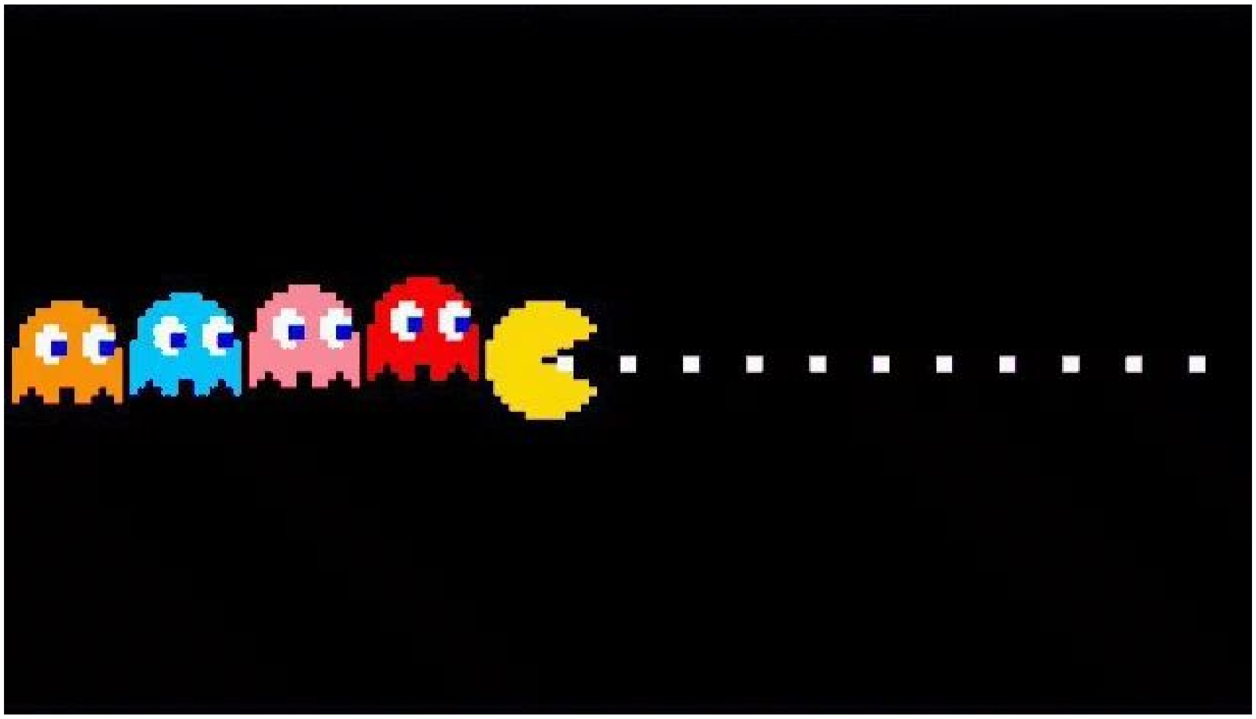
CSE 537
ARTIFICIAL INTELLIGENCE

Project 2 - PACMAN

REPORT

Submitted on 8th October 2018



Presented By**:**

Debjyoti Roy
112070373

Siddarth Harinarayanan
112026390

Q 1. Reflex Agent

One Ghost

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
ReflexAgent -k 1
Pacman emerges victorious! Score: 1189
Expanded Nodes :  1494
Average Score: 1189.0
Scores:        1189.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Two Ghosts

```
debjyotis-mbp:multiagent debroy$ python pacman.py --frameTime 0 -p ReflexAgent -k 2
Pacman emerges victorious! Score: 1566
Expanded Nodes:  1330
Average Score: 1566.0
Scores:        1566.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Open Classic Layout

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
ReflexAgent -l openClassic -k 2
Expanded Nodes :  520
Pacman emerges victorious! Score: 1250
Average Score: 1250.0
Scores:        1250.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Q 2. MiniMax Agent

Pacman Loses:

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py -p MinimaxAgent -l
minimaxClassic -a depth=4
Pacman died! Score: -495
Expanded Nodes:  21008
Average Score: -495.0
Scores:        -495.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

```
Pacman Wins:
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py -p MinimaxAgent -l
minimaxClassic -a depth=4
Pacman emerges victorious! Score: 516
Expanded Nodes:  6115
Average Score: 516.0
Scores:        516.0
Win Rate:      1/1 (1.00)
Record:        Win

On Trapped Classic (loses)
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py -p MinimaxAgent -l
trappedClassic -a depth=3
Pacman died! Score: -501
Expanded Nodes:  77
Average Score: -501.0
Scores:        -501.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Q 3 Alpha Beta Pruning

We are not pruning on equality , i.e. we are checking if alpha > beta is true, then we are not pruning the tree(exploring the child nodes) and we get a full score (5/5). If we modify the same to prune on equality (alpha>=beta), then we get a score 0/5 on running the autograder.

```
Pacman Wins (minimaxClassic)
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
AlphaBetaAgent -l minimaxClassic -a depth=3
Expanded Nodes :  653
Pacman emerges victorious! Score: 512
Average Score: 512.0
Scores:        512.0
Win Rate:      1/1 (1.00)
Record:        Win

Pacman Wins (smallClassic)
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
AlphaBetaAgent -l smallClassic -a depth=3
Pacman emerges victorious! Score: 1045
Expanded Nodes :  40060
Average Score: 1045.0
Scores:        1045.0
Win Rate:      1/1 (1.00)
```

```
Record:        Win
```

Pacman Loses

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
AlphaBetaAgent -a depth=3 -l smallClassic
Pacman died! Score: -135
Expanded Nodes :  8948
Average Score: -135.0
Scores:       -135.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Pacman on trappedClassic for AlphaBetaPruning 10 times (all losses as expected)

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py -p AlphaBetaAgent -l
trappedClassic -a depth=3 -q -n 10
Expanded Nodes :  28
Pacman died! Score: -501
Expanded Nodes :  56
Pacman died! Score: -501
Expanded Nodes :  84
Pacman died! Score: -501
Expanded Nodes :  112
Pacman died! Score: -501
Expanded Nodes :  140
Pacman died! Score: -501
Expanded Nodes :  168
Pacman died! Score: -501
Expanded Nodes :  196
Pacman died! Score: -501
Expanded Nodes :  224
Pacman died! Score: -501
Expanded Nodes :  252
Pacman died! Score: -501
Expanded Nodes :  280
Pacman died! Score: -501
Average Score: -501.0
Scores:       -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0
Win Rate:      0/10 (0.00)
Record:        Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
```

Q 4. Expectimax Agent

Pacman Wins

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py --frameTime 0 -p
ExpectimaxAgent -l smallClassic -a depth=3
Pacman emerges victorious! Score: 1265
Expanded Nodes :  154805
Average Score: 1265.0
Scores:        1265.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Pacman on trappedClassic for ExpectiMax Agent 10 times (all loses as expected)

```
E:\StonyBrook CS\AI\ai\multiagent>C:\Python27\python pacman.py -p ExpectimaxAgent -l
trappedClassic -a depth=3 -q -n 10
Expanded Nodes :  357
Pacman emerges victorious! Score: 532
Expanded Nodes :  714
Pacman emerges victorious! Score: 532
Expanded Nodes :  1071
Pacman emerges victorious! Score: 532
Expanded Nodes :  1428
Pacman emerges victorious! Score: 532
Expanded Nodes :  1785
Pacman emerges victorious! Score: 532
Expanded Nodes :  1874
Pacman died! Score: -502
Expanded Nodes :  2231
Pacman emerges victorious! Score: 532
Expanded Nodes :  2588
Pacman emerges victorious! Score: 532
Expanded Nodes :  2677
Pacman died! Score: -502
Expanded Nodes :  3034
Pacman emerges victorious! Score: 532
Average Score: 325.2
Scores:        532.0, 532.0, 532.0, 532.0, 532.0, -502.0, 532.0, 532.0, -502.0, 532.0
Win Rate:      8/10 (0.80)
Record:        Win, Win, Win, Win, Win, Loss, Win, Win, Loss, Win
```

Critical Analysis:

We infer the following points after working with solving the pacman puzzles for various layouts, and scenarios using various search agents:

1. Reflex agent’s evaluation function uses the manhattan distance from the pacman’s position to the nearest food as the heuristic in a greedy fashion and moves ahead. It also takes into account the position of the ghost for choosing the best action for the pacman to move in the next state. This evaluation is naive in the way that it does not look a few steps ahead to determine whether it can find an optimal path (like, not fall into a trap of ghost/s).
2. Minimax agent implementation involves choosing the best possible action for pacman and the worst possible action for the remaining agents (all the ghosts). This is a recursive function which makes this choice over all the levels of choices of pacman and other agents. This is an optimal solution for finding the path and eating all the food for pacman. The disadvantage of using this agent though, is that it expands all the possible nodes for checking the best possible solution. Hence, it is not practically feasible for a large of agents and their choices. For e.g. on minimaxClassic with depth=3, it expanded 21008 nodes.
3. AlphaBeta agent implementation improves the Minimax agent by using the trick of backtracking to pass back values to the parent nodes to check if some nodes need not be expanded further, thus reducing the expanded nodes space. The alpha and beta values calculated in the children nodes and passed onto the parent to check if its worth expanding the remaining nodes in the parent or not. This helps in reducing the search space as we can see in the run of Alpha Beta Agent on minimaxClassic with depth=3, it expanded just 653 nodes, which is way less than 21008, which were expanded by the MiniMax Agent.
4. Expectimax Agent implementation helps when the opponent does not always play optimally, i.e. when there is an element of chance involved in the choice of the opponent. Expectimax Agent takes an average of the all the possible moves for the opponent and MAX value for our pacman. This improves the movement of the pacman for searching food and winning the game as shown in the example of trappedClassic maze. The AlphaBeta agent is not able to win even in a single case, whereas Expectimax Agent wins in majority of the cases.

REFERENCES:

- Minimax Agent : we referred the minimax agent pseudocode from the Wikipedia page.