1. **Hiểu vì sao "Exploring Starts" (khởi tạo ngẫu nhiên cặp trạng thái-hành động) có thể gặp vấn đề trong các bài toán thực tế.**
    - Tức là nhận ra hạn chế của việc muốn cho agent bắt đầu mỗi episode từ mọi cặp (s, a) khả dĩ.
2. **Mô tả một phương pháp thay thế cho "Exploration" (khám phá) trong thuật toán Monte Carlo control.**
    - Tức là giới thiệu cách dùng chính sách "epsilon-soft" (đặc biệt là epsilon-greedy) để đảm bảo agent vẫn khám phá đủ khi cập nhật giá trị.

---

2. Tại sao "Exploring Starts" có thể không khả thi trong thực tế

2.1. Khái niệm "Exploring Starts"

- Trong Monte Carlo control, **Exploring Starts (ES)** là kỹ thuật bắt đầu mỗi episode bằng cách chọn ngẫu nhiên (uniform random) một **cặp trạng thái s_0 và hành động a_0**. Mục đích của ES là để đảm bảo:
    1. Mỗi cặp (s, a) trong không gian trạng thái–hành động đều có xác suất được thử ít nhất một lần.
    2. Từ đó, khi thu thập Return, ta luôn có dữ liệu cho mọi (s, a) để ước lượng $Q(s, a)$.
- Tuy nhiên, ES có **hai nhược điểm lớn** khi áp dụng vào bài toán thực:
    1. **Phải biết trước toàn bộ không gian trạng thái S và không gian hành động A.** Trong nhiều bài toán phức tạp (ví dụ, tự lái xe, robot, hay game có tiềm cơ vô hạn), rất khó để "liệt kê" tất cả kết hợp (s, a).
    2. **Khó khăn (và tốn kém) khi muốn cho agent thực sự khởi động từ một trạng thái "tùy ý".** Ví dụ: nếu đang huấn luyện một ô-tô tự lái, để thử nghiệm trạng thái "xe đang ở giữa đường", chúng ta phải dừng xe, đặt nó ở vị trí đó và reset tất cả cảm biến — điều này vừa kém an toàn vừa tốn thời gian, và càng không khả thi nếu trạng thái là "xe đang quay đầu, xung quanh có nhiều xe khác".

2.2. Ví dụ minh họa (self-driving car)

- Slide đưa ví dụ về **ô-tô tự lái (self-driving car)**:
    - Nếu muốn áp dụng ES, ta phải "ngẫu nhiên" chọn một trạng thái đầu ví dụ như: "Xe đang ở làn số 2, tốc độ 60 km/h, có chiếc xe khác cách 3 mét phía trước, v.v."
    - Sau đó, ta phải cho xe thực thi ngay một hành động nào đó (throttle, brake, rẽ trái, v.v.) từ trạng thái này.
    - Trong thực tế, để làm được thế, ta phải "đặt" chiếc xe vào đúng vị trí, đúng tốc độ, đúng hoàn cảnh giao thông—việc này không chỉ phức tạp mà còn rất **nguy hiểm** (vì xe có thể đang ở giữa đường đông), đồng thời **không thực tế** (đòi hỏi nhân lực, chi phí, và rủi ro quá cao).

Kết luận: “Exploring Starts” tuy đơn giản ở mặt ý tưởng (chọn ngẫu nhiên (s, a)) nhưng trong nhiều bài toán thực sự không làm được, hoặc nếu làm được thì rất tốn kém, nguy hiểm.

---

3. Cần tìm phương pháp khám phá khác (Exploration without Exploring Starts)

Slide đặt câu hỏi:

> **"Làm sao chúng ta có thể học tất cả các giá trị hành động (action values) mà không dùng Exploring Starts?"**

Câu trả lời được gợi ý: **sử dụng ý tưởng từ bài toán Bandit kết hợp Monte Carlo — cụ thể là chính sách epsilon-greedy (ε-greedy).**

3.1. Nhắc lại “Bandit với Monte Carlo”

- Trong bài toán **multi-armed bandit**, agent phải chọn một trong k “cánh tay” mỗi bước, và nhận payoff (reward).
- Để ước lượng giá trị kỳ vọng của mỗi arm, ta thu thập kết quả mỗi khi chọn arm đó, rồi tính trung bình. Tuy nhiên, nếu chỉ chọn arm có giá trị cao nhất (greedy), ta có nguy cơ **bỏ qua** các arm khác có tiềm năng chưa biết.
- Cách giải: dùng **ε-greedy** — tức là:
  - Với xác suất $1 - \varepsilon$, chọn arm greedy (có giá trị ước lượng cao nhất).
  - Với xác suất ε, chọn ngẫu nhiên một arm bất kỳ.

  Phương pháp này đảm bảo dù giá trị ước lượng đang nghiêng về một arm nhất định, agent vẫn **thỉnh thoảng** thử một arm khác để thu thập thêm dữ liệu.

3.2. Áp dụng cho Reinforcement Learning (Monte Carlo Control)

- Tương tự, khi agent đang ước lượng hàm $Q(s, a)$ cho mỗi trạng thái s và hành động a, nếu chỉ “greedy” (luôn chọn $a = \arg\max Q(s, a)$), thì một số hành động ở một trạng thái nhất định có thể **không bao giờ được thử** (vì chính sách luôn bỏ qua chúng). Khi đó, ta không thu được return để cập nhật Q cho những hành động đó, dẫn đến hội tụ vào **ngăn xếp con tối ưu cục bộ** (local optimum) chứ không phải global optimum.
- Giải pháp:
  - Sử dụng **ε-greedy** (hoặc rộng hơn là **ε-soft**) để đảm bảo agent vẫn **khám phá** hành động có giá trị ước lượng chưa cao, dù tần suất thấp.
  - Với ε-greedy trong RL:
    1. Tại trạng thái s, liệt kê tất cả hành động $a_1, a_2, \ldots, a_n$.
    2. Với xác suất $1 - \varepsilon$, chọn **hành động greedy** $a^* = \arg\max_a Q(s, a)$.

3. Với xác suất ε, chọn **ngẫu nhiên** một hành động trong toàn bộ tập $A(s)$ (có thể đều nhau).

- Do đó, mỗi hành động luôn có xác suất ít nhất $\varepsilon/|A(s)|$ được chọn tại bất cứ trạng thái nào.

---

4. Epsilon-Soft Policies (Chính sách "ε-mềm")

4.1. Định nghĩa

- **Chính sách ε-soft** là một lớp các chính sách ngẫu nhiên (stochastic), trong đó **mọi hành động** tại một trạng thái đều có **xác suất** được chọn **ít nhất** bằng $\varepsilon/|A(s)|$.
- Nói cách khác, ở mỗi trạng thái s, với tổng số hành động là n, thì

$$\pi(a \mid s) \geq \frac{\varepsilon}{n}, \quad \forall\, a \in A(s).$$

  Và phần còn lại (tức $1 - \varepsilon$) sẽ được dành cho việc chọn "greedy action" (hành động có giá trị ước lượng cao nhất).
- **Trái ngược** với "deterministic policy" (chỉ cho phép một hành động duy nhất có xác suất 1), "stochastic policy" (như ε-soft) cho phép đa hành động với các xác suất khác nhau, nhưng tất cả đều ≥ ε/n.

4.2. Ví dụ trong Grid World

Hãy tưởng tượng một Grid World (môi trường dạng lưới) với các ô, mỗi ô có 4 hành động khả dĩ: đi lên, đi xuống, đi trái, đi phải.

- **Chính sách deterministic**:
  - Ở mỗi ô, chính sách xác định sẵn một hành động duy nhất (ví dụ luôn đi sang phải). Nếu cứ chạy theo policy này, agent sẽ luôn đi theo một đường cố định và **không bao giờ** khám phá các đường khác.
- **Chính sách ε-greedy** (thuộc lớp ε-soft):
  - Ở mỗi ô, nếu ta định nghĩa giá trị ước lượng Q, ví dụ:
    - $Q(s, \uparrow) = 1.0,\ Q(s, \rightarrow) = 0.8,\ Q(s, \downarrow) = 0.5,\ Q(s, \leftarrow) = 0.2.$
  - Với $\varepsilon = 0.1$ và $n = 4$:
    - Xác suất chọn action ↑ (greedy) = $1 - 0.1 + 0.1/4 = 0.925.$
    - Mỗi action khác (→, ↓, ←) đều có xác suất $0.1/4 = 0.025.$
  - Nhờ đó, dù agent chủ yếu chọn action "lên" (↑), nó vẫn có một tỉ lệ nhỏ (2.5%) "thử" các hướng khác, để phát hiện con đường tốt hơn nếu có.
- **Hệ quả**:

- Mỗi episode, agent sẽ thường đi theo hướng "greedy", nhưng thỉnh thoảng "lạc chỗ" một chút do chọn ngẫu nhiên. Kết quả là nó sẽ có đường đi hơi khác nhau, và dần dần thu thập thêm dữ liệu về giá trị của những action chưa từng chọn.

---

5. Sơ đồ tổng hợp quy trình Monte Carlo Control với ε-greedy

Mặc dù slide không đưa thuật toán chi tiết, nhưng có thể tóm tắt quy trình Monte Carlo Control (thay thế Exploring Starts bằng ε-greedy) như sau:

1. **Khởi tạo**
   - Cho một policy π ban đầu (có thể là hoàn toàn ngẫu nhiên) sao cho π là ε-soft. Ví dụ: ε-greedy với $\varepsilon = 0.1$.
   - Khởi tạo ước lượng $Q(s,a)$ (thường đặt $Q(s,a) = 0$ cho mọi (s,a)).
   - Tạo cấu trúc lưu trữ:
     - `ReturnsSum(s,a) = 0`,
     - `ReturnsCount(s,a) = 0`,
       cho mọi (s,a).
2. **Lặp qua nhiều episode**
   - **Sinh episode**:
     - Bắt đầu từ trạng thái ban đầu (được môi trường định nghĩa).
     - Tại mỗi bước *t*: chọn action A_t theo chính sách ε-greedy dựa vào Q hiện tại:

$$\text{chọn } A_t = \begin{cases} \arg\max_a Q(S_t,a), & \text{với xác suất } 1-\varepsilon, \\ \text{random action}, & \text{với xác suất } \varepsilon. \end{cases}$$

     - Quan sát reward R_{t+1} và trạng thái kế tiếp S_{t+1}. Tiếp tục cho đến khi đạt trạng thái terminal, tạo ra toàn bộ episode:
       $(S_0, A_0, R_1,\ S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T, S_T)$.
   - **Tính Return** cho mỗi bước t trong episode:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T.$$

   - **Cập nhật giá trị hành động**:
     - Duyệt qua tất cả các cặp (S_t, A_t) trong episode (thường dùng **first-visit** – chỉ cập nhật khi (S_t, A_t) xuất hiện lần đầu trong episode đó).
     - Cho mỗi cặp (s,a) đó:
       1. `ReturnsSum(s,a) += G_t`,
       2. `ReturnsCount(s,a) += 1`,
       3. Cập nhật $Q(s,a) = ReturnsSum(s,a)/ReturnsCount(s,a)$.
   - **Cải thiện chính sách** cho mọi trạng thái đã xuất hiện trong episode:

- Đổi $\pi(s)$ thành hàm ε-greedy dựa trên ước lượng $Q(s, \cdot)$ vừa mới cập nhật.

3. **Lặp** cho đến khi Q và π hội tụ (không còn thay đổi đáng kể) hoặc đủ số episode.

Nhờ ε-greedy, dù chính sách có thiên về giá trị hiện tại, nhưng vẫn đảm bảo **mọi hành động đều được thử** với tần suất ít nhất $\varepsilon/|A(s)|$, nên về lâu dài $Q(s, a)$ sẽ được ước lượng chính xác hơn.

---

6. Tóm tắt (Summary)

Cuối slide, phần tóm tắt nhấn lại:

- **Vì sao Exploring Starts có thể không khả thi trong các bài toán thực tế?**
  - Khó "khởi tạo ngẫu nhiên" cho mọi cặp (s, a).
  - Điều kiện khắc nghiệt, nguy hiểm, hoặc tốn kém (ví dụ self-driving car).
- **Giải pháp thay thế: chính sách ε-soft (đặc biệt là ε-greedy).**
  - Đảm bảo agent **vẫn khám phá** các hành động chưa được thử, nhưng không phải khởi tạo từ mọi (s, a).
  - Mỗi hành động có xác suất ≥ $\varepsilon/|A|$ để được chọn, từ đó dần thu thập đủ dữ liệu cho $Q(s, a)$.
- Với **chính sách ε-greedy**, Monte Carlo Control vẫn có thể **ước lượng và cập nhật** $Q(s, a)$ rồi cải thiện policy (Generalized Policy Iteration), mà không cần thế "Exploring Starts" phức tạp.