Homework assignments for Class #5

Initial Q-table		Actions	
		Eat	Drink
States	Hungry	0	0
	Thirsty	0	0

1. Consider the agent discussed in the 1st simple example of the lecture. All the cells in the Q-table are initialized to zero values (Figure). Assume that $\alpha = 0.1$, $\delta = 0.9$ and (1) initially, an AGENT is in the state "Hungry" and executes the action "Eat"; (2) the next state of the AGENT will still be "Hungry" and it will execute the action "Drink"; and (3) after that the AGENT will stay "Thirsty" and will execute the action "Drink". What will be the resulting values in the Q-table, if they are updated using to the Bellman equation?

2. Explore the limitations of deep Q-learning by modifying the dqn_seqopt.py script and looking at how this affects the result. In particular, (1) replace the motif sequence with '000111' or other sequence of your choice, (2) increase the sequence length to 10 and (3) increase the number of iterations as needed. How many iterations will be needed in order to populate all the policy_enumerator slots? After this number of iterations, does the algorithm produce a 100% successful policy? If not, how many extra iterations will be needed in order to make the policy to 100% successful?

3. Executable release_train.py supports the command line option --min_length that specifies the lower limit on the length of the input SMILES strings that would be used by the training procedure. Using this option, compare the results of of training Generator on long SMILES strings, performed with LSTM and StackAugmentedGRU as the recurrent layers. Consider using for training only the SMILES strings > 120 tokens long, train for one epoch only and use the duration of the training and the reduction of the loss as the criteria for the comparison. NOTE: to speed up the training procedure, you may use up to 4 GPUs.