Report – Assignment 3

1. **Provide the learning curves for the above experiments. Clearly label the curves by the parameters used.**

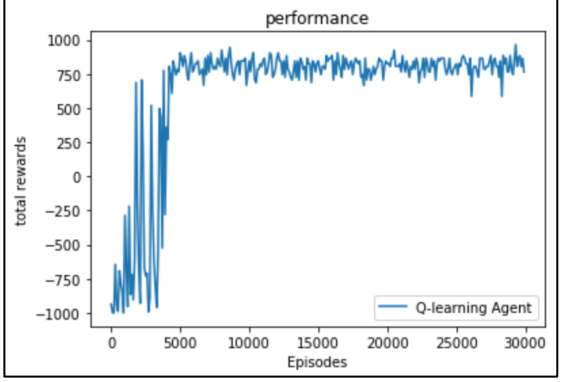
Use both SARSA and Q-Learning to learn policies for each of the two maps using learning rates of 0.001 and 0.1 (using the default value of ϵ)

30,000 episodes were used for map_DH and 100,000 episodes for map_16 as given in the instructions.

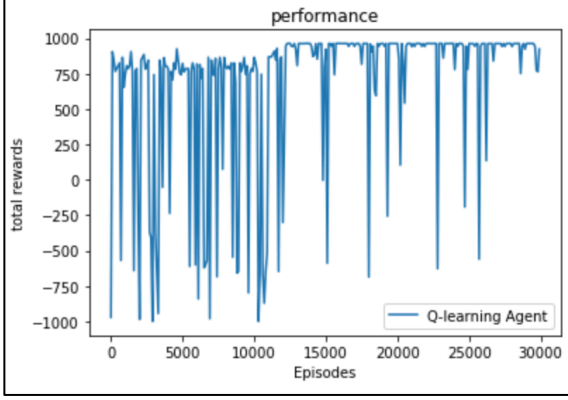
Q-Learning

(learning rate 0.1 and 0.001, epsilon 0.1, map_DH and map_16)

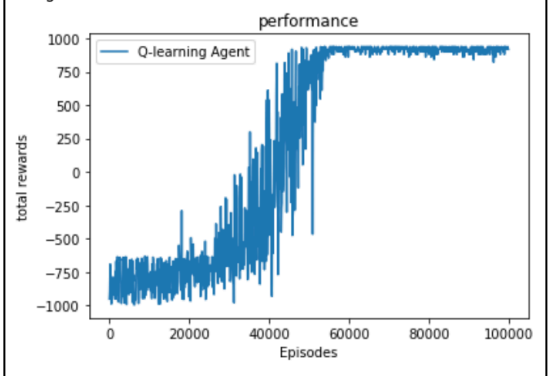
```
map_DH
epsilon 0.100   learning_rate 0.001
Learning time:

24.481563568115234

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



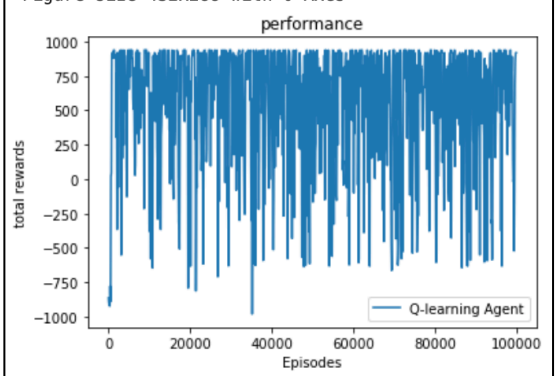

```
map_DH
epsilon 0.100   learning_rate 0.100
Learning time:

42.80897808074951

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_16
epsilon 0.100   learning_rate 0.001
Learning time:

394.0732567310333

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



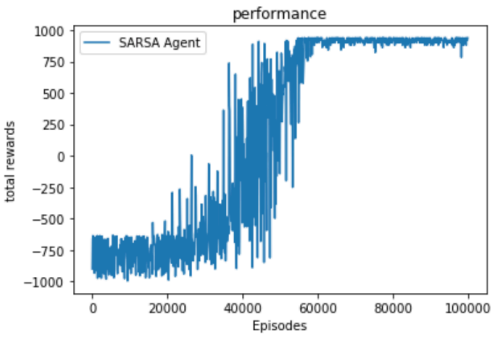

```
map_16
epsilon 0.100   learning_rate 0.100
Learning time:

290.8281874656677

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



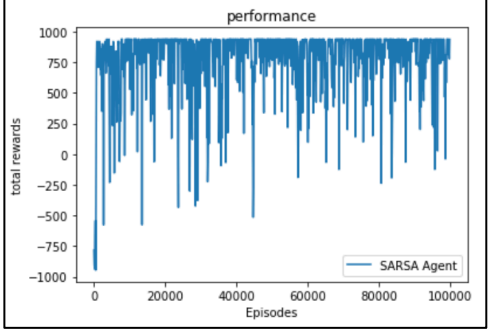
SARSA

(learning rate 0.1 and 0.001, epsilon 0.1, map_DH and map_16)

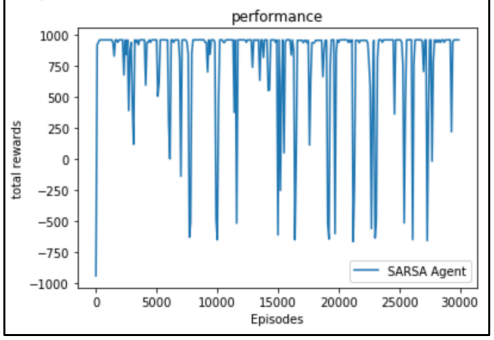
```
map_16
epsilon 0.100   learning_rate 0.001
Learning time:

434.8185350894928

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



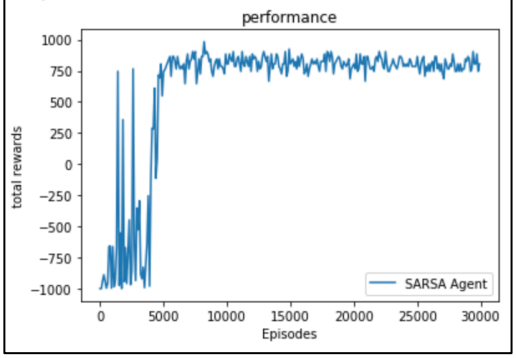

```
map_16
epsilon 0.100   learning_rate 0.100
Learning time:

189.22317361831665

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_DH
epsilon 0.100   learning_rate 0.100
Learning time:

58.42542290687561

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_DH
epsilon 0.100   learning_rate 0.001
Learning time:

27.795287132263184

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



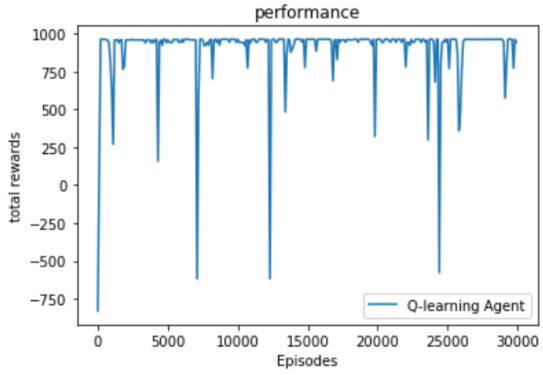
Use the better of the two learning rates to learning policies using SARSA and Q-Learning on each of the two maps using ϵ values of 0.3, 0.05.

The better learning rate was found to be 0.1 as it was able to reach the maximum reward sooner.

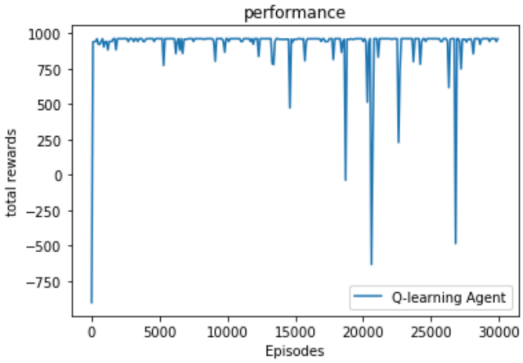
Q Learning

(learning rate 0.1, epsilon 0.3 and 0.05, map_DH and map_16)

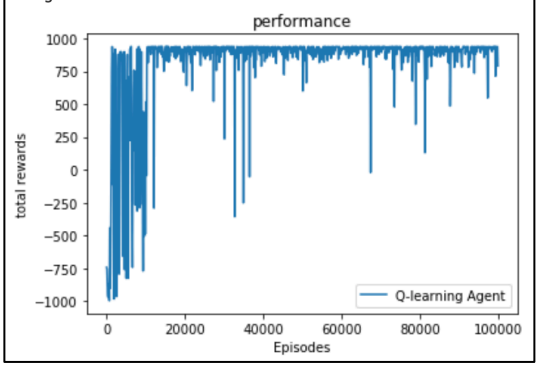
```
map_DH
epsilon 0.300   learning_rate 0.100
Learning time:

27.50800323486328

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



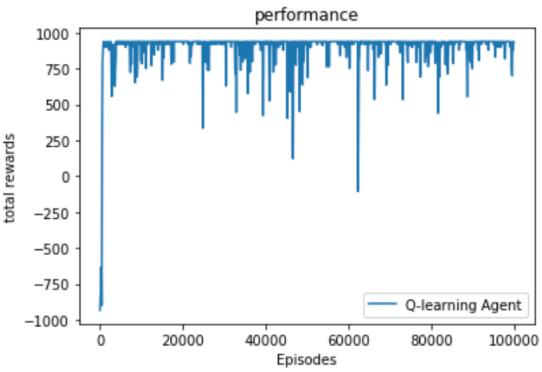

```
map_DH
epsilon 0.050   learning_rate 0.100
Learning time:

20.58450436592102

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_16
epsilon 0.300   learning_rate 0.100
Learning time:

137.7546741962433

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



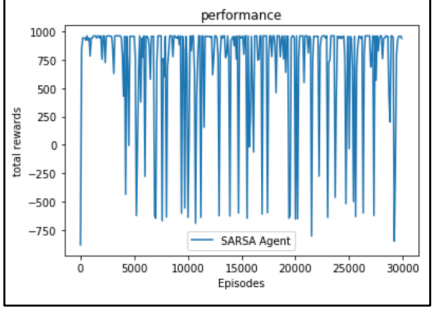

```
map_16
epsilon 0.050   learning_rate 0.100
Learning time:

96.860671043396

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



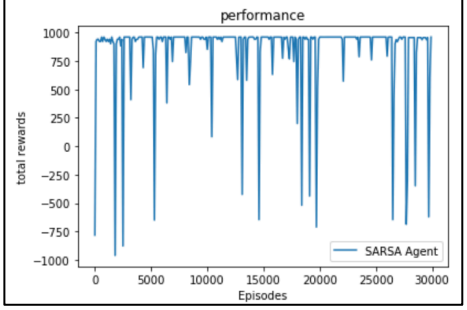
SARSA

(learning rate 0.1, epsilon 0.3 and 0.05, map_DH and map_16)

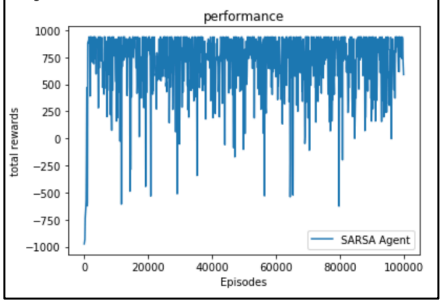
```
map_DH
epsilon 0.300   learning_rate 0.100
Learning time:

88.77084398269653

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



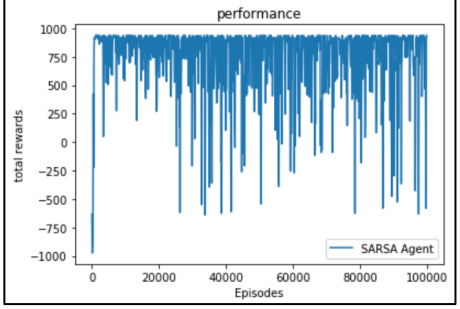

```
map_DH
epsilon 0.050   learning_rate 0.100
Learning time:

37.86541247367859

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_16
epsilon 0.300   learning_rate 0.100
Learning time:

238.16823554039001

Learning Performance:

<Figure size 432x288 with 0 Axes>
```




```
map_16
epsilon 0.050   learning_rate 0.100
Learning time:

202.34484434127808

Learning Performance:

<Figure size 432x288 with 0 Axes>
```



2. **Did you observe differences for SARSA when using the two different learning rates? If there were significant differences, what were they and how can you explain them?**

Yes, I did. Learning rate of 0.1 seems to perform better than 0.001. This can be observed in the performance graph. When using learning rate of 0.1, we can see that maximum reward is achieved much sooner.

With a lower learning rate, the local or global optima are reached much more slowly because the updates are in small increments. With a higher learning rate, local and global optima are reached sooner, but the risk can be overshooting the optima.

3. **Repeat (2) for Q-Learning.**

The same can be said for Q-Learning:

The learning rate of 0.1 performed better and achieved the maximum reward sooner.

With a lower learning rate, the local or global optima are reached much more slowly because the updates are in small increments. With a higher learning rate, local and global optima are reached sooner, but the risk can be overshooting the optima.

4. **Did you observe differences for SARSA when using different values of ϵ? If there were significant differences, what were they and how do you explain them?**

Yes, I did. The value functions derived by the two varied. The value function with 0.05 explored much less compared to that of 0.3.

Epsilon determines the exploration vs exploitation tradeoff. Lower the epsilon (closer to 0), lower the exploration (higher exploitation) and higher the epsilon (closer to 1), higher the exploration (lower exploitation).

5. **Repeat (4) for Q-Learning.**

The differences here weren't as significant as in SARSA, but there still were differences very similar to that in (4). The reasoning would be the same as (4) i.e. epsilon determines the exploration and exploitation.

6. **For the map "Dangerous Hallway" did you observe differences in the policies learned by SARSA and Q-Learning for the two values of epsilon (there should be differences between Q-learning and SARSA for at least one value)? If you observed a difference, give your best explanation for why Q-learning and SARSA found different solutions.**

Yes, there were differences.
Q-learning is an off-policy algorithm and it will find the optimal policy even if the exploration policy given is completely random. It is not sensitive to the action at s', while SARSA must be given the action at s'. SARSA on the other hand is an on-policy algorithm and the policy derived is not necessarily optimum. The differences in off and on-policy algorithms thus, would lead to different policies being derived.

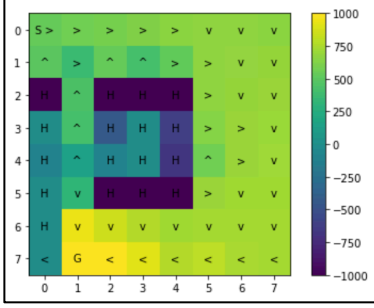
7. **Show the value functions learned by the distributed methods for the best policies learned with epsilon equal to 0.3 and compare to those of the single-core method. Run the algorithm for the recommended number of episodes for each of the maps. Did the approaches produce similar results?**

Distributed method:

```
map_DH
epsilon 0.300   learning_rate 0.100
Learning time:

88.77084398269653

Learning Performance:
```

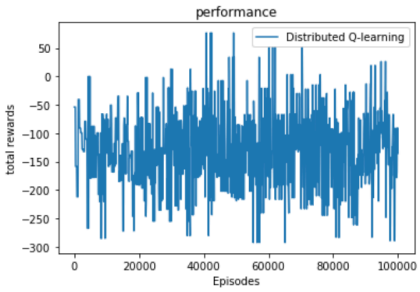
<Figure size 432x288 with 0 Axes>

Best Q-value and Policy:

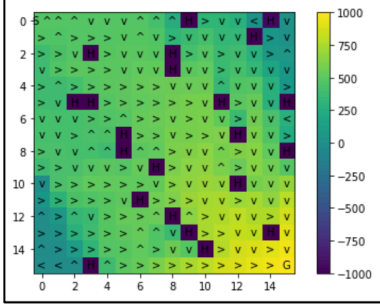


```
map_16
epsilon 0.300   learning_rate 0.100
Collectors:  4
Evaluators:  4
Learning time:

130.86214423179626

Learning Performance:
```

<Figure size 432x288 with 0 Axes>



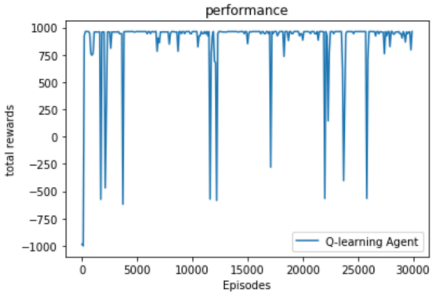
Best Q-value and Policy:



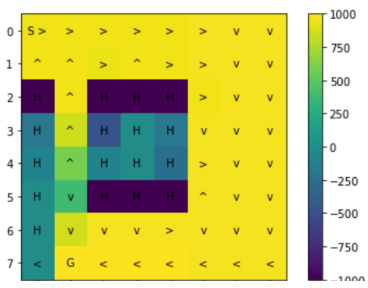
Single core:

```
map_DH
epsilon 0.300   learning_rate 0.100
Learning time:

29.798855781555176

Learning Performance:
```

<Figure size 432x288 with 0 Axes>

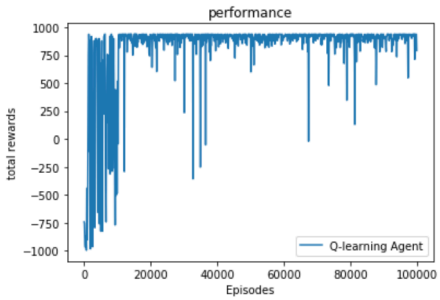


Best Q-value and Policy:

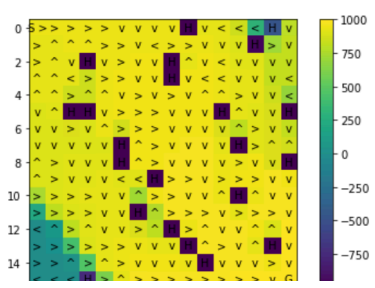


```
map_16
epsilon 0.300   learning_rate 0.100
Learning time:

137.7546741962433

Learning Performance:
```

<Figure size 432x288 with 0 Axes>



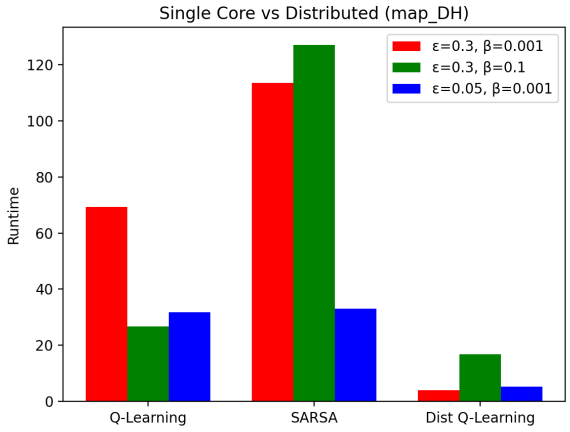
Best Q-value and Policy:



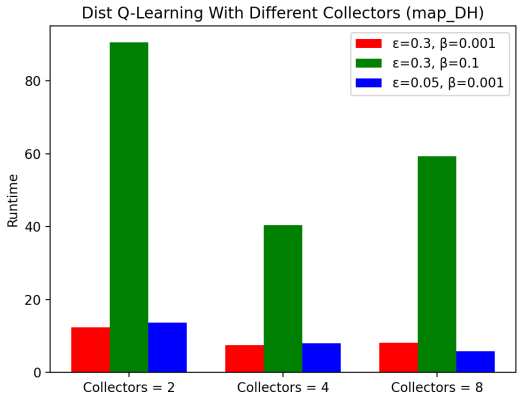
We can see that the policies found are similar when comparing single core Q-learning with its distributed variant. Both have explored most of the map. However, the learning time is significantly lower for both maps in distributed q-learning.

8. **Provide and compare the timing results for the single-core and distributed experiments, including the time to do the evaluations during learning. Describe the trends you observe as the number of workers increases.**

(I've shown these graphically for the ease of interpretation. Question (9) contains numerical results from a different test)



We can see that distributed-Q-learning performs significantly better than single core Q-learning and SARSA.



The performance increases when the number of cores increases from 2 to 4, **however the performance degrades when it reaches 8**. The communication overhead could be overwhelming the 8 core version thus, leading to poorer performance compared to 2 and 4 cores.
Even though the learning time was greater in map_16, it showed a similar pattern of results as described above.

9. **Provide and compare the timing results for the single-core and distributed experiments with the evaluation procedure turned off. That is, here you only want to provide timing results for the learning process without any interleaved evaluation. Compare the results with (8).**

For this step, I ran the results again (hence the difference in numbers from the graph given in (8). But this graph can also be used for question (8).

learning rate = 0.1, epsilon 0.3, evaluation cores = 4

Q-learning	**map_DH**		**map_16**	
Cores	**Train + Eval**	**Train**	**Train + Eval**	**Train**
Non-distributed	89.4322	21.6821	719.3412	744.3211
Dist 2 collectors	32.5647	4.9411	197.6412	34.8766
Dist 4 collectors	21.2431	4.3321	126.6762	25.9211
Dist 8 collectors	22.2984	4.3099	84.2123	18.1223

With evaluation turned off, all the results ended up with a much faster learning time.