

# Paradigmas da IA, Raciocínio Baseado em Casos e Aprendizado por Reforço

# Tipos de Aprendizado de Máquina

- Supervisionado
- Não Supervisionado
- Por Reforço

# Aprendizado Supervisionado

- É um aprendizado por indução ou seja do particular exemplos para o geral
- O algoritmo de aprendizado (indutor) recebe um conjunto de exemplos de treinamento para os quais os rótulos da classe associada são conhecidos
- Cada exemplo (instância ou padrão) é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada
- O objetivo do indutor é construir um classificador ou regressor que possa determinar corretamente a classe de novos exemplos ainda não conhecidos
- Exemplo Árvores de Decisão, Redes neurais, SVM, etc...

# Aprendizado Não Supervisionado

- O indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters
- Após a determinação dos agrupamentos, em geral, é necessário uma análise para determinar o que cada agrupamento significa no contexto do problema que está sendo analisado
- K-Médias, Clustering, Redes SOM

# Aprendizado por Reforço

- O indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters
- Após a determinação dos agrupamentos, em geral, é necessário uma análise para determinar o que cada agrupamento significa no contexto do problema que está sendo analisado
- Exemplo Q-Learning

# Os paradigmas da IA

- Simbolista
- Evolucionista
- Conexionista
- Estatístico

# O Paradigma Simbólico

- Neste paradigma um conceito é representado em uma estrutura simbólica, e o aprendizado é realizado através da apresentação de exemplos e contraexemplos, deste conceito.
- As estruturas simbólicas estão tipicamente representadas em alguma expressão lógica, regras de produção ou redes semânticas.
- Exemplos de técnicas que utilizam este paradigma são: Sistemas Especialistas e Árvores de Decisão.

# Um pouco sobre o paradigmas Simbolista

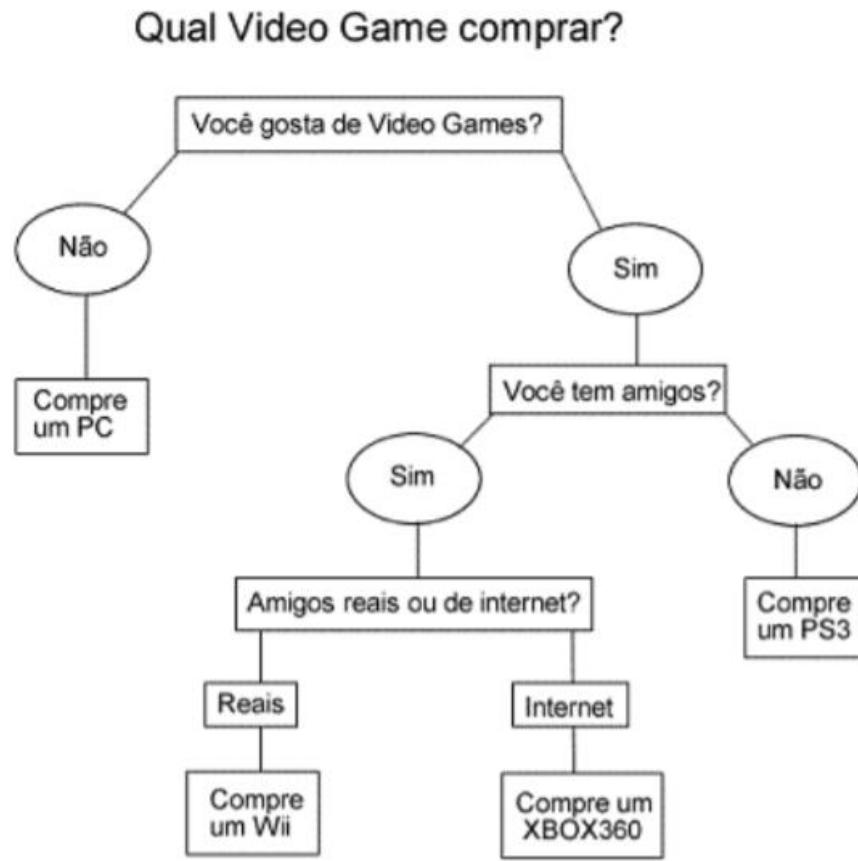
- Trata-se dos mecanismos de IA que efetuam transformações simbólicas (números, letras, palavras e símbolos), dentre estes mecanismos estão os programas de uso geral como sistemas ERP, workflows, planilhas, calculadoras (todos com alto determinismo, baixa generalização, baixo reconhecimento e nenhuma criatividade).
- Há ainda os sistemas de raciocínio que operam sobre as regras canônicas da lógica proposicional, os chamados Reasoners, onde bases de conhecimento lógico processual previamente preparada por especialistas (Ontologias Computacionais) são capazes de níveis superiores de generalização e reconhecimento se comparados aos sistemas mencionados no parágrafo anterior.
- Tais sistemas vêm se mostrando muito úteis em aplicações nas mais variadas áreas, entre elas: cálculo de rotas em aplicações como Google Maps, diagnóstico médico, composição de respostas a questões jurídicas



# Um pouco sobre o paradigmas Simbolista

- A IA simbólica está relacionada com a forma que o ser humano raciocina e foi popularizada com o surgimento dos Sistemas Especialistas e principalmente pela influência da linguagem Prolog.
- Se você não conhece, o Prolog é uma linguagem de programação totalmente baseada em lógica de predicados que opera por meio de regras e símbolos (clique aqui para acessar uma playlist que gravei sobre o Prolog).
- John McCarthy (criador da linguagem Lisp) e Newell (criador do Solucionador Geral de Problemas) foram os principais personagens dessa abordagem, que em resumo, apresenta um conhecimento explícito de um determinado problema e que também possui uma explicação do processo que levou a resposta dada pelo sistema de IA.
- O exemplo clássico da IA simbólica são os sistemas especialistas, que necessitam que o conhecimento sobre o problema seja definido manualmente no sistema para que ele possa raciocinar e tomar as decisões.

# Um modelo simbólico baseado em árvores



## 2. O Paradigma Estatístico

- No paradigma estatístico é utilizado um modelo estatístico que encontre uma hipótese que possua uma boa aproximação do conceito a ser induzido.
- O aprendizado consiste em encontrar os melhores parâmetros para o modelo.
- Estes modelo podem ser paramétricos (quando fazem alguma suposição sobre a distribuição dos dados, ou podem ser não paramétricos, quando não fazem suposição sobre a distribuição dos dados.
- Dentre os modelos estatísticos utilizados em aprendizagem de máquina podemos destacar os modelos Bayesianos e o Naive Bayes

# 3 - Paradigma Conexionista

- O nome conexionista vem da área de pesquisa de redes neurais artificiais.
- Uma rede neural artificial é um modelo computacional inspirado no funcionamento do cérebro humano.
- Uma RNA possui três componentes principais: unidade de processamento “os neurônios”, conexões “sinapses” e uma topologia.
- As redes neurais possuem como principal característica aprender através de exemplos e poder de generalização.
- As redes Multi Layer Perceptron (MLP) e Self Organizing Map (SOM) são exemplos de técnicas que utilizam este paradigma de aprendizado.

# 4 - Paradigma Evolutivo

- Este paradigma foi inspirado na teoria da evolução das espécies de Charles Darwin.
- O algoritmo inicia com uma população de indivíduos, onde cada indivíduo representa uma possível solução.
- Os indivíduos competem entre si, os indivíduos com menor desempenho são descartados, e os indivíduos com melhores desempenho são selecionados para reprodução (Crossover), os novos indivíduos gerados podem ou não sofrer mutação.
- A população evolui através de várias gerações, até que uma solução ótima seja encontrada.
- Algoritmos genéticos e Programação genética são exemplo do Paradigma Evolutivo

# Raciocínio Baseado em Casos

# Raciocínio baseado em casos (RBC)

- O **raciocínio baseado em casos (RBC)** é uma abordagem que busca resolver novos problemas adaptando soluções utilizadas para resolver problemas anteriores.
- Dentre as características do funcionamento de um sistema RBC estão:
  - A extração do conhecimento a partir de casos ou experiências com que o próprio sistema se depara.
  - A identificação das características mais significantes dos casos apresentados a fim de devolver uma melhor solução (resposta).
  - O armazenamento do caso e sua respectiva solução.
- A qualidade de um sistema RBC depende de sua experiência, ou seja, depende do número de casos relevantes que farão parte da base de casos.

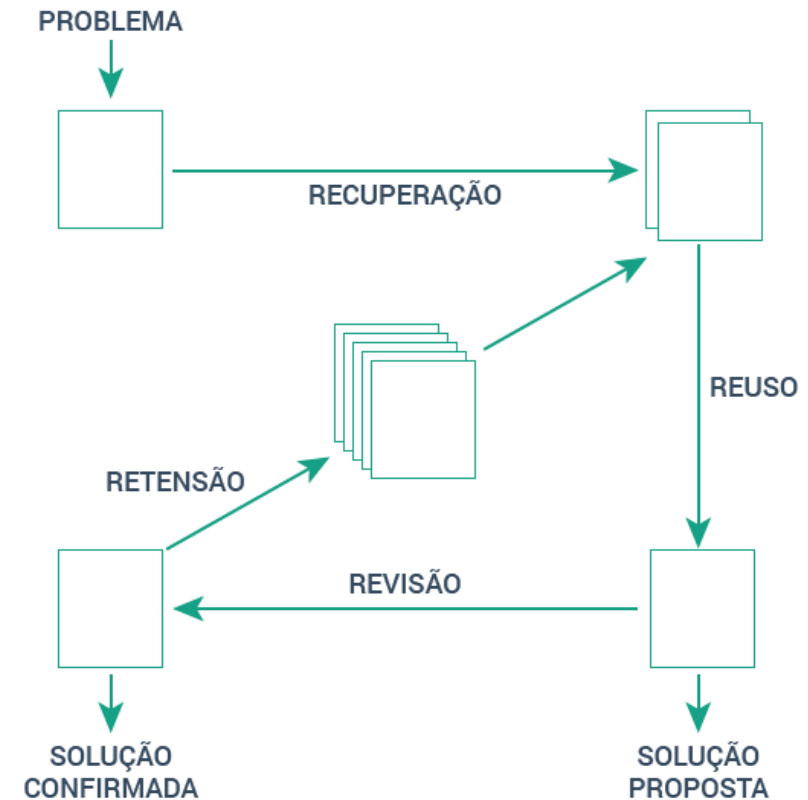
# O ciclo de funcionamento de um sistema RBC

- *Recuperação*: a partir da apresentação ao sistema de um novo problema é feita a recuperação na base de casos daquele mais parecido com o problema em questão. Isto é feito a partir da identificação das características mais significantes em comum entre os casos.
- *Reutilização*: a partir do caso recuperado é feita a reutilização da solução associada àquele caso. Geralmente a solução do caso recuperado é transferida ao novo problema diretamente como sua solução.
- *Revisão*: é feita quando a solução não pode ser aplicada diretamente ao novo problema. O sistema avalia as diferenças entre os problemas (o novo e o recuperado), quais as partes do caso recuperado são semelhantes ao novo caso e podem ser transferidas adaptando assim a solução do caso recuperado da base à solução do novo caso.
- *Retenção*: é o processo de armazenar o novo caso e sua respectiva solução para futuras recuperações. O sistema irá decidir qual informação armazenar e de que forma.



# Desenvolvimento de um sistema de raciocínio baseado em casos

- Uma vez que existe disponível uma base de dados para ser investigada, é possível investigá-la para extrair conhecimento a ser aplicado na tomada de novas decisões. A construção de um sistema de RBC a partir de uma base de dados passa pela definição de técnicas e formas de implementação de cada um dos componentes do sistema.
- As etapas mais importantes do processo de desenvolvimento de um sistema RBC são:
  - Seleção de uma base de informações (um sistema de banco de dados) que contenha implicitamente o conhecimento necessário na solução de problemas.
  - Definição de que atributos da informação são relevantes e podem ser utilizados para a solução do problema.
  - Definição de índices para os casos, para que seja possível a sua recuperação quando necessário. Na indexação deve-se decidir o que armazenar em um novo caso, encontrando uma estrutura apropriada para a descrição dos conteúdos dos casos e decidir como a base de casos deve ser armazenada.
  - Definição dos métodos de recuperação de casos para verificação da similaridade entre os casos contidos na base e os novos problemas.



Aprendizado por Reforço

# Lembrando os tipos de aprendizado

- O aprendizado de máquina (*Machine Learning*) é categoria de algoritmos de Inteligência Artificial que é subdividida em:
- Aprendizagem Supervisionada – quando o algoritmo é treinado com dados de entrada e as respectivas saídas
- Aprendizagem Não Supervisionada – quando o algoritmo é treinado somente os dados de entrada para determinar as saídas.
- Mas existe uma terceira categoria de aprendizagem, chamada de Aprendizagem por Reforço (ou *Reinforcement Learning*).
- Na Aprendizagem por Reforço os modelos de aprendizado de máquina são treinados para tomar uma sequência de decisões, como forma de atingir uma meta em um ambiente incerto e potencialmente complexo.
- Ao invés do treinamento com dados de entrada e saída o modelo emprega tentativas e erros para encontrar uma solução para o problema, recebendo recompensas ou penalidades pelas ações que executa, com o objetivo é maximizar a recompensa total.

# Sistema Baseados em Aprendizado por Reforço

- A idéia que se aprende interagindo com o ambiente é provavelmente a primeira a ocorrer quando se pensa na natureza do aprendizado.
- Essa conexão produz uma riqueza de aprendizados sobre causa e efeito, sobre as consequências das ações e sobre o que fazer para alcançar objetivos.
- Aprender com a interação é uma ideia fundamental subjacente a quase todas as teorias de aprendizado e inteligência (SULTON; BARTOS, 2018).

# Características do Aprendizado por Reforço

- O aprendizado por reforço envolve aprender o que fazer - como mapear situações para ações - de modo a maximizar um sinal numérico de uma recompensa.
- O modelo não é informado sobre quais ações executar, mas, em vez disso, deve descobrir quais ações geram mais recompensa, por meio de sucessivas tentativas (SULTTON; BARTOS, 2018).
- Em muito casos, as ações podem afetar não apenas a recompensa imediata, mas também a próxima situação e, com isso, todas as recompensas subsequentes.
- Essas duas características - pesquisa por tentativa e erro e recompensa atrasada - são as duas características distintivas mais importantes do aprendizado por reforço (SULTTON; BARTOS, 2018).
- O aprendizado por reforço, é simultaneamente um problema, uma classe de métodos de solução que funcionam bem no problema e o campo que estuda esse problema e seus métodos de solução (SULTTON; BARTOS, 2018).
- Assim como outros algoritmos de IA, o aprendizado por reforço consiste basicamente num agente inteligente e suas iterações com o ambiente (RUSSEL; NORVIG, 2013)

# Componentes do aprendizado por reforço

- Segundo Sultton e Bartos (2018) além do agente e do ambiente, é possível identificar quatro elementos principais de um sistema de aprendizado por reforço:
  - uma política,
  - um sinal de recompensa,
  - uma função de valor e,,
  - um modelo de ambiente (opcionalmente),

# Politica

- Uma **política** define a maneira de o agente de aprendizagem se comportar em um determinado momento.
- A política é o núcleo de um agente de aprendizado por reforço, no sentido de que por si só é suficiente para determinar o comportamento.
- Em geral, as políticas podem ser estocásticas, especificando probabilidades para cada ação.

# Sinal de recompensa

- Um **sinal de recompensa** define o objetivo de um problema de aprendizado por reforço.
- Em geral, os sinais de recompensa podem ser funções estocásticas do estado do ambiente e das ações tomadas.



# Função de valor

- Enquanto o sinal de recompensa indica o que é bom em um sentido imediato, uma **função de valor** especifica o que é bom a longo prazo.
- Enquanto as recompensas determinam a conveniência imediata e intrínseca dos estados ambientais, os valores indicam a conveniência a longo prazo dos estados após levar em conta os estados que provavelmente seguirão e as recompensas disponíveis nesses estados.

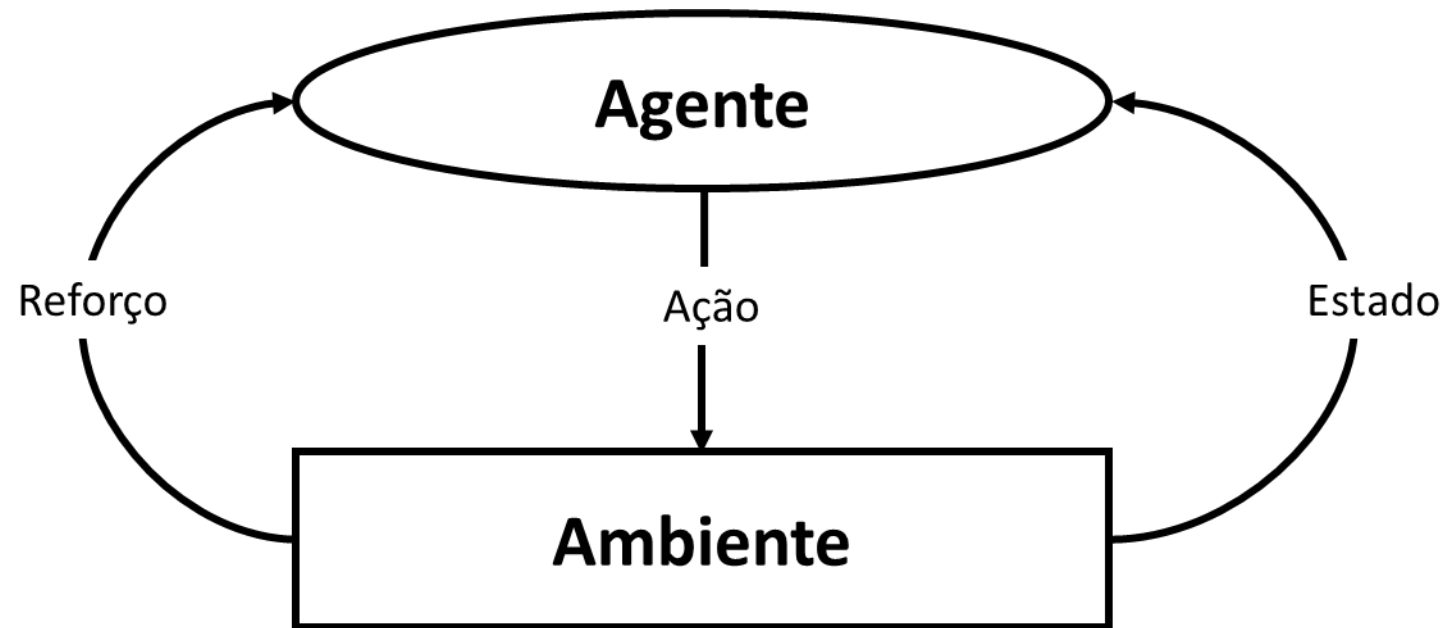
# Função de valor x Recompensas

- Um estado sempre pode gerar uma recompensa imediata baixa, mas ainda tem um valor alto porque é seguido regularmente por outros estados que produzem recompensas altas.
- Para fazer uma analogia humana, as recompensas são um pouco como o prazer (se alto) e a dor (se baixo),
- Enquanto os valores correspondem a um julgamento mais refinado e previdente do quanto estamos satisfeitos ou descontentes com o fato de nosso ambiente estar em um estado particular.
- As recompensas são, em certo sentido, primárias, enquanto os valores, como previsões de recompensas, são secundários.
- Sem recompensas, não poderia haver valores, e o único objetivo de estimar valores é obter mais recompensa.
- Buscamos ações que gerem estados de maior valor, e não de maior recompensa, porque essas ações obtêm a maior quantidade de recompensa a longo prazo.

# Modelo de ambiente

- O quarto e último elemento de alguns sistemas de aprendizado por reforço é um **modelo do ambiente**.
- Os métodos para resolver problemas de aprendizado por reforço que usam modelos e planejamento são chamados de métodos baseados em modelo, em oposição a métodos mais simples, sem modelos, que são explicitamente aprendizes de tentativa e erro - vistos quase o oposto do planejamento.
- O aprendizado por reforço abrange o espectro, desde o aprendizado de baixo nível por tentativa e erro até o planejamento deliberativo de alto nível.

# Elementos centrais do aprendizado por reforço



# Processo de Decisão de Markov - PDM

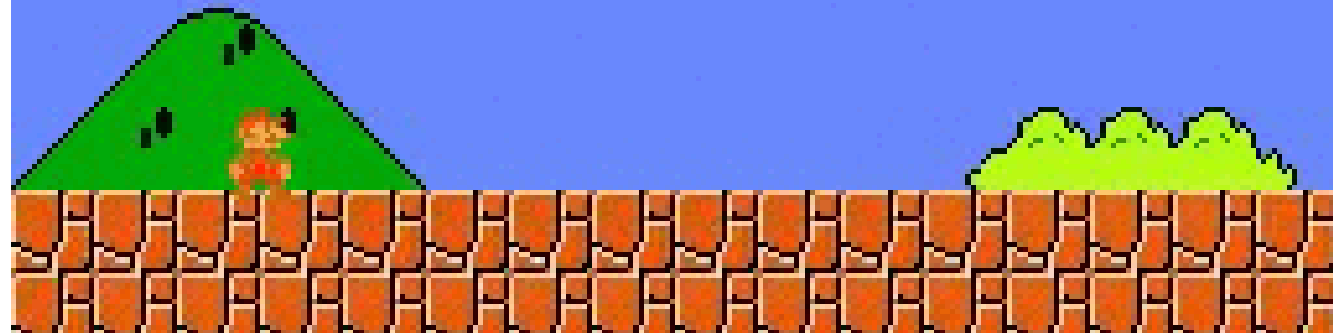
- O aprendizado por reforço é baseado em processos de decisão de Markov (PDM) , que consiste nos seguintes elementos (SUTTON; BRATO, 2018):
  - $S$ : Conjunto finito de estados do sistema
  - $s_t$  : Estado do agente no instante  $t$
  - $A$ : Conjunto finito de ações do sistema
  - $a_t$ : Ação tomada no instante  $t$
  - Em  $s_t$ , ação  $a_t$  é tomada. No instante seguinte a recompensa  $r_{t+1}$  é recebida e o estado muda para  $s_{t+1}$
  - $P$ : Modelo de transição de estados
  - Probabilidade do próximo estado:  $P(s_{t+1} \mid s_t, a_t)$
  - $R$ : Função de recompensas  $R$
  - Distribuição de recompensas:  $p(r_{t+1} \mid s_t, a_t)$
  - Estado(s) inicial(is), estado(s) objetivos
  - Episódios de ações do estado inicial ao estado final.

MARIO  
000000

000000

WORLD  
1-1

TIME



# O que é Cadeia de Markov?

- Um Processo Markoviano ou Cadeia de Markov é um **processo estocástico**, ou seja, é totalmente aleatório, onde o **Estado futuro** depende apenas de seu **Estado atual**.
- Vamos o Jogo do Mário Bros
- Nosso herói possa fazer apenas três coisas: **andar**, **ficar parado/esperando**, ou **pular**. Cada uma dessas três coisas é um **Estado**.



# O que é Cadeia de Markov?

- Segundo é previsto pela Cadeia de Markov, **o próximo estado em que Mario vai estar apenas depende do estado atual dele.**
- Ou seja, não importa se Mario ficou pulando nos últimos 5 minutos, se nesse momento ele está parado, é tudo que precisamos saber.





# O que é Cadeia de Markov?

- Por que isso é importante?
- Simples, já que os Estados anteriores não são interessantes para nós, podemos saber qual o próximo Estado de Mario ao calcularmos a probabilidade de transição dele.
- **Em outras palavras, podemos saber a chance que Mario tem de ir do estado parado para o estado andando, por exemplo.**
- Podemos fazer isso com um elemento chamado Matriz de Transição.

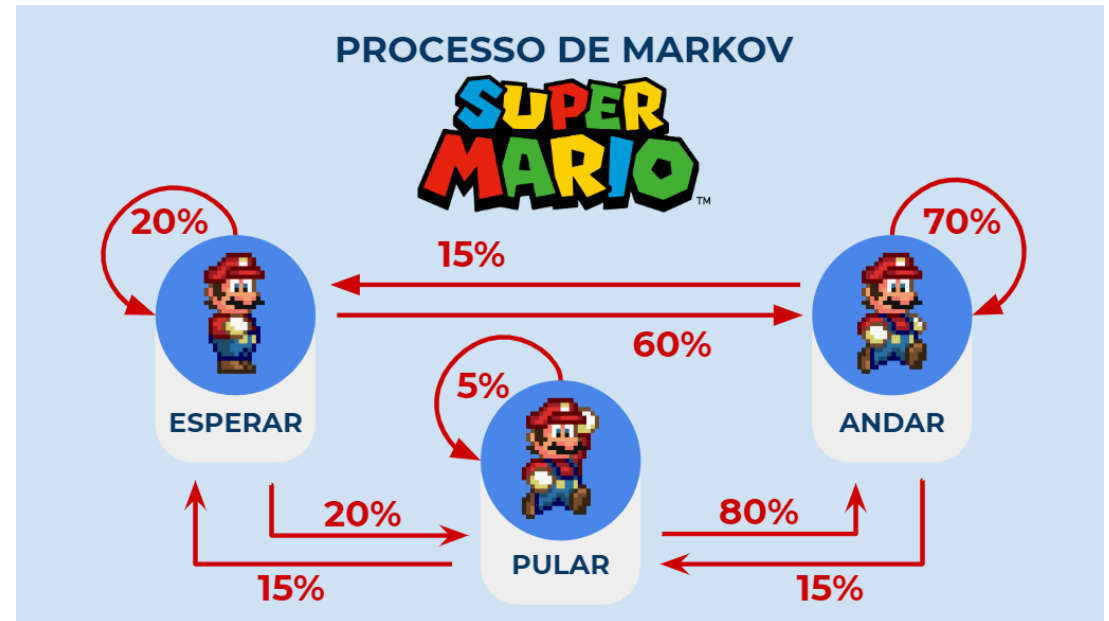
# A Matriz de Transição

- Você pode imaginar uma Matriz de Transição como uma tabela onde cada linha é um possível Estado do Mario.
- Em nosso caso, teremos uma tabela com três linhas.
- Assim como temos um estado em cada linha, também teremos uma coluna para cada estado.
- No fim das contas, você acaba com uma matriz cruzada.

	Matriz de transição Mario		
	Andar	Esperar	Pular
Andar	70%	15%	15%
Esperar	60%	20%	20%
Pular	80%	15%	5%

# Transição de Estados

- Supondo alguns valores na planilha acima, **se Mario estiver no estado Andar, ele tem 70% de probabilidade de continuar no Estado Andar;**
- da mesma forma, ele tem 15% de chance de ir para o estado Esperar ou Pular.
- Caso o Mário esteja no estado Pular, ele tem apenas 5% de probabilidade de continuar nesse estado.



# Conjunto finito de Estados (denotado pela letra S)

- Esse conjunto de estados podem ser os que falamos acima: **andar**, **esperar** e **pular**.
- Contudo, cada projeto terá seu próprio conjunto de Estados, como por exemplo, a informação se uma porta está aberta ou fechada, ou a posição geográfica de algo.

# Conjunto finito de ações (A)

- Ações são justamente o que parece: são ações que, em nosso caso, Mario pode realizar em cada Estado.
- Ação aqui pode estar no ato de apertar um botão para Mario passar de um Estado para outro, **como apertar o botão para frente ou para trás para fazer o Mário sair do Estado parado para andando.**

# Modelo de probabilidade (P)

- É a probabilidade de uma Ação levar Mario do seu Estado atual para um Estado futuro.

# Recompensa (R)

- A Recompensa é um valor número que Mario irá receber após realizar aquela Ação naquele Estado.
- Quando esse número é positivo, quer dizer que a Ação que Mario fez naquele Estado foi benéfica; caso o número seja negativo, Mario será punido por aquela Ação.
- O objetivo principal de Mario é realizar o conjunto de Ações que maximizam a Recompensa até seu objetivo final: completar a fase.

# Fator de desconto ( $\gamma$ )

- O fator de desconto é um número escalar, geralmente entre 0 e 1. Ele influencia no total de recompensa futura que o seu Agente irá receber.
- Por exemplo, um fator de desconto de 0,9 indica que quanto mais avançado no futuro está seu Agente, maior será a recompensa dele.
- Em outras palavras, Mario irá preferir percorrer caminhos na fase que irão garantir uma recompensa maior no futuro, ao invés de uma Recompensa imediata.

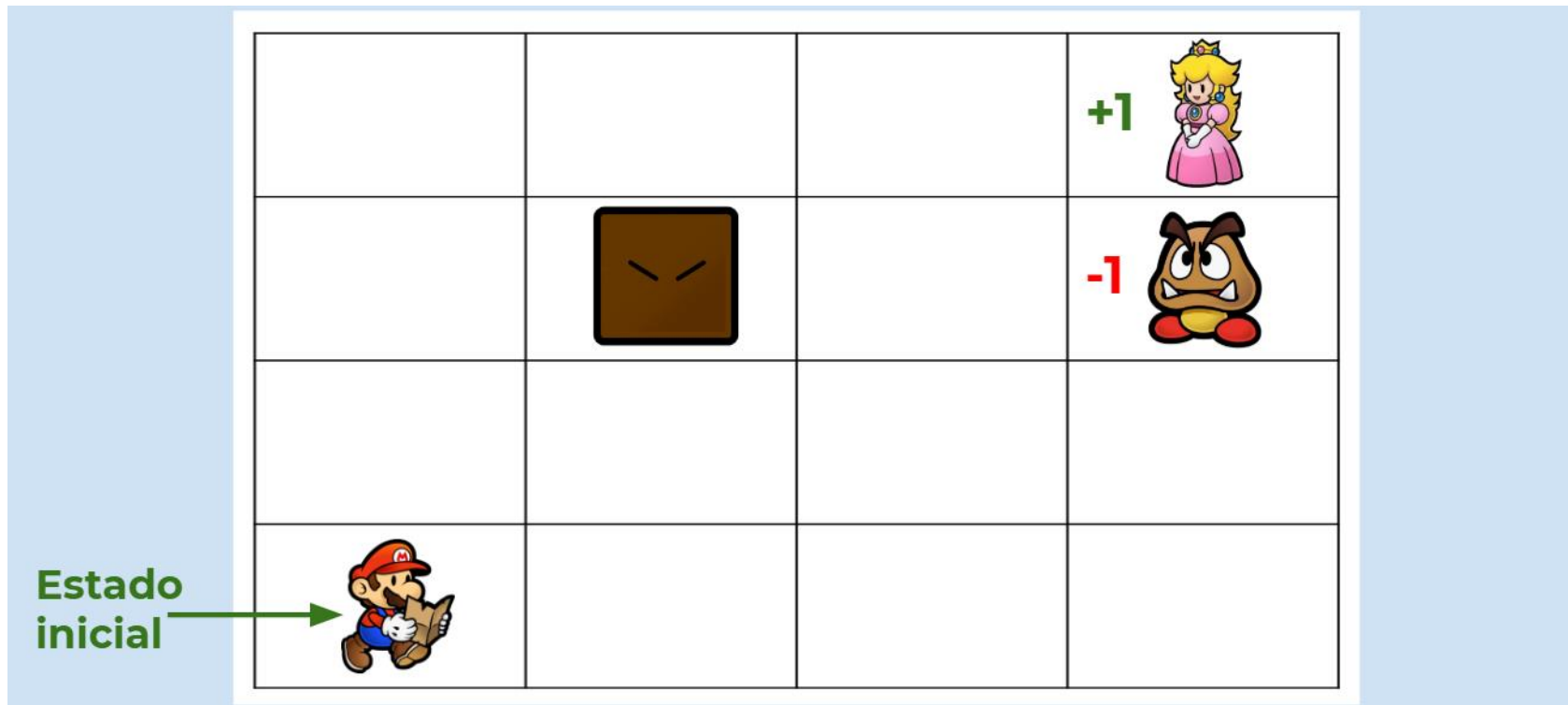


# Entendendo como funciona

- Com base nessas cinco propriedades base, um algoritmo de RL é capaz de aprender, através de tentativa e erro, quais Ações (A) ele deve fazer em cada Estado (S) a fim de que sua Recompensa (R) seja a maior possível para atingir um Objetivo.
- **Tenha em mente que, durante o treinamento, o Agente irá tentar todas as possibilidades e selecionar a que garantir um melhor resultado no final.**

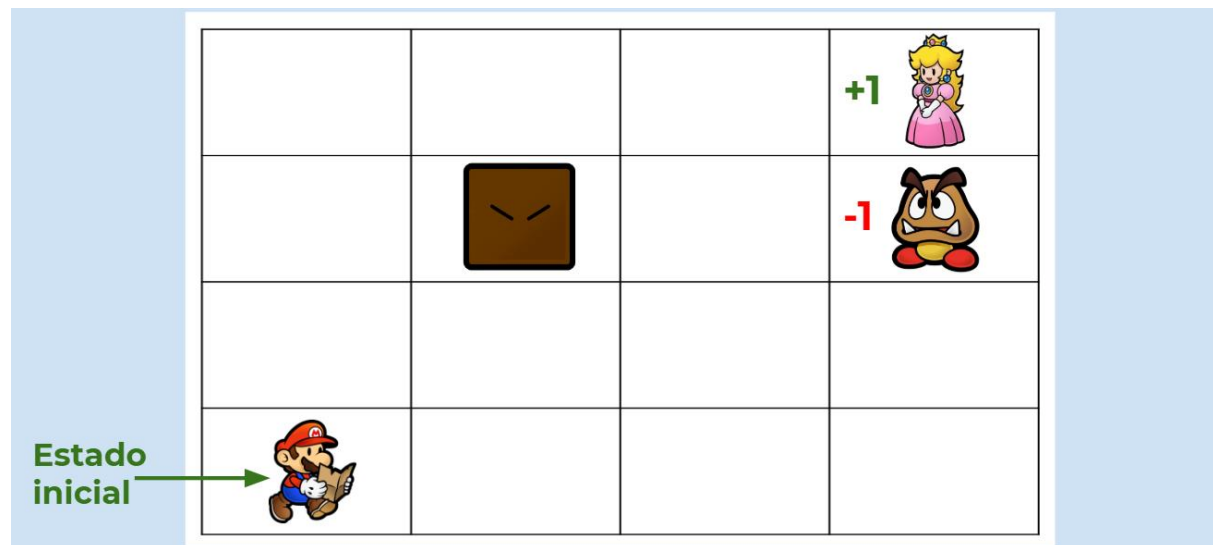
# Entendendo como Funciona

- Vamos aprimorar nosso exemplo com o clássico mundo em grade.
- Mario iria interagir com ambiente da seguinte forma.



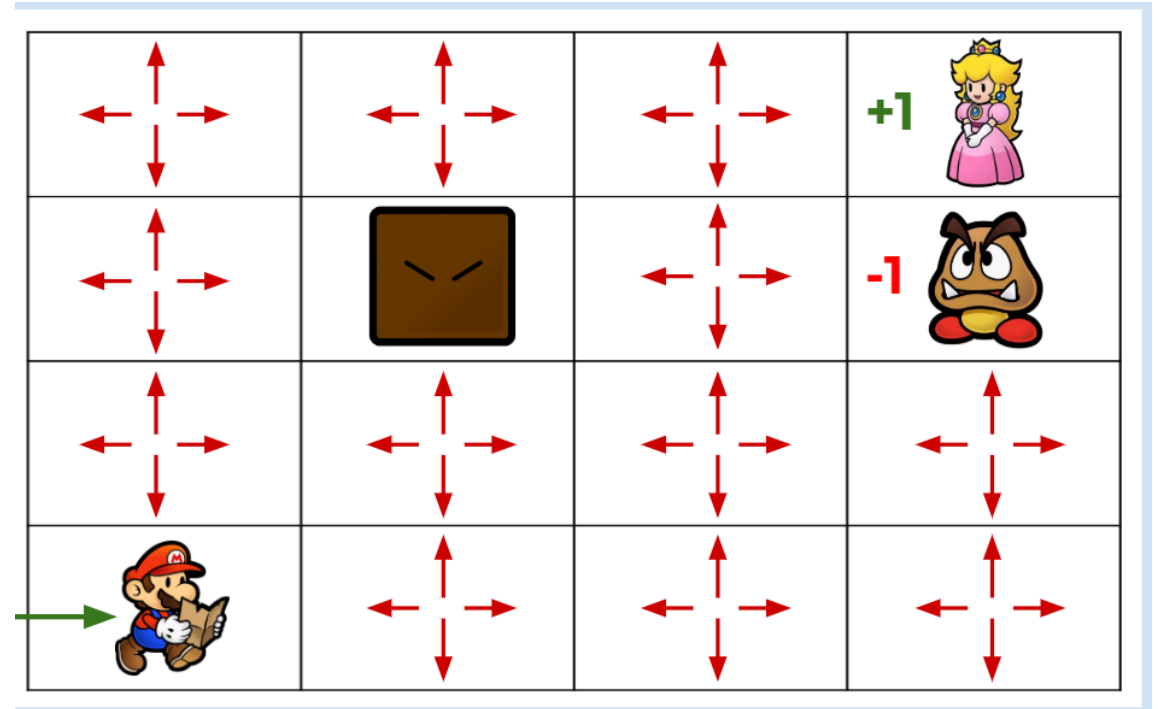
# Entendendo como Funciona

- Imagine que esse mundo em grade é a fase de nosso jogo.
- No canto inferior esquerdo está Mario em seu estado inicial — onde ele começará a fase.
- No canto superior direito está a princesa Peach, seu objetivo; logo abaixo dela está um inimigo de Mario, que ele precisa evitar; e, por fim, um obstáculo é encontrado na segunda linha, segunda coluna, bloqueando a passagem de Mario por aquele quadrado.



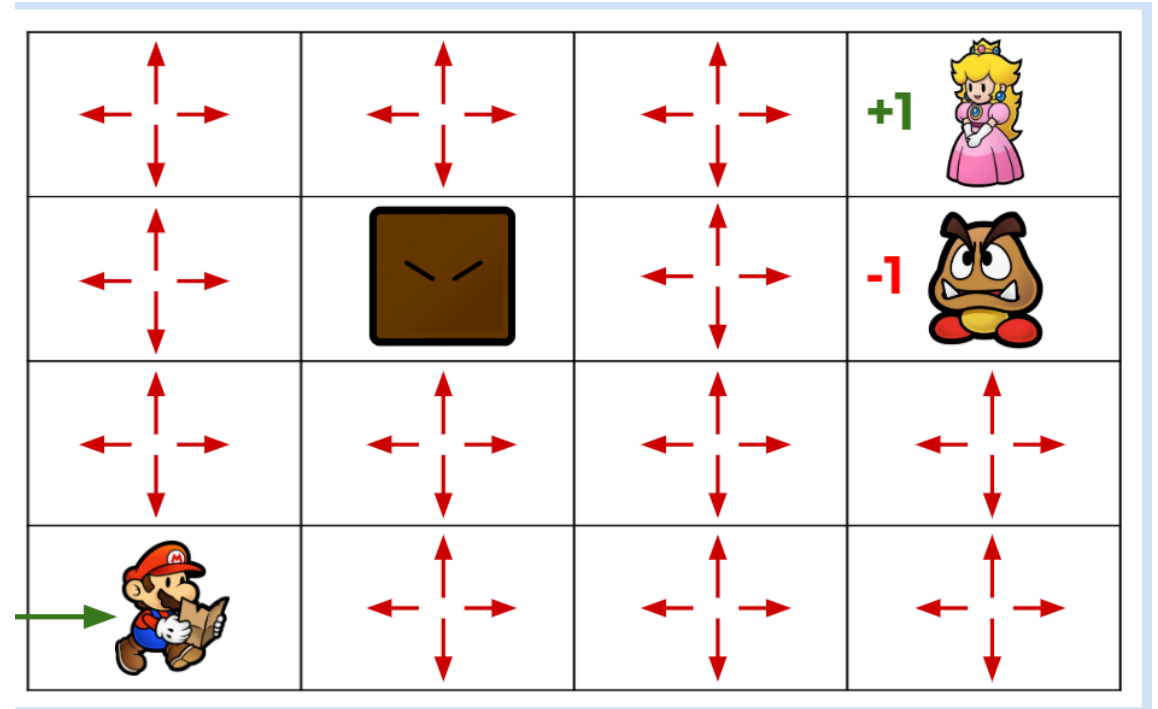
# Entendendo como Funciona

- Cada quadrado da nossa fase gradeada é um Estado em que Mario poderá estar.
- Agora, para nosso exemplo, imagine que Mario pode fazer uma de quatro Ações:
- ir para cima,
- para baixo,
- para esquerda e
- para direita.
- Em outras palavras, nosso conjunto de Ações (A) é quatro







# Entendendo como Funciona

- O principal objetivo de nosso Agente (Mario) é seguir a **melhor política**, ou seja, **escolher as melhores Ações em cada Estado para coletar o máximo de Recompensas no caminho.**
- Mas, como o Mario sabe o que é uma Ação boa?



# Recompensas e punições em MDP

- Mario saberá quais as melhores Ações para se realizar em cada Estado ao receber uma Recompensa por ela.
- Recompensas nada mais são que números inteiros ou flutuantes, e o Agente sempre irá tentar coletar o máximo possível de Recompensa, o que faz o processo de definição do valor da Recompensa uma verdadeira arte.

			+1 
			-1 
			

Recompensas (R) e Punições (P) em MDP:

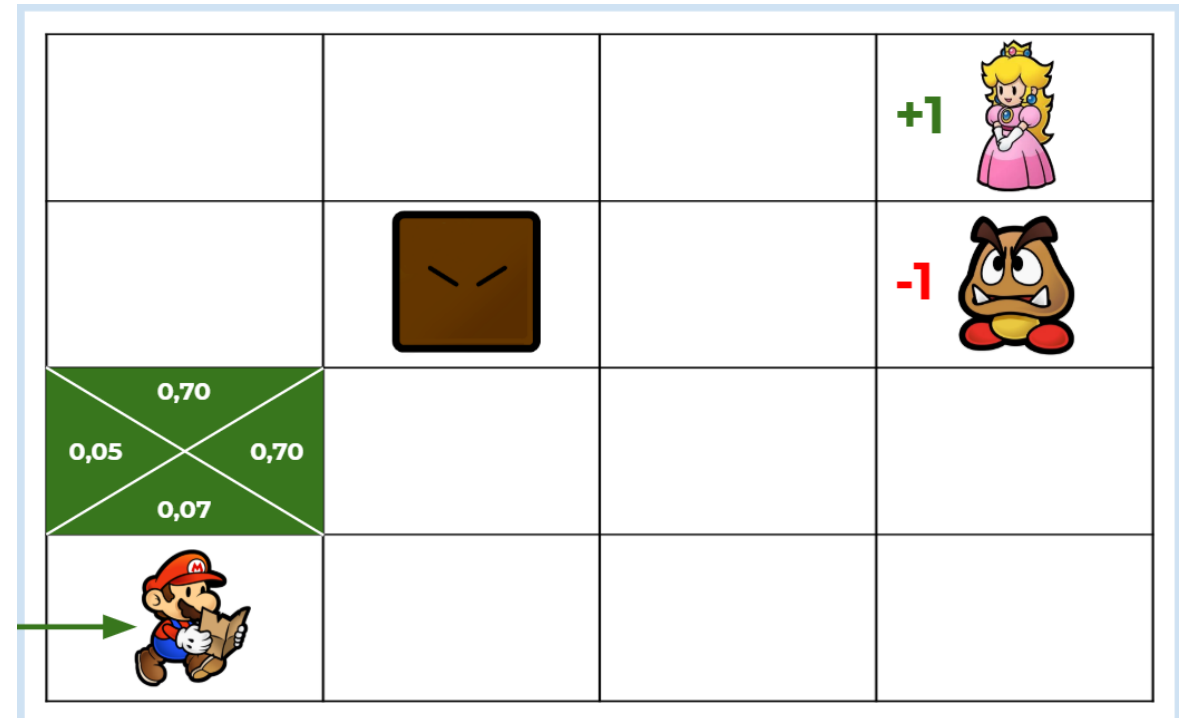
- Recompensa: +1 (Princess Peach)
- Punição: -1 (Koopa)
- Recompensas (R) e Punições (P) em MDP:

Recompensas (R) e Punições (P) em MDP:

- Recompensa: +1 (Princess Peach)
- Punição: -1 (Koopa)
- Recompensas (R) e Punições (P) em MDP:

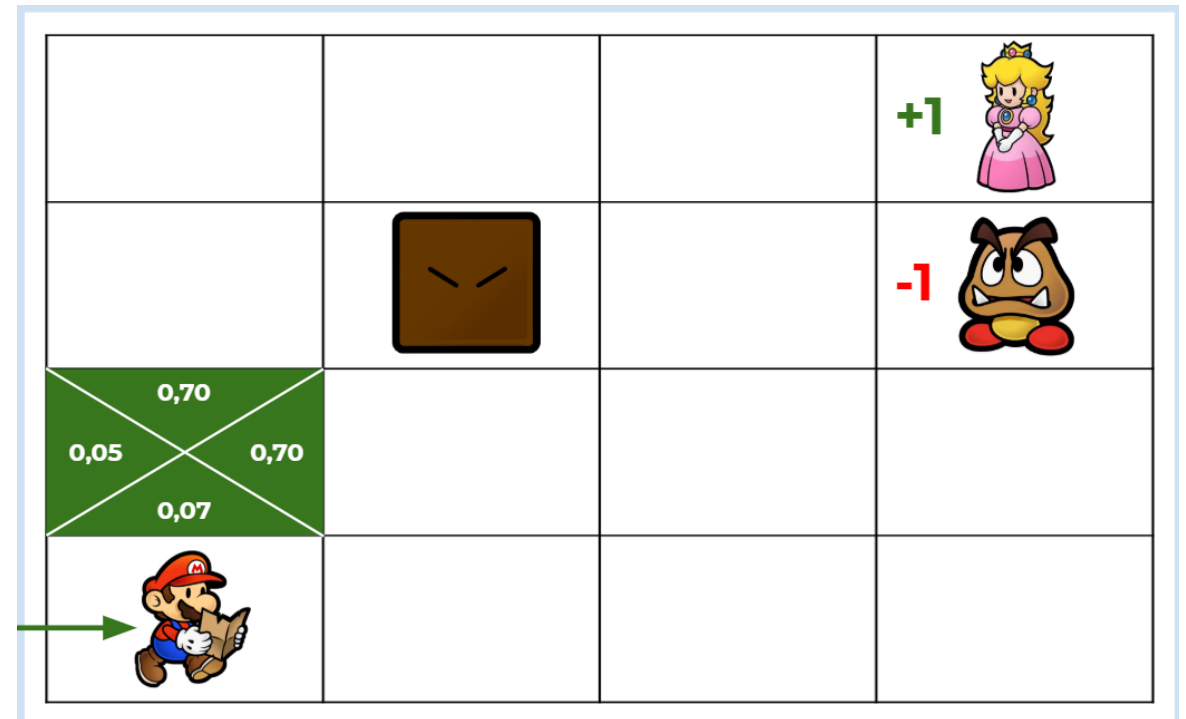
# Recompensas e punições em MDP

- Na imagem ao lado.
- O quadrado verde indica qual será a Recompensa esperada no final se ele escolher uma das Ações.
- Por exemplo: **se Mario escolher ir para cima, a Recompensa que ele receberá no final do Episódio (fase) será 0,70.**
- Caso ele escolha ir para a esquerda, ele vai receber uma Recompensa menor de 0,05, afinal, ele está batendo na parede.



# Recompensas e punições em MDP

- Na imagem ao lado.
- O quadrado verde indica qual será a Recompensa esperada no final se ele escolher uma das Ações.
- Por exemplo: **se Mario escolher ir para cima, a Recompensa que ele receberá no final do Episódio (fase) será 0,70.**
- Caso ele escolha ir para a esquerda, ele vai receber uma Recompensa menor de 0,05, afinal, ele está batendo na parede.





# Recompensas e punições em MDP

- Na prática, **todos os quadrados estarão preenchidos da mesma forma,**
- Onde cada Ação em um Estado resultará em uma Recompensa final diferente.
- O **Agente** sempre escolherá as **Ações** que maximizam a Recompensa final ao chegar na Princesa.

