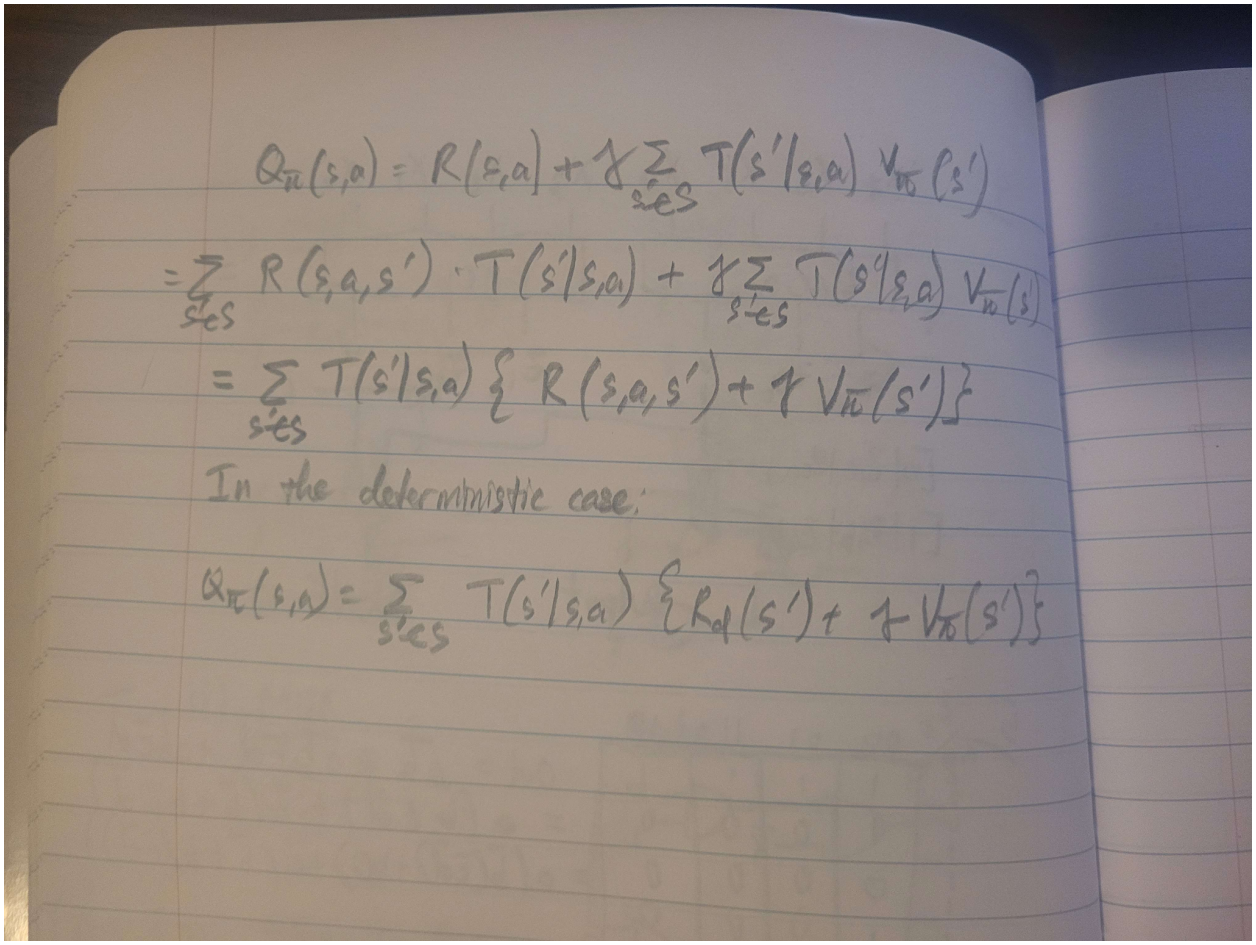
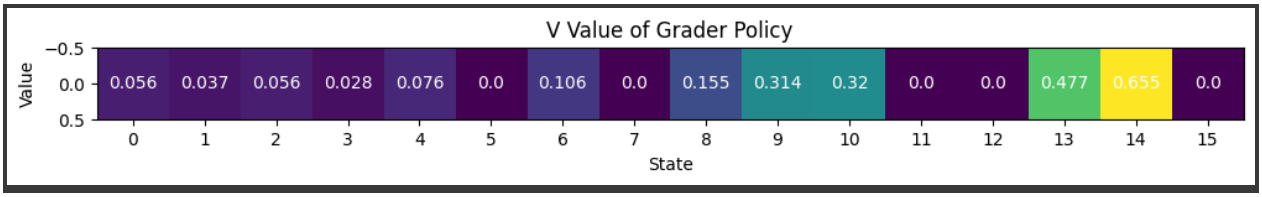
1.1



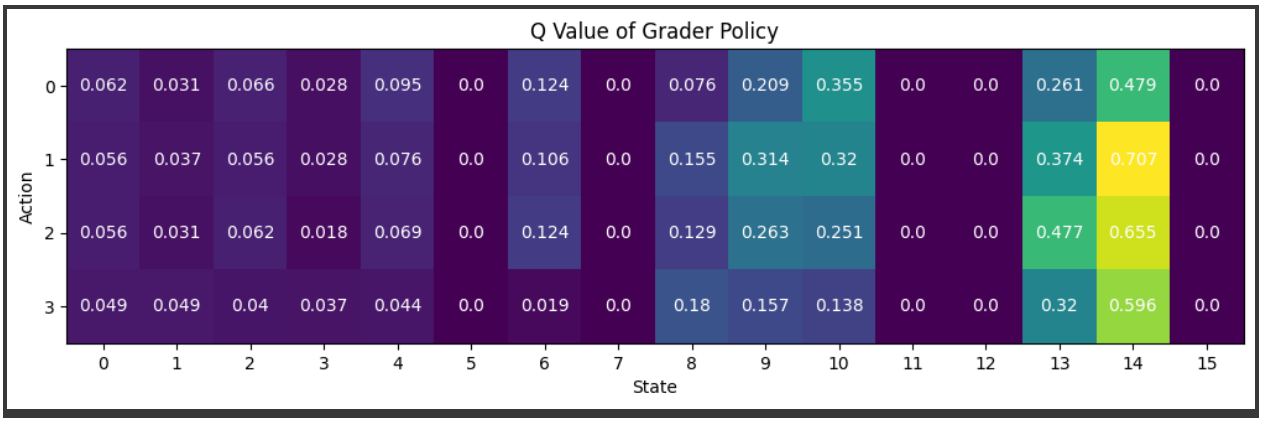
1.2

The V value of policy A is correct to at least 2 decimal values. The V value of policy B is correct to at least 2 decimal values. The V value of policy C is correct to at least 2 decimal values.



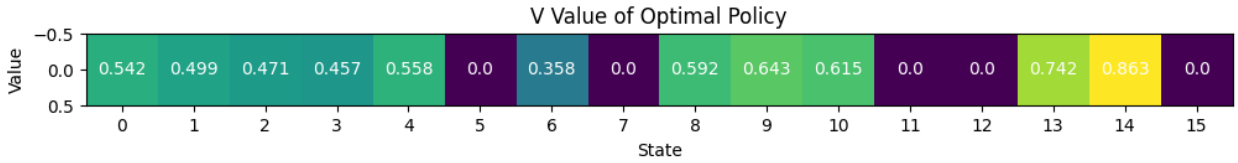
1.3

The Q value of policy A is correct to at least 2 decimal values. The Q value of policy B is correct to at least 2 decimal values. The Q value of policy C is correct to at least 2 decimal values.

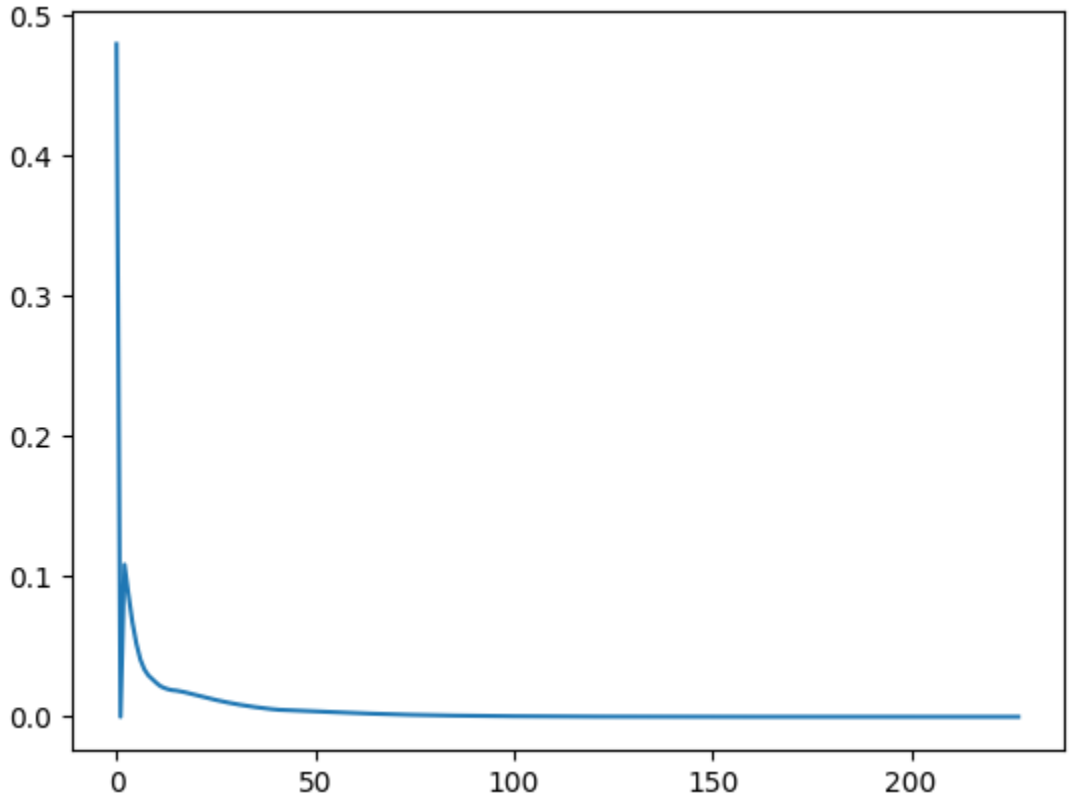


2.1

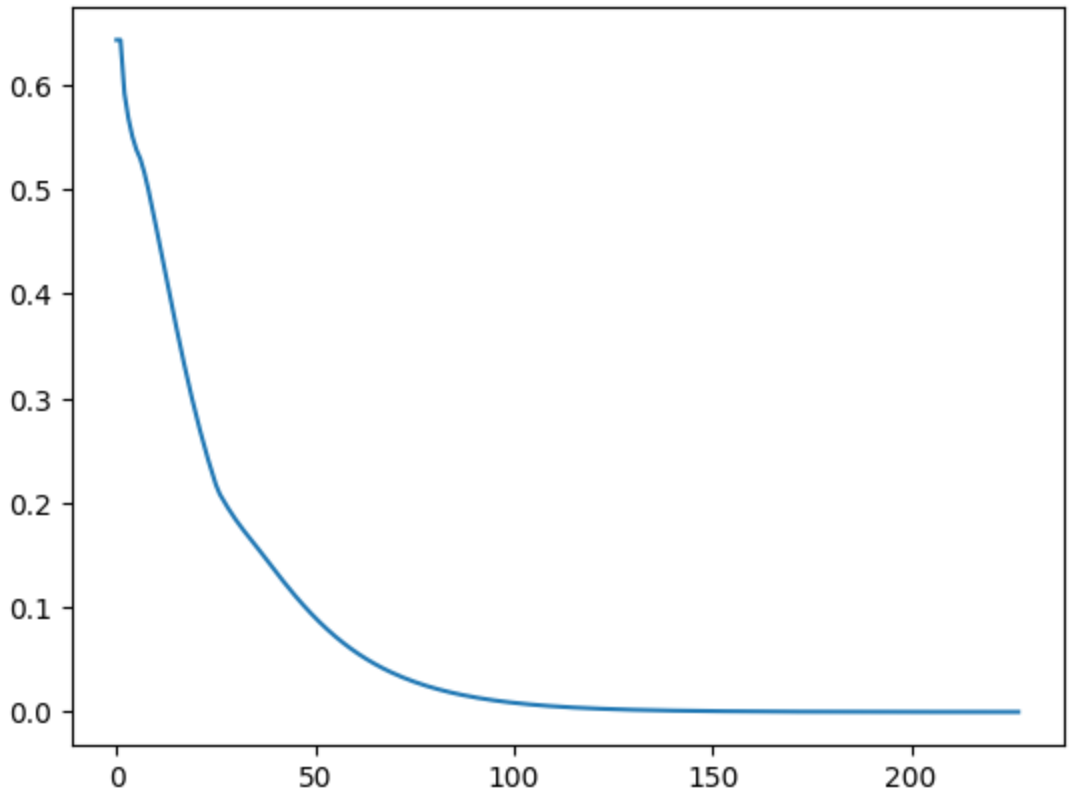
```
The optimal V value for gamma = 0.0 is correct. The optimal V value for
gamma = 0.9 is correct.
```



maxs |V (s) − Vprevious_iteration(s)| with respect to iteration number:



maxs |V (s) − Vfinal_iteration(s)| with respect to iteration number

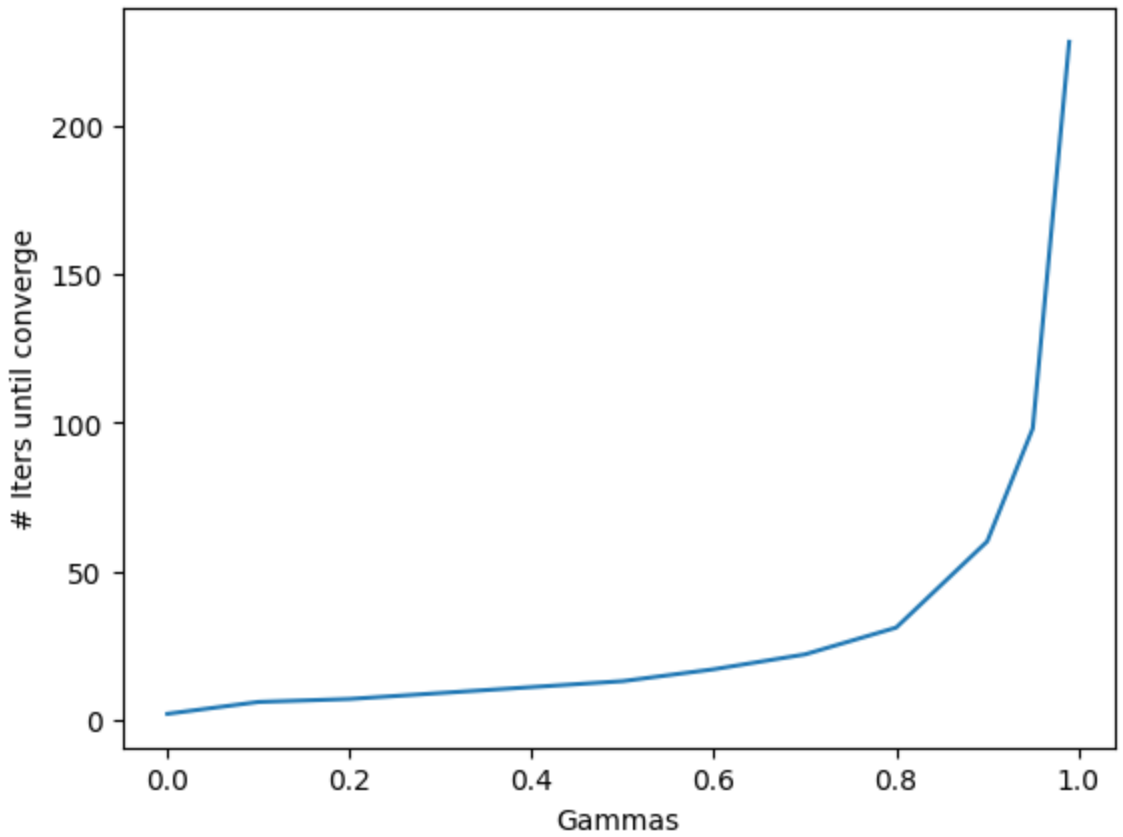


2.3

```
228 iterations
```

We can see that in the 2nd plot for 2.2, that the V value has converged sufficiently to a) satisfy the termination condition and b) create a “tail” at the right end (> 100 iters) of the chart to show that the V value does not change too much at later iterations. Since V is the expected return (what we want to approximate to have a good policy), we can say that our algorithm has converged to a policy that is based on the optimal expected returns (i.e. the optimal policy).

2.4



As gamma increases, reward values farther away from the current timestep are discounted. This means that too high a gamma would push the model to favor immediate rewards when learning a policy, reducing importance on long-term relationships between states. For many problems, this is detrimental. Also, by setting a low gamma, the model approaches a greedy bandit solution (as gamma approaches 0, we're just picking highest expected value), which makes the optimization problem much simpler, enabling the policy to converge faster.

2.5

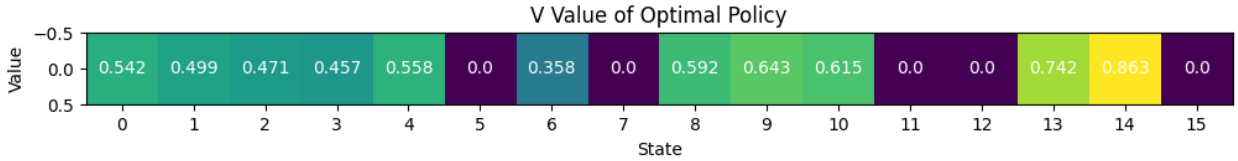
```
Without guess policy: 228 iterations
With guess policy: 155 iterations
```

3.1

|S| states, |A| possible actions in each state → |A|^|S| policies, since deterministic policies only depend on state, and have only one possibility per action.

3.2

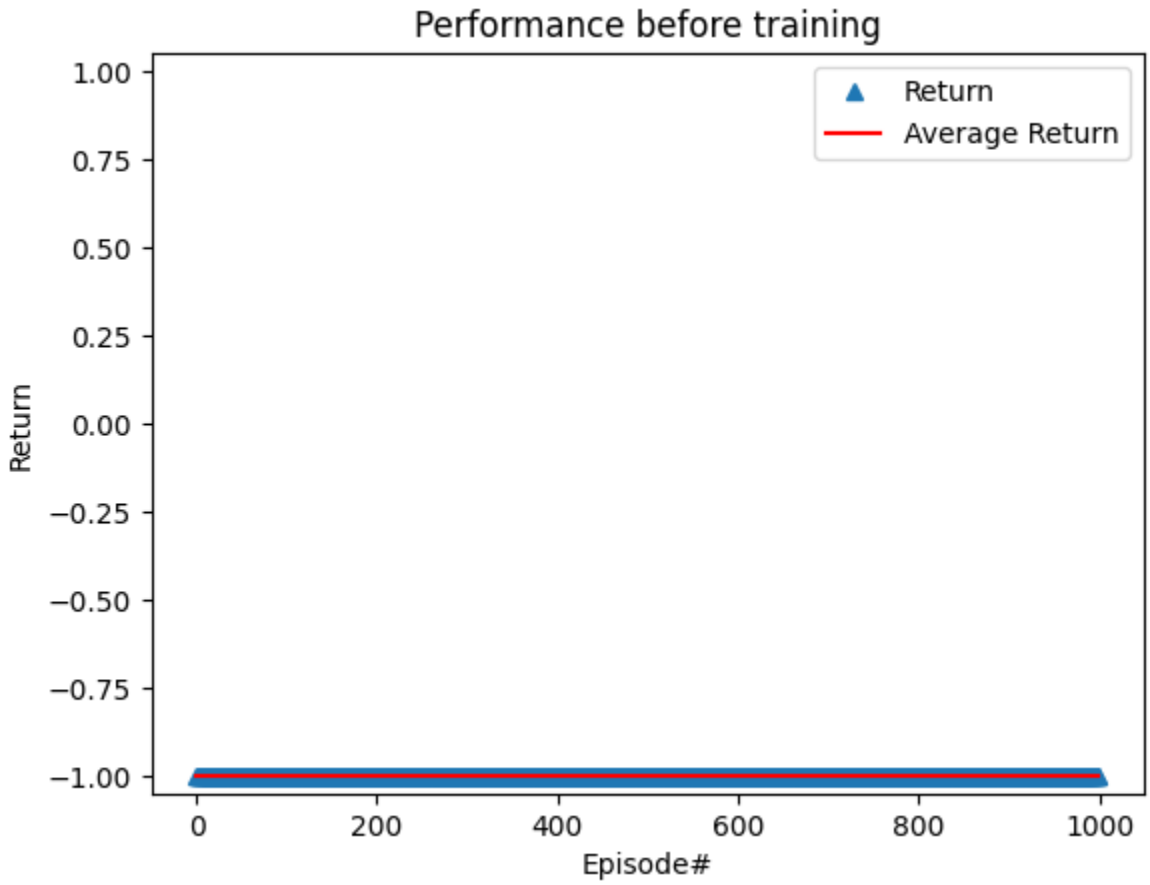
```
The optimal V value for gamma = 0.0 is correct.
The optimal V value for gamma = 0.9 is correct.
```



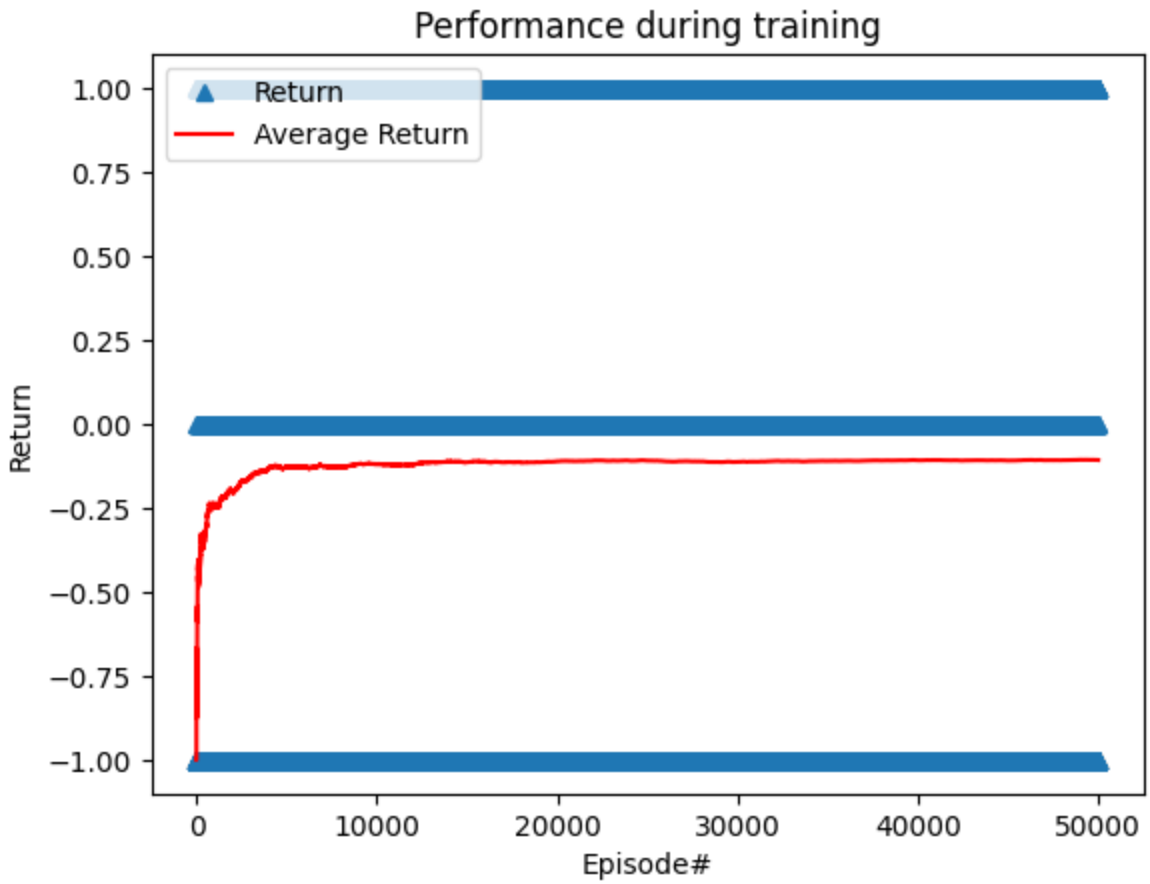
3.3

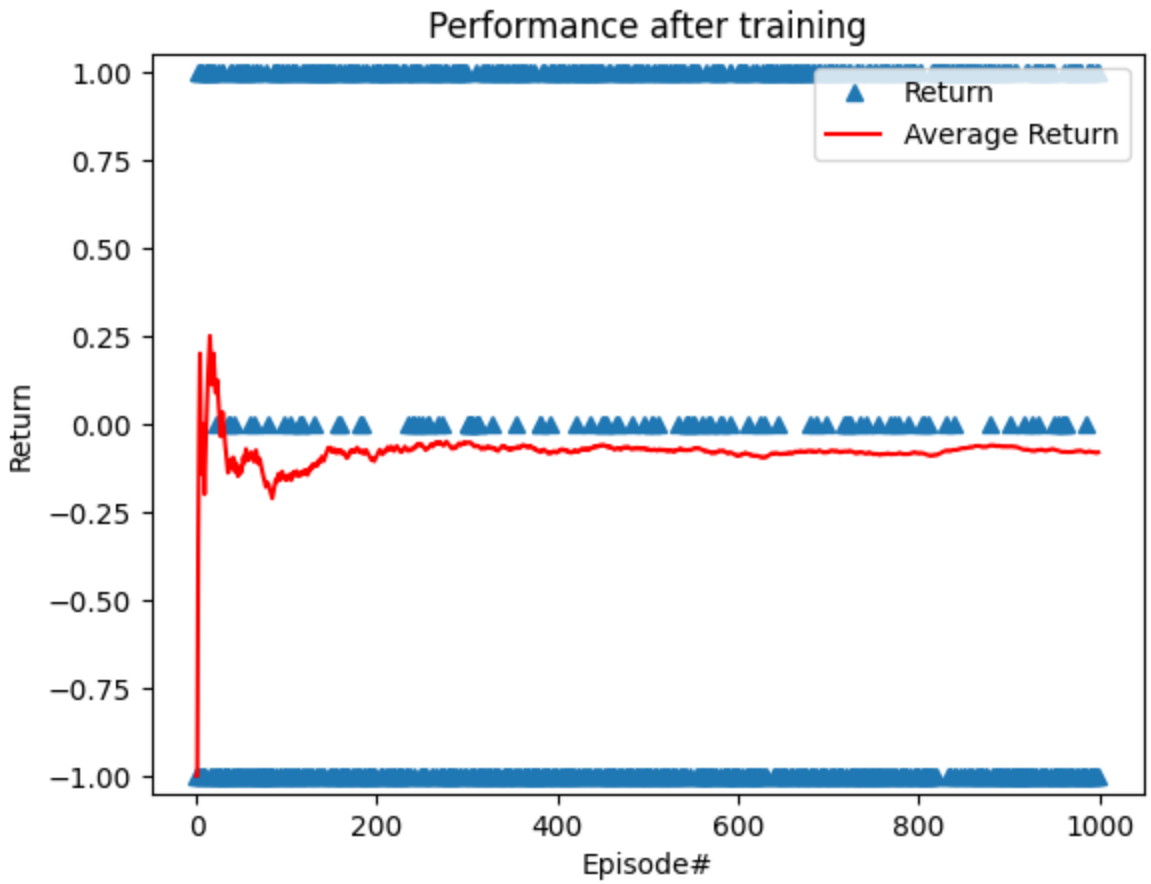
For gamma=0.9, we get 6 iterations, with one policy evaluation per iteration. This is much more efficient than searching through all the policies (4^16). This method can be more efficient because we understand that the optimal policy maximizes expected reward, and we are sampling values by taking (a majority of the time) actions that maximize expected reward, meaning that we are only searching in a subspace around an area of policies with high and increasing reward.

4.



```
Average reward before training -1.0
```

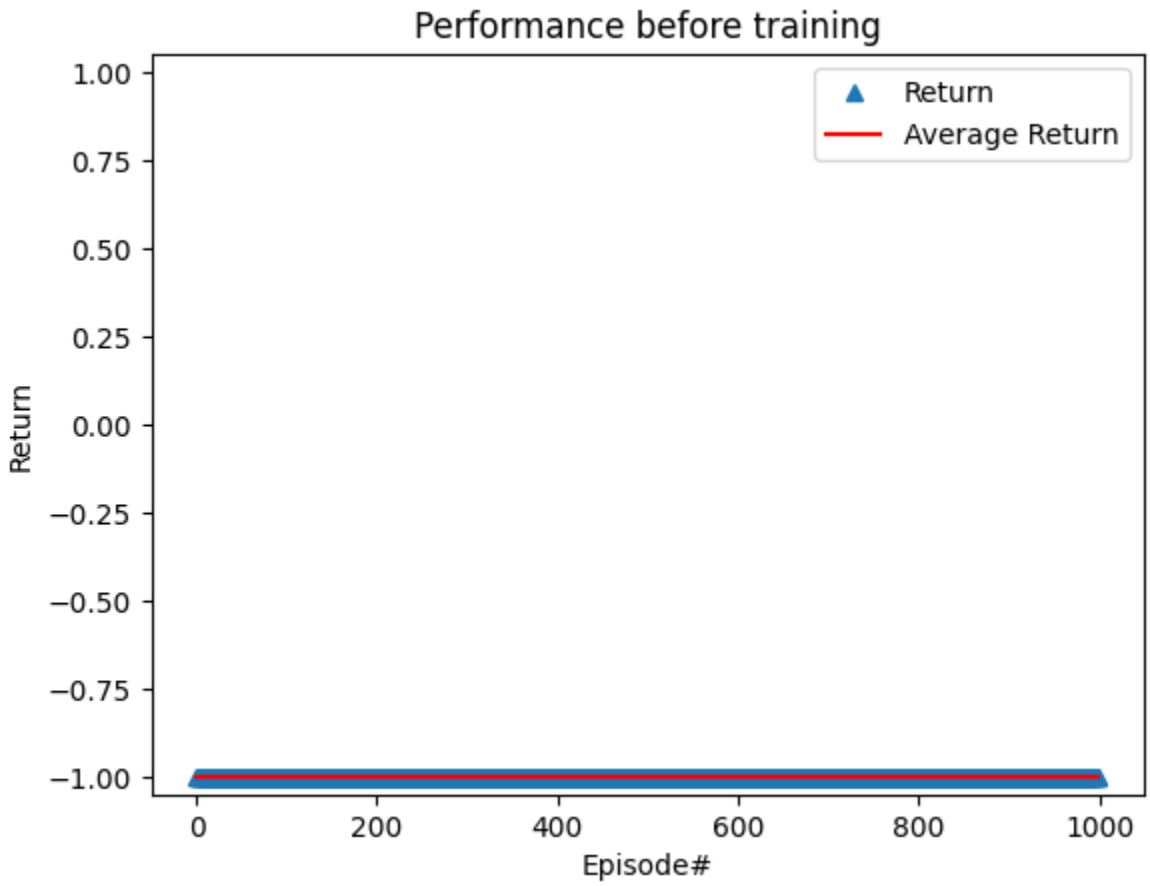

Performance during training
Return
Average Return
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Return
0
10000
20000
30000
40000
50000
Episode#

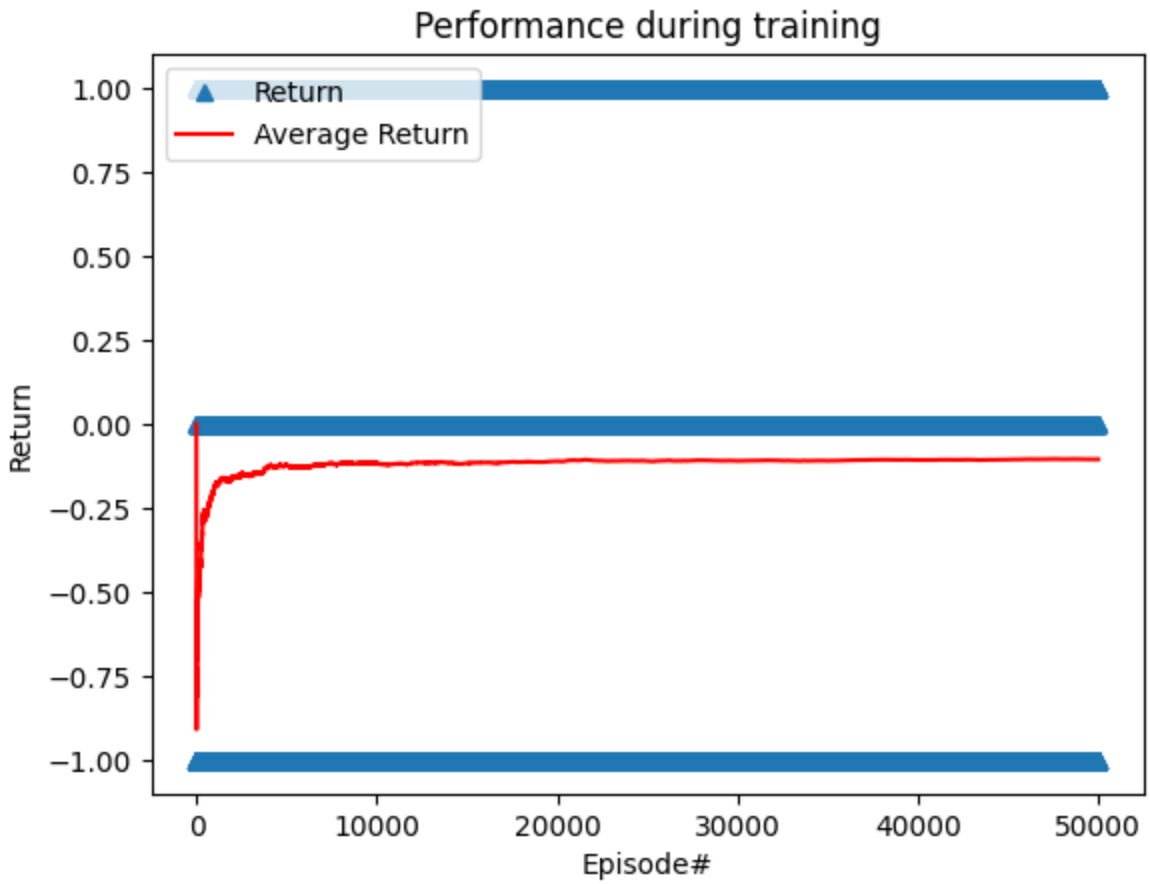


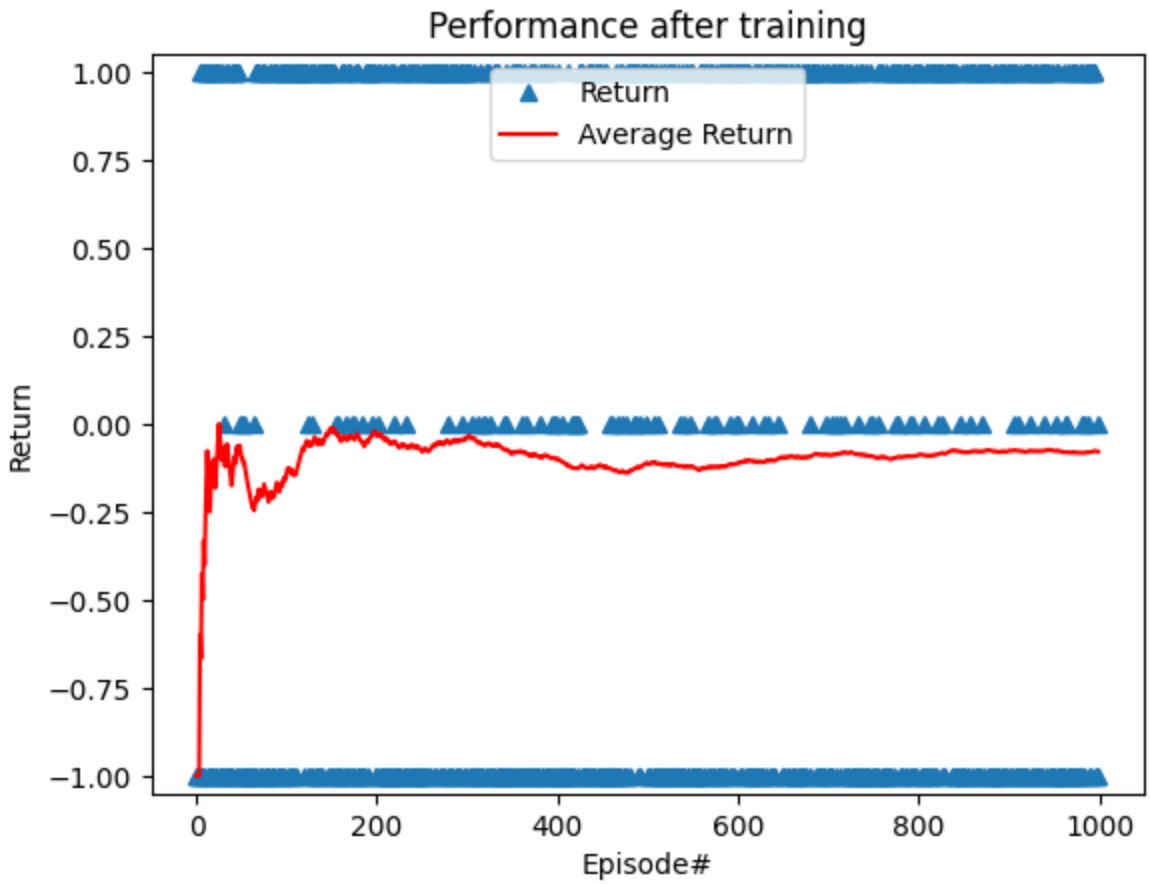

```
Average reward after training -0.081
```

5.

```
Average reward before training -1.0
```




Performance during training
Return
Average Return
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Return
0
10000
20000
30000
40000
50000
Episode#




```
Average reward after training -0.079
```