**ASSIGNMENT 2: REPORT**

* **Q1: Evaluation function:**
* The parameters for the evaluation function are as follows:
* Always avoid the ghosts. Among the given pacman positions choose the one, whose distances to the ghosts are the maximum.
* With respect to a given pacman position, find out the food nearest to him. Nearer the food in a specific direction, the pacman will move in that direction
* Return the weighted sum of the heuristics

**C:\Python27\python.exe C:/Users/mahathi/Documents/artificial\_intelligence/multiagent/multiagent/pacman.py -p ReflexAgent -l testClassic**

**Pacman emerges victorious! Score: 489**

**Average Score: 489.0**

**Scores: 489.0**

**Win Rate: 1/1 (1.00)**

**Record: Win**

* **Q2: MiniMax Algorithm**
* The algorithm for the minimax algorithm as defined in Wikipedia is as follows:

**function** minimax(node, depth, maximizingPlayer)

**if** depth = 0 **or** node is a terminal node

**return** the heuristic value of node

**if** maximizingPlayer

bestValue := -∞

**for each** child of node

val := minimax(child, depth - 1, FALSE)

bestValue := max(bestValue, val)

**return** bestValue

**else**

bestValue := +∞

**for each** child of node

val := minimax(child, depth - 1, TRUE)

bestValue := min(bestValue, val)

**return** bestValue

This has been generalized to accommodate the following changes:

* For the min player, generate all possible moves of all the ghosts existing.
* Unless and until all the moves of the ghosts are generated for a given pacman position, the turn of pacman does not come.

Modified MiniMax Algorithm:

**function** minimax(node, depth, turnOfPlayer)

**if** depth = 0 **or** node is a terminal node

**return** the self.evaluationFunction()

**if** **turnofPacman**

bestValue := -∞

**for each** child of node

val := minimax(child, depth - 1, turnOfGhost)

bestValue := max(bestValue, val)

**return** bestValue

**if turnOfGhosts**

bestValue := +∞

if **all ghostsuccessors generated**

**for each** child of node

val **:=** minimax(child, depth - 1, turnOfPacman)

bestValue := min(bestValue, val)

else

**for each** child of node

val := minimax(child, depth, turnOfGhosts)

bestValue := min(bestValue, val)

**return** bestValue

Output:

C:\Python27\python.exe C:/Users/mahathi/Documents/artificial\_intelligence/multiagent/multiagent/pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4

nodes explored :

370

nodes explored :

740

nodes explored :

999

nodes explored :

1075

nodes explored :

1196

nodes explored :

1453

Pacman emerges victorious! Score: 514

Average Score: 514.0

Scores: 514.0

Win Rate: 1/1 (1.00)

Record: Win

* **Q3: MiniMax with alpha beta pruning**

The algorithm for alpha beta pruning as defined in Wikipedia is as follows:

01 **function** alphabeta(node, depth, α, β, maximizingPlayer)

02 **if** depth = 0 **or** node is a terminal node

03 **return** the heuristic value of node

04 **if** maximizingPlayer

05 v := -∞

06 **for each** child of node

07 v := max(v, alphabeta(child, depth - 1, α, β, FALSE))

08 α := max(α, v)

09 **if** β ≤ α

10 **break** *(\* β cut-off \*)*

11 **return** v

12 **else**

13 v := ∞

14 **for each** child of node

15 v := min(v, alphabeta(child, depth - 1, α, β, TRUE))

16 β := min(β, v)

17 **if** β ≤ α

18 **break** *(\* α cut-off \*)*

19 **return** v

* The modified minimax algorithm, has been accommodated in the alpha beta pruning algorithm.

01 **function** alphabeta(node, depth, α, β, turnOfPlayer)

02 **if** depth = 0 **or** node is a terminal node

03 **return** the heuristic value of node

04 **if** **turnOfPacman**

05 v := -∞

06 **for each** child of node

07 v := max(v, alphabeta(child, depth - 1, α, β, FALSE))

08 α := max(α, v)

09 **if** β ≤ α

10 **break** *(\* β cut-off \*)*

11 **return** v

12 **if turnOfGhosts:**

13 v := ∞

if **all ghostsuccessors generated**

**for each** child of node

**v =** alphabeta(child, depth - 1, α, β ,turnOfPacman)

β := min(β, v)

17 **if** β ≤ α

18 **break** *(\* α cut-off \*)*

19 **return** v

else

**for each** child of node

v = alphabeta (child, depth, α, β ,turnOfGhosts)

β := min(β, v)

17 **if** β ≤ α

18 **break** *(\* α cut-off \*)*

**return** v

Output :

nodes expanded

22464

Pacman died! Score: 74

Average Score: 74.0

Scores: 74.0

Win Rate: 0/1 (0.00)

Record: Loss

nodes expanded

18008

Pacman died! Score: 252

Average Score: 252.0

Scores: 252.0

Win Rate: 0/1 (0.00)

Record: Loss

nodes expanded

13779

Pacman emerges victorious! Score: 958

Average Score: 958.0

Scores: 958.0

Win Rate: 1/1 (1.00)

Record: Win

|  |  |  |  |
| --- | --- | --- | --- |
| Execution Time | Eval Function | MiniMax | AlphaBeta |
| Classic Layout | 30 seconds |  |  |
| Medium Classic |  | 3 seconds |  |
| Maze |  |  | 59 secs |

References:

* <http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning>
* <http://en.wikipedia.org/wiki/Minimax>