Group Project

- Experiment 0: Successfully implemented value iteration algorithm on the world given on the midterm.
- Changed the format a bit, but our implementation produces the correct results as found on the midterm.

WE
NEED
GROCERIES

The following MDP world consists of 5 states and 3 actions:

Midterm World

(1, 1) Actions: down, right	(1, 2) Action: Exit = 1
(2, 1) Actions: down, right	(2, 2) Action: Exit = -10
(3, 1) Action: Exit = 10	

When taking action down, it is successful with probability 0.75, otherwise you go right.

When taking action right, it is successful with probability 0.75, otherwise you go down.

When taking action Exit, it is successful with probability 1.0.

The only reward is when taking action Exit, and there is no discounting.

Calculate the value of states using Value Iteration algorithm for time step K=0, 1, 2, 3

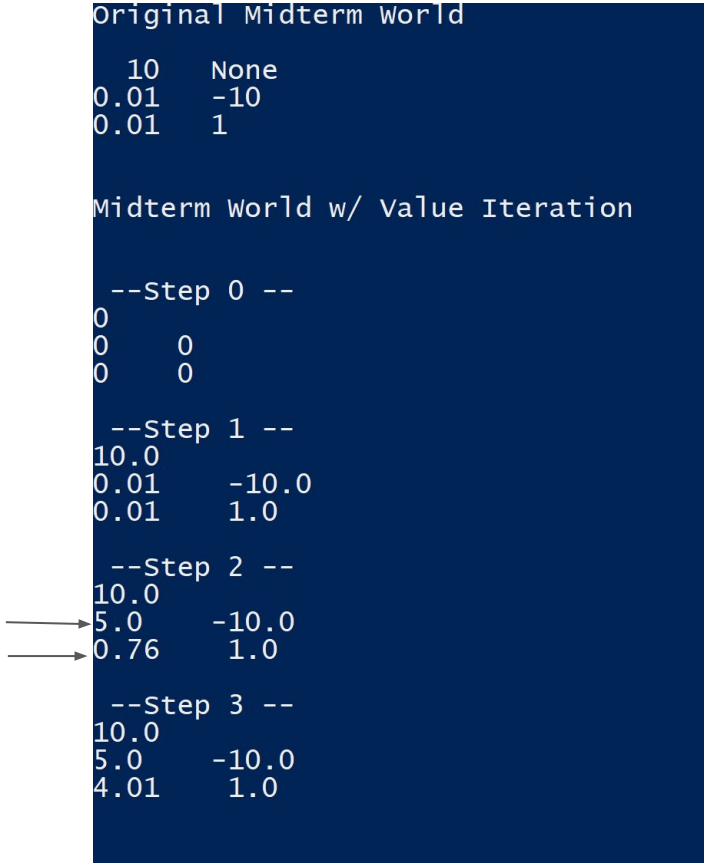
EXPERIMENT 0: Results

Calculate the value of states using Value Iteration algorithm for time step K=0, 1, 2, 3

We start with the given values.

At time step 0 all values begin with 0.

Now at step one the values are initialized.
At step 2 value iteration finds correct values for states (0,0) & (1,0).



RESULTS EXPERIMENT 0

In step 4 the delta value becomes smaller than the gamma value and the algorithm converges and is done.

```
Midterm World w/ Value Iteration


 --Step 0 --
0
0    0
0    0
delta:  10.0   gamma: 0.999

 --Step 1 --
10.0
0.01    -10.0
0.01    1.0
delta:  4.995   gamma: 0.999

 --Step 2 --
10.0
5.0    -10.0
0.76    1.0
delta:  3.2479987500000003   gamma: 0.999

 --Step 3 --
10.0
5.0    -10.0
4.01    1.0
delta:  0   gamma: 0.999
```

```python
def value_iteration2(mdp, epsilon=0.0001):

    U1 = {s: 0 for s in mdp.states}
    R, T, gamma = mdp.R, mdp.T, mdp.gamma
    i = 0
    print("Midterm World w/ Value Iteration", '\n')
    while True:
        U = U1.copy()
        delta = 0
        # add by Group START
        print('\n',"--Step", i, "--")
        print(rounder(U1[(0, 2)], 2))
        print(rounder(U1[(0, 1)], 2), "  ", rounder(U1[(1, 1)], 2))
        print(rounder(U1[(0, 0)], 2), "  ", rounder(U1[(1, 0)], 2))
        i+=1
        for s in mdp.states:
            U1[s] = R(s) + gamma * max(sum(p * U[s1] for (p, s1) in T(s, a))
                                       for a in mdp.actions(s))
            delta = max(delta, abs(U1[s] - U[s]))

        # print convergence check
        print("delta: ", delta, "  gamma:", gamma)
        if delta <= epsilon * (1 - gamma) / gamma:
            return U
```

EXPERIMENT 1

- We changed our original mdp slightly and produced a world to represent the original idea.
- .75 probability of going up or .25 we go right.

Problem:
How to save the most money.
How many stores should we go to.
Should we just go to COSTCO?

Let's find the best policy.

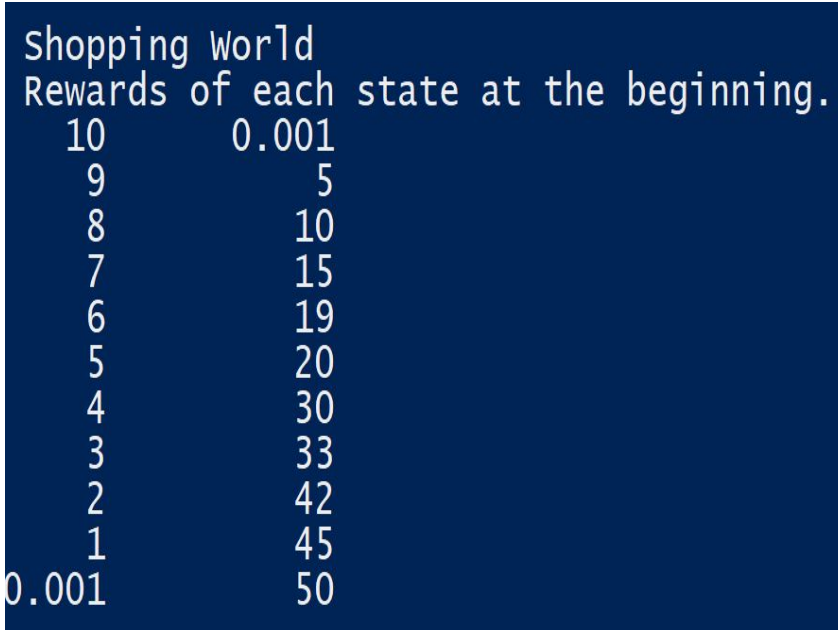
Original Idea

(1,1) AT HOME WITH NO GROCERIES (START) Actions: Costco(R) / Store1(D) Reward: 0	(1,2) AT COSTCO BUYING 9 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(2,1) AT STORE1 BUYING ITEM Actions: Costco(R) / Store2(D) Reward: Cost + Variables = ?	(2,2) AT COSTCO BUYING 8 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(3,1) AT STORE2 BUYING ITEM Actions: Costco(R) / Store3(D) Reward: Cost + Variables = ?	(3,2) AT COSTCO BUYING 7 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(4,1) AT STORE3 BUYING ITEM Actions: Costco(R) / Store4(D) Reward: Cost + Variables = ?	(4,2) AT COSTCO BUYING 6 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(5,1) AT STORE4 BUYING ITEM Actions: Costco(R) / Store5(D) Reward: Cost + Variables = ?	(5,2) AT COSTCO BUYING 5 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(6,1) AT STORE5 BUYING ITEM Actions: Costco(R) / Store6(D) Reward: Cost + Variables = ?	(6,2) AT COSTCO BUYING 4 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(7,1) AT STORE6 BUYING ITEM Actions: Costco(R) / Store7(D) Reward: Cost + Variables = ?	(7,2) AT COSTCO BUYING 3 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(8,1) AT STORE7 BUYING ITEM Actions: Costco(R) / Store8(D) Reward: Cost + Variables = ?	(8,2) AT COSTCO BUYING 2 ITEMS Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(9,1) AT STORE8 BUYING ITEM Actions: Costco(R) / Store9(D) Reward: Cost + Variables = ?	(9,2) AT COSTCO BUYING 1 ITEM Actions: Go Home / Get Gas Reward: Cost + Variables = ?
(10,1) AT STORE9 BUYING LAST ITEM Actions: Costco(R) / Go Home / Get Gas Reward: Cost + Variables = ?	(10,2) STOP FOR GAS Actions: Go Home Reward: Cost + Variables = ?
(11,1) MADE IT HOME WITH GROCERIES (END) Actions: None Reward: Stress minimized = BONUS REWARD!	NOTE: Each transition includes a living reward! If an item is sold out at StoreX we are forced to go to Costco.

EXPERIMENT 1: The World

We began by producing a representation of our shopping world.
We knew that we were successful with implementing the value iteration algorithm on the midterm problem.

Now we can try to use it on the shopping world...



EXPERIMENT 1: Results

```
 Shopping World
 Rewards of each state at the beginning.
   10       0.001
    9           5
    8          10
    7          15
    6          19
    5          20
    4          30
    3          33
    2          42
    1          45
0.001          50
```

```
 State   Money Saved
 (0,10)   39.88   Hit all 10 stores, NO COSTCO

 (0,9)    40.13   Went to 9 stores then COSTCO

 (0,8)    40.56   Went to 8 stores then COSTCO

 (0,7)    41.14   Went to 7 stores then COSTCO

 (0,6)    41.57   Went to 6 stores then COSTCO

 (0,5)    41.14   Went to 5 stores then COSTCO

 (0,4)    42.32   Went to 4 stores then COSTCO

 (0,3)    42.95   Went to 3 stores then COSTCO

 (0,2)    45.25   Went to 2 stores then COSTCO

 (0,1)    47.19   Went to 1 store then COSTCO

 (0,0)    49.93   Went to directly to COSTCO

Best Policy Found
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
>  to COSTCO.
>  to COSTCO.
>  to COSTCO.
```

Here you can see that value iteration finds that we save the most money by going directly to Costco in this world. I had the world set to where if you go to all stores, you spend about 50 and if you go to COSTCO in a state you save. Sooner you go to COSTCO the better.

EXPERIMENT 2: Results

In a different shopping experience, it might be best to visit 5 stores for an item than get the other 5 at COSTCO. Here is what that would look like. If we make the original reward for going to COSTCO greater, we find a **different policy**.

```
Shopping World
Rewards of each state at the beginning.
  10      0.001
   9          5
   8         10
   7         15
   6         19
   5         50
   4         30
   3         33
   2         42
   1         45
0.001        40
```

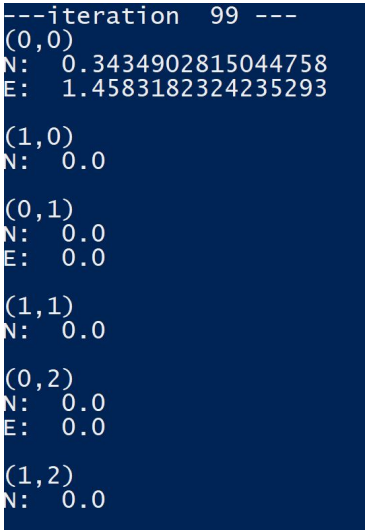
Make state (1,5) greater at start

```
State   Money Saved
(0,10)    39.88    Hit all 10 stores, NO COSTCO
(0,9)     40.13    Went to 9 stores then COSTCO
(0,8)     40.56    Went to 8 stores then COSTCO
(0,7)     41.14    Went to 7 stores then COSTCO
(0,6)     41.57    Went to 6 stores then COSTCO
(0,5)     55.77    Went to 5 stores then COSTCO
(0,4)     53.28    Went to 4 stores then COSTCO
(0,3)     51.16    Went to 3 stores then COSTCO
(0,2)     50.82    Went to 2 stores then COSTCO
(0,1)     50.32    Went to 1 store then COSTCO
(0,0)     47.69    Went to directly to COSTCO

Best Policy Found
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
>  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
^  to COSTCO.
```

Project Progress

With experiments 0-2 we found that we can use value iteration with our idea if we know the value of the states. But I don't think that it would work too well with our original idea. We would basically need to know the answer to the problem to provide good input rewards. The next experiments we were going to try is utilizing a Q-Learning agent, but we haven't gotten that far yet. Hopefully we will be able to produce some interesting results!



WE STILL NEED GROCERIES