# CS 7.603
# Reinforcement Learning

**Tejas Bodas**

Assistant Professor, IIIT Hyderabad

# RECAP

- Markov chains, $P, \bar{\mu}$
- Finite dimensional distributions
- Chapman-Kolmogorov equations
- Irreducible chains, Transient/Recurrent states
- Limiting and stationary distribution distribution

# Markov Chain as recursions

## Theorem

*Consider the recursion* $X_{n+1} = f(X_n, U_n), n \geq 0$ *where* $f : \mathcal{S} \times [0,1] \to \mathcal{S}$ *and* $\{U_n, n \geq 0\}$ *is an i.i.d sequence of random variables. Then the process* $\{X_n, n \geq 0\}$ *defines a Markov chain with tpm* $P_{ij} = P(f(i, U) = j)$*. Conversely, every Markov chain can be represented by such a recursion for some* $f$ *and* $\{U_n, n \geq 0\}$*.*

- **Proof**: The forward part is trivial.
- Converse follows from inverse transform method (simulation).
- Given a Markov chain with TPM $P$, we want to identify $f(.,.)$.
- Given $X_n = i$, how would you sample $X_{n+1}$?
- Let $F_i(x)$ denote the CDF of the $i^{th}$ row of $P$.
- $X_{n+1}$ can be seen as a sample from $F_i(\cdot)$ when $X_n = i$.

# Markov Chain as recursions

### Theorem

*Consider the recursion $X_{n+1} = f(X_n, U_n), n \geq 0$ where $f : \mathcal{S} \times [0,1] \to \mathcal{S}$ and $\{U_n, n \geq 0\}$ is an i.i.d sequence of random variables. Then the process $\{X_n, n \geq 0\}$ defines a Markov chain with tpm $P_{ij} = P(f(i,U) = j)$. Conversely, every Markov chain can be represented by such a recursion for some $f$ and $\{U_n, n \geq 0\}$.*

- To generate $X_{n+1}$, draw $U_n$ from $U[0,1]$ and set $X_{n+1} = F_i^{-1}(U_n)$.
- When $F_i$ is discrete we have $F_i^{-1}(y) := min\{x : F_i(x) \geq y\}$
- Clearly, $f(i, U_n) := F_i^{-1}(U_n)$
- $X_{n+1} = f(X_n, U_n) = F_{X_n}^{-1}(U_n)$. $\square$
- Here $U_n$ is $U[0,1]$. Can $U_n \sim G$ where $G$ is arbitrary? (HW)

# Markov Reward Process

- Consider a Markov Chain $\{X_n, n \geq 0\}$ on $\mathcal{X}$ with $|\mathcal{X}| = M$.
- Suppose reward $r(X_t)$ is obtained when in state $X_t$ at time $t$.
- Let $\beta \in (0,1)$ be the discount factor for the rewards.
- We want to characterize the cumulative expected discounted reward $V(x)$ conditioned on starting in state $x$.

$$V(x) = \mathbb{E}_x \left[ \sum_{t=0}^{\infty} \beta^t r(X_t) \right]$$

- Here $\mathbb{E}_x$ denotes conditional expectation on starting in $x$.
- Note that $V : \mathcal{X} \to \mathbb{R}$. Since $|\mathcal{X}| = M$, we have $V \in \mathcal{R}^{M \times 1}$

# Markov Reward Process

## Lemma

*$V(x)$ is a unique solution to $V(x) = \beta(PV)(x) + r(x)$ for $x \in \mathcal{X}$ where $(PV)(x)$ denotes the $x^{th}$ row in the $P \times V$ vector.*

**Proof**[1]
$$
\begin{aligned}
V(x) &= \mathbb{E}_x\left[\sum_{t=0}^{\infty} \beta^t r(X_t)\right] \\
&= r(x) + \beta\mathbb{E}_x\left[\sum_{t=1}^{\infty} \beta^{t-1} r(X_t)\right] \\
&= r(x) + \beta\mathbb{E}_x\mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} r(X_t)\Big| X_1\right] \quad \text{Tower rule} \\
&= r(x) + \beta\mathbb{E}_x\mathbb{E}_{X_1}\left[\sum_{t=1}^{\infty} \beta^{t-1} r(X_t)\right] \\
&= r(x) + \beta\mathbb{E}_x V(X_1) \text{ (Def of } V(.)) \\
&= r(x) + \beta\sum_{x_1} P_{xx_1} V(x_1) \\
&= r(x) + \beta(PV)(x). \quad \square
\end{aligned}
$$

[1] Refer Neil Walton's notes for proof of uniqueness

# Markov Reward Process

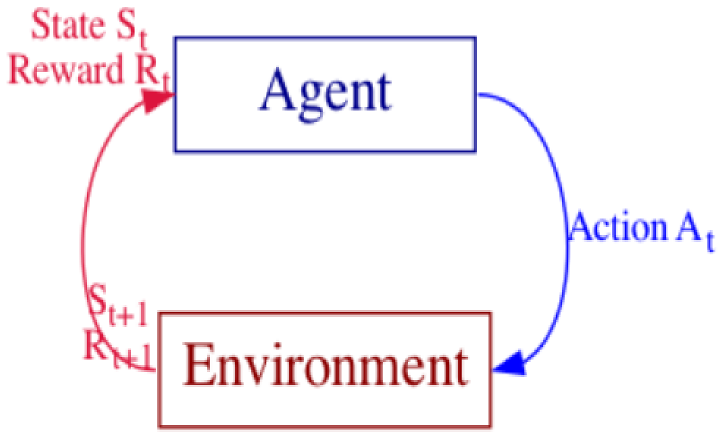
**Lemma**

*$V(x)$ is a unique solution to $V(x) = \beta(PV)(x) + r(x)$ for $x \in \mathcal{X}$ where $(PV)(x)$ denotes the $x^{th}$ row in the $P \times V$ vector.*

- In vector notation this is written as $V = r + \beta PV$.
- This implies $V = [I - \beta P]^{-1} r$
- This is $O(M^3)$ operation.
- What if the state space is discrete but infinite dimensional?
- What if the state space is continuous? HW

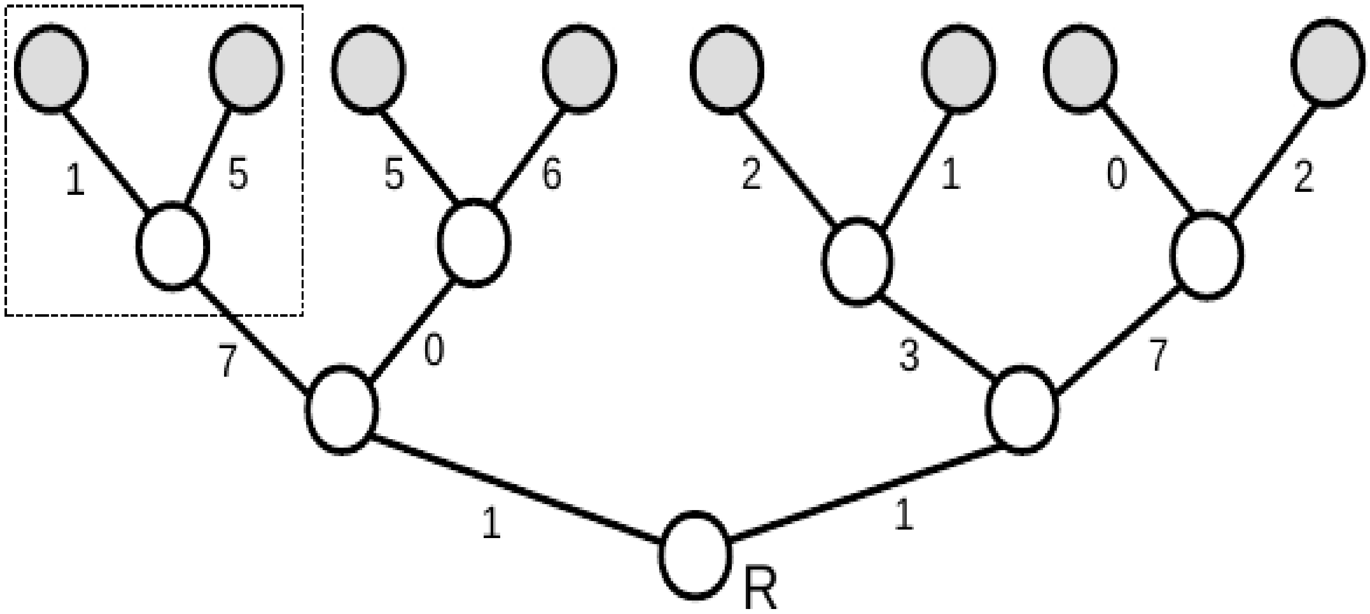
# Deterministic Dynamic Programming

# Basic Idea




- In a deterministic dynamic program, the environment is governed by what is called a deterministic plant equation.
- You can interact with the environment by taking actions.
- Actions lead to rewards.
- We want to find the sequence of actions that maximize the sum of rewards collected.
- Lets formalize this with an example.

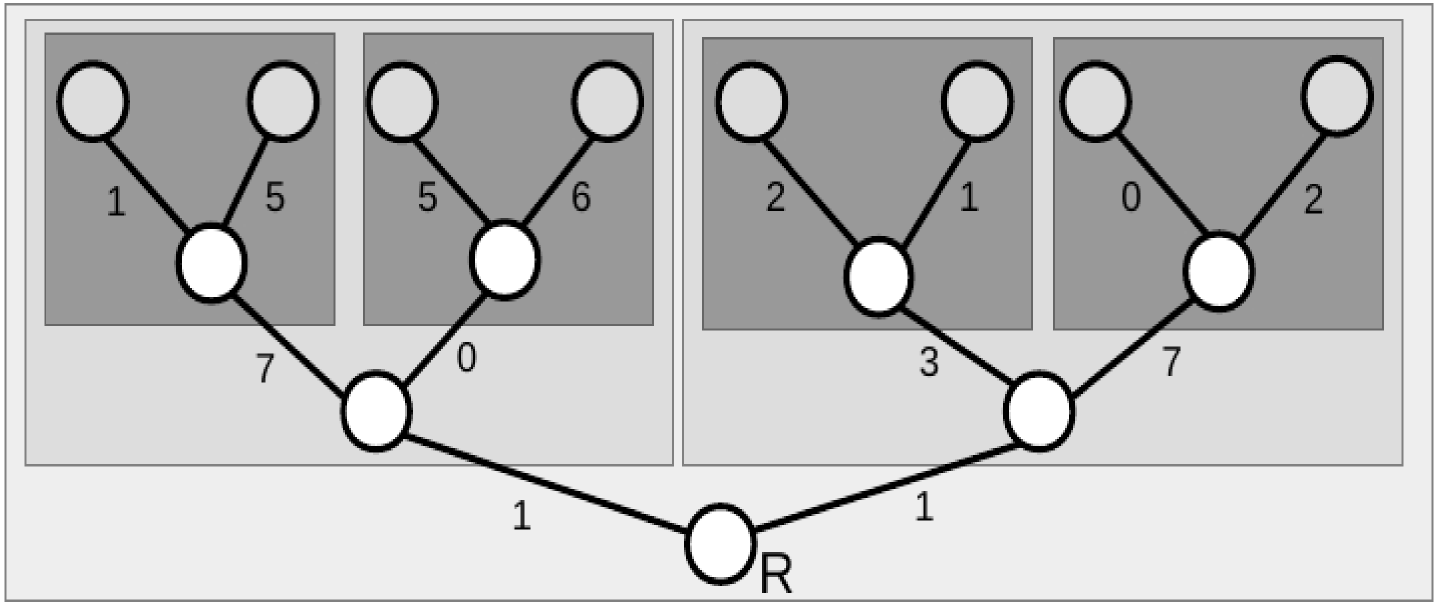
# A shortest path example




- We want to find the least costly path from $R$ to any leaf. [2]
- Naive way is to enumerate all paths and then choose.
- We want to take advantage of the recursive structure and possibly breakdown to smaller and smaller problems.

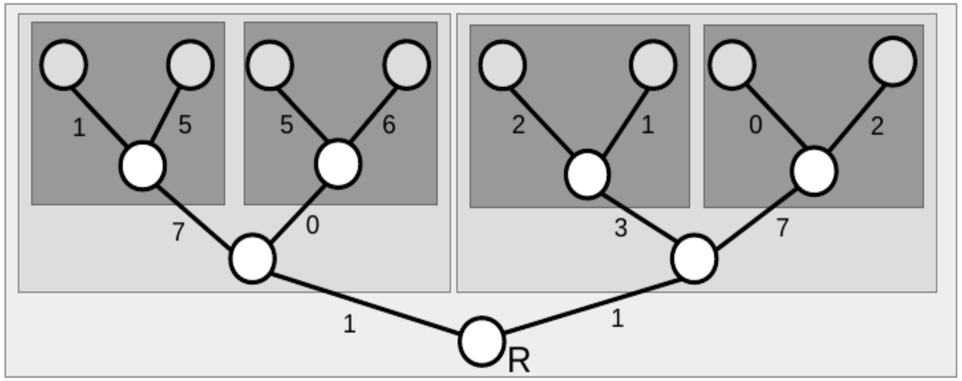
[2] Example from Neil Walton's notes on Stochastic Control

# An Example



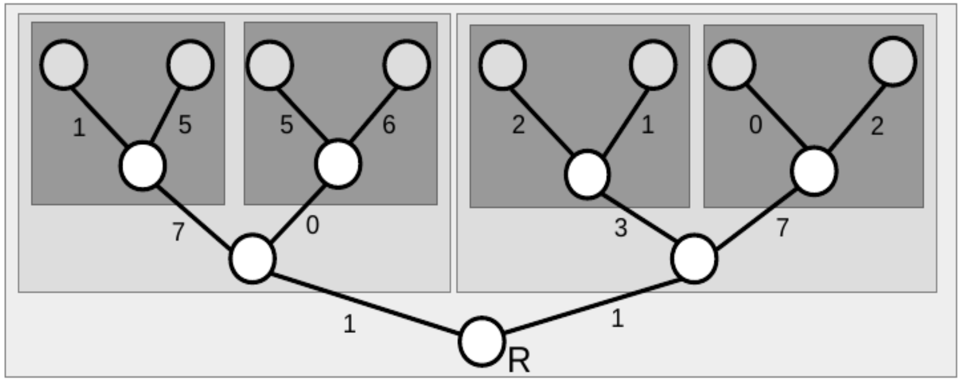

- The original problem can be broken down into sequence of simpler problems (shaded boxes)
- At any (non-leaf) node, we have two actions. $\mathcal{A} = \{lhs, rhs\}$
- Let $\mathcal{S}$ denote the state space, in this case the set of 15 nodes.

# An Example



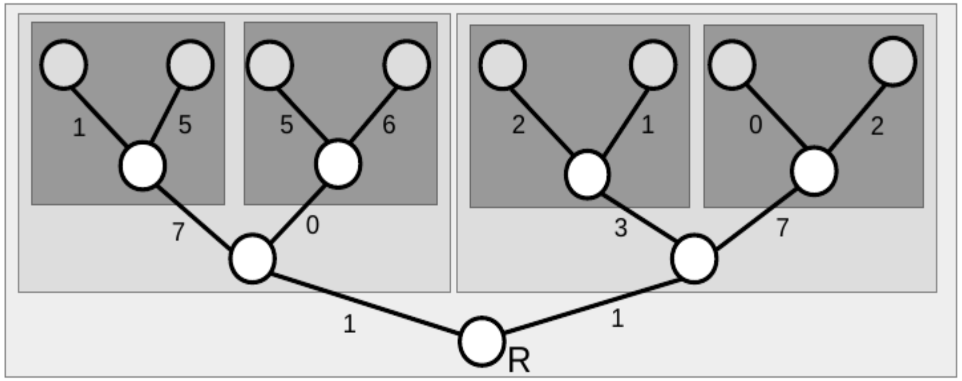

- For any node $s \in \mathcal{S}$, let $V(s)$ denote the least cost path to any leaf, starting from node $s$.
- Let $f(s, a)$ denote the next state reached from node $s \in \mathcal{S}$ after taking action $a \in \mathcal{A}$.
- $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ (plant equation)
- Let $c(s, a)$ denote the edge cost incurred by taking action $a$ from node $s$.
- Objective: Determine $V(R)$ and also the strategy/policy to follow, that will help you realize it.

# An Example




- Can you write an expression for $V(R)$ in terms of $V(s')$ where $s' = f(R, a)$ for $a \in \mathcal{A}$ ?
- $V(R) = \min_{a \in \{lhs, rhs\}} \{c(R, a) + V(s')\}$ where $s' = f(R, a)$.
- Bellman-type Equation: Solve recursively backwards
- But how do you get the policy/route with the least cost?
- Let $\pi : \mathcal{S} \to \mathcal{A}$ denote a policy that maps states to actions.

# An Example




- Let $\pi : \mathcal{S} \to \mathcal{A}$ denote a policy that maps states to actions.
- There are many policies. How do we find the best policy $\pi^*$ corresponding to $V(R)$?
- Recall $V(R) = \min_{a \in \{lhs, rhs\}} \{c(R, a) + V(s')\}$ where $s' = f(R, a)$.
- Infact, $\pi^*(R) = \underset{a \in \{lhs, rhs\}}{argmin} \{c(R, a) + V(s')\}$.
- Doing this at every node will give you the optimal policy for that node.

# Dynamic programming: State, Action, Reward, State

- Lets consider discrete set of times $t = 0, 1, \ldots, T$
- Let $\mathcal{S}$ denote the state space and $\mathcal{A}$ denote the action space.
- In most cases, these will be countable spaces unless specified.
- $s_t \in \mathcal{S}$ denotes state of the system at time $t$.
- $s_0$ denotes starting state.
- $r(s_t, a_t)$ for reward at time $t$ when in state $s_t$ and action $a_t$.
- $r_T(s_T)$ denotes the reward for terminating in $s_T$ at time $T$.
- $c(s, a)$ for cost formulation
- $s_{t+1} = f(s_t, a_t)$ is the deterministic plant equation.
- These transitions are Markovian in an MDP.

# DP: policy, cumulative reward, value function

- A policy $\pi = (\pi_t : t = 0, 1, \ldots, T-1)$ specifies action $\pi_t \in \mathcal{A}$ to be taken at time $t$.
- Sequence of states following this policy: $s_{t+1} = f(s_t, \pi_t)$.
- How good is this policy? Measured using cumulative reward
- $V^\pi(s_0) := r(s_0, \pi_0) + r(s_1, \pi_1) + \ldots + r_T(s_T)$.
- How do we get the best policy $\pi^*$ from the set of policies $\Pi$ ?
- Optimization problem: $\max_{\pi \in \Pi} V^\pi(s_0)$.

# Definition of a Dynamic Program

Given initial state $s_0$, a dynamic program is the following optimization problem:

$$\begin{aligned} V(s_0) \quad &:= \quad \max_{\pi \in \Pi} V^{\pi}(s_0) \\ \text{such that} \quad & \quad s_{t+1} = f(s_t, \pi_t) \text{ for } t = 0, \ldots, T-1. \end{aligned}$$

- How do we obtain an expression for $V(s_0)$
- Except for toy problems, it is hard to get an explicit expression for $V(s_0)$ in terms of the parameters of the problem.
- At best, we can expect an implicit or recursive equation for $V(s_0)$ in terms of smaller and smaller sub-problems.
- These are known as Bellman equations.