**Purpose**

There were several purposes to this assignment. One detail to note first is that we changed the reward for the apple to 75 instead of 50. The 50 reward was not large enough to give a converging solution immediately (not changing live reward/gamma), holding all else constant. We will discuss the purpose of each experiment:

* The first is to implement the MDP to run on the gridWorld we were given. The purpose for this was to see how value iteration and policy iteration solve certain problems given an MDP (with bellman equations, transition probabilities, ect.).
* We then changed some variables in the gridWorld (live rewards) and MDP (gamma). The purpose for this was to see how varying values would change the solution.
* The last task was to modify the MDP to include a jumping action for the horse. The purpose for this was to add another action and see how this new, jumping action would effect the solution (shorter or “better” solutions, ideally).

**Procedure**

The overall procedure to find a solution: We ran the value/policy iteration in combination with some of the auxiliary functions (best\_policy, policy\_evaluation was implemented automatically in policy\_valuation) to find utilities for each state, and then create a dictionary of the best transition from state-to-state. Thus, we were able to look up the dictionary and connect the states to find a converging solution, if there was one. If there was not a solution to the apple, i.e. run into a wall and stay in current position, we checked for an invalid state; running into a wall will produce a transition to a state that is not available to move to (a wall).

We will discuss procedure in each experiment in the same format and order as purpose:

* After formatting our grid into a matrix with a terminal state (apple goal-state), running value iteration and policy iteration we converged to a solution. This solution was: [(0,0),(1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(6,1),(6,2),(6,3),(6,4),(7,4),(7,5),(8,5),(9,5),(9,6),(9,7)].

The only modifications used were the modification of the matrix, so that it could be made into a GridMDP object.

* For the live rewards a different solution was reached when we changed the live rewards to positive values, .9 and 1. Otherwise, (.1 - .8) the solution was unvaried. For negative values, we converged to a different solution on each of the values, -.1 to -1, though some of these different solutions were the same, but they all varied from the original solution, given in the bullet point above. More specifically, -.1 to -.3 returned a solution where we actually reached the terminal, apple state. For -.4 to -1 the utility was greater if we stayed at a barn (i.e. just keep running into the wall). Each out come is specified in my program, along with how the path differs with the original solution.

For the gamma change we did not converge to a solution for any of the modified values, other than our original .9. Some of the values that were close to .9 (i.e. .8, .7, .6) were closer to reaching a solution versus smaller values. For value 1, there was an infinite solution (i.e. never stop moving). The gamma change was run with variables that matched the original solution problem. There may have been different outcomes if we mixed the experiments, i.e. ran different gammas with different living values.

* We assigned a whole new action for the jumping, with the specified transition probability of .5 successes and .5 stay in original state. We also implemented a “jump” function in the utils.py file (utils2.py in the jump folder). One can find this change on line 525, or next to the other orientation functions (turn\_right, turn\_left).

**Data**

The data used started with the original matrix for the horse gridWorld. The initial living rewards, were 0, with mountains giving a -1 reward and snakes a -.5 reward. As mentioned in the Purpose section, we changed the apple value to 75 to converge to a solution immediately, without changing any other values. The first two experiments (living reward and gamma) involved changing the values of the living reward (-.1 to 1) and then gamma (.1 – 1). The gamma directly effected the bellman equation, while living reward had a subtler effect, but still could drastically change the solution (especially with negative values).

**Results**

Some of the results were discussed in the procedure portion, yet we will summarize them in more detail:

* For the initial solution, with only implementing the horse gridWorld and holding all else constant, we converged to a solution of: [(0,0),(1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(6,1),(6,2),(6,3),(6,4),(7,4),(7,5),(8,5),(9,5),(9,6),(9,7)]. We particularly found it interesting that on several occasions going through the mountain was preferred versus the snake, though we needed one extra step to go through the snake. We can conclude from this that because the negative value of the mountains and snake only varied by .5, that this small variation had only a small effect on the utility.
* For negative values, -.1 to -.3 returned a solution where we actually reached the terminal state, and -.4 to -1 the utility was greater if we stayed at a barn (i.e. just keep running into the wall). As for positive values, .9 and 1 produced different solutions, while .1 - .8 did not effect the original solution (solution was the same). All of the different solutions for each value, or lack there of, are outputted when running the program.
* The horse jump achieved a solution of: [(0,0),(1,0),(3,0),(5,0),(7,0),(7,2),(7,4),(9,4),(9,6),(9,7)]. The 1-step area sometimes vary between value and policy iteration. Though, the general path does not vary (only when to take a 1-step since the matrix has odd rows). We concluded that this occurs because the utility of certain solutions are equal, i.e. taking 1-step from a state at one position on the grid are equal to taking 1-step at a different position on the gird. Transition probabilities (also mentioned in procedure) were .5 for a successful jump and .5 for stay in state. The original transitions for just 1-step were .8 for success, and .1 for right-step/left-step.