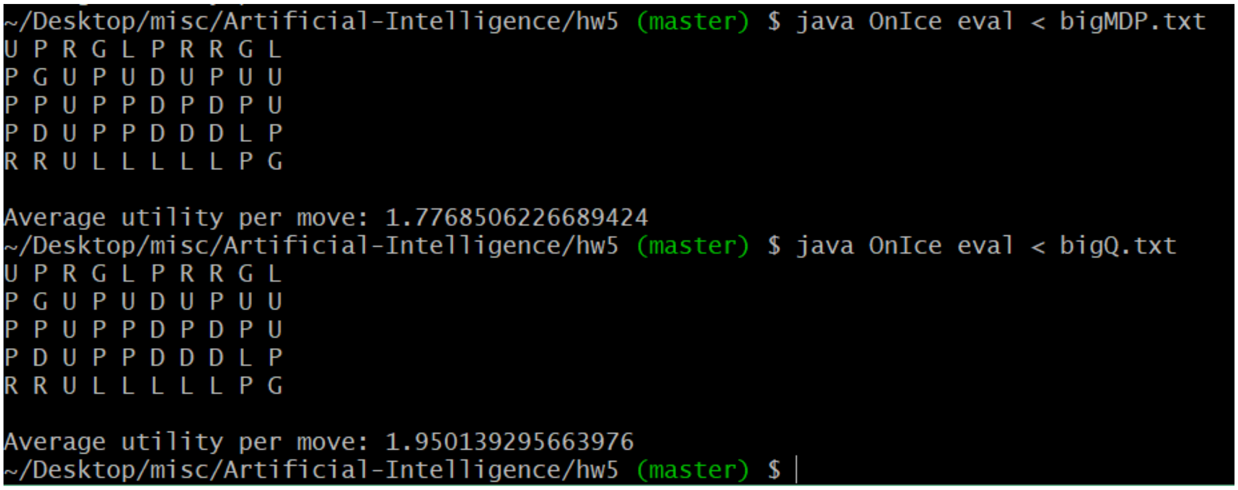

```
~/Desktop/misc/Artificial-Intelligence/hw5 (master) $ java OnIce eval < bigMDP.txt
U P R G L P R R G L
P G U P U D U P U U
P P U P P D P D P U
P D U P P D D D L P
R R U L L L L L P G

Average utility per move: 1.7768506226689424
~/Desktop/misc/Artificial-Intelligence/hw5 (master) $ java OnIce eval < bigQ.txt
U P R G L P R R G L
P G U P U D U P U U
P P U P P D P D P U
P D U P P D D D L P
R R U L L L L L P G

Average utility per move: 1.950139295663976
~/Desktop/misc/Artificial-Intelligence/hw5 (master) $ |
```

Using the seed provided (5100) for random, I got 1.78 for the average utility per move for the policy created by MDP and 1.95 for the average utility per move for the policy created by Q-Learning. This result was unexpected especially because **after 10,000 iterations, the policy produced by MDP and Q-Learning were identical**. This difference in utility was probably just a result of the random starting points; with enough iterations, these two average utility per move should converge because they have the exact same policy. In general, the average utility per move should be higher for the MDP than the Q-Learning because MDP uses value iteration and has access to the actual world and rewards. Q-Learning uses less information (does not know the world) but is more efficient than MDP.