# 2102470 Học máy

# Bài giảng: Giới thiệu về học tăng cường

Chương 4: Học tăng cường



VTPham@IUH, 2024

# Ôn lại bài học trước

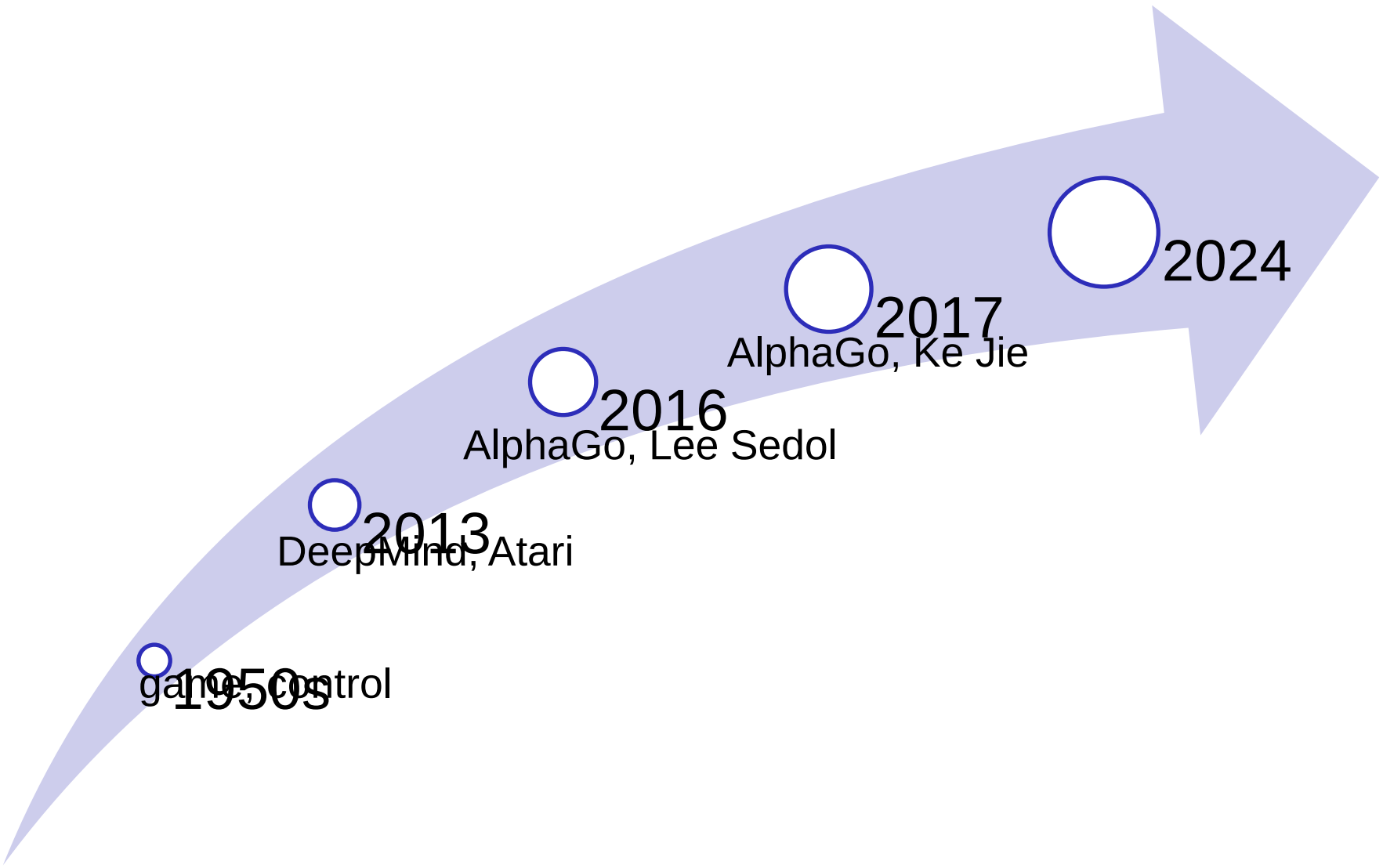
- Bạn có nhớ ? % ?

# Nội dung chính

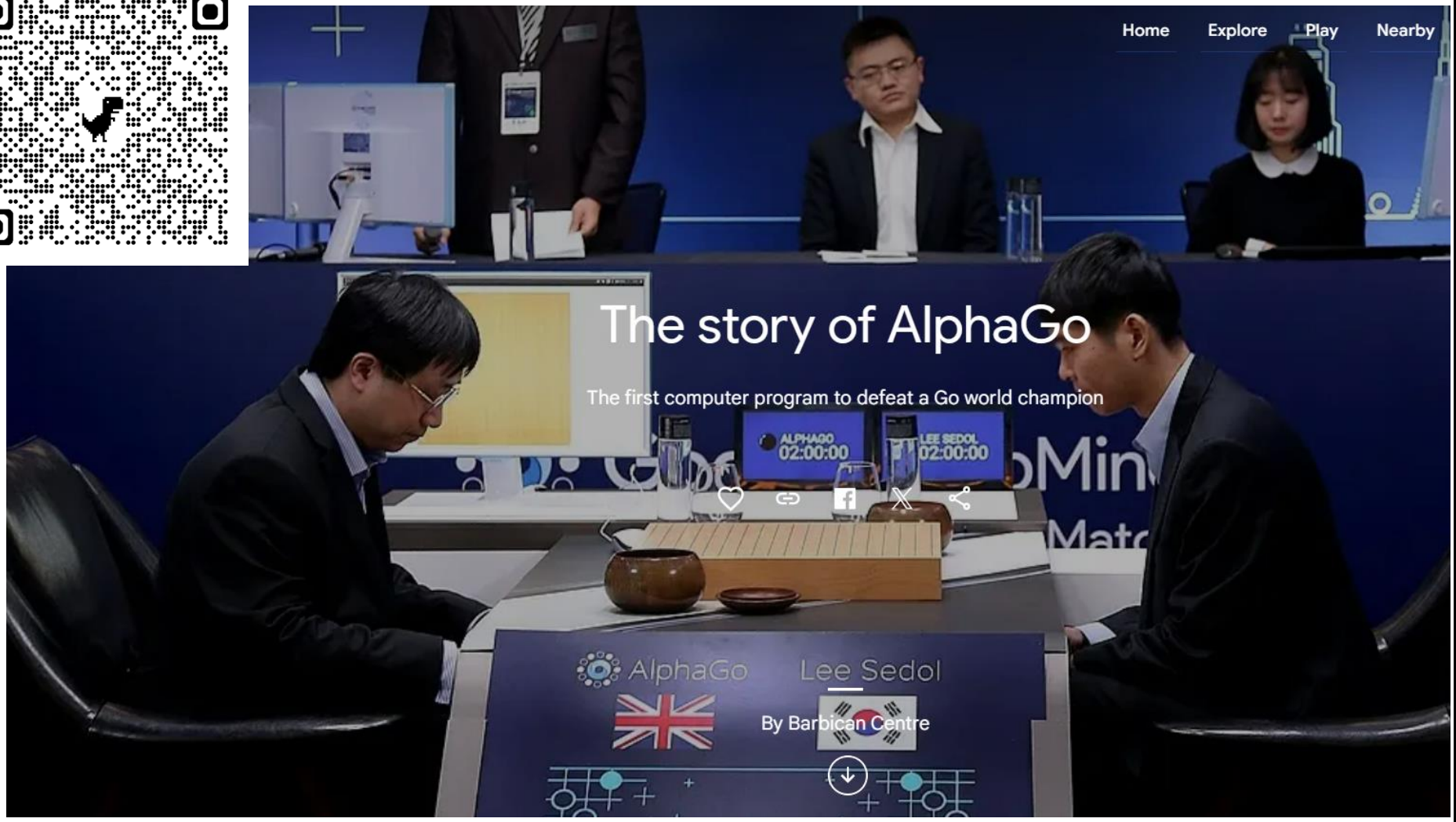
- 4.1 Giới thiệu học tăng cường
  - 4.1.1 Giới thiệu bài toán học tăng cường
  - 4.1.2 Xây dựng môi trường và hàm mục tiêu
- 4.2 Các thuật toán học tăng cường
- 4.3 Ví dụ về học tăng cường

# 4.1 Giới thiệu học tăng cường

# Học tăng cường



# AlphaGo




https://artsandculture.google.com/story/the-story-of-alphago-barbican-centre/kQXBk0X1qEe5KA?hl=en&src_trk=em6698d04f252c73.215288571849908387

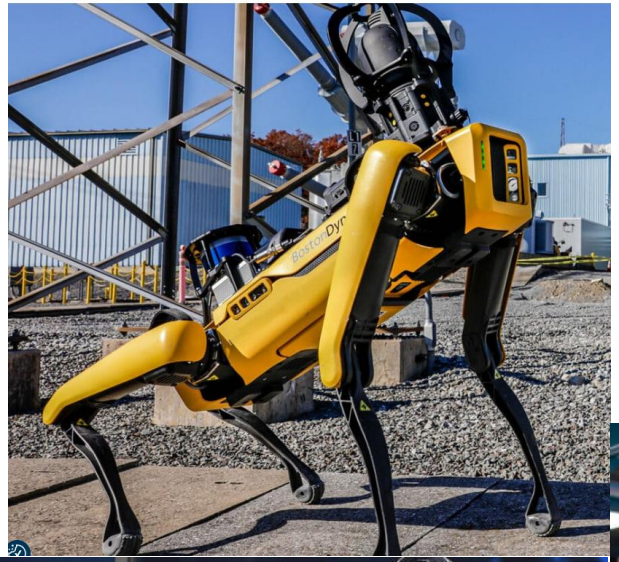
# Tesla

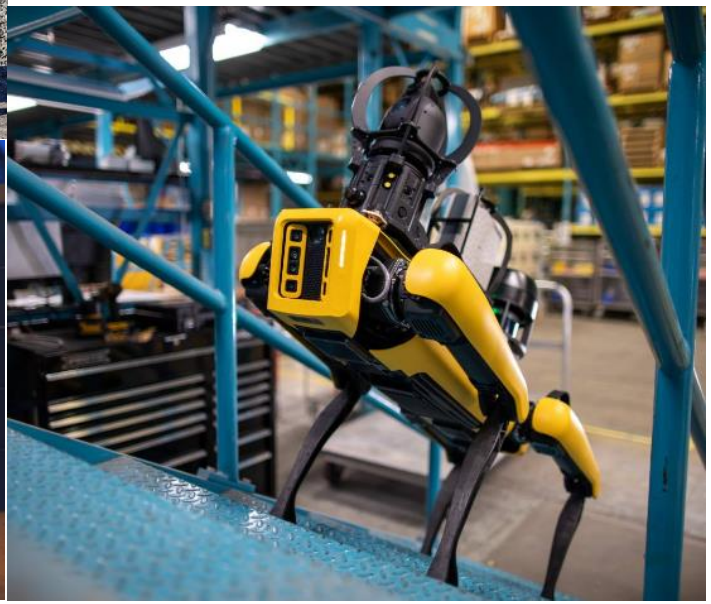
- **Tesla Autopilot and Full Self-Driving**


https://www.tesla.com/support/autopilot?src_trk=em6698d04f252c73.215288571849908387

# Boston Dynamics

- Robotics

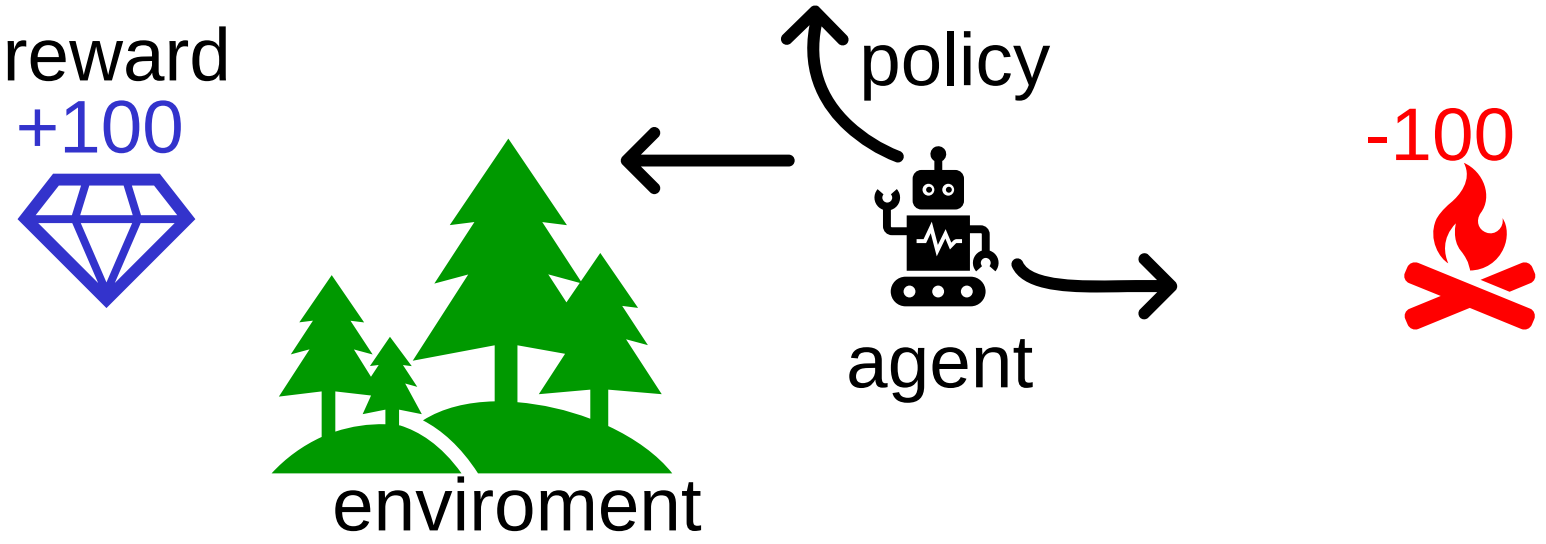




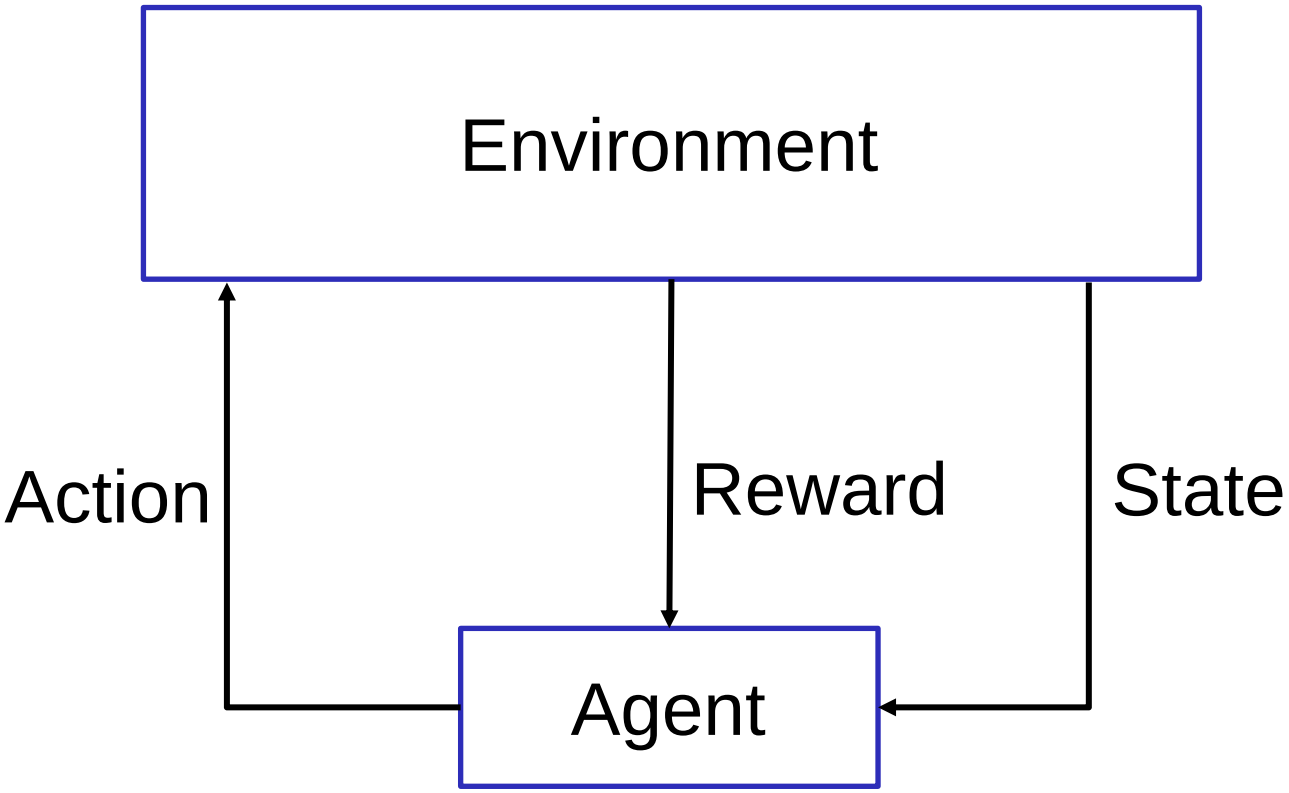
https://bostondynamics.com/?src_trk=em6698d04f252c73.215288571849908387

# Học tăng cường (RL)

- Học tăng cường là một cách tiếp cận tính toán nhằm để hiểu và tự động hóa quá trình học trực tiếp hướng tới mục tiêu và đưa ra quyết định



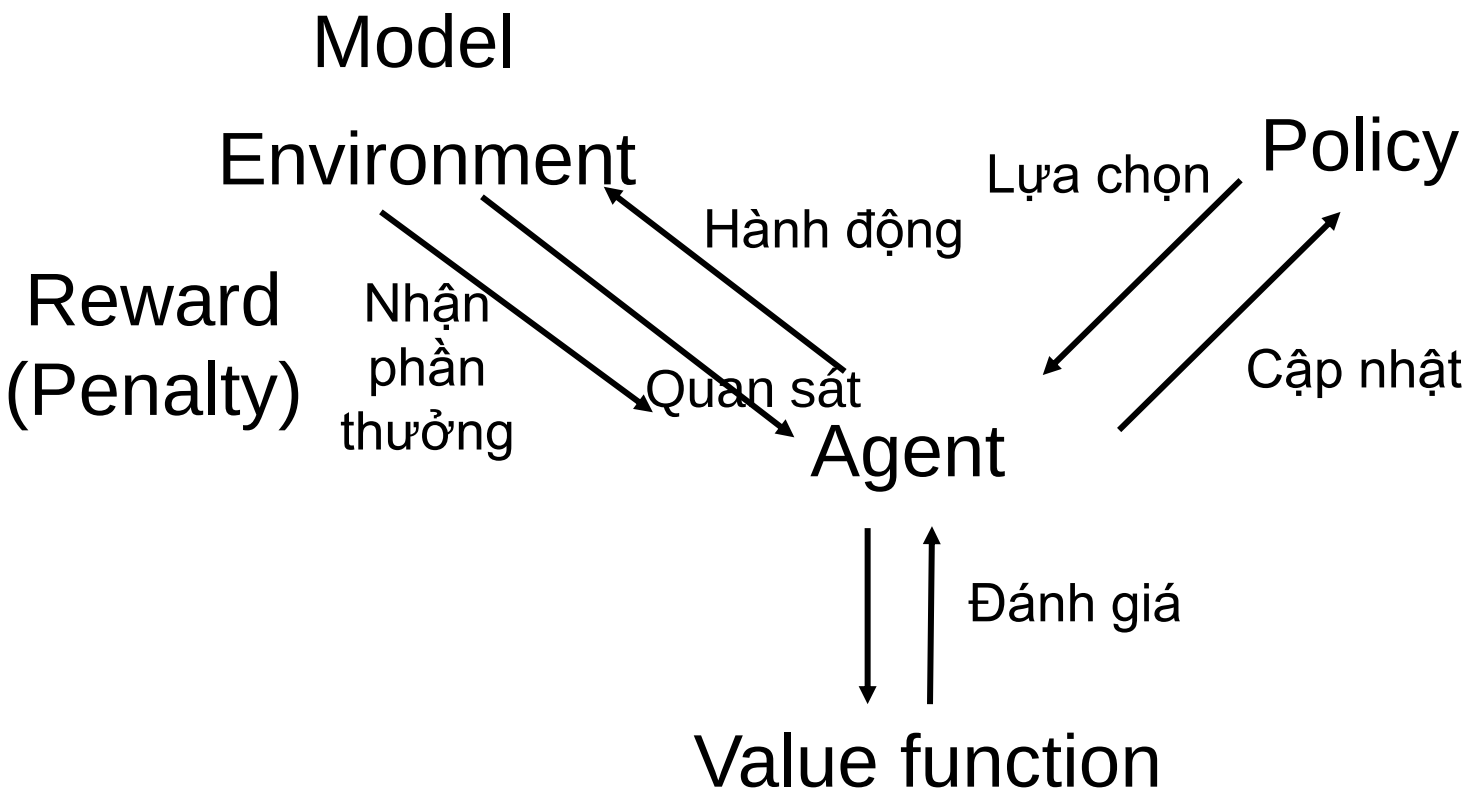
# Học tăng cường



# Học tăng cường

- Khác với các cách tiếp cận khác
  - RL nhấn mạnh vào quá trình học của 1 agent từ việc tương tác trực tiếp với môi trường không cần yêu cầu giám sát hoặc mô hình hoàn thiện của môi trường
  - RL là việc học điều cần làm
    - Bằng cách nào có thể ở một tình huống thực hiện hành động, sao cho tối đa hóa phần thưởng nhận được

# Học tăng cường



# Học tăng cường

- Policy
  - Định nghĩa cách hành xử của agent tại một thời điểm xác định
    - Với 1 trạng thái của môi trường thì những hành động nào cần thực hiện
  - Trong một số trường hợp policy là một hàm đơn giản hoặc một bảng tìm kiếm
  - Trong trường hợp khác policy có thể bao gồm việc tính toán mở rộng, như một quá trình tìm kiếm

# Học tăng cường

- Reward signal
  - Định nghĩa mục đích của vấn đề học tăng cường
  - Ở mỗi bước thời gian, môi trường gửi tới cho agent một số gọi là phần thưởng
    - Mục tiêu duy nhất của agent là tối đa hóa tổng phần thưởng nhận được trong một khoảng thời gian dài
    - Tín hiệu phần thưởng là phần cơ bản cho việc thay đổi policy
      - Nếu 1 hành động được lựa chọn bởi policy nhận được phần thưởng thấp => sau đó policy có thể được thay đổi để agent lựa chọn hành động khác khi ở tình huống tương tự trong tương lai

# Học tăng cường

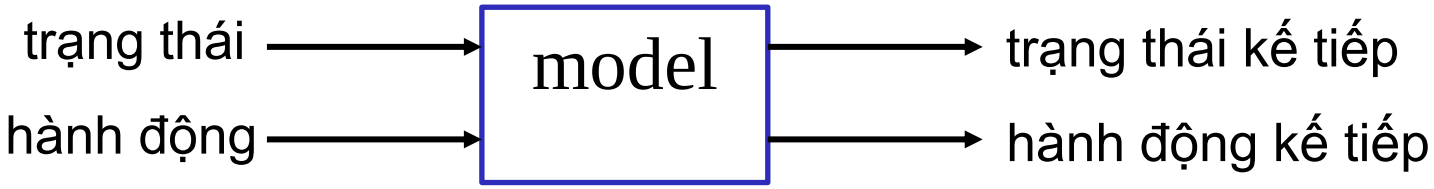
- Value function
  - Cho biết điều gì tốt trong khoảng thời gian dài
  - Giá trị của 1 trạng thái là tổng phần thưởng 1 agent mong đợi thu thập được trong tương lai, bắt đầu tính từ trạng thái này
  - Việc lựa chọn hành động được thực hiện dựa trên đánh giá giá trị
    - Tìm kiếm các hành động mang tới các trạng thái có giá trị cao nhất

# Học tăng cường

- Value function
  - Việc xác định value khó hơn rất nhiều việc xác định reward
    - Reward về cơ bản được đưa ra trực tiếp bởi môi trường
    - Value phải được ước lượng và tái đánh giá từ 1 chuỗi những quan sát mà agent làm trong toàn bộ thời gian tồn tại của nó
  - Một phần quan trọng trong hầu hết các thuật toán RL là 1 phương pháp ước lượng hiệu quả value

# Học tăng cường

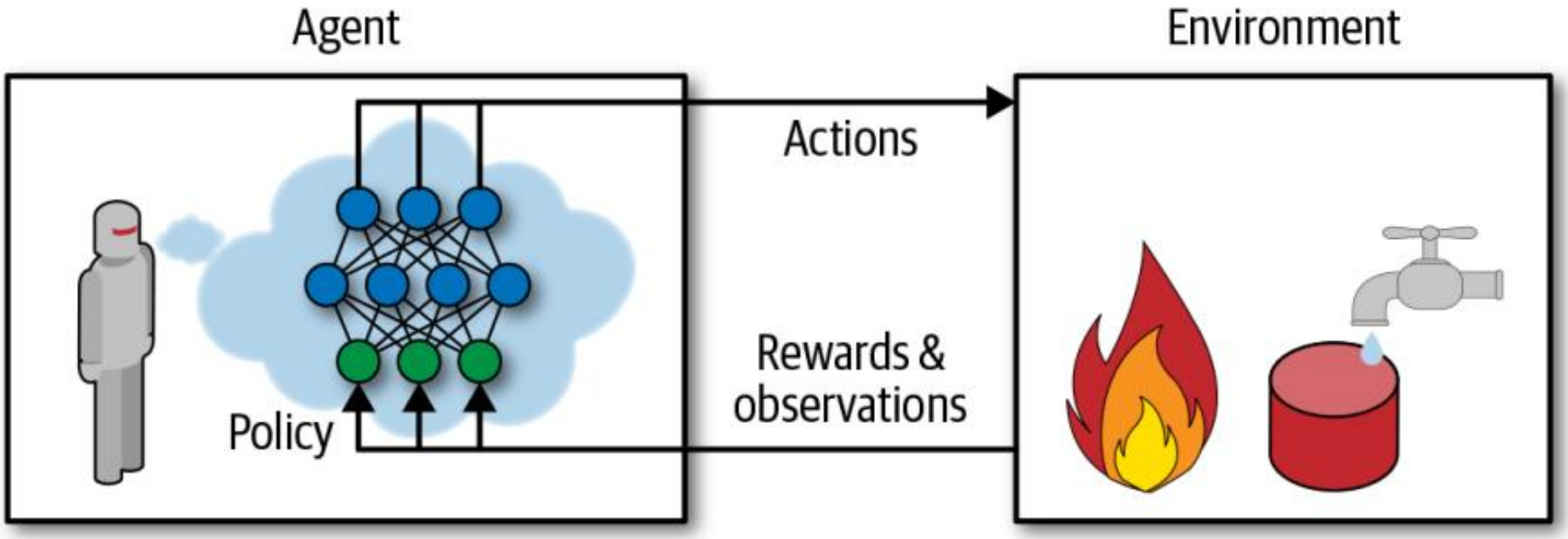
- Enviroment Model
  - Giả lập thái độ của môi trường, cho phép việc suy luận có thể được thực hiện về cách môi trường sẽ hoạt động như thế nào




  - Từ đó chia các phương pháp giải quyết các vấn đề RL ra
    - Các phương pháp dựa trên mô hình
    - Các phương pháp không dùng mô hình
      - Thử sai

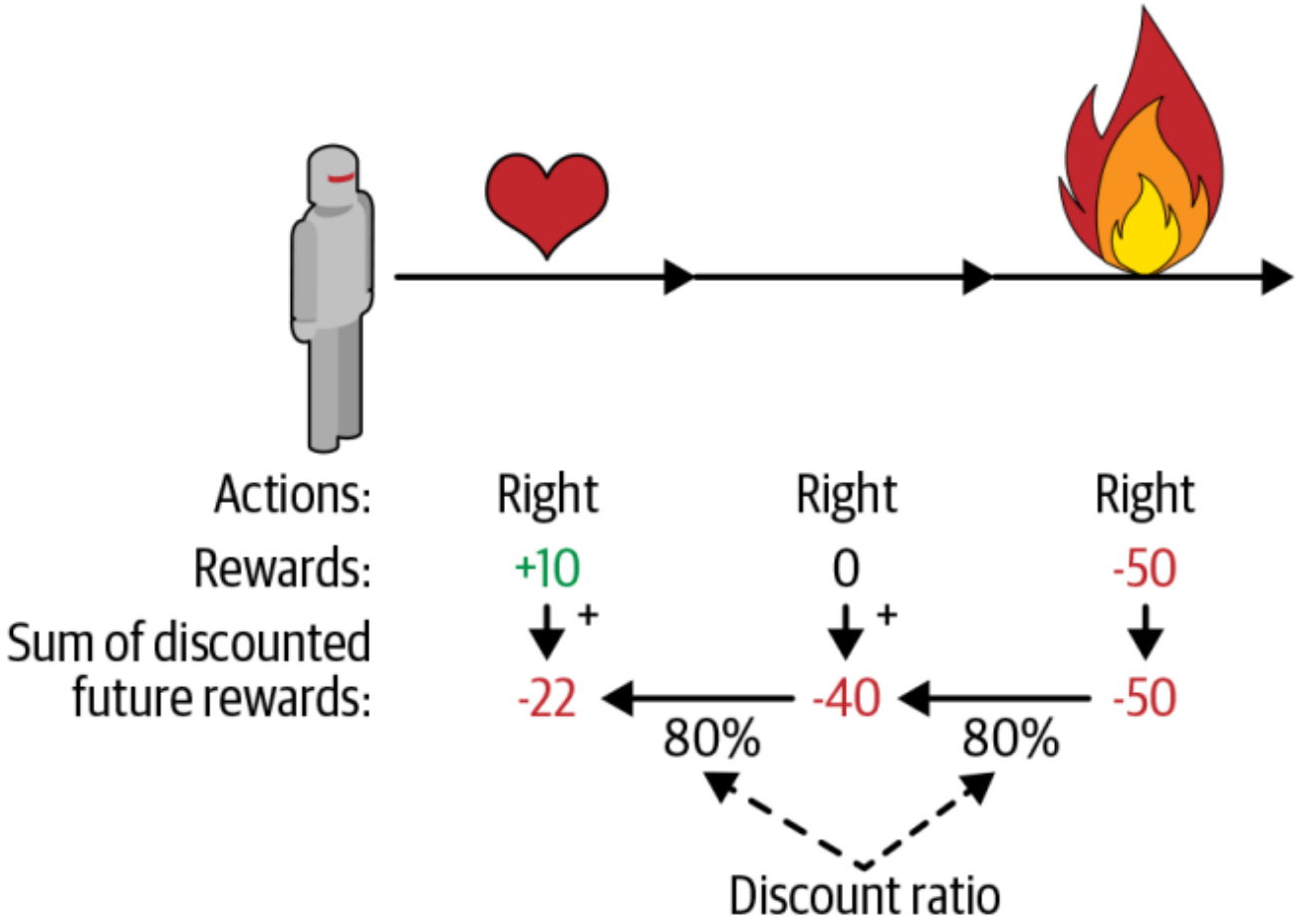
# 4.1.1 Mô tả bài toán học tăng cường

# Evaluating Actions



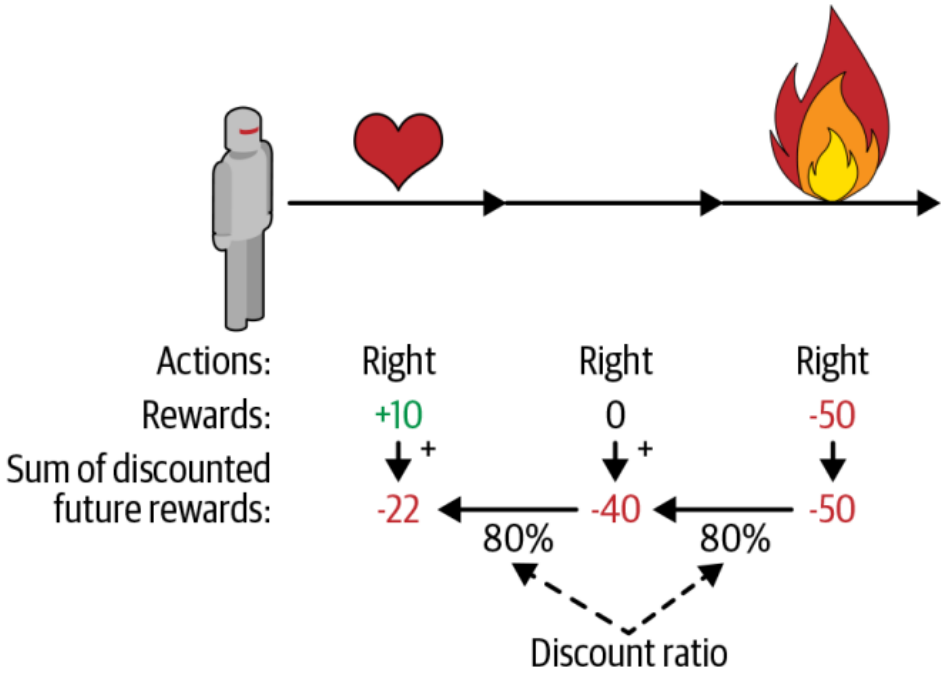
# Credit Assignment Problem

- Trong RL, hướng dẫn duy nhất mà agent nhận được là thông qua phần thưởng,
  - phần thưởng thường ít và chậm trễ.



# Credit Assignment Problem

- Khi agent nhận được phần thưởng, rất khó để biết những hành động nào nên được ghi nhận (hoặc bị đổ lỗi) cho phần thưởng đó
  - Một chiến lược phổ biến là đánh giá một hành động dựa trên tổng của tất cả các phần thưởng đi kèm sau đó, thường áp dụng với hệ số chiết khấu ở mỗi bước.
    - Tổng phần thưởng chiết khấu này được gọi là lợi nhuận của hành động.




$\gamma = 0,8$ discount factor

=> return

$$10 + \gamma \times 0 + \gamma^2 \times (-50) = -22$$

| | $\gamma = 0$ | $\gamma = 1$ |
|---|---|---|
| => return | 10 | -40 |

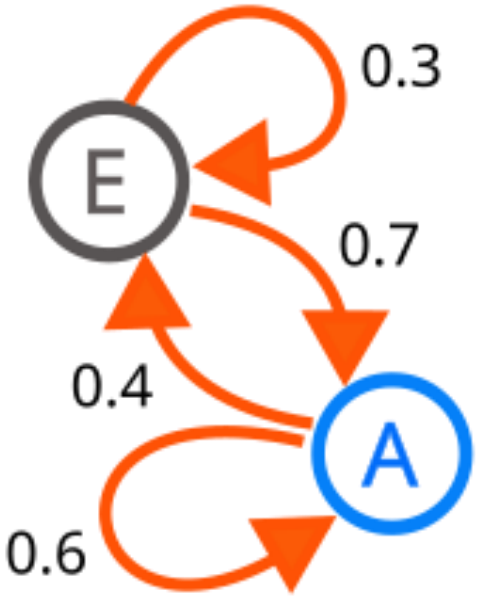
# Credit Assignment Problem

- Nếu hệ số chiết khấu gần bằng 0, thì phần thưởng trong tương lai sẽ không được tính nhiều so với phần thưởng tức thời
- Ngược lại, nếu hệ số chiết khấu gần bằng 1, thì phần thưởng trong tương lai xa sẽ được tính gần bằng phần thưởng tức thời
- Hệ số chiết khấu thông thường thay đổi từ 0,9 đến 0,99
  - Với hệ số chiết khấu là 0,95, phần thưởng sau 13 bước trong tương lai được tính gần bằng một nửa phần thưởng tức thời
  - Với hệ số chiết khấu là 0,99, phần thưởng sau 69 bước trong tương lai được tính bằng một nửa phần thưởng tức thời

$$0,95^{13} \approx 0,5 \qquad 0,99^{69} \approx 0,5$$

# Chuỗi Markov

Andrey Andreyevich Markov
(1856 –1922)




https://en.wikipedia.org/wiki/Andrey_Markov

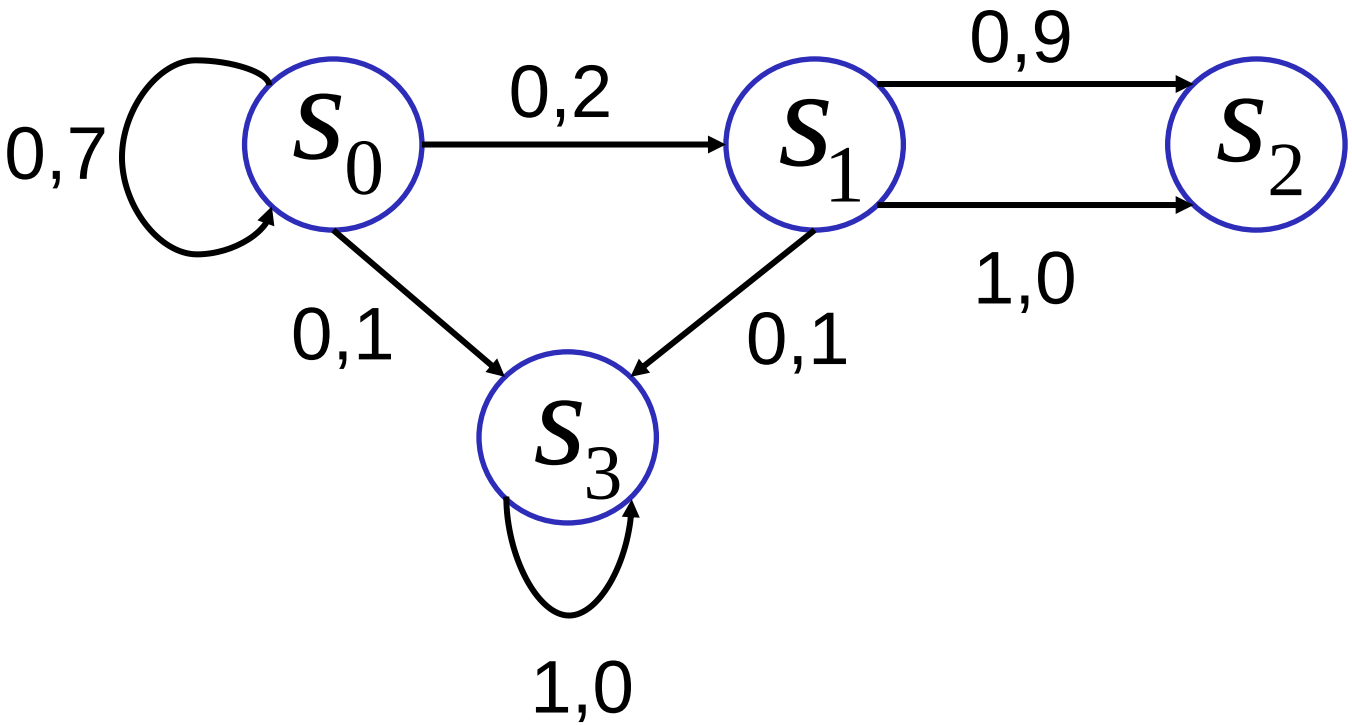
https://en.wikipedia.org/wiki/Markov_chain

# Chuỗi Markov

- Vào đầu thế kỷ 20, nhà toán học Andrey Markov đã nghiên cứu các quá trình ngẫu nhiên không có nhớ, được gọi là chuỗi Markov
  - Một quá trình như vậy có một số trạng thái cố định và nó tiến hóa ngẫu nhiên từ trạng thái này sang trạng thái khác ở mỗi bước
  - Xác suất để nó tiến hóa từ trạng thái *s* sang trạng thái *s′* là cố định và nó chỉ phụ thuộc vào cặp (*s*, *s′*), không phụ thuộc vào các trạng thái trong quá khứ (=> hệ thống không có nhớ)

# Chuỗi Markov

- Xem xét một chuỗi Markov với 4 trạng thái



# Chuỗi Markov

- Ví dụ: Khi Cj buồn, cô ấy sẽ đi chạy bộ, ăn kem hoặc ngủ trưa (từ dataset của Cj)

NEXT STATE

CURRENT STATE

| | SLEEP | RUN | ICE CREAM |
|---|---|---|---|
| SLEEP | 0.2 | 0.6 | 0.2 |
| RUN | 0.1 | 0.6 | 0.3 |
| ICE CREAM | 0.2 | 0.7 | 0.1 |

1/ Vẽ chuỗi Markov. Chú ý: Cần ghi rõ nhãn cho từng trạng thái (Sleep, Run, Ice Cream) và hiển thị xác suất chuyển đổi giữa các trạng thái

2/ Giả sử mỗi bước là một ngày. Tính xác suất Cj sẽ chuyển đến trạng thái “Run“ sau hai ngày buồn bã, biết rằng cô ấy bắt đầu ở trạng thái “Sleep“.

https://www.datacamp.com/tutorial/markov-chains-python-tutorial

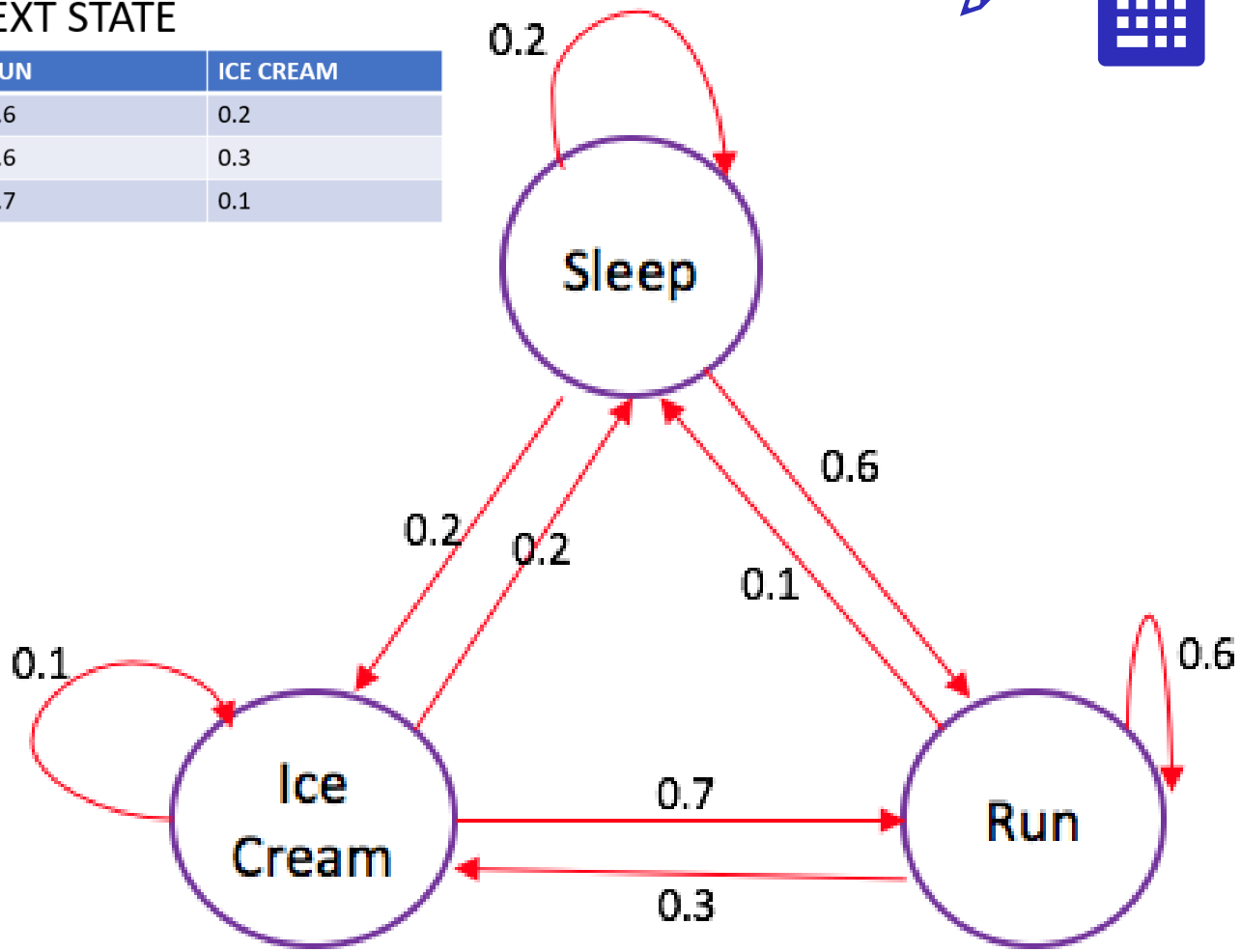
# Chuỗi Markov

- Ví dụ: Vẽ biểu diễn chuỗi Markov

CURRENT STATE

NEXT STATE

| | SLEEP | RUN | ICE CREAM |
|---|---|---|---|
| SLEEP | 0.2 | 0.6 | 0.2 |
| RUN | 0.1 | 0.6 | 0.3 |
| ICE CREAM | 0.2 | 0.7 | 0.1 |




https://www.datacamp.com/tutorial/markov-chains-python-tutorial

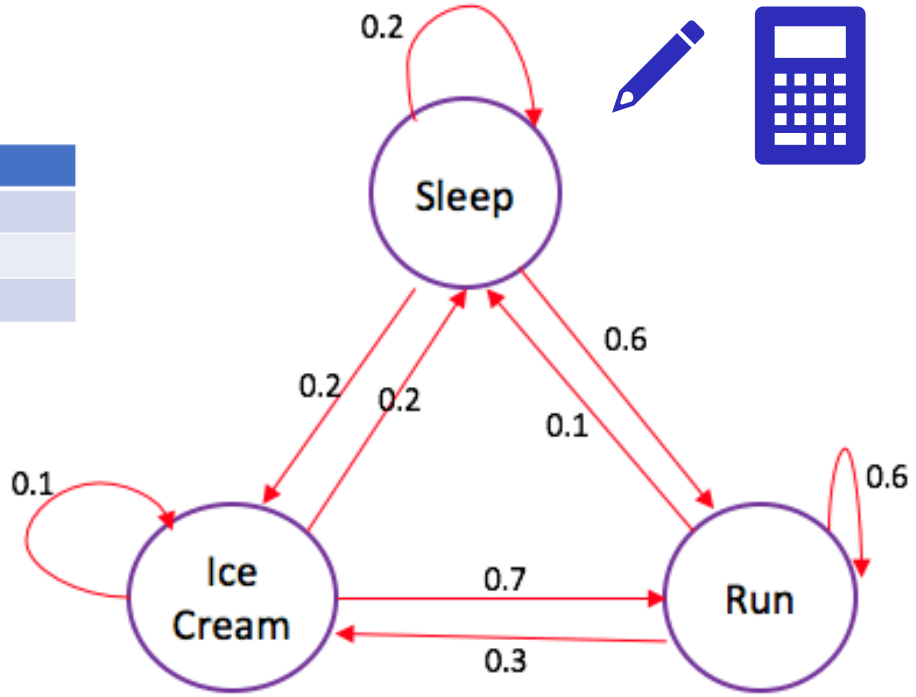
# Chuỗi Markov

- Ví dụ: Tính xác suất

CURRENT STATE

NEXT STATE

| | SLEEP | RUN | ICE CREAM |
|---|---|---|---|
| SLEEP | 0.2 | 0.6 | 0.2 |
| RUN | 0.1 | 0.6 | 0.3 |
| ICE CREAM | 0.2 | 0.7 | 0.1 |




bước 1 ~ ngày 1

Sleep => Sleep: 0,2

Sleep => Run: 0,6

Sleep => Ice Cream: 0,2

bước 2 ~ ngày 2

Sleep => Run: 0,6

Run => Run: 0,6

Ice Cream => Run: 0,7

Xác suất sau 2 ngày

Sleep => Run: **0,62**

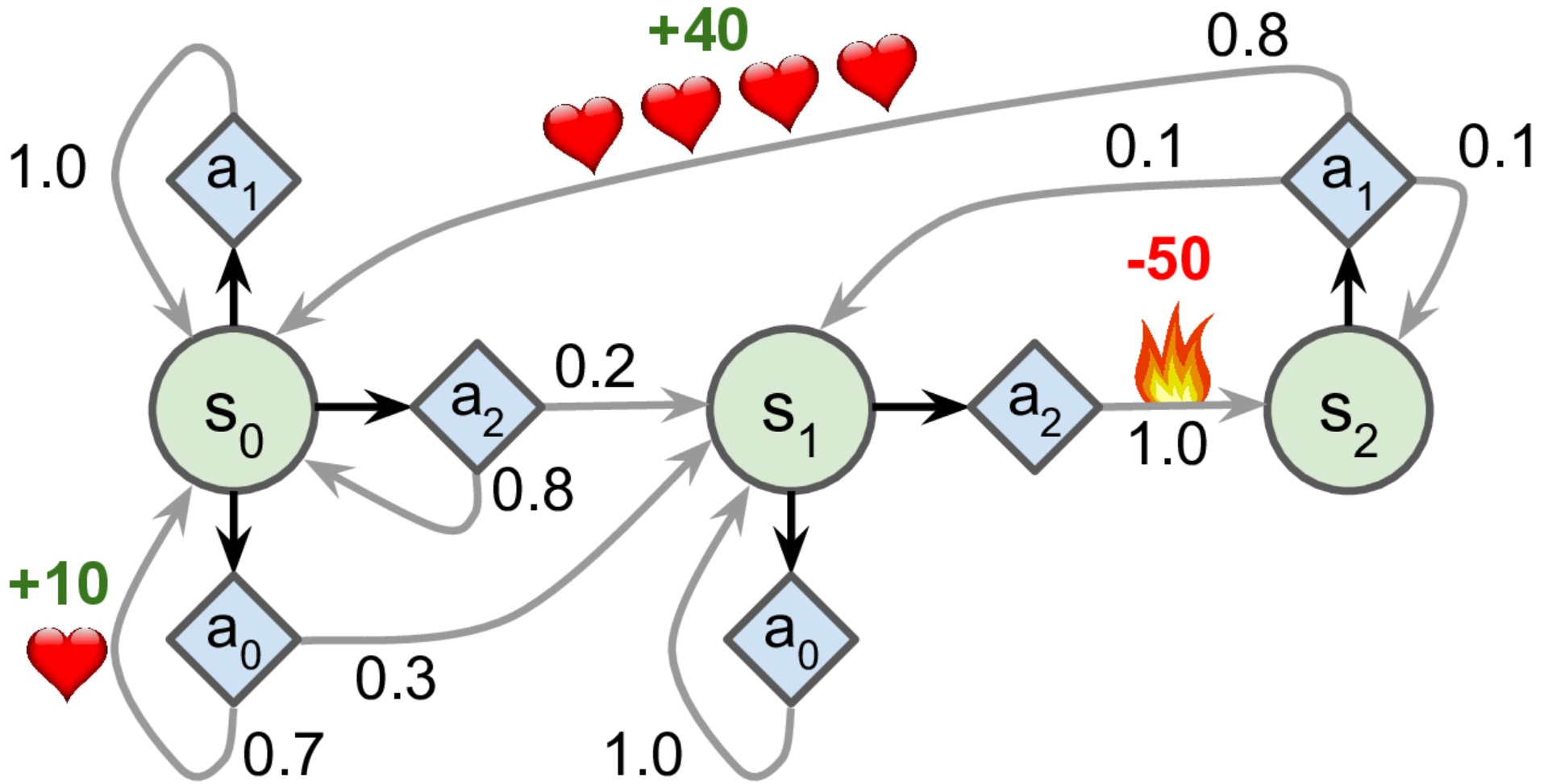
https://www.datacamp.com/tutorial/markov-chains-python-tutorial

# Markov Decision Process (MDP)

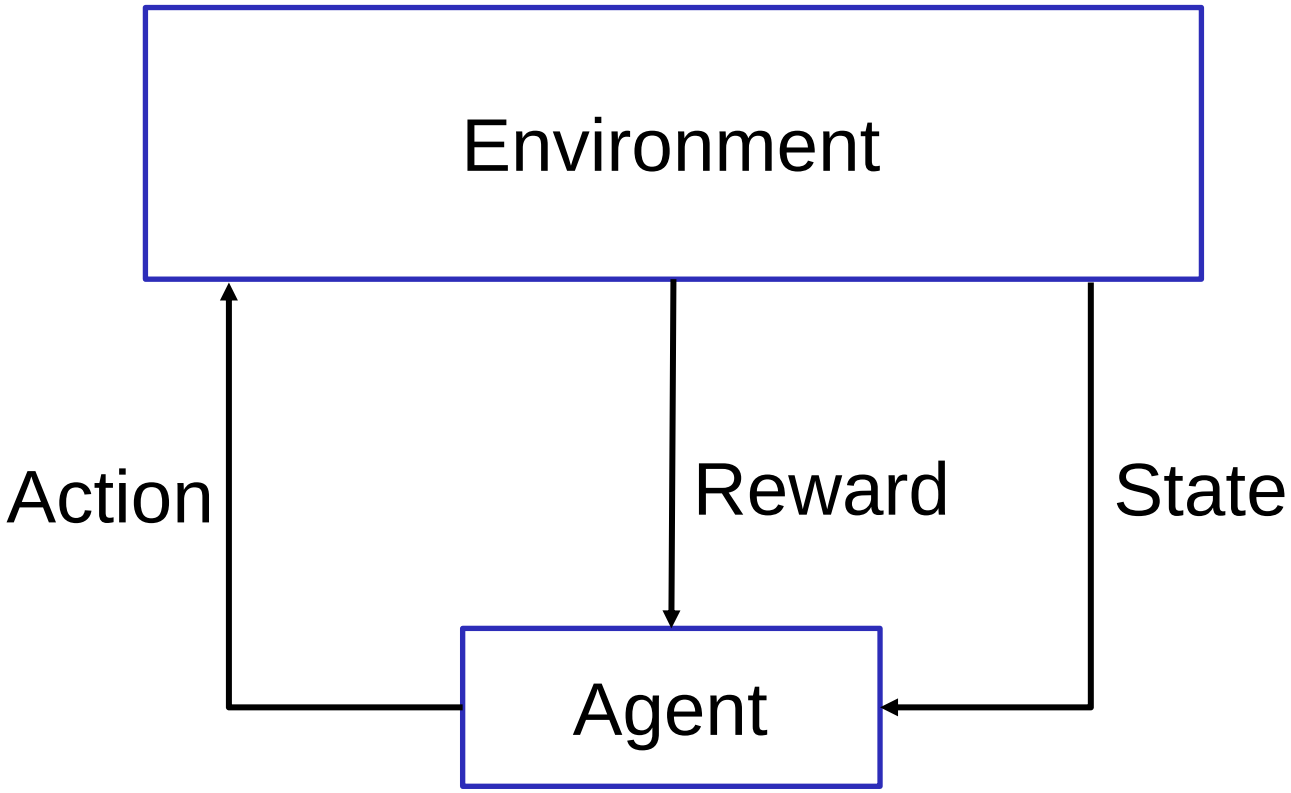
- Các quá trình quyết định Markov (MDP) lần đầu tiên được mô tả vào những năm 1950 bởi Richard Bellman
  - MDP giống với chuỗi Markov nhưng có điểm khác biệt:
    - Ở mỗi bước, một tác tử có thể chọn một trong một số hành động khả thi và xác suất chuyển đổi phụ thuộc vào hành động đã chọn
    - Hơn nữa, một số chuyển đổi trạng thái trả về một số phần thưởng (tích cực hoặc tiêu cực) và mục tiêu của tác tử là tìm ra một chính sách/chiến lược (policy) sẽ tối đa hóa phần thưởng theo thời gian

# Markov Decision Process

- Ví dụ: Xét một MDP, có thể đoán được chiến lược nào sẽ mang lại phần thưởng lớn nhất theo thời gian không?



# 4.1.2 Xây dựng môi trường và hàm mục tiêu



# Bellman optimality equation

- Bellman đã tìm ra cách để ước tính giá trị trạng thái tối ưu của bất kỳ trạng thái *s* nào, ký hiệu và $V^*(s)$
  - $V^*(s)$ là tổng của tất cả các phần thưởng trong tương lai được chiết khấu mà tác tử có thể mong đợi trung bình sau khi đạt đến trạng thái *s*, giả sử rằng tác tử hoạt động tối ưu

Richard Ernest Bellman
(1920- 1984)

https://en.wikipedia.org/wiki/Richard_E._Bellman

# Bellman optimality equation

- Phương trình tối ưu Bellman

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

cho tất cả các trạng thái *s*

$T(s,a,s')$ là xác suất chuyển đổi từ trạng thái *s* sang trạng thái *s'*, với điều kiện là tác tử đã chọn hành động *a*

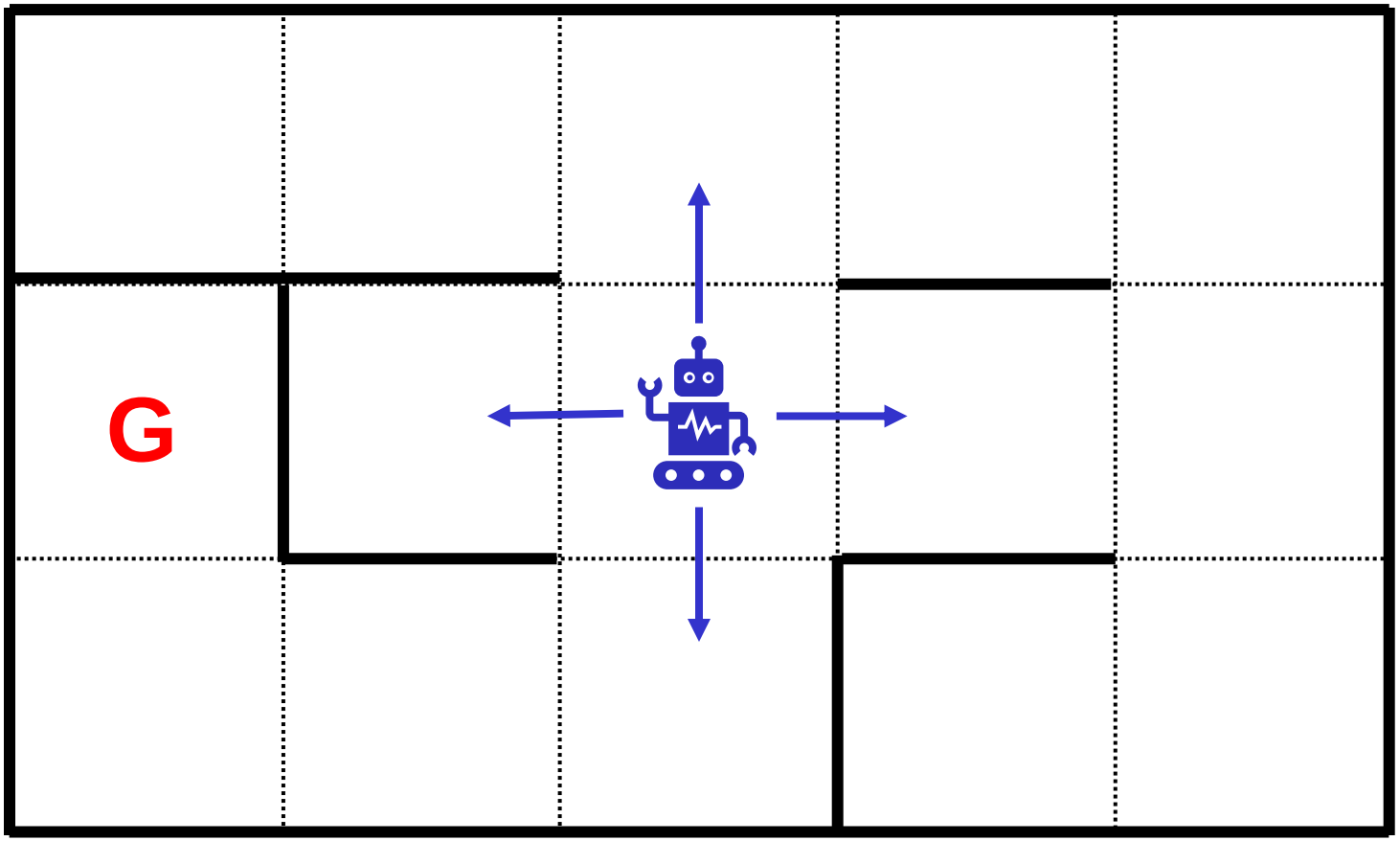
$R(s,a,s')$ là mà tác tử nhận được khi chuyển từ trạng thái *s* sang trạng thái *s'*, với điều kiện là tác tử đã chọn hành động *a*

$\gamma$ là hệ số chiết khấu

=> nếu tác tử hành động tối ưu, thì giá trị tối ưu của trạng thái hiện tại bằng với phần thưởng mà nó sẽ nhận được trung bình sau khi thực hiện một hành động tối ưu, cộng với giá trị tối ưu dự kiến của tất cả các trạng thái tiếp theo có thể mà hành động này có thể dẫn đến

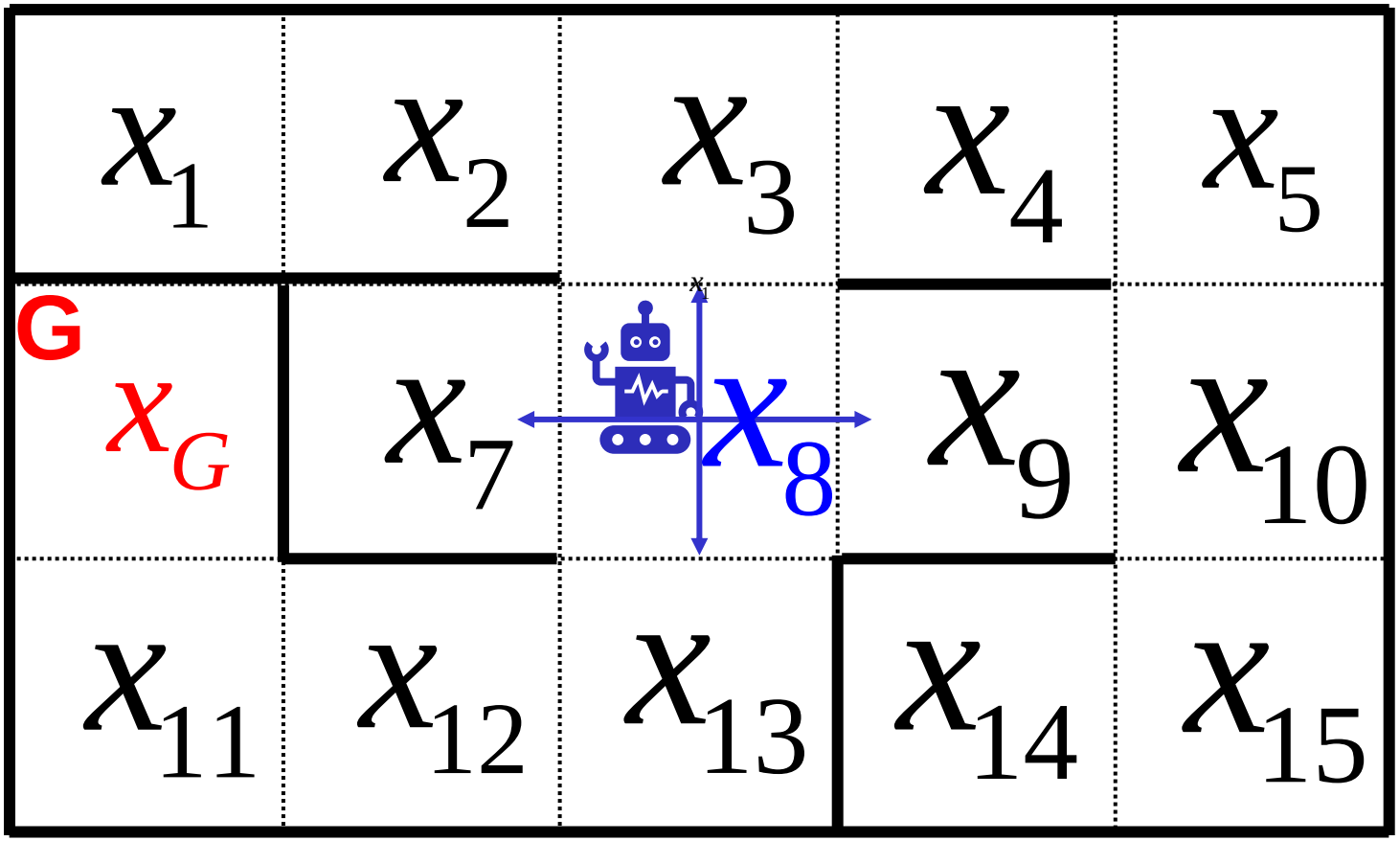
# Ví dụ

- Ví dụ: Bài toán robot di chuyển trong mê cung
  - Xác định: môi trường, trạng thái, tác tử, hành động, phần thưởng
  - Di chuyển đến đích G

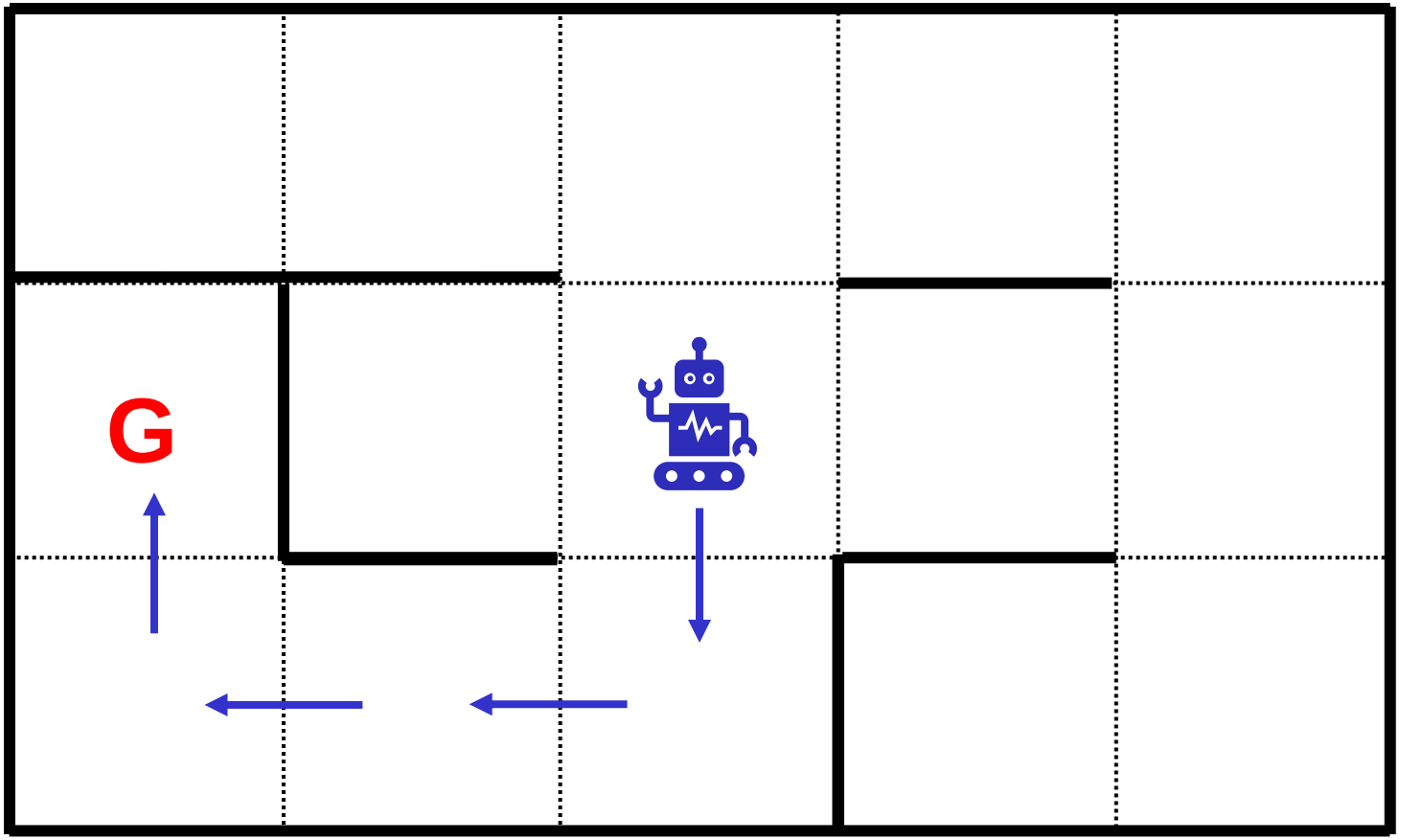


# Ví dụ

- Ví dụ: Bài toán robot di chuyển trong mê cung
  - Xác định: môi trường, trạng thái, tác tử, hành động, phần thưởng
  - Di chuyển đến đích G



# Ví dụ



# Thảo luận 1

- Sinh viên thảo luận về những ưu điểm và nhược điểm của học tăng cường

# Thảo luận 2

- Sinh viên thảo luận về
  - Trong những trường hợp nào chúng ta nên sử dụng học tăng cường
  - Trong những trường hợp nào chúng ta không nên sử dụng học tăng cường

# Tổng kết

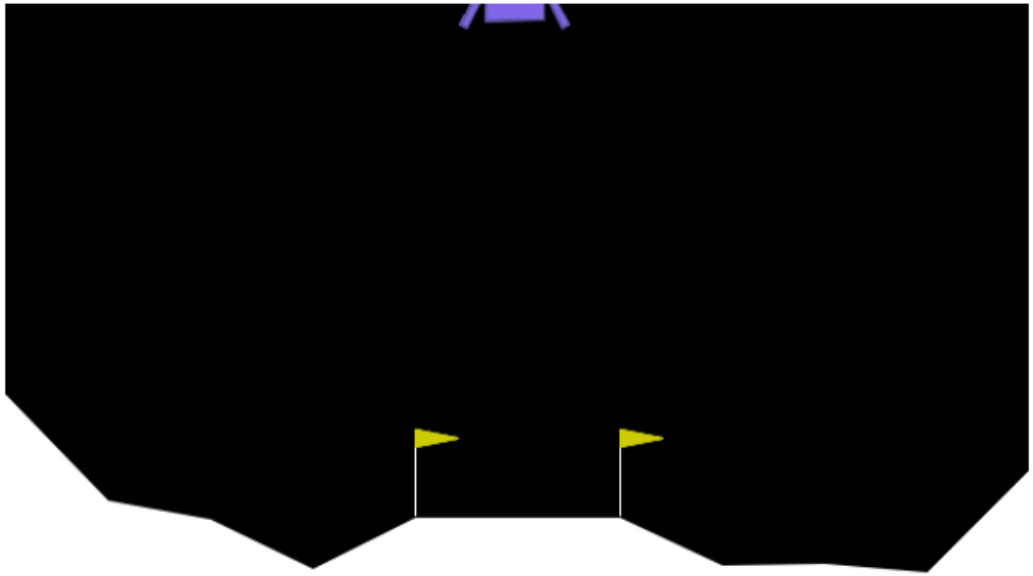
- Sinh viên nắm được học tăng cường và ứng dụng thực thế
- Từ ưu điểm và nhược điểm của học tăng cường, sinh viên biết cách lựa chọn:
  - trong những trường hợp nào nên sử dụng học tăng cường
  - và trong những trường hợp nào không nên sử dụng học tăng cường

# Hoạt động sau buổi học

- Cài đặt và tìm hiểu OpenAI Gym

Gymnasium

**An API standard for reinforcement learning with a diverse collection of reference environments**



# Chuẩn bị cho buổi học tiếp theo

- Sinh viên tìm hiểu về các thuật toán học tăng cường

# Tài liệu tham khảo

- Markov Chains explained visually
- https://gymnasium.farama.org/