Solution Bài 3

1. Phương pháp thiết kế thuật toán

Phương pháp tổng thể áp dụng cho bài toán này là **greedy**, với ý tưởng chọn bước đi sao cho chi phí hiện tại là nhỏ nhất tại mỗi trạng thái. Trong bài toán, Dijkstra được xem như một dạng greedy mở rộng để giải quyết bài toán chi phí tối thiểu trên đồ thị trọng số.

Dijkstra mở rộng được áp dụng như sau:

1. **Khởi tạo:** Mỗi Pokémon có nhiều thuộc tính, do đó mỗi trạng thái được định nghĩa là (Pokémon, thuộc tính). Ma trận chi phí $dist[i][j]$ được khởi tạo vô cùng lớn, ngoại trừ trạng thái bắt đầu $dist[1][j] = 0$ cho tất cả thuộc tính j của Pokémon đầu tiên.

2. **Sử dụng priority queue:** Tạo priority queue lưu trữ các tuple (chi phí, Pokémon, thuộc tính). Lúc nào cũng chọn trạng thái có chi phí nhỏ nhất ra xử lý tiếp.

3. **Cập nhật trạng thái kế tiếp và unlock:** Từ trạng thái hiện tại (x, t):

   - Di chuyển đến Pokémon tiếp theo theo cùng thuộc tính: chi phí bằng chi phí hiện tại cộng chênh lệch thuộc tính.
   - **Unlock Pokémon:** nếu Pokémon chưa được unlock, có thể trả chi phí unlock để mở khóa. Đây cũng là một hành động greedy vì luôn chọn cách mở khóa ngay khi có thể, nhằm cập nhật chi phí tối ưu cho các trạng thái tiếp theo.
   - Nếu chi phí mới nhỏ hơn giá trị trong $dist$, cập nhật và push vào queue.

4. **Lặp lại:** Tiếp tục chọn trạng thái có chi phí nhỏ nhất trong priority queue và cập nhật các trạng thái kế tiếp cho đến khi đạt Pokémon cuối cùng.

5. **Kết thúc:** Khi trạng thái Pokémon cuối cùng được xử lý, chi phí tối thiểu sẽ là giá trị nhỏ nhất trong các trạng thái tương ứng.

Dijkstra ở đây là **greedy cục bộ**, vì tại mỗi bước luôn chọn trạng thái có chi phí nhỏ nhất trước khi mở rộng sang các trạng thái khác, bao gồm cả hành động unlock để cập nhật chi phí tốt hơn.

—

2. Tính phù hợp của phương pháp

Phương pháp greedy kết hợp Dijkstra là phù hợp vì:

- Bài toán yêu cầu tìm chi phí tối thiểu từ Pokémon 1 đến Pokémon n, với chi phí giữa các trạng thái luôn dương.
- Mỗi Pokémon có nhiều thuộc tính, tạo ra nhiều trạng thái khác nhau. Dijkstra mở rộng giúp tính toán chi phí tối thiểu giữa tất cả các trạng thái này.
- Greedy cục bộ đảm bảo luôn chọn trạng thái có chi phí nhỏ nhất tại mỗi bước, từ đó mở rộng đến các trạng thái khác chính xác.
- Chi phí unlock Pokémon được cộng khi cần thiết, phù hợp với cơ chế greedy để đảm bảo chi phí tổng thể là tối thiểu.

—

3. Phân tích độ phức tạp

Độ phức tạp thời gian:

- Gọi n là số Pokémon, m là số thuộc tính.
- Mỗi Pokémon có m trạng thái, tổng số trạng thái là $O(n \cdot m)$.
- Mỗi trạng thái có thể có tối đa $O(m)$ cạnh chuyển tiếp (unlock hoặc di chuyển theo thuộc tính).
- Sử dụng priority queue, độ phức tạp Dijkstra là $O(E \log V)$, với $V = n \cdot m$, $E = O(n \cdot m^2)$.
- Tổng độ phức tạp thời gian: $O(n \cdot m^2 \log(n \cdot m))$.

Độ phức tạp không gian:

- Lưu ma trận chi phí $dist[n][m]$ và priority queue.
- Tổng không gian sử dụng: $O(n \cdot m)$.