinMaxAgent` taking in consideration the worst-case scenario in this "Alternative Game" the worst-case scenario (even with minor probability) is the game stop suddenly and with a boom and score of 0. the `MinMaxAgent` will not works as we wished since all steps will recieved score of 0 and then it choose step randomly and will play as RandomAgent. On the contrary `ExpectimaxAgent` will not be effected so much from this change since the probability of "BOOM" is minor and the Agent works on expectation of the opponent steps.

2. In our implementation the expectimax player does not calculate the exact expectation but assumes that all outcomes have equal probability. Use this fact to create a new simple game where the minimax player outperforms this not-exactly-expectimax player.

> In Nim the barnching factor of the opponent is very small what will make the running of the `MinMaxAgent` more faster and therefore the `MinMaxAgent` could run in greater depths. In addition its known that in Nim if you start the game you can always win and the `MinMaxAgent` has the exact ability to think forward and win the game. While the `ExpectimaxAgent` will not be able to thinks due to it's needed to take in considerations all the opponent steps. The `ExpectimaxAgent` gives bad and good actions same probability even when a player wont take a bad action if it have a good one to take , therfore the assumption of `ExpectimaxAgent` will hurt him , while the minmax agent will take the best action he can allways . therfore in the nim game he will be superior. (Tic Tac Toe will work as well).