Solution

1. Phương pháp thiết kế thuật toán

Phương pháp thiết kế thuật toán được sử dụng là **Thuật toán Dijkstra** trên một đồ thị được xây dựng đặc biệt bằng kỹ thuật **Sử dụng Đỉnh ảo và Sắp xếp (Virtual Nodes and Sorting)** để tối ưu hóa việc tính toán trọng số cạnh.

Mục đích của việc xây dựng đồ thị này là để ánh xạ tất cả các điều kiện thắng $a_{i,j} \geq a_{p,j}$ cho một thuộc tính j thành một tập hợp nhỏ các cạnh, thay vì phải tính toán riêng biệt $O(N^2)$ lần.

2. Tính phù hợp của phương pháp

Phương pháp này phù hợp vì:

1. **Chuyển đổi Ràng buộc thành Cạnh:** Mỗi lần chuyển trạng thái $(p \to i)$ cần i thắng p dựa trên thuộc tính j. Chi phí là $c_i + \max(0, a_{p,j} - a_{i,j})$.
2. **Tối ưu hóa Chi phí Thuộc tính (Attribute Cost):**
    - Đối với một thuộc tính j cố định, ta có thể nhận thấy điều kiện chiến thắng phụ thuộc vào giá trị thuộc tính.
    - Giải pháp đã phân tách việc tính toán này bằng cách:
        - **Sắp xếp:** Sắp xếp tất cả các giá trị thuộc tính $a_{i,j}$ $(i = 1..n)$ theo thứ tự tăng dần.
        - **Đỉnh ảo X_i và Y_i:** Tạo $2N$ đỉnh ảo cho mỗi thuộc tính.
        - **Ánh xạ Tăng Thuộc tính:** Các cạnh giữa các đỉnh ảo $X_i \to X_{i+1}$ (hoặc tương tự) có trọng số bằng **sự khác biệt** giữa các giá trị thuộc tính đã được sắp xếp $\text{val}_{i+1} - \text{val}_i$. Điều này mô hình hóa chi phí cần thiết để đạt đến mức thuộc tính tiếp theo trong danh sách.
        - **Ánh xạ Chiến thắng:** Cạnh từ $Y_{\text{rank}_i} \to i$ với trọng số c_i mô hình hóa việc Pokémon i chiến thắng sau khi đã đạt được mức thuộc tính cần thiết.
3. **Giảm Độ phức tạp:** Phương pháp này thay thế $O(N^2)$ cạnh trong đồ thị đầy đủ bằng $O(N)$ cạnh ảo cho mỗi thuộc tính, giảm đáng kể tổng số cạnh của đồ thị.

3. Phân tích độ phức tạp thời gian và không gian

1. Độ phức tạp thời gian (Time Complexity)

1. **Tiền xử lý/Xây dựng đồ thị:**
   - Ta lặp qua M thuộc tính.
   - Với mỗi thuộc tính:
     - **Sắp xếp:** Sắp xếp N giá trị thuộc tính $a_{i,j}$. Thời gian: $O(N \log N)$.
     - **Thêm đỉnh và cạnh:** Ta thêm $O(N)$ đỉnh ảo (X_i, Y_i) và $O(N)$ cạnh giữa chúng.
   - Tổng thời gian xây dựng đồ thị: $O(M \cdot (N \log N))$.
2. **Thuật toán Dijkstra:**
   - **Số đỉnh $|V|$:** N đỉnh thực (Pokémon) $+M \times 2N$ đỉnh ảo. $|V| = O(N \cdot M)$.
   - **Số cạnh $|E|$:**
     - Cạnh giữa các đỉnh ảo: $O(N)$ cạnh/thuộc tính $\implies O(N \cdot M)$ cạnh.
     - Cạnh từ $X_i \to Y_i$ và $Y_i \to i$: $O(N)$ cạnh/thuộc tính $\implies O(N \cdot M)$ cạnh.
     - Cạnh từ $i \to X_{\text{rank}_i}$ (chuyển từ đỉnh thực vào hệ thống ảo): $O(N)$ cạnh/thuộc tính $\implies O(N \cdot M)$ cạnh.
     - Tổng số cạnh: $|E| = O(N \cdot M)$.
   - Áp dụng Dijkstra với Hàng đợi Ưu tiên: $O(|E| \log |V|)$.

$$\text{Thời gian Dijkstra} = O(NM \log(NM))$$

Do $\log(NM) = \log N + \log M$, ta có thể viết là $O(NM(\log N + \log M))$.

- Tổng độ phức tạp thời gian:

$$O(NM \log N + NM \log(NM)) = O(NM \log(NM))$$

2. Độ phức tạp không gian (Space Complexity)

- **Lưu trữ Đồ thị:** Số lượng đỉnh và cạnh đều là $O(NM)$.
- **Mảng Khoảng cách (Distance Array):** Cần lưu khoảng cách cho $O(NM)$ đỉnh.
- **Tổng độ phức tạp không gian:** $O(NM)$.