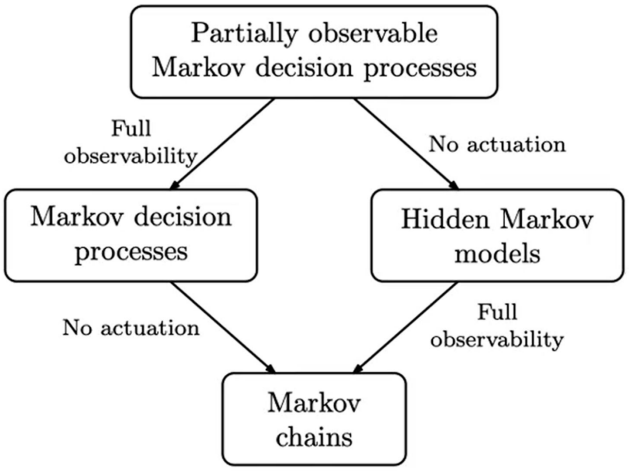
➔ Markov Chain
- $\mathcal{X}$ - set of possible states
- P - transition probability matrix

➔ Hidden Markov Model (HMM)
- $\mathcal{X}$ - set of possible states
- P - transition probability matrix
- $\mathcal{Z}$ - set of possible observations
- O - observation probability matrix

➔ Markov Decision Process (MDP)
- $\mathcal{X}$ - set of possible states
- $\mathcal{A}$ - set of possible actions
- $[P_a]$ - transition probability matrices for $a \in \mathcal{A}$
- C - immediate cost function ($\mathcal{X} \times \mathcal{A}$)

➔ Markov Decision Problem
- $\mathcal{X}$ - set of possible states
- $\mathcal{A}$ - set of possible actions
- $[P_a]$ - transition probability matrices for $a \in \mathcal{A}$
- C - immediate cost function
- Y - discount factor

➔ Partially Observable MDP (POMDP)
- $\mathcal{X}$ - set of possible states
- $\mathcal{A}$ - set of possible actions
- $\mathcal{Z}$ - set of possible observations
- $[P_a]$ - transition probability matrices for $a \in \mathcal{A}$
- $[O_a]$ - observation probability matrices for $a \in \mathcal{A}$
- C - immediate cost function



➔ Belief MDP: For a given POMDP, the belief MDP is an equivalent MDP model whose state space is the set of possible distributions over the state space of the original POMDP (also known as the belief space). Belief MDPs are thus a “bridge” between the theory of MDPs and that of POMDPs, allowing us to use MDP solution methods such as value and policy iteration to solve POMDPs.

➔ P ,Transition probability matrix: $\mathcal{X}$x $\mathcal{X}$, where the element (x1,x2) represents the probability of the agent going from state x1 to state x2. When there are multiple actions then we have P_a, which represents the probability of the agent going from state x1 to state x2 if it selected action a.

➔ Π (ou π), Policy matrix: $\mathcal{X}$x $\mathcal{A}$, where a row represents the probability of choosing action a in state x. It serves to determine the action to choose.

➔ c, Cost matrix: $\mathcal{X}$x $\mathcal{A}$, where the element (x,a) represents the cost of choosing action a when on state x. In this subject all costs are in interval [0,1].

➔ Q^{Π}, Q function: $\mathcal{X}$x $\mathcal{A}$, where the element (x,a) represents the average cost-to-go for policy Π, given initial state x and action a.

➔ O, Observation matrix: $\mathcal{X}$x $\mathcal{Z}$, where the element (x,o) represents the probability of witnessing Observation o, when at state x. When there are multiple actions then we have O_a, which represents the probability of the agent witnessing Observation o, when at state x if it selected action a.

➔ J^{Π}, Cost-to-go function: $\mathcal{X}$x 1, where the element (x) represents the total cost of reaching the goal starting from state x, while following the policy Π.

➔ b_t, Belief: $\mathcal{X}$x 1, which represents the stationary distribution over χ at the instant t. As it is a distribution all of it’s values must add up to 1 and are all between [0,1]

➔ **Markov Property**: The state at instant *t* is enough to predict the state at *t* + 1

- i.e. depends only on the last state

➔ **Irreducibility**: A chain is **irreducible** if any state y can be reached from any state x

- State y can be reached from a state x if $P_t(y \mid x) > 0$
- We can split the state space X in **communicating classes** (sets of states that are mutually reachable) - basically componentes fortemente ligadas

➔ **Aperiodicity**

- The period of a state x is the **greatest common divisor** of all time steps in which x can be visited if the chain departs from x
- A state is aperiodic if its $p_x = 1$
- A chain is **aperiodic** if all states are aperiodic, and **periodic** otherwise
- When a chain is **irreducible**, all states have the same period
- If all nodes have self loops, the chain is aperiodic

➔ **Stationary** distribution

- Let μ be a **distribution** over X
- μ is a row vector (1 x *m*) such that $\mu = [\mu(x_1)\ \mu(x_2)\ ...\ \mu(x_{|X|})]$
- Component μ(x) is the probability of x according to μ
- If the state at time t follows the distribution μ and μ is stationary, then the state at time t+1 also follows the distribution μ, i.e. **if μ is stationary, μ * P = μ**
- The stationary distribution corresponds to the stable behavior of the chain

➔ An **irreducible** and **aperiodic** Markov Chain possesses a stationary distribution

- **Positive chain** - if x_t is distributed according to μ, then so is x_{t+1}

➔ For an irreducible and aperiodic Markov chain with a stationary distribution will eventually converge to this distribution independently of the initial distribution

- No matter how the initial state is selected, if we let the chain run for a sufficient amount of time, or make predictions far enough in the future, then those predictions will eventually converge to the stationary distribution
- **Ergodic chain** - eventually reaches the stationary distribution

➔ Filtering: given a sequence of observations, estimate the **final state**

- Forward algorithm

➔ Smoothing: given a sequence of observations, estimate the **sequence of states**

- We want to estimate **the most likely sequence**, i.e.
  - $x^*_{0:T} = \text{argmax}\ P\mu_0[x_{0:T} = x_{0:T} \mid z_{0:T} = z_{0:T}]$
- Maximizing forward mapping
- Viterbi algorithm

➔ Marginal smoothing: given a sequence of observations, estimate **some state** in the middle

- Forward-backward algorithm

➔ Prediction: given a sequence of observations, predict **future states**

- Compute $\mu_{T|0:T}$ using the forward algorithm
- Use the Markov property
  - $\mu_{T+1|0:T} = \mu_{T|0:T} * P$

➔ Forward algorithm

- Forward mapping: maps states to real numbers
  - a_t: X ➔ R
  - $a_t(x) = P\mu_0[x_t = x, z_{o:t} = z_{o:t}]$
  - Each component x represents the joint probability of being in state x at timestamp t given the observations from timestamp 0 to timestamp t
- Algorithm:
  - Multiply initial distribution by diag($O(z_0|:)$)
  - At each time step:
    - Multiply current distribution by P
    - Multiply by diag($O(z_t|:)$)
  - Normalize

Note: If there are no observations, all you have to do is multiply the by P_A , being A the action that was chosen.

Marginal Smoothing: Given a sequence of observations, estimate some state in the middle.
➔ Forward-backward algorithm

- Backward mapping: maps states to real numbers
  - b_t: X ➔ R
  - $b_t(x) = P\mu_0[z_{t+1:T} = z_{t+1:T} \mid x_t = x]$
- Algorithm:
  - Multiply initial distribution by $O(z_0|:)$, $b_T = 1$
  - for t = 0, …, t do (**forward update**)
    - Current distribution * P * diag($O(z_t|:)$)
  - for t = T-1, …, t do (**backward update**)
    - $b_t = P * diag(O(z_t|:)) * b_{t+1}$
  - Combine and normalize ➔ a_t x b_t / (a_tT * b_t)

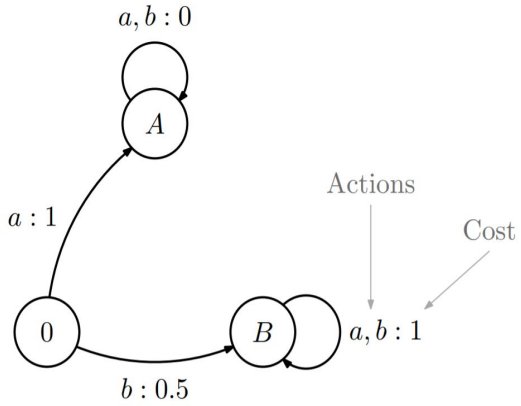
Joint Smoothing : Given a sequence of observations and/or actions, estimate the whole sequence of states (most likely sequence).
➔ Viterbi algorithm

- $m_0 = diag(O_{:,z0}) * \mu_0T$ // just μ_0T if there is no initial observation, i.e at t = 0
- for t = 1, …, T do
  - $m_t = diag(O_{:,zt}) * max\_row\_vector(PT * diag(m_{t-1}))$ // dá um row vector
  - i_t = indices dos max_row_vector() escolhidos // row vector, indices de 1-N
- x^*_t = indice do max_row() de m_t // inteiro, um indice de 1-N
- for t = T-1, …, 0 do
  - $x^*_t = i_{t+1}(x^*_{t+1})$ // backtrack com indices, DM me se não perceberem
- return $x^*_{0:t}$
- Example iteration $\boldsymbol{m}_1 = \mathrm{diag}(\mathbf{O}_{:,u}) \max\{\mathbf{P}^\top \mathrm{diag}(m_0)\}$

Decision Trees: The objective is to minimize the cost

- c = - u (where c is cost and u is utility)
- So the objective is to maximize the utility

Both policy iteration and value iteration are algorithms used to find the optimal policy for a certain MDP.
Now to explain both policy iteration and value iteration and I'm going to use the following example:



- States:
  - $\mathcal{X} = \{0, A, B\}$
- Actions:
  - $\mathcal{A} = \{a, b\}$

- Transition probabilities:

$$\mathbf{P}_a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_b = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Cost:

$$\mathbf{C} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Value Iteration

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$;

1: Initialize $k = 0$, $J^{(0)} \equiv 0$
2: **repeat**
3: $\quad J^{(k+1)} \leftarrow \mathsf{T} J^{(k)}$
4: $\quad k \leftarrow k + 1$
5: **until** $\left\| J^{(k-1)} - J^{(k)} \right\|_\infty < \varepsilon$.
6: Compute π^* using (†)
7: **return** π^*

In step 1, we initialize k = 0 and $J^{(0)}$ as Vector with dimensions X x 1
In step 2, we will now start a loop where that will stop when the distance between $J^{(K-1)}$ and $J^{(K)}$ is lesser than ε, which is our tolerance.
In step 3, we will apply the operator T to $J^{(K)}$. The operator T is actually just for each action a in de MDP's Actions do the following formula:

$$\boldsymbol{J}^{(1)} = \min_a \left[\boldsymbol{c}_a + \gamma \mathbf{P}_a \boldsymbol{J}^{(0)} \right]$$

In our example, we will do it for action a and for action b, since we only have 2 actions.

- For action *a*:

$$\begin{bmatrix}1\\0\\1\end{bmatrix} + 0.99 \begin{bmatrix}0 & 1 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\1\end{bmatrix}$$

- For action *b*:

$$\begin{bmatrix}0.5\\0\\1\end{bmatrix} + 0.99 \begin{bmatrix}0 & 0 & 1\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0.5\\0\\1\end{bmatrix}$$

And then we will see which one is the minimum in each line between the two, which will be the rows of our $J^{(1)}$

$$\min\left(\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0.5\\0\\1\end{bmatrix} \right) = \begin{bmatrix}0.5\\0\\1\end{bmatrix}$$

In step 4, we increment variable k, with k = k+1.
In step 5, we verify the stop condition.
In step 6, we will compute the policy $\pi^{(*)}$ to do this we will look at the example:

$$\min\left(\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0.5\\0\\1\end{bmatrix} \right) = \begin{bmatrix}0.5\\0\\1\end{bmatrix}$$

For each row of this matrix let's see which column has the lowest value.

In row 0, associated with state 0, column 1 has the lowest value, therefore in our policy $\pi^{(*)}[0][1]=1$

In row 1, due to the fact that both have the same value, both have the lowest value, therefore in our policy, $\pi^{(*)}[1][0]=1$ and $\pi^{(*)}[1][1]=1$. However all rows of a policy must add up to 1, so we must normalize this, by dividing it by the total sum of the row's values, resulting in $\pi^{(*)}[1][0]=0.5$ and $\pi^{(*)}[1][1]=0.5$.

In row 2, due to the fact that both have the same value, both have the lowest value, therefore in our policy, $\pi^{(*)}[2][0]=1$ and $\pi^{(*)}[2][1]=1$. However all rows of a policy must add up to 1, so we must normalize this, by dividing it by the total sum of the row's values, resulting in $\pi^{(*)}[2][0]=0.5$ and $\pi^{(*)}[2][1]=0.5$

This results in $\pi^{(*)} = \begin{bmatrix} 0 & 1 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$

In step 7 we return $\pi^{(*)}$

Policy Iteration

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$

1: Initialize $k = 0$, $\pi^{(0)}$ randomly
2: **repeat**
3: $\quad \boldsymbol{J} \leftarrow (\boldsymbol{I} - \gamma \boldsymbol{P}_{\pi^{(k)}})^{-1} \boldsymbol{c}_{\pi^{(k)}}$ — Policy evaluation
4: $\quad \pi^{(k+1)} \leftarrow \pi_g^J$ — Policy improvement
5: $\quad k \leftarrow k + 1$
6: **until** $\pi^{(k-1)} = \pi^{(k)}$
7: **return** $\pi^{(k)}$

In step 1, we initialize the variable k as 0 and we initialize the policy $\pi^{(0)}$ randomly. Remember that $\pi^{(0)}$ is a matrix with dimension X ✕ A, and that each row must add up to 1.

In step 2, we will now start a loop that will last until $\pi^{(k-1)}=\pi^{(k)}$. This means that our policy can no longer be improved, the loop will stop.

In step 3, we will calculate the J value for this iteration.

I is the identity matrix, with dimension X ✕ X, where X is the number of states in the MDP's state space, $\mathcal{X}$. γ is the MDP's discount rate. But first, we need to calculate $P\pi^{(k)}$. To calculate $P\pi^{(k)}$ we need to for each action A, **do the product of the diagonal matrix of $\pi^{(k)}$'s A column with the Probability Matrix associated with action A and then sum them all.** Second we need to calculate $c\pi^{(k)}$, to do that we need to for each action A, **do the product of the diagonal matrix of $\pi^{(k)}$'s A column with the Cost Matrix's A column and then sum them all.** After we have all necessary components we can now calculate J, using the formula on the picture's step 3.

To make step 3 clearer I'll use the example:

In our example, our $\pi^{(0)}$ is the uniform policy, which means $\pi^{(0)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$

Now let's calculate $P\pi^{(k)}$, the formula for $P\pi$ is $diag(\pi^{(k)}[:,a]).P_a$,where is $diag(\pi^{(k)}[:,a])$ is the matrix where the diagonal is the a column of $\pi^{(k)}$ and we do this for each action.

Let's show it on the example:

$diag(\pi^{(0)}[:,a]) = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$ and $diag(\pi^{(0)}[:,b]) = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$

Now let's multiply each by their respective actions's probability transition matrix:

For action a: $diag(\pi^{(0)}[:,a]).P_a = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$ For action b: $diag(\pi^{(0)}[:,b]).P_b = \begin{bmatrix} 0 & 0 & 0.5 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$

Now if we sum both we get $P\pi = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Now let's calculate $c\pi^{(k)}$, the formula for $c\pi$ is $diag(\pi^{(k)}[:,a]).c[:,a]$, where is $diag(\pi^{(k)}[:,a])$ is the matrix where the diagonal is the a column of $\pi^{(k)}$ and $c[:,a]$ is the a column of c.

For action a: $diag(\pi^{(0)}[:,a]).c[:,a] = \begin{bmatrix} 0.5 \\ 0 \\ 0.5 \end{bmatrix}$ For action a: $diag(\pi^{(0)}[:,b]).c[:,b] = \begin{bmatrix} 0.25 \\ 0 \\ 0.5 \end{bmatrix}$

Now if we sum both we get $c\pi = \begin{bmatrix} 0.75 \\ 0 \\ 1 \end{bmatrix}$

Now that we have both $c\pi$ and $P\pi$ we can finally calculate J. The formula for J is:

$J\pi = (I - gamma*P\pi)^{-1}c\pi$, whose result is $\begin{bmatrix} 50.25 \\ 0 \\ 100 \end{bmatrix}$.

In step 4, We will now calculate the Q-Function for each action, using the J value we calculated in step 3. The Q-Function for the action, A is calculated by: $c[:,A] + \gamma * P[A]J$, where $c[:,A]$ is the A^{th} column of the cost matrix and P[A] is the probability transition Matrix of action A.

For action a: $Q_a = \begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix}$ For action b: $Q_b = \begin{bmatrix} 99.5 \\ 0 \\ 100 \end{bmatrix}$

Which will result in a $Q^{\pi}=\begin{bmatrix} 1 & 99.5 \\ 0 & 0 \\ 100 & 100 \end{bmatrix}$.
Now we will choose for each row the column that has the lowest value.
For row 0, column 0 has the lowest value, therefore $\pi^{(1)}[0][0] = 1$.
For row 1, both columns have the same value, therefore $\pi^{(1)}[1][0] = 1$ and $\pi^{(1)}[1][1] = 1$, however because rows need to add up to zero, we need to normalize it by dividing each value on the row by the sum of the row's value, so in actually $\pi^{(1)}[1][0] = 0.5$ and $\pi^{(1)}[1][1] = 0.5$.
For row 2, both columns have the same value, therefore $\pi^{(1)}[2][0] = 1$ and $\pi^{(1)}[2][1] = 1$, however because rows need to add up to zero, we need to normalize it by dividing each value on the row by the sum of the row's value, so in actually $\pi^{(1)}[2][0] = 0.5$ and $\pi^{(1)}[2][1] = 0.5$.

So in the end we get $\pi^{(1)} = \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$, concluding step 4.
In step 5, we increment the value of k, using k = k + 1
In step 6, we verify if we reached our stopping point. If we didn't we will go back to step 3.
And in step 7, we return $\pi^{(k)}$

➔ How to verify if a cost-to-go function is optimal?
To verify if a cost-to-go is optimal, do one iteration of value iteration using it. If it remains unchanged then it's optimal, otherwise it isn't.

➔ How to verify if a policy is optimal?
To verify if a policy is optimal, do one iteration of policy iteration using it. If it remains unchanged then it's optimal, otherwise it isn't.

Heuristics

In time step t, we have a belief b, and also the optimal MDP policy

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}(x_1) & \boldsymbol{b}(x_2) & \dots & \boldsymbol{b}(x_N) \end{bmatrix}$$

➔ **MLS** (Most Likely State)

- We choose the most likely state from the belief
- After that we will execute the corresponding action: $\pi_{mdp}(x_2)$.
- Basically $\pi_{\mathrm{MLS}}(\boldsymbol{b}) = \pi_{\mathrm{MDP}}(\operatorname*{argmax}_{x \in \mathcal{X}} \boldsymbol{b}(x))$

➔ **AV**

- States vote proportionally to their probability
- Each action a will have a number of votes.
- Each state x will calculate the corresponding action $\pi_{mdp}(x)$ and add b(x) to action's number of votes. The action that has the highest number of votes will be executed.

➔ **Q-MDP**

- No partial observability
- Start by creating a row vector Q_{MDP} with dimensions A x 1, with zeros in every element
- for each b(x) in b
    - multiply it by $Q_{MDP}(x, :)$
    - Q_{MDP} += b(x) * $Q_{MDP}(x, :)$
- Select argmin(Q_{MDP})

Another way to calculate it directly is to transpose b, obtaining b^T and multiply it by Q_{mdp}.In that case we have q_mdp = $b^T Q_{mdp}$

➔ **FIB**

- Factor out the belief but include partial observability

Sauce

➔ Compute **optimal policy** for MDP

- $\pi^*(x) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} Q^*(x, a)$
- For each row in Q*, select the smallest value
- i.e. for $\mathbf{Q}^* = \begin{bmatrix} 1.45 & 1.9 \\ 1.31 & 0.0 \\ 1.0 & 1.9 \end{bmatrix}$, the optimal policy is $\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \end{bmatrix}$

➔ Suppose that, at time step t = 0, agent chooses by Q-MDP. Compute the **action selected**

- Q-MDP prescribes $a_0 = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} b_0(x) Q^*(x, a)$, and i.e. $b_0\mathbf{Q}^* = \begin{bmatrix} 1.25 & 1.27 \end{bmatrix}$, we choose a_1

➔ Suppose that agent observes z_1 = u. Compute the **resulting belief** at t = 1

- Belief update rule
- $\boldsymbol{b}_1 = \rho\, \boldsymbol{b}_0 \mathbf{P}_a \operatorname{diag}(\mathbf{O}_{a,u})$, where ρ is a normalization constant

➔ Compute **optimal action** at t = 0 (Γ is gifted to us by the elves, α-vectors)

- $J^*(\boldsymbol{b}_0) = \min_{\boldsymbol{\alpha} \in \Gamma} \boldsymbol{b}_0 \cdot \boldsymbol{\alpha}$
- Cada b_0 * α dá um número real; escolhemos o mais pequeno e a ação correspondente

➔ Get the **distribution over states** at t = 2, assuming that the agent makes no observations:

- $\mu_2 = b_0 * P_{a0} * P_{a1}$

➔ Compute the **most likely state** at t = 2

- If they give us the first state, μ_0 is a column vector with 1 on that state
- $\mu_2 = \mu_0 * P_{a0} * P_{a1}$
- Forward algorithm
- If they ask for t = 0, just need to do le backwards (don't forget $\boldsymbol{\mu}_{0|0:3} = \boldsymbol{\alpha}_0 \otimes \boldsymbol{\beta}_0 / (\boldsymbol{\alpha}_0^\top \boldsymbol{\beta}_0)$)

➔ Compute the **most likely sequence of states** given actions and observations

- Viterbi

➔ Show that **b_0 is a stationary distribution** for the policy

- Obtain the transition probability matrix P
  - For each row of the policy, see which action it chooses and get the corresponding row from said action
- Then we just have to prove that $\boldsymbol{b}_0 \mathbf{P} = \boldsymbol{b}_0$

➔ Calculate chain **periodicity**

- Draw it from the transition probability matrix

➔ Is the chain **ergodic**?

- "The chain is periodic with period N. Therefore, for any initial distribution b ≠ b_0, the state distribution will oscillate with a period of N and will not converge to b_0. It follows that the chain is not ergodic."

➔ Is the chain **irreducible**?

- No, since there are multiple communicating classes

➔ Indicate one advantage of Q-MDP over MLS and AV

- Both the MLS and AV heuristics use directly the optimal MDP policy and, therefore, prescribe only actions that are optimal for the underlying MDP. Q-MDP, on the other hand, may prescribe actions that the underlying MDP policy never selects. This was observed, for example, in the Tiger problem, where the Q-MDP did select the action "Listen", while neither MLS nor AV were able to select such action.

➔ Compute the **cost-to-go** associated with policy π

- $\boldsymbol{J}^{\pi} = (\boldsymbol{I} - \gamma \mathbf{P}_{\pi})^{-1} \boldsymbol{c}_{\pi}$

➔ Compute the **greedy policy** with respect to the cost-to-go function…

- We compute the Q-values for each action a ∈ A
- $\boldsymbol{Q}^{\pi}_{:,A} = \boldsymbol{c}_{:,A} + \gamma \mathbf{P}_{A} \boldsymbol{J}^{\pi}$ for action A, for example
- The resulting greedy policy is, therefore, $\boldsymbol{\pi}_g^{J^{\pi}}$ (same dimension as each of the Q-values)
  - To get it, just put probability 1 on the smallest value of the actions for a given row
  - And 0 in all others

➔ Do you believe that / is the policy π is optimal?

- We have Jπ, so we perform one step of value iteration (or policy iteration as alternative)
  - $\boldsymbol{Q}^{\pi}_{:,a} = \mathbf{C}_{:,a} + \gamma \mathbf{P}_{a} \boldsymbol{J}^{\pi}$ for each action
  - $\boldsymbol{J}_{\text{new}} = \min_{a} \boldsymbol{Q}_{:,a}$, we create a row vector with the smallest values for each of the rows in the Q things computed for each action
  - Since J_{new} = Jπ, we can conclude that Jπ = J* and thus π = π*
  - And vice versa if J_{new} != Jπ
- "The policy π is, most likely, not optimal, since the greedy policy obtained from Jπ is different from π, which can be observed by noting that the transition probabilities resulting from $\boldsymbol{\pi}_g^{J^{\pi}}$ are different from $P_{π}$. Therefore, since the greedy policy is what we would get after one step of policy iteration, π is, most likely, not optimal."

➔ Perform one step of **policy iteration** (using policy π)

- $\boldsymbol{Q}_a = \boldsymbol{c}_a + \gamma \mathbf{P}_a \boldsymbol{J}^\pi$ for each action
- π_{new}, we create a row vector with the smallest values for each of the rows in the Q things computed for each action
- If $\pi_{new} = \pi$, then π is optimal