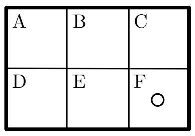
CS 188 Summer 2021 Introduction to Artificial Intelligence Midterm Review MDPs

Q1. MDP

Pacman is using MDPs to maximize his expected utility. In each environment:

- Pacman has the standard actions {North, East, South, West} unless blocked by an outer wall
- There is a reward of 1 point when eating the dot (for example, in the grid below, $R(C, South, F) = 1$)
- The game ends when the dot is eaten

(a) Consider a the following grid where there is a single food pellet in the bottom right corner (F). The **discount** factor is 0.5. There is no living reward. The states are simply the grid locations.



(i) What is the optimal policy for each state?

State	$\pi(state)$
A	
B	
C	
D	
E	

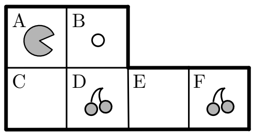
(ii) What is the optimal value for the state of being in the upper left corner (A)? Reminder: the discount factor is 0.5.

$V^*(A) =$

(iii) Using value iteration with the value of all states equal to zero at k=0, for which iteration k will $V_k(A) = V^*(A)$?

$k =$

(b) Consider a new Pacman level that begins with cherries in locations D and F. Landing on a grid position with cherries is worth 5 points and then the cherries at that position disappear. There is still one dot, worth 1 point. The game still only ends when the dot is eaten.

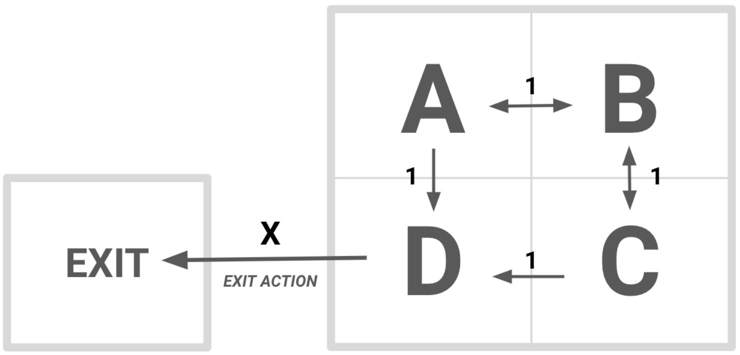


(i) With no discount ($\gamma = 1$) and a living reward of -1, what is the optimal policy for the states in this level's state space?

(ii) With no discount ($\gamma = 1$), what is the range of living reward values such that Pacman eats exactly one cherry when starting at position A?

Q2. Strange MDPs

In this MDP, the available actions at **state A, B, C** are *LEFT, RIGHT, UP,* and *DOWN* unless there is a wall in that direction. The only action at **state D** is the *EXIT ACTION* and gives the agent a **reward of** x. The **reward for non-exit actions is always 1**.



(a) Let all actions be deterministic. Assume $\gamma = \frac{1}{2}$. Express the following in terms of x.

$V^*(D) =$ $V^*(C) =$

$V^*(A) =$ $V^*(B) =$

(b) Let any non-exit action be successful with probability $= \frac{1}{2}$. Otherwise, the agent stays in the same state with reward $= 0$. The *EXIT ACTION* from the **state D** is still deterministic and will always succeed. Assume that $\gamma = \frac{1}{2}$.

For which value of x does $Q^*(A, DOWN) = Q^*(A, RIGHT)$? Box your answer and justify/show your work.

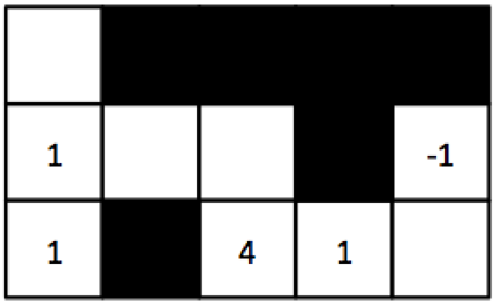
(c) We now add one more layer of complexity. Turns out that the reward function is not guaranteed to give a particular reward when the agent takes an action. Every time an agent transitions from one state to another, once the agent reaches the new state s', a fair 6-sided dice is rolled. If the dices lands with value x, the agent receives the reward $R(s, a, s') + x$. The sides of dice have value $1, 2, 3, 4, 5$ and 6.

Write down the new bellman update equation for $V_{k+1}(s)$ in terms of $T(s, a, s')$, $R(s, a, s')$, $V_k(s')$, and γ.

Q3. MDPs: Reward Shaping

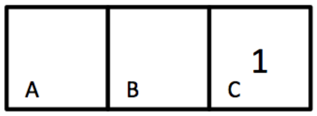
PacBot is in a Gridworld-like environment E. It moves deterministically Up, Down, Right, or Left, or at any time it can exit to a terminal state (where it remains). If PacBot is on a square with a number written on it, it receives a reward of that size **on Exiting**, and it receives a reward of 0 for Exiting on a blank square. Note that when it is on any of the squares (including numbered squares), it can either move Up, Down, Right, Left or Exit. However, it only receives a non-zero reward when it Exits on a numbered square.

(a) Draw an arrow in **each** square (including numbered squares) in the following board to indicate the optimal policy PacBot will calculate with the discount factor $\gamma = 0.5$. (For example, if PacBot would move Down from the square in the middle, draw a down arrow in that square.) If PacBot's policy would be to exit from a particular square, draw an X in that square.



In order to speed up computation, Pacbot computes its optimal policy in a new environment E' with a different reward function $R'(s, a, s')$. If $R(s, a, s')$ is the reward function in the original environment E, then $R'(s, a, s') = R(s, a, s') + F(s, a, s')$ is the reward function in the new environment E', where $F(s, a, s') \in \mathbb{R}$ is an added "artificial" reward. If the artificial rewards are defined carefully, PacBot's policy will converge in fewer iterations in this new environment E'.

(b) To decouple from the previous question's board configuration, let us consider that Pacbot is operating in the world shown below. Pacbot uses a function F defined so that $F(s, a, s') = 10$ if s' is closer to C relative to s, and $F(s, a, s') = 0$ otherwise (consider C to be closer to C than B or A). Let us also assume that the action space is now restricted to be between Right, Left, and Exit only.



In the diagram above, indicate by drawing an arrow or an X in each square, as in part (a), the optimal policy that PacBot will compute in the new environment E' using $\gamma = 0.5$ and the modified reward function $R'(s, a, s')$.

(c) PacBot's utility comes from the discounted sum of rewards **in the original environment**. What is PacBot's expected utility of following the policy computed above, starting in state A if $\gamma = 0.5$?

(d) Find a non-zero value for x in the table showing $F(s, a, s')$ drawn below, such that PacBot is guaranteed to compute an optimal policy that maximizes its expected true utility for **any** discount factor $\gamma \in [0, 1)$.

	Value
$F(A, \text{Right}, B)$	10
$F(B, \text{Left}, A)$	x
$F(B, \text{Right}, C)$	10
$F(C, \text{Left}, B)$	x

$x =$ ________________________________