CS 4320/5314
PA3: Solving MDPs
DUE: Sunday, Dec 3 at 11:59 PM

Groups: You may optionally work in a group of two for this assignment. Groups of three that have worked together on the first two assignments will be allowed to continue working together. Turn in only one copy per group, clearly listing all team members. In your writeup you should including a brief discussion of what the contributions of each individual team member were towards the final submission. It is expected that each team member will contribute significantly towards the design, coding, testing, and documentation aspects of the project. Paired coding and similar approaches are encouraged.

Objective: To experiment with some of the basic algorithms for solving MDPs on a simple domain.

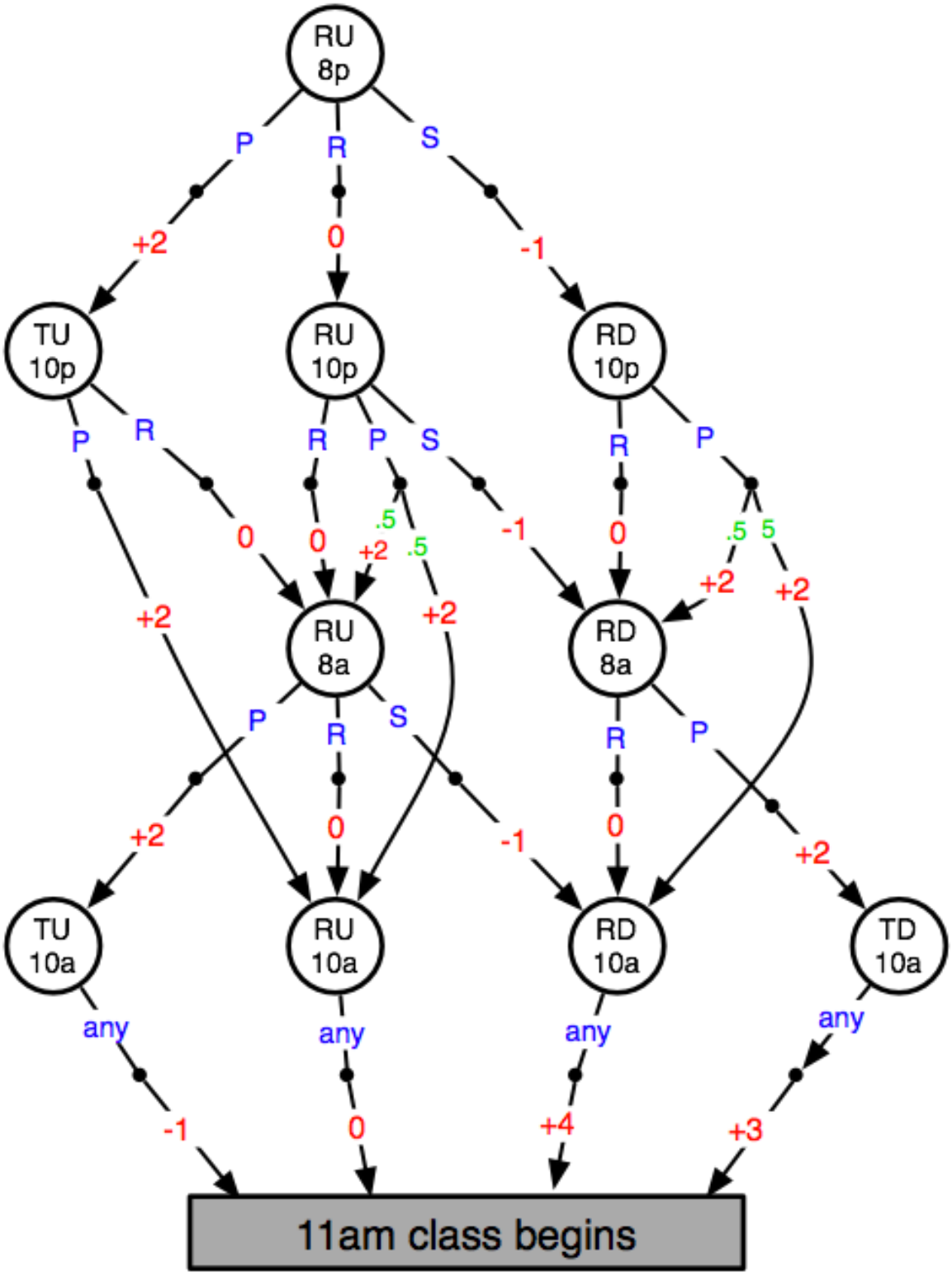
Domain: The domain is based on a simple MDP originally designed by Rich Sutton at the University of Alberta. The example describes a Markov Decision Process that models the life of a student and the decisions one must make to both have a good time and remain in good academic standing.

States:
R = Rested
T = Tired
D = homework Done
U = homework Undone
8p = eight o'clock pm

Actions:
P = Party
R = Rest
S = Study
any means any action has the same effect
note: not all actions are possible in all states

Red numbers are rewards
Green numbers are transition probabilities (all those not labeled are probability 1.0)
The gray rectangle denotes a *terminal* state.

See the following page for the diagram of the MDP.


RU
8p
P
R
S
+2
0
-1
TU
10p
RU
10p
RD
10p
P
R
R
P
S
R
P
0
0
.5
+2
.5
-1
0
.5
5
+2
+2
+2
+2
RU
8a
RD
8a
P
R
S
R
P
0
0
-1
+2
+2
TU
10a
RU
10a
RD
10a
TD
10a
any
any
any
any
-1
0
+4
+3
11am class begins

Part I: Monte Carlo

Implement a program that models the MDP above. Assume that the agent follows a random equiprobable policy (i.e. the probability of picking a particular action while in a given state is equal to 1 / number of actions that can be performed from that state). Run your program for 50 episodes. For each episode, have your program print out the agent's sequence of experience (i.e. the ordered sequence of states/actions/rewards that occur in the episode) as well as the sum of the rewards received in that episode *in a readable form*.

Perform first-visit Monte-Carlo updates after each episode to update the values of all states visited during the run. Use an alpha (learning rate) value of 0.1. Print out the values of all of the states at the end of your experiment along with the average reward for each episode, also in a readable form.

Part II: Value Iteration

Implement the value iteration algorithm and use it to find the optimal policy for this MDP. Set all value estimates to 0 initially. Use a discount rate (lambda) of 0.99. Each time you update the value of a state, print out the previous value, the new value, the estimated value of each action, and the action selected. Continue to update each state until the maximum change in the value of ***any*** state in a single iteration is less than 0.001. At the end, print out the number of iterations (i.e., the number of times you updated each state), the final values for each state, and the final optimal policy.

Part III: Q-Learning

Implement Q-learning and use it to find the optimal policy for this MDP. Note that for this algorithm you will need the Q values, which are values for state/action pairs. Similar to before, you will run episodes repeatedly until the maximum change in any Q value is less than 0.001. Use an initial learning rate (alpha) of 0.2, and a discount rate (lambda) of 0.99. Decrease alpha after each episode by multiplying the current value of alpha by 0.995. Use the same random equiprobable policy as in part I throughout the learning process (recall that the Q-learning updates and convergence are independent of the policy being followed, so it should converge as long as every state/action pair continues to be selected).

Each time you update a Q value, print out the previous value, the new value, the immediate reward, and the Q value for the next state. At the end, print out the number of episodes, the final Q values, and the optimal policy.

Submit via blackboard in a single zip file:

- The traces from each of your runs for parts I-III in readable form, and clearly labeling the final values.
- Your source code