Q Learning

CHARANTEJA SAKAMURI
1530974

Introduction:

Reinforcement learning is learning about an environment by interacting with it. The environment can be static or dynamic. Each position is represented by a state. The agent moves from one state to another by taking an action. The appropriate action to be taken at a state is determined by the policy. The agent receives a reward or penalty depending on the efficiency of action. The efficiency of a state is measured in terms of Utility. The good states will have higher utilities to encourage the agent to take those paths and the bad states will have penalties to discourage them from taking the path. As the agent continues to interact more with the environment, he takes more promising paths frequently. This is a sign that the agent is learning from the environment.

The Utility of a state is calculated using the formula:

$$Q(a,s) \leftarrow (1-\alpha)*Q(a,s) + \alpha*[R(s',a,s)+ \gamma*\max_{a'}Q(a',s')]$$

where
Q is the current utility, initially zero
α is the learning rate
s is the current state being evaluated
R is the immediate reward function for reaching state s' by taking action a in state s
γ is the discount rate
$\max_{a'}Q(a',s')$ is the max utility obtainable from all valid actions after taking action a in state s

The agent always tries to move towards states with higher utility values with the assumption that it leads to the goal state. Generally, in reinforcement learning we need to build a model or learn policy functions. Q-learning is a special kind of reinforcement learning approach which is model free. We are given three policies for this experiment. All the three experiments are run using different combinations of these policies as follows:

PRANDOM: If pickup and dropoff is applicable, choose this operator; otherwise, choose an applicable operator randomly.
• **PEPLOIT**: If pickup and dropoff is applicable, choose this operator; otherwise, apply the applicable operator with the highest q-value (break ties by rolling a dice for operators with the same q-value) with probability 0.85 and choose a different applicable operator randomly with probability 0.15.

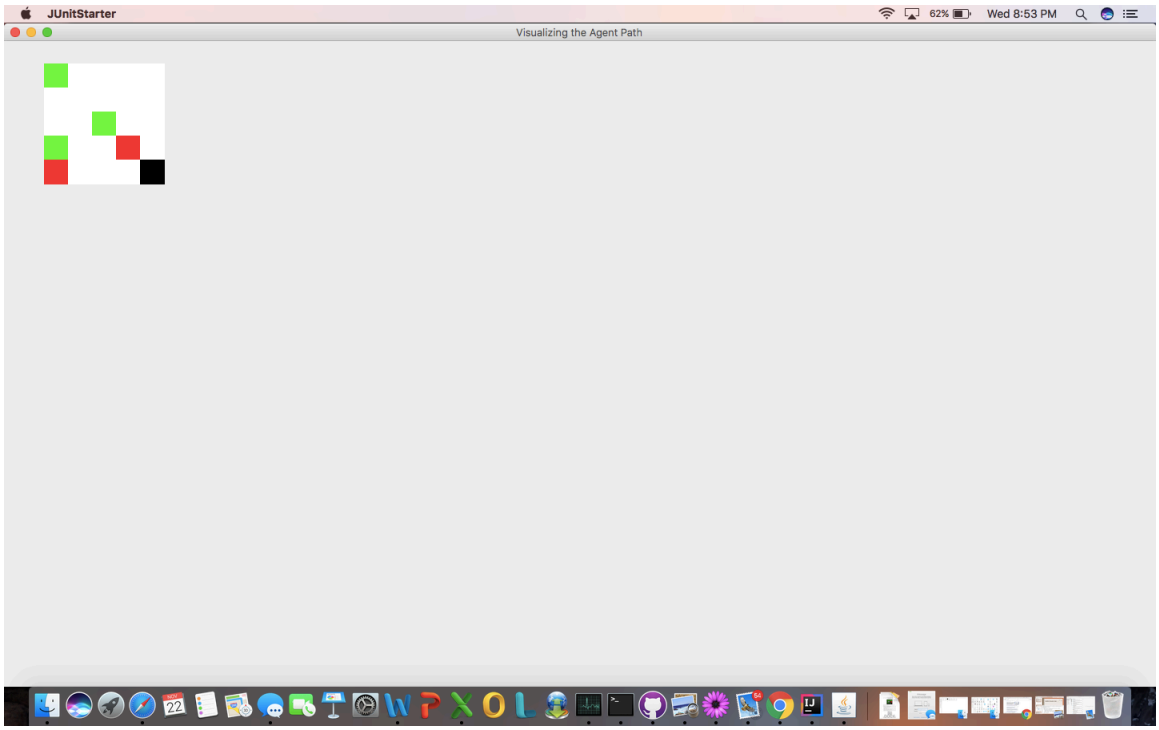
• **PGREEDY**: If pickup and dropoff is applicable, choose this operator; otherwise, apply the applicable operator with the highest q-value (break ties by rolling a dice for operators with the same q-value).

In all the three experiments, the learning rate(α) and discount rate(γ) are set to 0.3 and 0.5 respectively. All the entries of Q-Table are initialized to zeros. We have set two seeds for this project.

SEED 1 = 115
SEED 2 = 2241

EXPERIMENTAL SETUP:

Screenshot of Agent path:



Here, the pick up locations are indicated by GREEN color and dropoff locations are indicated by RED color. The agent is indicated by BLACK color if he doesn't carry a block, otherwise BLUE.

EXPERIMENT1:

In this experiment, PRANDOM is ran for the first 3000 iterations and PGREEDY for the next 3000 iterations. During the first 3000 iterations, the agent is learning the Q-values more rapidly. During the next 3000 iterations, Q-values won't change significantly.

The most important finding is that the agent reached the terminal state in 810 steps for the first time. Although the agent has some knowledge about the world, it still ended up in taking 810 steps for the second and third times. This is because the agent is taking the next move randomly and the information available in Q-Table is not at all useful.

In the later 3000 iterations, the terminal state is reached in less number of steps. The agent gets improved in each and every step, so the terminal state should be reached in less number of steps each time. But, sometimes the agent takes more steps to reach terminal. This is because of the implementation of greedy policy. In greedy policy, when two actions have the same value for a given state, we randomly pick the action. If the randomly picked action is bad, it will take more time to reach terminal state.

RESULTS:

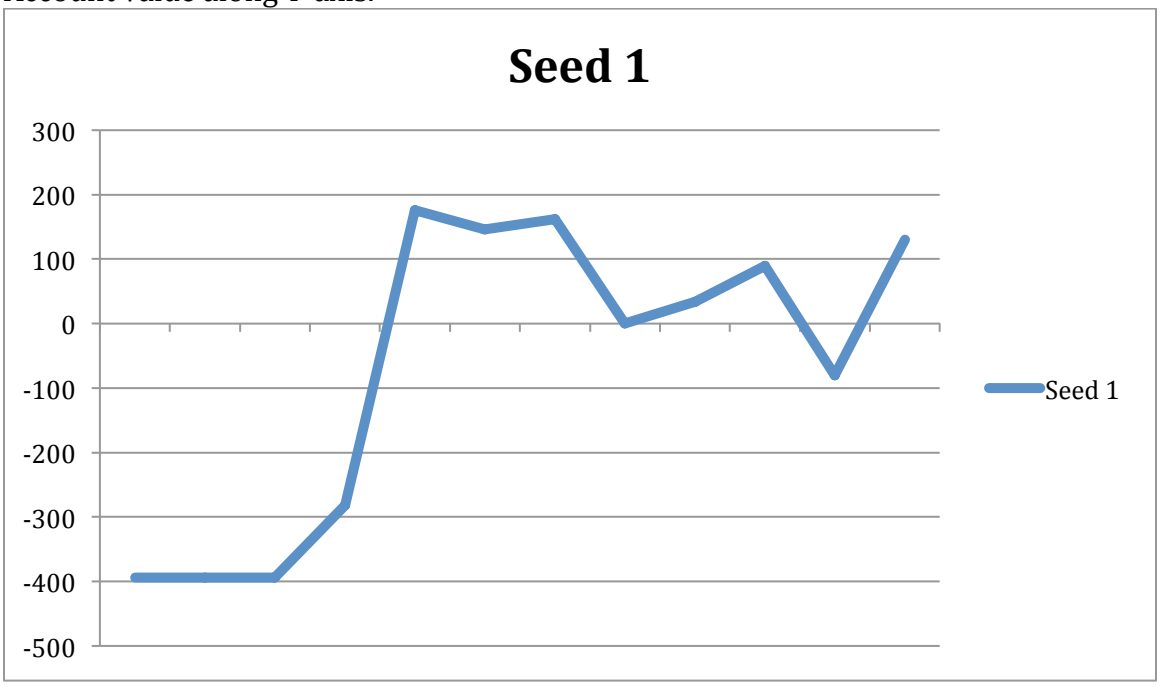
Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
115	22	810	-394
115	22	810	-394
115	22	810	-394
115	22	698	-282
115	10	240	176
115	10	270	146
115	14	254	162
115	8	416	0
115	10	382	34
115	12	326	90
115	8	496	-80
115	14	286	130

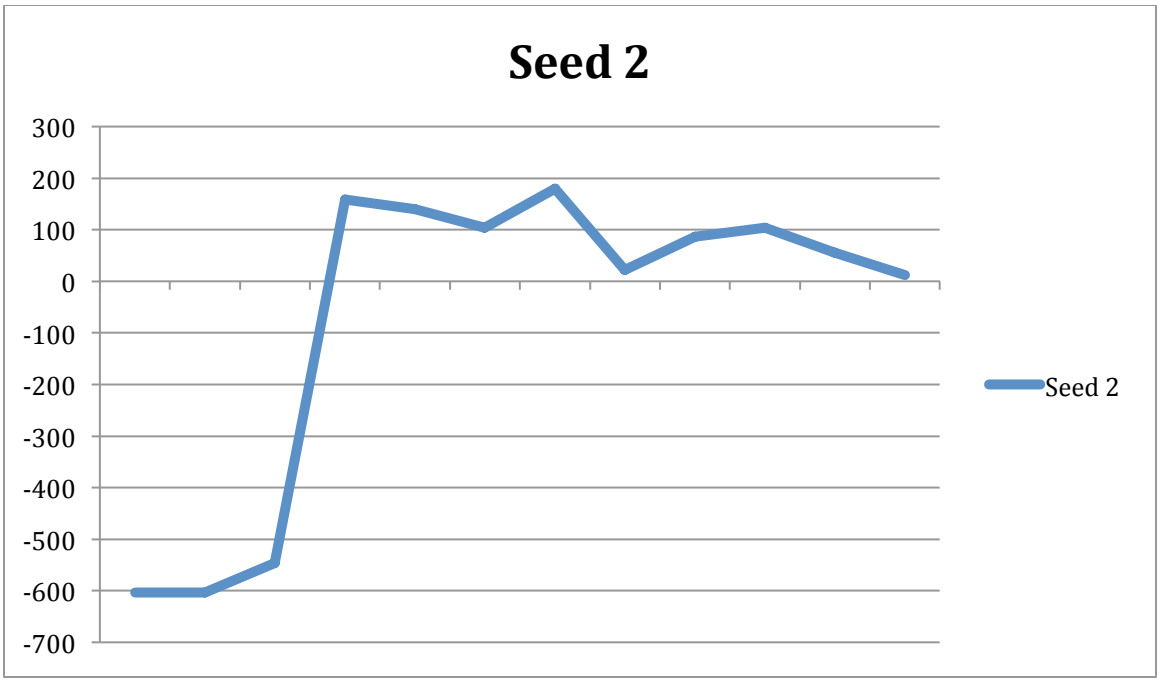
Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
2241	28	1020	-604
2241	28	1020	-604
2241	28	962	-546
2241	12	258	158
2241	8	276	140
2241	18	312	104
2241	8	236	180
2241	32	394	22
2241	12	330	86
2241	8	312	104

2241	16	360	56
2241	10	404	12

BANK ACCOUNT:

Here, term numbers(the count of terminal state) are taken along X- axis and Bank Account value along Y-axis.

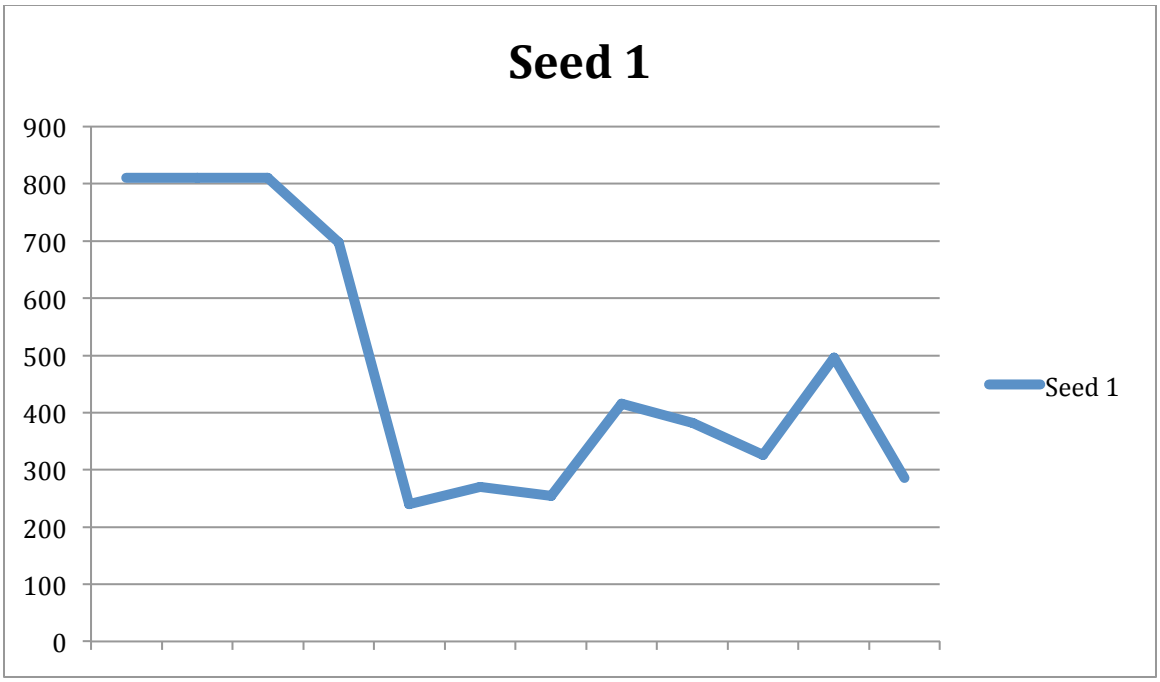


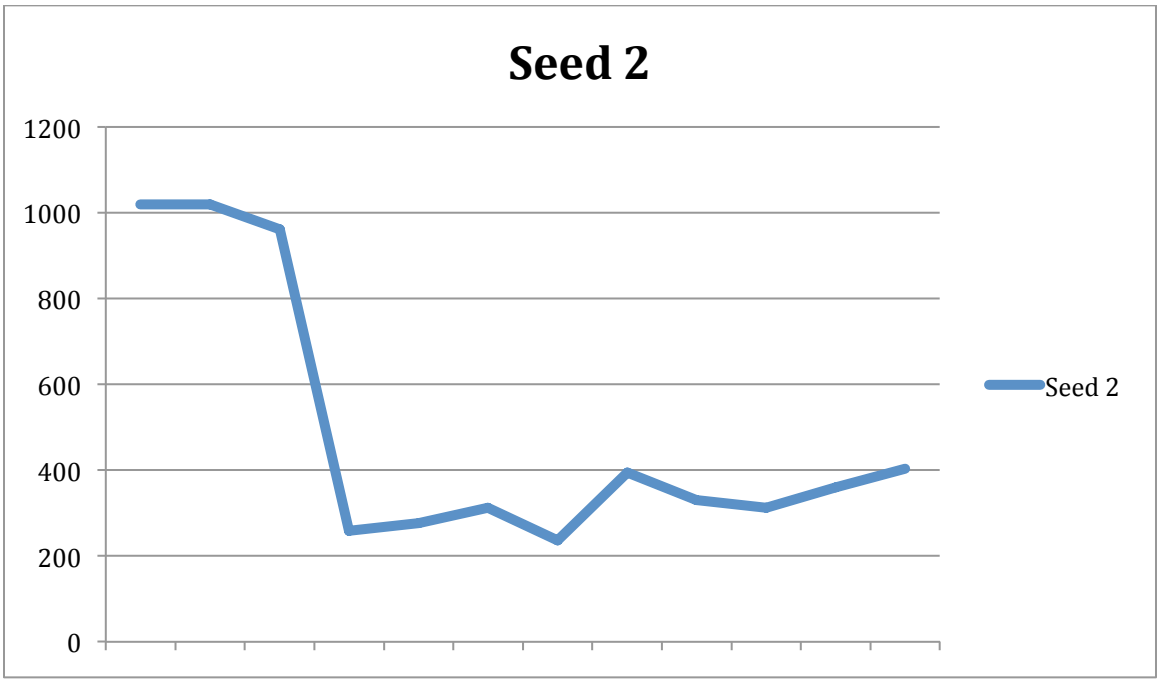


From the graphs of Bank Account, we can infer that the agent is improving because the Bank Account increased over time except at some points where a bad move is taken due to the nature of greedy policy we implemented.

STEPS TAKEN TO REACH TERMINAL STATE:

Here, term numbers(the count of terminal state) are taken along X- axis and the step count to reach terminal along Y-axis.





EXPERIMENT2:

In this experiment, PRANDOM is ran for the first 200 iterations and PEXPLOIT for the next 5800 iterations. During the first 200 iterations, the agent moves randomly, because there is no information available in the Q-Table. After that, the agent

switches to Exploit policy with the probability of choosing the random action with 15% and the rest greedily .

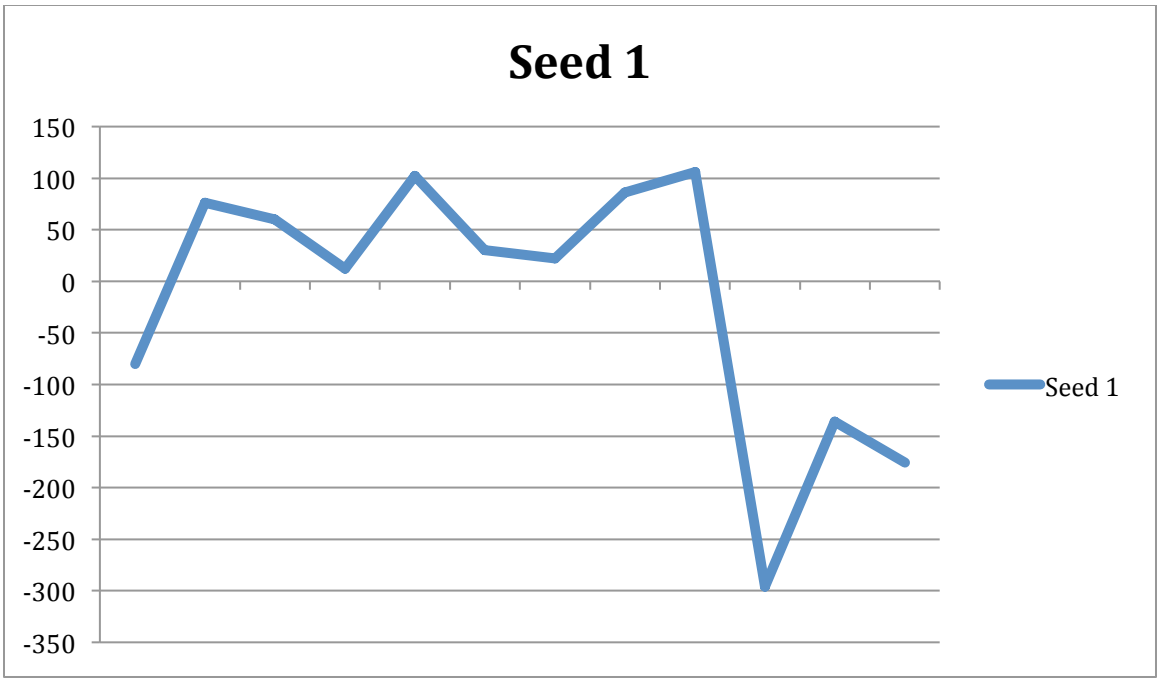
For seed 1, the agent reached terminal state for the first time in 496 steps. It reached the terminal state in 340 steps for the second time. The agent reached the terminal state 12 times. The bank account is positive for 8 times and negative for 4 times. These results indicate that the agent is performing good.

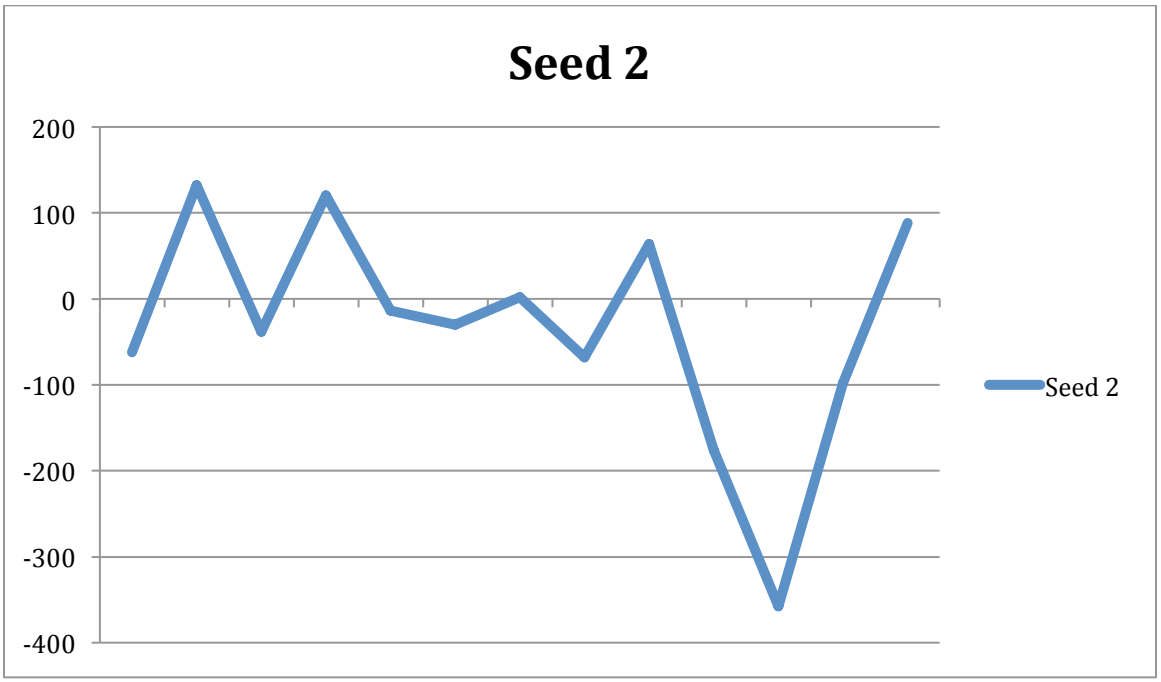
Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
115	22	496	-80
115	14	340	76
115	8	356	60
115	30	404	12
115	16	314	102
115	8	386	30
115	18	394	22
115	16	330	86
115	8	310	106
115	14	712	-296
115	40	552	-136
115	14	592	-176

Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
2241	28	478	-62
2241	16	284	132
2241	10	454	-38
2241	16	296	120
2241	10	430	-14
2241	24	446	-30
2241	10	414	2
2241	24	484	-68
2241	12	352	64
2241	10	592	-176
2241	8	774	-358
2241	18	514	-98
2241	22	328	88

BANK ACCOUNT:

Here, term numbers(the count of terminal state) are taken along X- axis and Bank Account value along Y-axis.

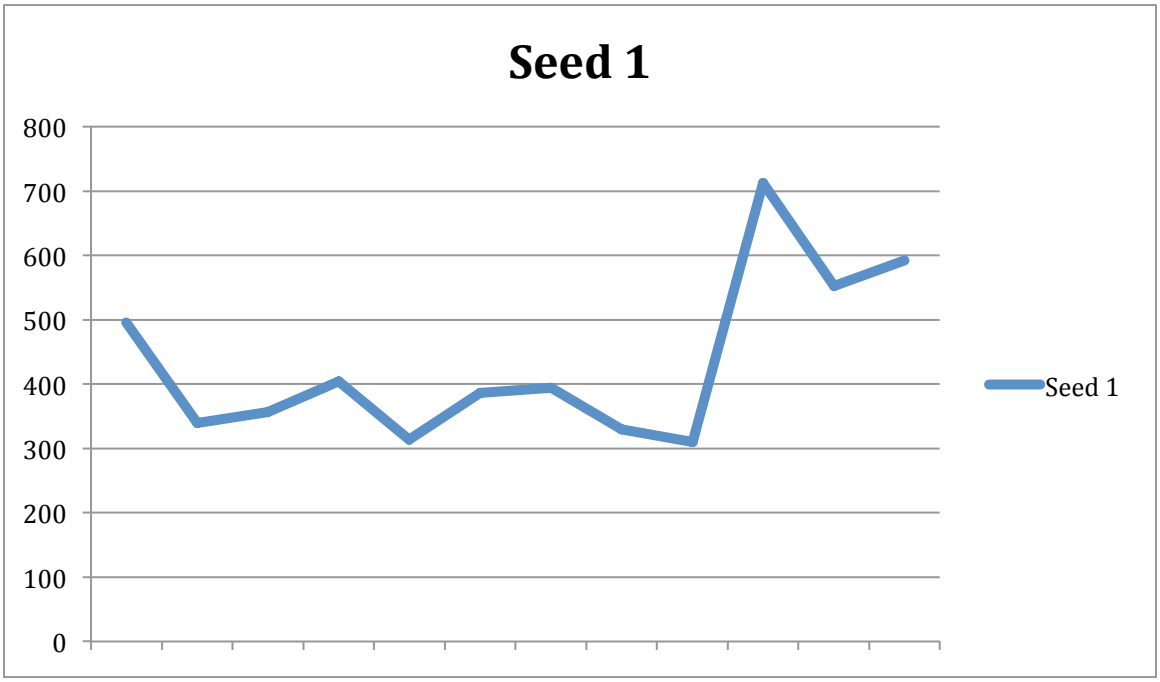


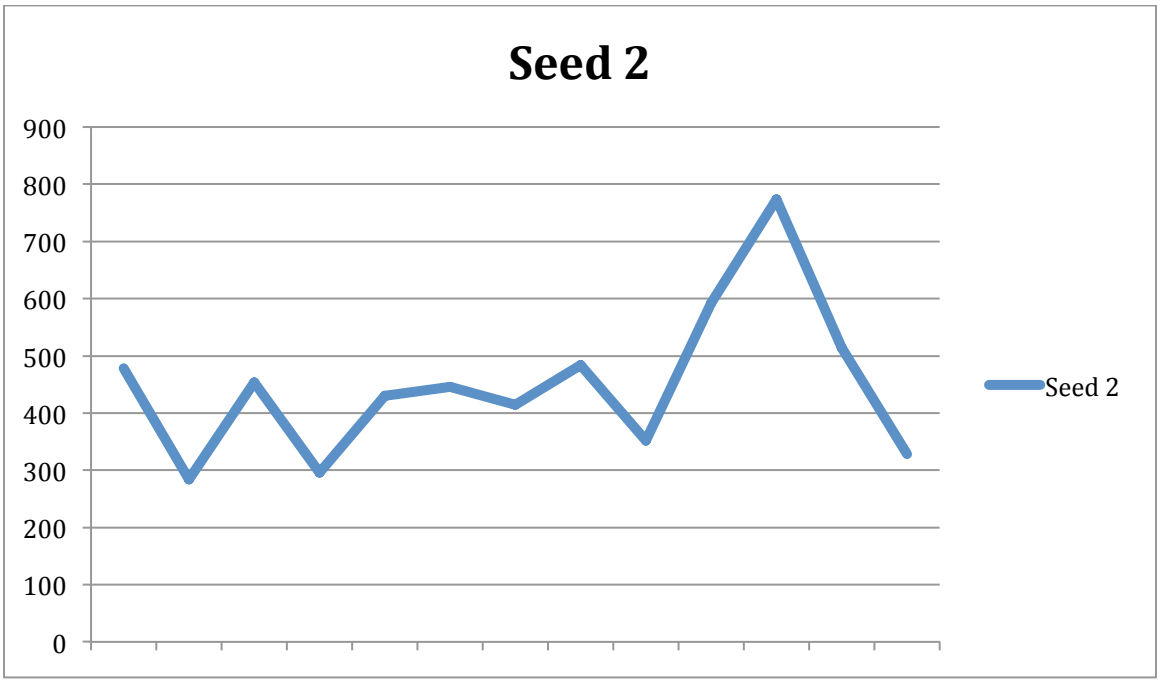


The sudden drop in Bank account value is due to the selection of random action with 15% probability. But, it won't drop often because the probability of selection random action is very less.

STEPS TAKEN TO REACH TERMINAL STATE:

Here, term numbers(the count of terminal state) are taken along X- axis and the step count to reach terminal along Y-axis.





EXPERIMENT3:

In this experiment, PRANDOM is ran for the first 200 iterations and PEXPLOIT for the next 5800 iterations using SARSA. The agent took a lot of iterations to reach the first terminal state. The number of times the agent reaches the terminal state reduces as the experiment progresses.

For seed 1, the agent reached terminal state 13 times. The bank account is positive for 9 times and negative for 4 times. For seed 2, the terminal state is reached 12 times, out of which the bank account is positive for 6 times.

RESULTS:

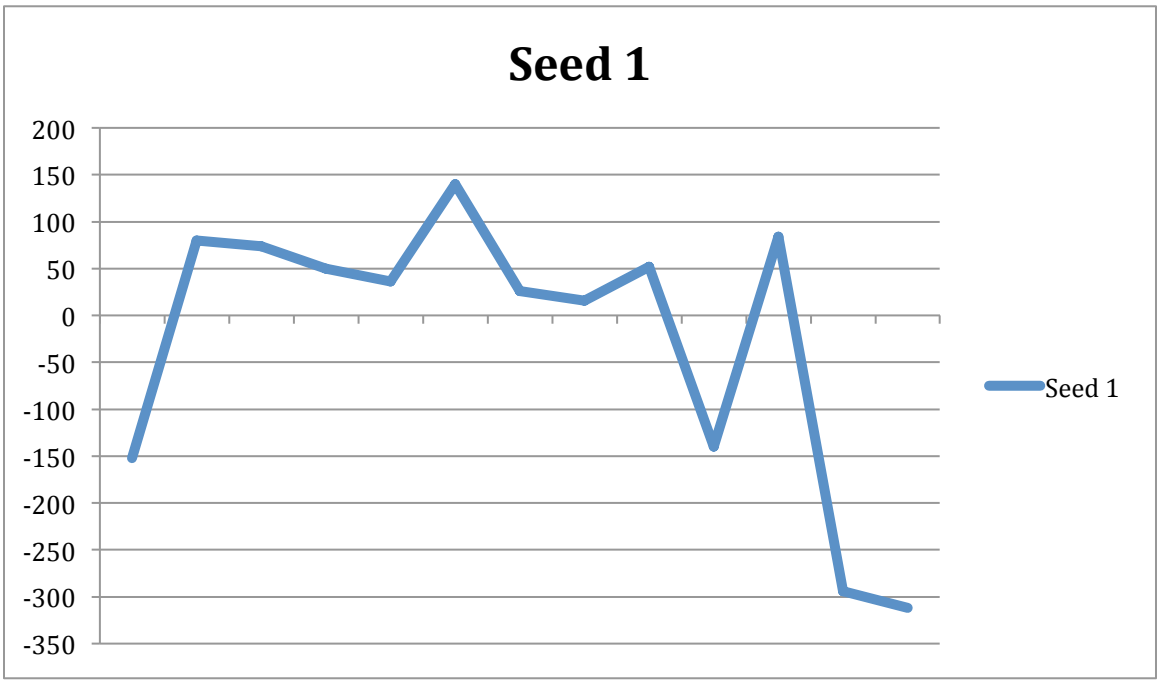
Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
115	84	568	-152
115	16	336	80
115	10	342	74
115	16	366	50
115	8	380	36
115	20	276	140
115	44	390	26
115	10	400	16
115	18	364	52
115	8	556	-140
115	12	332	84
115	8	710	-294
115	30	728	-312

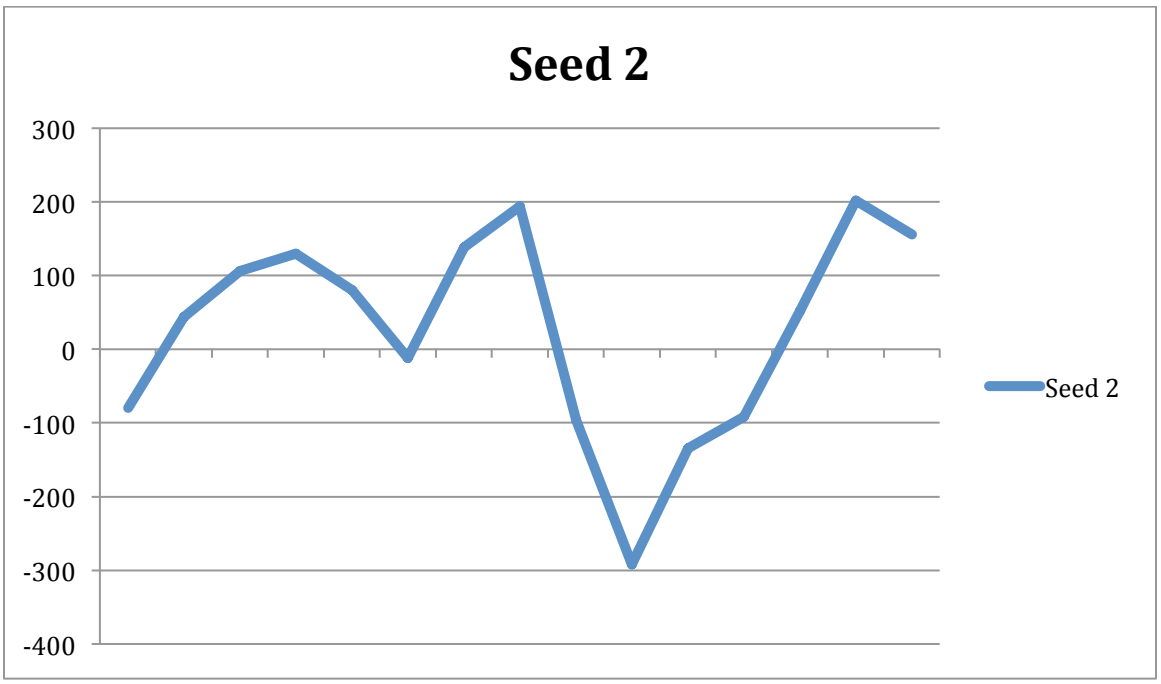
Seed	First drop off iteration	Total number of steps to reach terminal state	Bank Account
2241	16	496	-80
2241	16	372	44
2241	14	310	106
2241	16	286	130
2241	8	336	80
2241	20	428	-12
2241	12	278	138
2241	12	222	194
2241	12	512	-96
2241	12	708	-292
2241	20	550	-134
2241	22	508	-92
2241	8	364	52
2241	16	214	202

2241	8	260	156

BANK ACCOUNT:

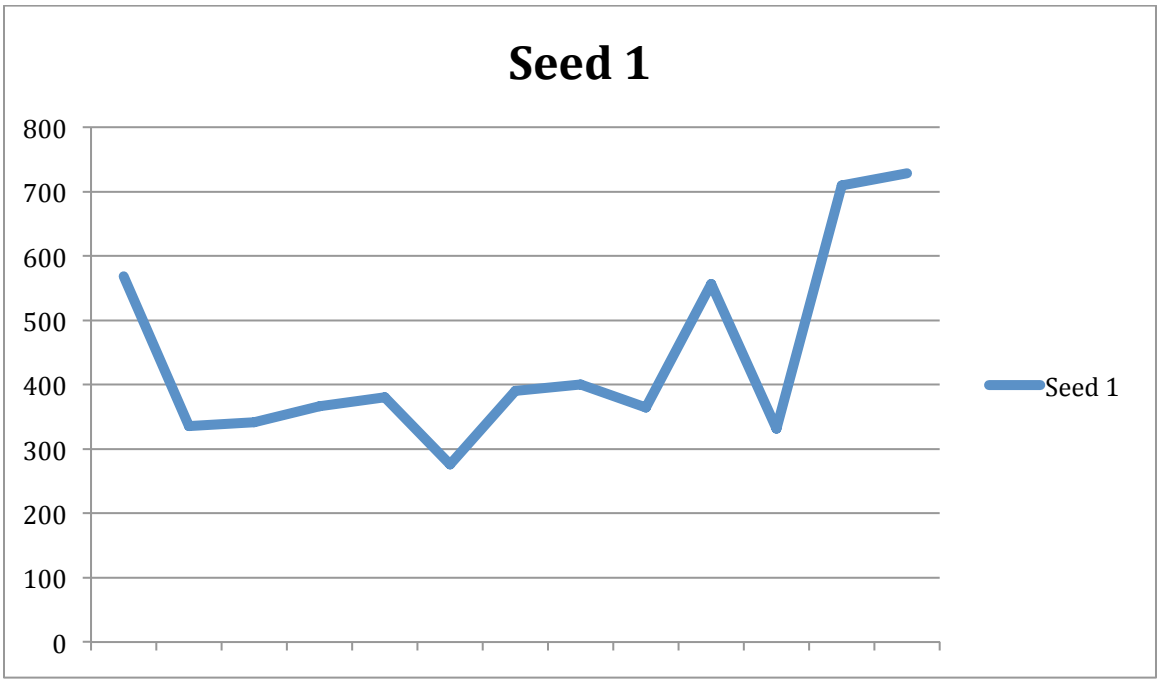
Here, term numbers(the count of terminal state) are taken along X- axis and Bank Account value along Y-axis.

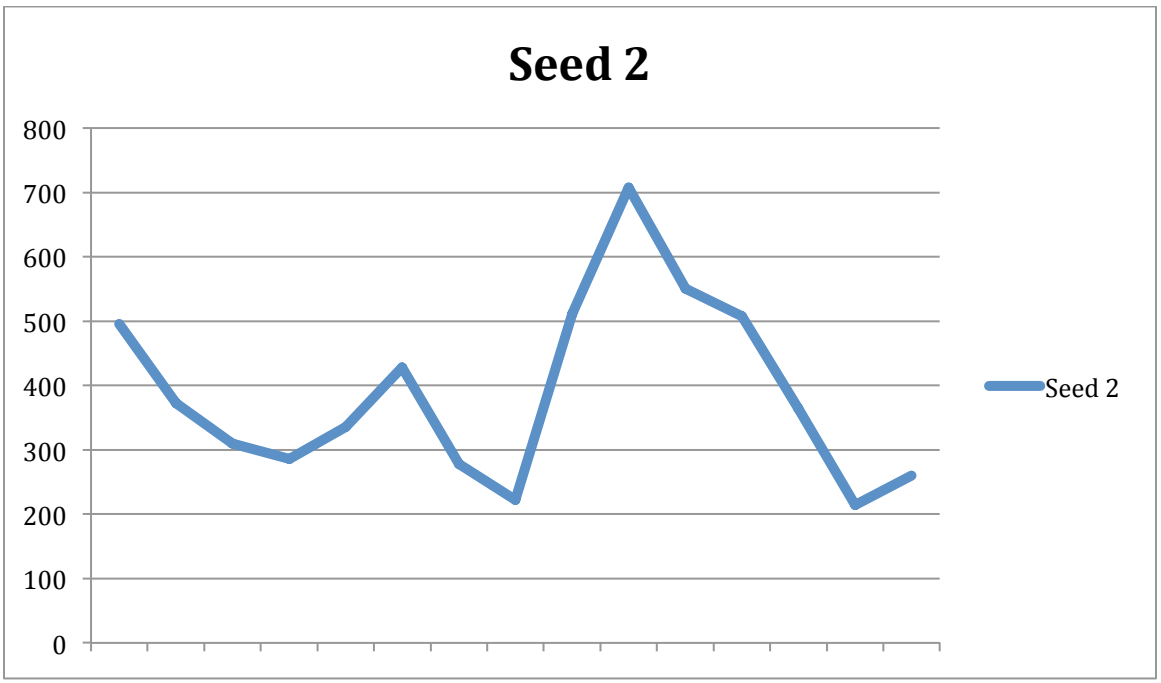




STEPS TAKEN TO REACH TERMINAL STATE:

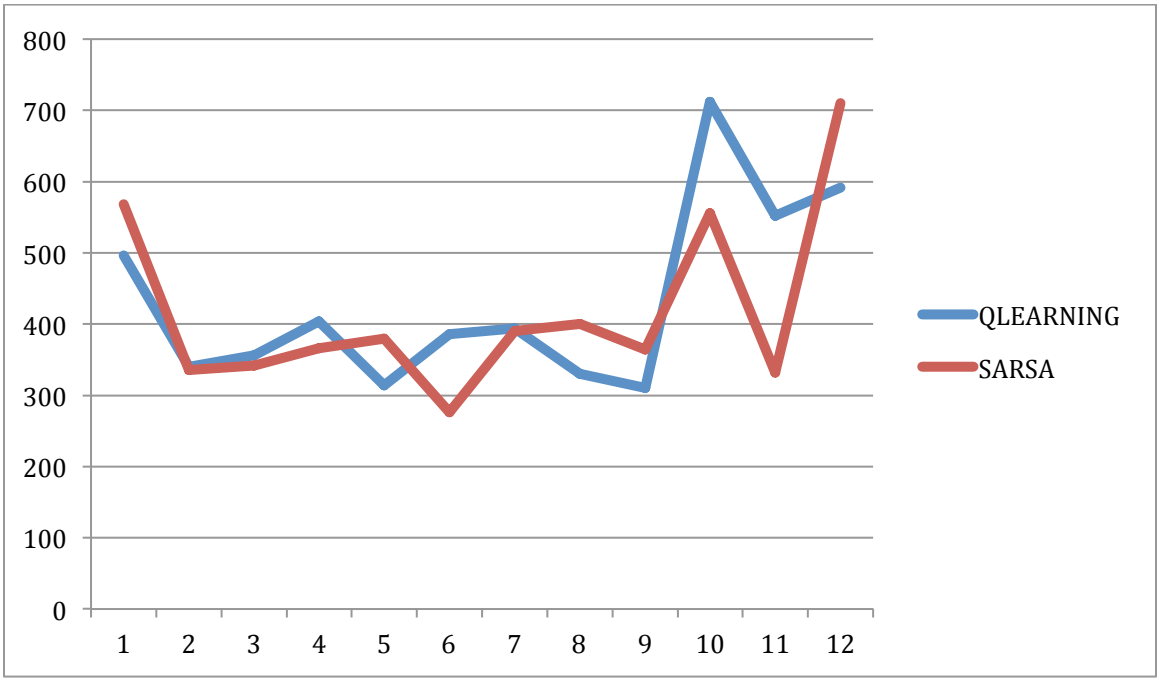
Here, term numbers(the count of terminal state) are taken along X- axis and the step count to reach terminal along Y-axis.





Q Learning vs SARSA:

Comparing Number of steps taken to reach terminal state of Q Learning and SARSA for SEED 1:



We can't pick the better one among these. It depends on the environment and the policy that is followed by the agent. When updating the Q value of a state-action pair, Q-Learning follows greedy policy. It always takes the maximum or best action of the next state i.e. the state- action pair with maximum value.

On the other hand, while updating the Q value of a state-action pair, SARSA follows the policy underneath i.e. if it is following random policy, then any random action is picked from the next state and it is used to update the current Q value.

In our environment, SARSA is better than Q-Learning. This is because greedy policy has the characteristic of getting struck in loops. Also, the graph indicates that SARSA is better. If the agent follows greedy policy, then Q-Learning and SARSA are similar.

CONCLUSION:

In the initial stage, when the agent has no idea about the environment, it is better to use PRANDOM policy because it improves the Q-values. After the agent has learnt sufficient enough, use PGREEDY or PEXPLOIT policy to reach the terminal state fast. This can be evidenced from the the BANK ACCOUNT and THE NUMBER OF ITERATIONS TO REACH TERMINAL STATE values of all the experiments. The bank account in all the experiments kept on increasing indicating that the agent is learning.

Following a single policy doesn't yield good results. An experiment must contain a right mixture of different kind of policies. The combinations PRANDOM+PGREEDY, PRANDOM+PEXPLOIT and PRANDOM+PGREEDY+PEXPLOIT are good. Also, you should run these policies for sufficiently right number of steps to achieve good results.