Báo cáo nhóm 7

Đặng Phú Duy-24520010, Mai Quốc Anh-24520002

Đề bài (tóm tắt)

Bạn có n Pokémons, ban đầu Pokémon thứ 1 đang đứng trong đấu trường; bạn muốn để Pokémon n đứng trong đấu trường. Mỗi Pokémon i có m thuộc tính $a_{i,1}, \ldots, a_{i,m}$ và chi phí thuê c_i. Bạn có thể làm hai thao tác bất kỳ nhiều lần:

- Chọn $i, j, k > 0$ và tăng $a_{i,j}$ lên k vĩnh viễn, tốn chi phí k.
- Chọn i, j và thuê Pokémon i để đấu với Pokémon hiện tại dựa trên thuộc tính j. Pokémon i thắng nếu $a_{i,j} \geq$ thuộc tính j của Pokémon đang đứng; chi phí thuê là c_i. Sau đấu chỉ kẻ thắng đứng lại.

Tìm chi phí nhỏ nhất để Pokémon n đứng trong đấu trường.

(Đầu vào có nhiều test; tổng $n \cdot m \leq 4 \cdot 10^5$.)

Ý tưởng chính và xây dựng đồ thị

Chúng ta mô tả trạng thái và các thao tác bằng một đồ thị có trọng số, rồi chạy Dijkstra để tìm chi phí nhỏ nhất từ nút n tới nút 1.

Quan sát: một thao tác "tăng thuộc tính j của Pokémon u lên k rồi thuê Pokémon v để đánh" có chi phí $k + c_v$, và điều kiện để thuê v sau khi tăng là

$$a_{v,j} \geq a_{\text{current},j} + k.$$

Ta có thể biểu diễn mọi thao tác như một đường đi trong đồ thị nếu xây dựng hợp lý các nút ảo theo từng thuộc tính.

Cách xây dựng (theo từng thuộc tính x độc lập):

1. Với thuộc tính x lấy tất cả các cặp $(a_{i,x}, i)$, sắp xếp theo giá trị tăng dần. Giả sử thứ tự sau khi sắp xếp là $p_1, p_2, \ldots, p_n$ với giá trị tương ứng $v_1 \leq v_2 \leq \cdots \leq v_n$.
2. Tạo cho thuộc tính này hai chuỗi n nút ảo: $X_1, \ldots, X_n$ và $Y_1, \ldots, Y_n$ tương ứng với các vị trí trong thứ tự sắp xếp.
3. Thêm các cạnh (có hướng, trọng số):
   - $X_i \to X_{i+1}$ với trọng số $v_{i+1} - v_i$ (việc đi dọc chuỗi X tương ứng với việc tăng giá trị thuộc tính – chi phí là tổng các hiệu).
   - $Y_{i+1} \to Y_i$ với trọng số 0 (di chuyển trong chuỗi Y theo chiều giảm index miễn phí).

- $X_i \to Y_i$ với trọng số c_{p_i} (tại vị trí thứ i ta có thể thuê chính Pokémon tương ứng, tốn chi phí c).
- Nối nút thực p_i (nút đại diện cho chính Pokémon) **tới** X_i bằng cạnh trọng số 0.
- Nối Y_i **tới** nút thực p_i bằng cạnh trọng số 0.

4. Làm tương tự cho mọi thuộc tính x và ghép tất cả các cạnh, nút vào một đồ thị to chung. Các nút thực (Pokémons) là $1 \dots n$; ngoài ra có các nút ảo X, Y cho mỗi thuộc tính.

Giải thích trực quan:

- Nếu từ Pokémon u muốn "tăng thuộc tính x lên k"rồi thuê v (thứ tự thứ r_u và r_v trong sắp xếp), ta đi từ nút u tới X_{r_u} (chi phí 0), đi dọc chuỗi X tới X_{r_v} (chi phí chính là $v_{r_v} - v_{r_u} = k$ cần tăng), sau đó từ X_{r_v} sang Y_{r_v} (chi phí c_v) rồi từ Y_{r_v} tới nút v (0). Tổng là $k + c_v$ — đúng mô tả thao tác.
- Các cạnh $Y_{i+1} \to Y_i$ cho phép kết hợp nhiều lần thuê/nhảy mà không cần cộng thêm chi phí tăng.

Vậy bài toán quy về: tìm đường đi có tổng trọng số nhỏ nhất từ nút n đến nút 1 trong đồ thị có hướng, trọng số dương (Dijkstra phù hợp).

Minh chứng đúng

Mọi chuỗi thao tác hợp lệ (các lần tăng thuộc tính và các lần thuê) có thể ánh xạ sang một đường đi từ nguồn n tới đích 1 theo quy tắc trên. Ngược lại, mỗi đường đi từ n tới 1 trong đồ thị tương ứng với một chuỗi thao tác hợp lệ vì:

- Các cạnh chỉ cho phép tăng giá trị theo thứ tự tăng của các giá trị gốc (xây dựng X) và thuê Pokémon tương ứng (cạnh $X_i \to Y_i$ có chi phí c), sau đó quay về các nút thực.
- Cạnh $X_i \to X_{i+1}$ bắt buộc phải trả đúng chênh lệch giá trị để "tăng"đủ tới ngưỡng tiếp theo.

Vì mọi thao tác cơ bản đều được mô tả và không xuất hiện đường đi tương ứng với thao tác bất hợp lệ, tìm đường đi ngắn nhất sẽ cho chi phí tối ưu.

Phương pháp thiết kế (0.4)

Sử dụng **mô hình hoá trạng thái bằng đồ thị** và áp dụng **thuật toán Dijkstra** để tìm chi phí nhỏ nhất. Đây là phương pháp kết hợp tư duy đồ thị (mô tả trạng thái và chuyển trạng thái bằng cạnh có trọng số) và thuật toán đường đi ngắn nhất trên đồ thị có trọng số không âm.

Tính phù hợp (0.3)

Phương pháp này phù hợp vì mỗi thao tác (tăng thuộc tính, thuê) tương ứng tự nhiên với một chuyển trạng thái có chi phí xác định; khi ta biểu diễn chính xác các trạng thái trung gian (các mức giá trị sau khi sắp xếp) dưới dạng nút ảo, tổng chi phí của một chuỗi thao tác là tổng trọng số các cạnh trên một đường đi. Do trọng số đều không âm, Dijkstra sẽ tìm ra đường đi tối ưu.

Phân tích độ phức tạp thời gian và không gian (0.3)

Gọi tổng số phần tử là $S = \sum(n \cdot m)$ (theo đề bài $S \leq 4 \cdot 10^5$). Với cách xây dựng trên:

- Số nút thực là n. Với mỗi thuộc tính ta thêm $2n$ nút ảo, tổng số nút xấp xỉ $N = n + 2nm = O(S)$ (vì $n \cdot m = S$).
- Số cạnh: với mỗi thuộc tính ta thêm khoảng $5n$ cạnh (các cạnh giữa các X, Y, các nối tới/từ nút thực), tổng cạnh $M = O(nm) = O(S)$.
- Chạy Dijkstra trên đồ thị có N nút và M cạnh có độ phức tạp $O((N+M)\log N) = O(S \log S)$.
- Không gian lưu trữ đồ thị và hàng đợi ưu tiên là $O(N + M) = O(S)$.

Do $S \leq 4 \cdot 10^5$, thuật toán chạy nhanh trong giới hạn thời gian và bộ nhớ (cận 512 MB) của bài toán.