Universidad del Valle de Guatemala
Inteligencia Artificial
Sec. 20

Proyecto No. 2

Inteligencia artificial

Javier Chavez 21016
Marco Ramirez 21032
Andres Quezada 21085
Josué Morales 21116

Problema #1: Reflexión estratégica

- Explicación de la implementación

Para implementar este algoritmo hicimos lo siguiente: Como grupo propusimos que en lugar de planificar acciones a largo plazo, como en un agente basado en búsqueda, este agente elige acciones en cada punto de decisión evaluando directamente el estado actual.

En el método getAction, el agente examinará las acciones legales disponibles y calculará puntajes para cada una usando una función de evaluación. Esta función de evaluación está definida en el método evaluationFunction. Aquí, se calculan tres componentes principales:

- **Evaluación de la comida:** Calcula la distancia de Manhattan entre la nueva posición del Pac-Man y cada comida restante en el tablero. Luego, se promedian para obtener una evaluación general de la distribución de la comida.
- **Evaluación de los fantasmas:** Similar al cálculo de la comida, se calcula la distancia de Manhattan entre el Pac-Man y cada fantasma en el tablero. Nuevamente, se promedian para obtener una evaluación general de la proximidad de los fantasmas.
- **Evaluación del tiempo asustado de los fantasmas:** Calcula el tiempo restante en que los fantasmas permanecerán asustados después de que el Pac-Man haya comido una pastilla de poder.

Orden de prioridad:

Puntuación actual del juego:
Este componente tiene el mayor peso en la función de evaluación (* 4). Esto indica que maximizar la puntuación actual del juego es la prioridad más alta para Pac-Man.

Distancia a los alimentos:
Aunque este componente tiene un factor de penalización (* -1.25), sigue siendo una prioridad significativa debido a que Pac-Man necesita recolectar alimentos para aumentar su puntuación total. Sin embargo, la penalización puede hacer que Pac-Man priorice la recolección de alimentos sobre la seguridad o la evasión de fantasmas, pero sigue siendo una prioridad alta.

Factor de tiempo asustado de los fantasmas:
Aunque tiene un peso significativo (* 3), este componente puede variar dependiendo del tiempo restante para que los fantasmas dejen de estar asustados. Es importante para evitar a los fantasmas mientras están asustados, pero puede ser menos crucial que la recolección de alimentos y la puntuación del juego.

Distancia a los fantasmas:
Este componente tiene un peso menor en comparación con los anteriores (* 1). Aunque es importante evitar a los fantasmas para evitar ser capturado, parece ser menos prioritario en comparación con la recolección de alimentos y la puntuación del juego.

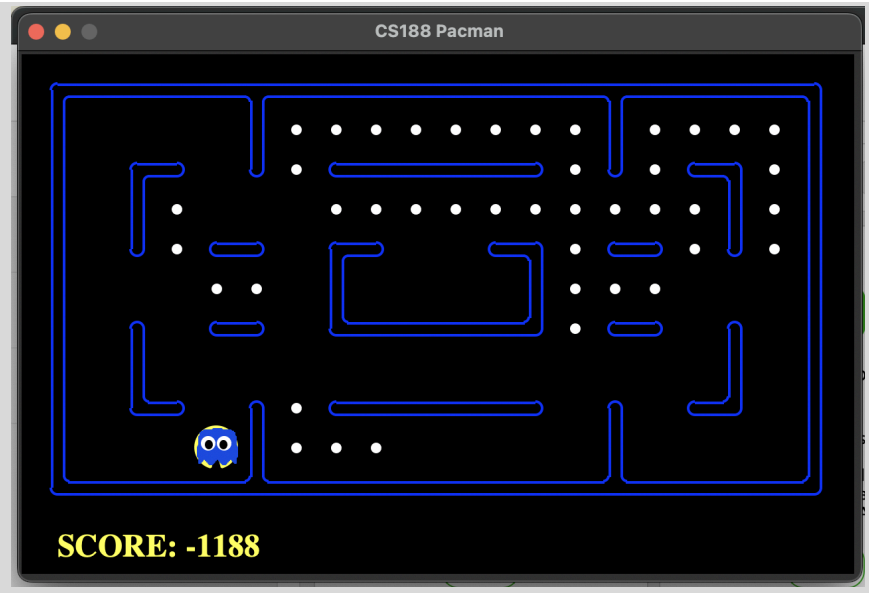
Indicar cómo se desempeña el Pac-Man con 1 fantasma:

- El comportamiento es el esperado, cuando solamente hay 1 fantasma siempre logra ganar. Agarrando toda la comida sin perder. Logra esquivarlo cuando se acerca pero casi siempre está alejado.

- Ejemplo de corrida

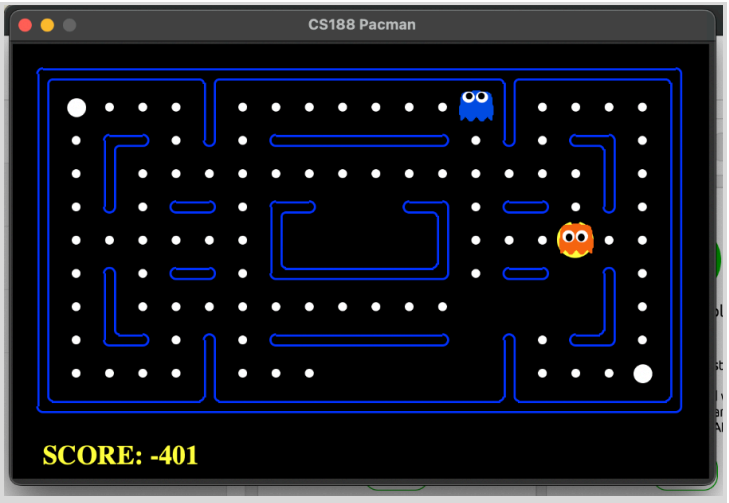
python pacman.py -p ReflexAgent -l testClassic

python pacman.py --frameTime 0 -p ReflexAgent -k 1




```
(Python38) python pacman.py --frameTime 0 -p ReflexAgent -k 1
Pacman died! Score: -1188
Average Score: -1188.0
Scores:        -1188.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

python pacman.py --frameTime 0 -p ReflexAgent -k 2




```
(Python38) python pacman.py --frameTime 0 -p ReflexAgent -k 2
Pacman died! Score: -401
Average Score: -401.0
Scores:        -401.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

python autograder.py -q q1 --no-graphics

```
(Python38) python autograder.py -q q1 --no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 5-5 at 17:37:24

Question q1
===========

Pacman emerges victorious! Score: 1182
Pacman emerges victorious! Score: 1155
Pacman emerges victorious! Score: 1172
Pacman emerges victorious! Score: 1158
Pacman emerges victorious! Score: 1246
Pacman emerges victorious! Score: 1226
Pacman emerges victorious! Score: 1219
Pacman emerges victorious! Score: 1206
Pacman emerges victorious! Score: 1218
Pacman emerges victorious! Score: 1218
Average Score: 1200.0
Scores:        1182.0, 1155.0, 1172.0, 1158.0, 1246.0, 1226.0, 1219.0, 1206.0, 1218.0, 1218.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases/q1/grade-agent.test (4 of 4 points)
***     1200.0 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (0 of 0 points)
***         Grading scheme:
***          < 10:  fail
***         >= 10:  0 points
***     10 wins (2 of 2 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  0 points
***         >= 5:  1 points
***         >= 10:  2 points

### Question q1: 4/4 ###


Finished at 17:37:25

Provisional grades
==================
Question q1: 4/4
================
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Problema #2: Optimización de decisiones

- Explicación de la implementación: Para este caso implantamos un agente Minimax para el jueg. Ahora, en lugar de evaluar directamente el estado actual como lo hace el agente de reflexión, con el agente Minimax buscamos encontrar la mejor acción considerando posibles futuros de los estados del juego. En el método getAction seleccionamos la mejor acción disponible para el Pac-Man en su estado actual, utilizando una búsqueda Minimax con una profundidad limitada definida por self.depth.

    - Con el método getAction exploramos todas las acciones legales disponibles para el Pac-Man en el estado actual.
    - Por cada acción, calculamos el valor Minimax llamando al método getValue, que simulará recursivamente las jugadas de los agentes Pac-Man y fantasmas hasta alcanzar la profundidad límite o un estado terminal.
    - En los niveles de profundidad par, el agente Minimax (Pac-Man) buscamos maximizar su valor, mientras que en los niveles impares, los agentes fantasmas buscan minimizarlo.
    - Al final de la búsqueda, el Pac-Man elige la acción que maximiza su valor esperado.
    - Con los métodos minValue y maxValue simulamos el comportamiento de los agentes fantasmas y Pac-Man, respectivamente, mientras que el método getValue coordina la alternancia entre ellos y gestiona la profundidad de la búsqueda.

- Ejemplo de corrida

python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3


```
(Python38) python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3
Pacman died! Score: -501
Average Score: -501.0
Scores:        -501.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4


```
(Python38) python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4
Pacman died! Score: -495
Average Score: -495.0
Scores:        -495.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

python autograder.py -q q2 --no-graphics

```
(Python38) python autograder.py -q q2 --no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 5-5 at 17:59:10

Question q2
==========

*** PASS: test_cases/q2/0-eval-function-lose-states-1.test
*** PASS: test_cases/q2/0-eval-function-lose-states-2.test
*** PASS: test_cases/q2/0-eval-function-win-states-1.test
*** PASS: test_cases/q2/0-eval-function-win-states-2.test
*** PASS: test_cases/q2/0-lecture-6-tree.test
*** PASS: test_cases/q2/0-small-tree.test
*** PASS: test_cases/q2/1-1-minmax.test
*** PASS: test_cases/q2/1-2-minmax.test
*** PASS: test_cases/q2/1-3-minmax.test
*** PASS: test_cases/q2/1-4-minmax.test
*** PASS: test_cases/q2/1-5-minmax.test
*** PASS: test_cases/q2/1-6-minmax.test
*** PASS: test_cases/q2/1-7-minmax.test
*** PASS: test_cases/q2/1-8-minmax.test
*** PASS: test_cases/q2/2-1a-vary-depth.test
*** PASS: test_cases/q2/2-1b-vary-depth.test
*** PASS: test_cases/q2/2-2a-vary-depth.test
*** PASS: test_cases/q2/2-2b-vary-depth.test
*** PASS: test_cases/q2/2-3a-vary-depth.test
*** PASS: test_cases/q2/2-3b-vary-depth.test
*** PASS: test_cases/q2/2-4a-vary-depth.test
*** PASS: test_cases/q2/2-4b-vary-depth.test
*** PASS: test_cases/q2/2-one-ghost-3level.test
*** PASS: test_cases/q2/3-one-ghost-4level.test
*** PASS: test_cases/q2/4-two-ghosts-3level.test
*** PASS: test_cases/q2/5-two-ghosts-4level.test
*** PASS: test_cases/q2/6-tied-root.test
*** PASS: test_cases/q2/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q2/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q2/7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q2/8-pacman-game.test

### Question q2: 5/5 ###
```

```
Finished at 17:59:10

Provisional grades
==================
Question q2: 5/5
----------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Problema #3: Optimización de busqueda

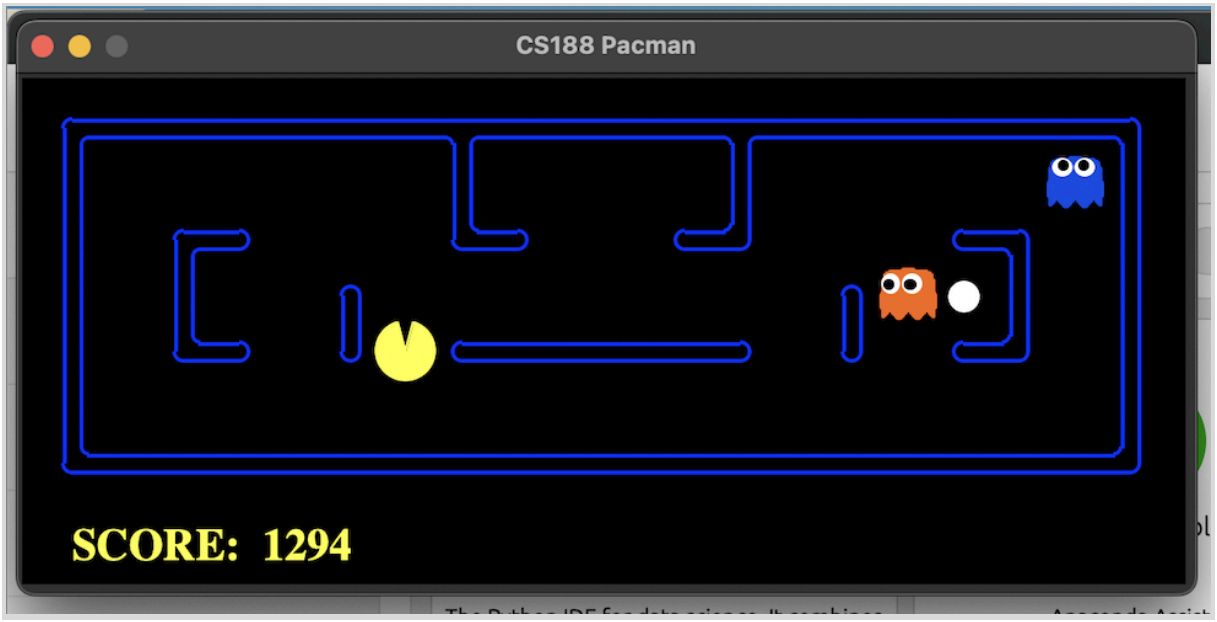
- Explicación de la implementación:

Para implementar este algoritmo hicimos lo siguiente: Este caso es muy parecido al anterior, implementamos de igual forma un agente Minimax con poda alfa-beta para el juego. Al igual que el agente Minimax sin poda alfa-beta, con este agente buscamos encontrar la mejor acción considerando posibles futuros estados del juego, pero con una eficiencia mejorada gracias a la implementación de poda alfa-beta, que elimina ramas del árbol de búsqueda que no afectan la decisión final.

Con el método getAction que funciona de manera similar al del agente Minimax, ahora con el agregado de los parámetros de alfa y beta que se utilizan en la poda alfa-beta. Este método buscamos realizar una búsqueda recursiva llamando al método getValue, que simula las jugadas de los agentes Pac-Man y fantasmas, y poda las ramas del árbol de búsqueda que no son prometedoras.

- Ejemplo de corrida

python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic




```
(Python38) python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic
Pacman emerges victorious! Score: 1294
Average Score: 1294.0
Scores:        1294.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python autograder.py -q q3 --no-graphics

```
(Python38) python autograder.py -q q3 --no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 5-5 at 19:46:09

Question q3
==========

*** PASS: test_cases/q3/0-eval-function-lose-states-1.test
*** PASS: test_cases/q3/0-eval-function-lose-states-2.test
*** PASS: test_cases/q3/0-eval-function-win-states-1.test
*** PASS: test_cases/q3/0-eval-function-win-states-2.test
*** PASS: test_cases/q3/0-lecture-6-tree.test
*** PASS: test_cases/q3/0-small-tree.test
*** PASS: test_cases/q3/1-1-minmax.test
*** PASS: test_cases/q3/1-2-minmax.test
*** PASS: test_cases/q3/1-3-minmax.test
*** PASS: test_cases/q3/1-4-minmax.test
*** PASS: test_cases/q3/1-5-minmax.test
*** PASS: test_cases/q3/1-6-minmax.test
*** PASS: test_cases/q3/1-7-minmax.test
*** PASS: test_cases/q3/1-8-minmax.test
*** PASS: test_cases/q3/2-1a-vary-depth.test
*** PASS: test_cases/q3/2-1b-vary-depth.test
*** PASS: test_cases/q3/2-2a-vary-depth.test
*** PASS: test_cases/q3/2-2b-vary-depth.test
*** PASS: test_cases/q3/2-3a-vary-depth.test
*** PASS: test_cases/q3/2-3b-vary-depth.test
*** PASS: test_cases/q3/2-4a-vary-depth.test
*** PASS: test_cases/q3/2-4b-vary-depth.test
*** PASS: test_cases/q3/2-one-ghost-3level.test
*** PASS: test_cases/q3/3-one-ghost-4level.test
*** PASS: test_cases/q3/4-two-ghosts-3level.test
*** PASS: test_cases/q3/5-two-ghosts-4level.test
*** PASS: test_cases/q3/6-tied-root.test
*** PASS: test_cases/q3/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q3/8-pacman-game.test

### Question q3: 5/5 ###
```

```
Finished at 19:46:09

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Problema #4: Implementación de Expectimax en Pac-Man

- Explicación de la implementación: Para el caso del agente Expectimax no consideramos a los agentes oponentes que son los fantasmas que juegan de manera óptima en cada paso. Si no que, modelamos a los fantasmas como jugadores que eligen uniformemente al azar entre sus movimientos legales.

  El método getAction selecciona la mejor acción disponible para el Pac-Man en el estado actual, utilizando una búsqueda Expectimax con una profundidad limitada definida por self.depth. En lugar de maximizar o minimizar el valor como en Minimax, el agente Expectimax calcula un valor esperado para cada acción, considerando todas las posibles respuestas de los fantasmas.

- Ejemplo de corrida

python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3


```
(Python38) python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3
Pacman died! Score: -498
Average Score: -498.0
Scores:        -498.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

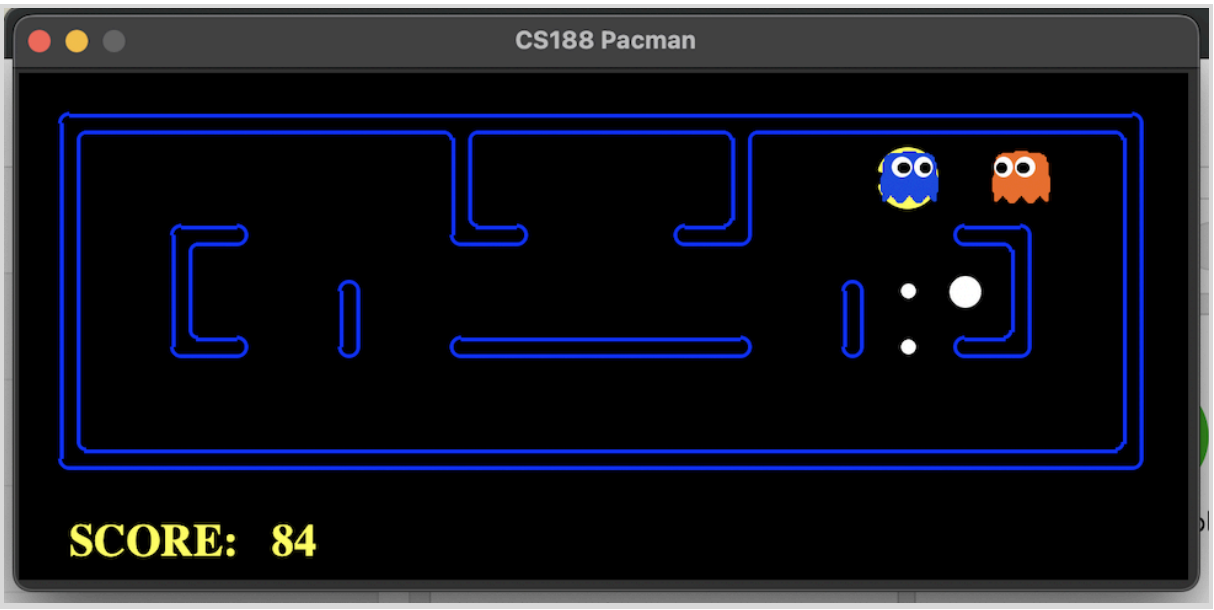
python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10

```
(Python38) python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
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
```

python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10

```
(Python38) python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Average Score: -88.4
Scores:        -502.0, 532.0, -502.0, -502.0, -502.0, 532.0, -502.0, 532.0, 532.0, -502.0
Win Rate:      4/10 (0.40)
Record:        Loss, Win, Loss, Loss, Loss, Win, Loss, Win, Win, Loss
```

python autograder.py -q q4



```
(Python38) python autograder.py -q q4
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 5-5 at 19:49:41

Question q4
==========

*** PASS: test_cases/q4/0-eval-function-lose-states-1.test
*** PASS: test_cases/q4/0-eval-function-lose-states-2.test
*** PASS: test_cases/q4/0-eval-function-win-states-1.test
*** PASS: test_cases/q4/0-eval-function-win-states-2.test
*** PASS: test_cases/q4/0-expectimax1.test
*** PASS: test_cases/q4/1-expectimax2.test
*** PASS: test_cases/q4/2-one-ghost-3level.test
*** PASS: test_cases/q4/3-one-ghost-4level.test
*** PASS: test_cases/q4/4-two-ghosts-3level.test
*** PASS: test_cases/q4/5-two-ghosts-4level.test
*** PASS: test_cases/q4/6-1a-check-depth-one-ghost.test
*** PASS: test_cases/q4/6-1b-check-depth-one-ghost.test
*** PASS: test_cases/q4/6-1c-check-depth-one-ghost.test
*** PASS: test_cases/q4/6-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q4/6-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q4/6-2c-check-depth-two-ghosts.test
*** Running ExpectimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running ExpectimaxAgent on smallClassic after 11 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q4/7-pacman-game.test

### Question q4: 5/5 ###


Finished at 19:49:52

Provisional grades
==================
Question q4: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```