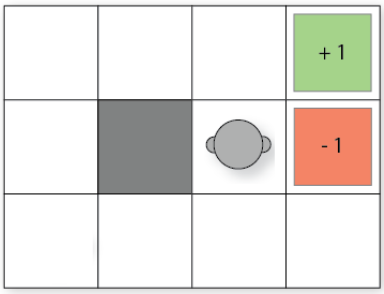
# Problem Set VIII: Monte-Carlo Tree Search & n-step TD

**Aim** The purpose of this workshop is to help you get a better understanding of Monte-Carlo Tree Search for solving MDPs in an online manner. In addition, we will also cover the left question in last workshop (Reinforcement Learning).

## Tasks

In this workshop, you will consider the example from the lectures of the agent that moves in a 2D grid world. Remember that if the agent tries to move in a particular direction, there is an 80% of success, and a 10% chance of it going to the left or right.




1. The agent is at cell (2,1), in which 2 is the x-coordinate and 1 the y-coordinate (both start from 0). It samples the following 5 iterations of MCTS, where E (East) goes right, W (West) goes left, N (North) goes up, and S (South) goes down:

| **Iter** | **Trace** | |
|---|---|---|
| 1 | $N$ | $simulate(succ) = -1$ |
| 2 | $N \xrightarrow{succ} E$ | $simulate(slip(S)) = -1$ |
| 3 | $E$ | $simulate(succ) = -1$ |
| 4 | $W$ | $simulate(succ) = 1$ |
| 5 | $N \xrightarrow{succ} S$ | $simulate(slip(W)) = 1$ |

   Here, $N \xrightarrow{succ} E$ means that we select $N$, then select that the outcome of $N$ was successful, and then select $E$. The notation $N \xrightarrow{slip(E)} S$ means that we select $N$ and the agent went East instead; then select South.

   The notation $simulate(X) = r$ means having just expanded a node, simulate from the outcome $X$, which will return reward $r$.

   Draw the MCTS tree for this, assuming $\lambda = 1.0$. Label the lines on the tree with the actions & outcomes and label the nodes with the value $V(s)$ for each state and $N$ (the number of times this has been visited).

2. Based on your tree, calculate the action with the highest expected return.

3. Based on your tree, which of action, North, South, East, or West, would be more likely to be chosen if we use UCT to probabilistically select the next action? Show your working. Assume that $C_p = \frac{1}{2}$.

4. The following question uses the same problem as the one in Reinforcement Learning problem set. Recall the following description from Problem Set VII:

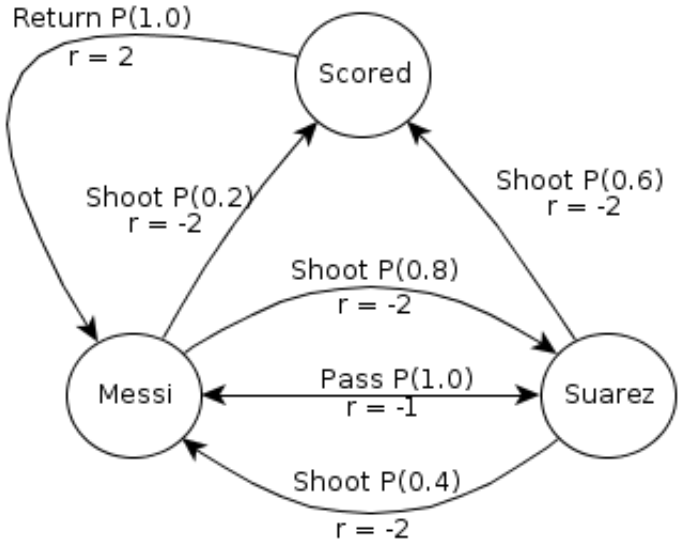
   Consider two football-playing robots: Messi and Suarez.

   They play a simple two-player cooperate game of football, and you need to write a controller for them. Each player can pass the ball or can shoot at goal.

   The football game can be modelled as a discounted-reward MDP with three states: *Messi*, *Suarez* (denoting who has the ball), and *Scored* (denoting that a goal has been scored); and the following action descriptions:

   - If Messi shoots, he has 0.2 chance of scoring a goal and a 0.8 chance of the ball going to Suarez. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
   - If Suarez shoots, he has 0.6 chance of scoring a goal and a 0.4 chance of the ball going to Messi. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
   - If either player passes, the ball will reach its intended target with a probability of 1.0. Passing the ball incurs a cost 1 (or a reward of -1).
   - If a goal is scored, the only action is to return the ball to Messi, which has a probability of 1.0 and has a reward of 2. Thus the reward for scoring is modelled by giving a reward of 2 when *leaving* the goal state.

   The following diagram shows the transition probabilities and rewards:




   Given the following trace from a historical game feed from last season:

   " Suarez passes the ball to Messi, Messi dribbles around all of his opponents, shoots and scores yet another goal! Barcelona F.C 10 - 0 Real Madrid! The ball is returned to Messi for kickoff. After he passes the ball to Suarez, the referee blew the final whistle. End of the game, the ball is taken by Messi to remember the match forever."

   Show the 3-step SARSA update for the above feed. Do you think the 1-step update is more accurate or the 3-step update? Does it indicate more steps is always better? Explain why.