# Lab 12 - Markov Decision Process (MDP)

## I. Introduce

**Markov Decision Process (MDP)** consists of a tuple of 5 elements $< \mathcal{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma >$

- $\mathcal{S}$: set of states $\rightarrow$ At each time step, state of the environment is an element $s \in \mathcal{S}$.
- $\mathscr{A}$: set of actions $\rightarrow$ At each time step, agent takes an action $a \in \mathscr{A}$.
- $\mathscr{P}$: state transition model (matrix)

  $$\mathscr{P}^a_{x \rightarrow x'} = P[S_{t+1}|S_t = s, A_t = a]$$

  Probability of transition to next state $x'$ after taking action $a$ in current state $x$.
- $\mathscr{R}$: reward model (matrix)

  $$\mathscr{R}^a_x = E[R_{t+1}|S_t = s, A_t = a]$$

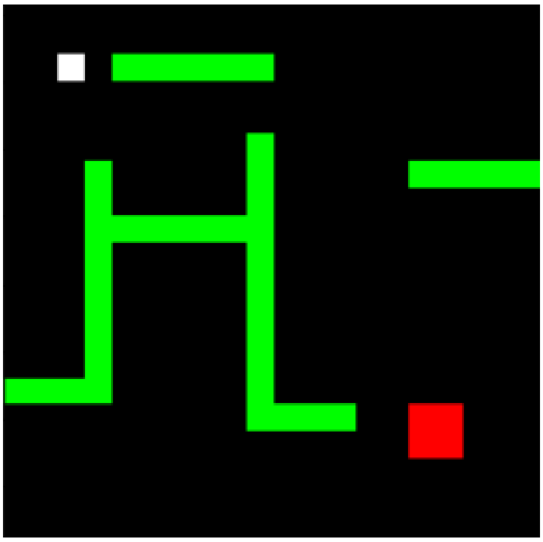
  Reward obtained after taking action $a$ in current state $x$. (to be more general, $\mathscr{R}^a_{x \rightarrow x'} = E[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']$ )
- $\gamma$: discount factor $\rightarrow$ Control the importance of future rewards.

In this lab, we are about to find a shortest path with collision avoidance using MDP. We will model the problem as a MDP problem and use **value iteration** or **policy iteration** algorithm to solve it.

## II. Task Description

### (1) Environment Data

- Environment: `map_matrix.npy` has environment data. You need to use `numpy` to load it.
  - White block: an agent, for example, a robot
  - Red block: destination
  - Green block: obstacle



- $\mathscr{R}$ Reward: reward is implemented in code and it only concerns the next state:
  - wall: $-1$

    - destination: 0
    - else: $-0.1$
- $\mathscr{P}$ State transformation: next state is deterministic when taking an action under a certain state.
- $\pi$ Initial policy: each direction (up, right, bottom, left) has equal probability.

## (2) Display

We have several methods for you to display policy, state value and path on map

```
def display_policy()
def display_v()
def display_path()
```

# III. Lab Requirement

Please finish the **Exercise** and answer **Questions**.

## (1) Exercise

You should implement value or policy iteration to solve this problem.

**Metrics**

1. Arrive destination successfully with collision avoidance.
2. Take the least number of steps to reach destination.

**Submit**

1. File: your code and images of policy, state value and path you take
2. Report: include your results and brief comments

## (2) Questions

1. What are the relationships between MDP and RL?
2. What are the requirements for the dynamic programming based MDP; when does it perform poorly?
3. What makes the dynamic programming based MDP a good candidate for the planning/decision problem, if you have enough knowledge about the problem?