The focus of this program is to analyze the effects of TD(lambda) with varying degrees of lambda. The chosen task was a racetrack where a car had to reach from C to F. The '|' were defined as off course and moving onto these would cause a crash that resulted in a penalty of -99. Initially a penalty of -4 was used for crashing (as suggested by the problem definition, Sutton and Barto Exercise 5.4), but this led to the car sometimes determining that crashing was the optimal action to make. Further, each time-step had a penalty of -1. The car was allowed to accelerate either +1 or -1 in both up and right directions at every time-step. For the purpose of saving time, the max velocity in both directions was 5 and the minimum was 0. In addition, at every half time-step, the car moved up or right towards the finish line. Without this, the car would often get stuck at velocity 0 and fail to learn even the most basic track within a reasonable duration.

The four racetracks that we analyzed on were (available in the tracks directory)

1. basic, a straight path upwards
2. quick, a staright path rightwards
3. race, a path first straight, then right
4. diag, a path that is diagonal

The lambda's that were compared were (0 0.1 0.25 0.5 0.75 0.875 0.9 0.925 0.95 0.98 0.99 1).

For each track 1000 iterations were allowed, and an exploration constant of 0 was provided. In addition the initial q values were set to 0, to allow for optimistic learning.

Note to view the track just use the viewtrack script from the race-track directory

usage: ./viewtrack <track> <lambda>

Red locations signify the how many choices the Agent is still not sure of at those location.

White number signify the desired velocity the agent aims towards

Upon viewing the Figure 1, we notice that for all tracks except basic, a lambda value of 1 provides the fastest learning, meaning MC updates provided the best learning. The reason for this is probably because the half-step highly favors the upward motion. This way, the basic path reaches its goals regardless of the the right direction is in a minimum of 24 time steps. Meanwhile for the quick track, even though it is straight right, it is only able to go right automatically once it has reached the top of the track. This may also explain the why when lambda is 0, the agent is not able to learn anything for sure, only that moving up in the top row is bad.

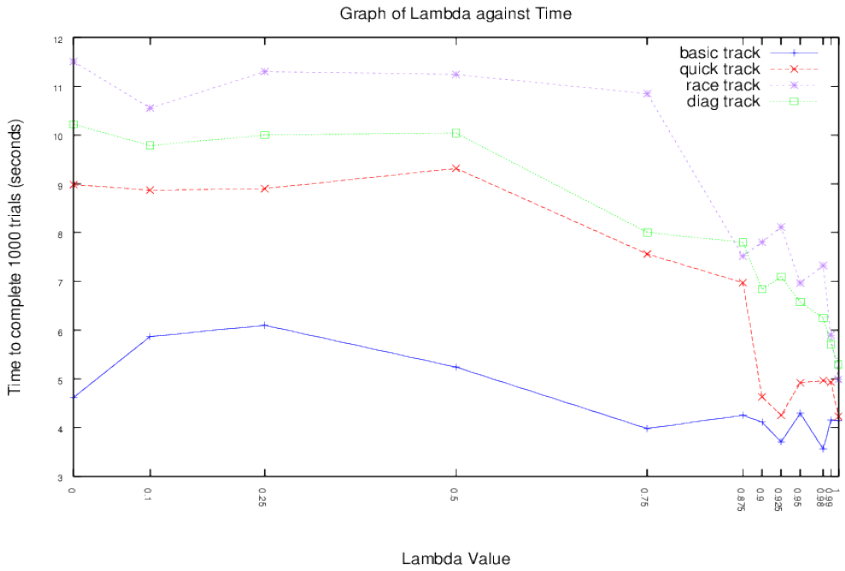
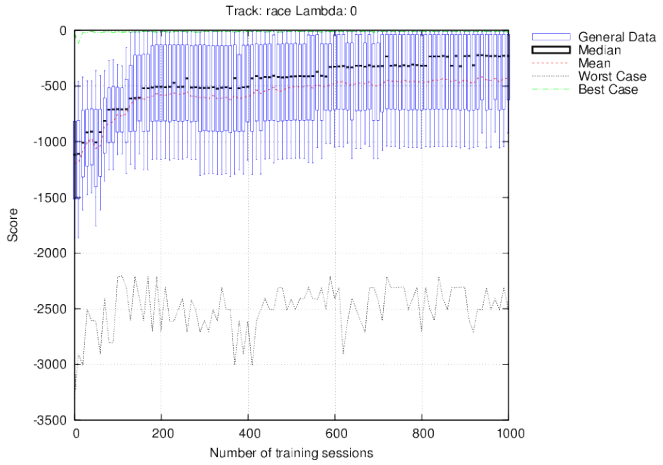
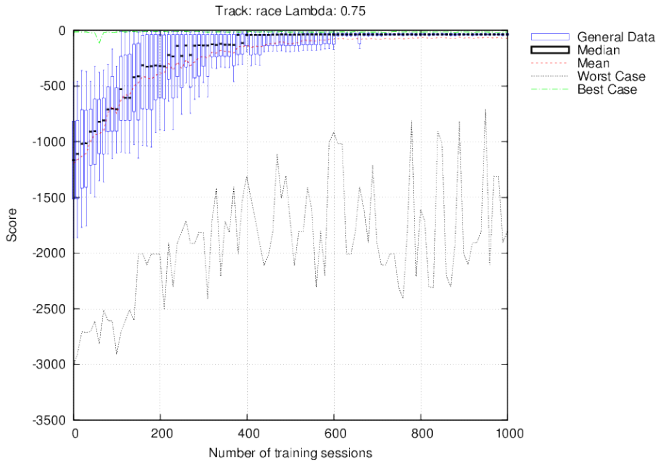
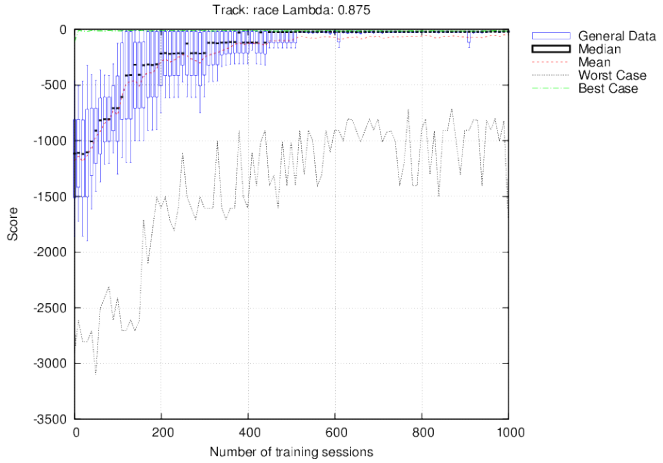
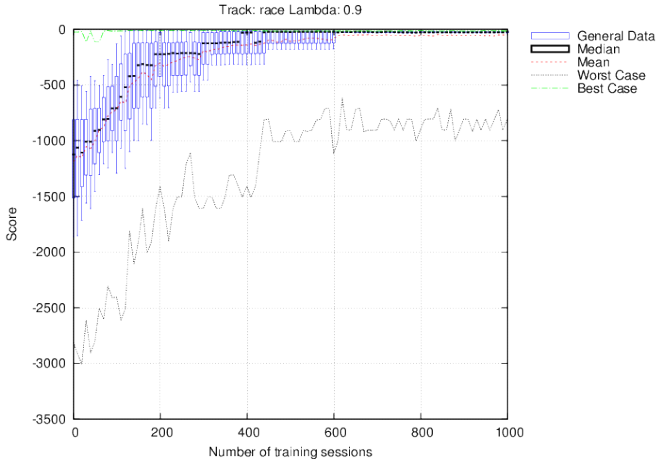


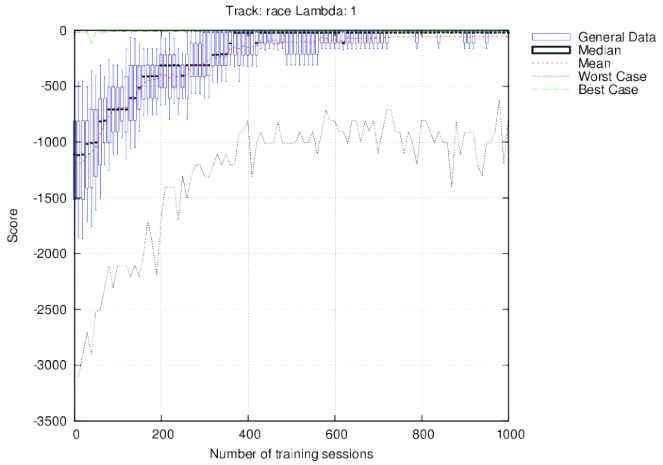
Figure 1: Graph of total time to iterate through 1000 trials











Upon looking at the graphs for the race track and lambda values (0, 0.75, 0.875, 0.9, 1), we notice the following. Though at TD(0), we are learning, but the rate of learning is very slow. Given more trials it should converge to the optimal track, but as of now, we can expect that 75% of the data will only be crash less than 10 times while going through the track. Interestingly, though at TD(0.75) the track seems to have converged towards a minimal path after 600 trials, it still has very poor recognition of which paths are poor as the variance of the worst case track is extremely large. The difference between TD(0.825) and TD(0.75) is earlier convergence of TD(0.875), approximately 100 less trials, and the start of recognizing poor tracks around trial 150. However, at TD(0.925), though the convergence of appears again at ~500 trials, it again loses the ability to recognize poor tracks. At TD(0.95), we again converge at ~500 trials, but compared to TD(0.875), TD(0.95) has two large jumps in recognizing poor tracks, the first at ~100 trials, and another at ~275 trials, after which it stabilizes in its learning of poor tracks. Though TD(1) takes a lot longer to converge, ~700 trials, it seems to learn which tracks are bad much faster. However, before 400 trials, TD(1) has 50% of its trials converging towards the minimal run.

If we analyze the tracks produced by them as well we again notice that TD(1) has the least number of unsure (red) locations. However, its also interesting why it learns to crash in the first few top-left

locations of the Up Policy and the side of the track for the Right Policy. A possible explanation could be the lack of visits that occurred at those locations since as the policy converged, it would not approach those locations.

If we look at the graphs of basic and compare TD(0.25) and TD(1) to TD(0.98), we notice that again the largest factor for time is how early the policy is able to discern poor policies. One thing that is still interesting is why TD(0.925) takes such little time. It takes longer to converge that TD(1), and fails to recognize any of the poor paths. This may be just be a program error as its not able to recognize minute time differences.

The remaining graphs are available in the results/graphs directory. There is nothing unique for the quick or diag path besides what is already observed in basic and race.

One thing that is critical to understand is that the “best” lambda value here was determined by total time; however if we observed the number of trials until convergence, then we would arrive at a different “best” lambda value. In addition, analysis was not done for differing initial Q values or exploration constants.