Nghia Lam

1001699317

Homework 7

Task 1

value_iteration.py 'environment2.txt' -0.04 1 20

```
utilities:
 0.812  0.868  0.918  1.000
 0.762  0.000  0.660 -1.000
 0.705  0.655  0.611  0.388

policy:
     >      >      >      o
     ^      X      ^      o
     ^      <      <      <
```

value_iteration.py 'environment2.txt' -0.04 0.9 20

```
utilities:
 0.509  0.650  0.795  1.000
 0.399  0.000  0.486 -1.000
 0.296  0.254  0.345  0.130

policy:
     >      >      >      o
     ^      X      ^      o
     ^      >      ^      <
```

Task 2

In chess, a win, a loss, or a tie would be the final results. We can choose to set win at 1, lose at -1, and draw at 0. We can put the Discount factor very near to 1 since we want short-term incentives to be just as important as long-term ones. You could wind up losing the entire game because the reason is reliant on the position's immediate reward. Take a piece, for instance, when a checkmate is reached in five or six moves. Since the game of chess is built on long-term advantage, we would also like non-terminal rewards to be very low. Thus, it would be very nearly 0.

Task 3.a)

For position 2,2 based on the image, I would compute it based on the Bellman equation below

Ex: For the 1 iteration

$U(s)=R(s)+\gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$

1. This is the utility value of each direction currently: (If the direction is a wall then the utility value is based on the utility of the position)

    uVal = 1

    dVal = -1

    lVal = 0

    rVal = 0

2. Next calculate each directions value based on the probability of .8 and .1 for each direction.

    states["upState"] = ntr + gamma * (.8 * uVal + .1*lVal + .1*rVal)

    states["downState"] = ntr + gamma * (.8 * dVal + .1*lVal + .1*rVal)

    states["rightState"] = ntr + gamma * (.8 * rVal + .1*uVal + .1*dVal)

    states["leftState"] = ntr + gamma * (.8 * lVal + .1*uVal + .1*dVal)

3. Once each state is calculated get the max value and that would determine your new utility value in your map

    So, the max utility value in this case for the second iteration would be .68 for going UP.

Task 3.b)

For up to be the suboptimal direction the

(r + gamma of UP) need to be lower than (r + gamma of DOWN) or any other direction

In this case the UP will never be the suboptimal path as .8* 1 = .8 and .8 * -1 = -.8, thus no matter what the non terminal state is it will never get a value that is greater that UP in this situation for 2,2. .8 > -.8. In addition of left and right it is the same situation as it will have -1 affecting it.