CSE 410 - AI: Homework 7
Markov Decision Processes

In this homework you will implement algorithms that are built around the transition model of Markov Chains, which are a special case of a Bayes Nets that happen to be extremely useful in practice.

1) **(30) Silly Game**

In this problem you will compute the optimal strategy for deciding to play a silly game of chance. As discussed in class, someone offers you a great deal on something you'd like to buy. For only $100 you get something that is worth $150 everywhere else. However, you must pay now. Since you are short on funds, you decide to try your hand in a (silly) game of chance. The rules are:

- playing costs $1
- with %50 chance you lose and get nothing
- with chance ϵ you get your dollar back
- with chance $0.5 - \epsilon$ you get two dollars

At each time you have the option of playing or stopping, in which case you get cache out and take all your money and use it. Since owning $100 is basically worth $150 to you, the reward function of caching out at a $100 is $50 more than the face value of cash. This results in a non-linear reward function, see `MPDSillyGame`.

Model this game as a Markov process where the state is the amount of money you currently have and solve for the optimal strategy using value iteration. The first step is to make a transition matrix that encodes the outcomes of playing the game. Then use value iteration until it converges and extract the optimal policy.

With $\epsilon = 0.05$ what is the cutoff value for when you should play? Store the answer in the `cutoff` filed of the silly game class.

2) **(70) MDP for maze solving** We will use the same maze class from last time and use it to figure out optimal paths for unreliable robots.

Like the gambling example, we need a transition model that encodes the outcomes of actions. Here the actions are up,left,down,right, and stop.

The robot succeeds with $1 - \epsilon$ probability in executing the action and with ϵ probability executes are random legal move (like the random walk from the last homework), except stopping which works perfectly.

- Generate the 5 transition matrices for encoding the actions. This can be done easily using the matrix notation and making a linear combination of matrices, i.e. $A_{\text{up}} = (1 - \epsilon)A_{\text{up prefect}} + \epsilon A_{\text{random}}$. Write a function `computeTransitionMatrices` that computes the 5 matrices.
- Use the discount factor $\gamma = 0.9$ and implement one step of value iteration `valIter()` that updates the `value` of the MDP problem.
- Finally, use these functions to implement `computePolity()`. After it is run, the `policy` field of the MDP class should have a mapping from states to actions, i.e. a list where each position holds one of the 5 actions.

2) **(10) Extra Credit (written)** For what value of ϵ do the shorter or longer paths become preferable? What is the role of γ and why did we not use it in the game?