Lời giải chi tiết bài tập nhóm 7

Phân tích và thiết kế thuật toán

Nguyễn Duy Khang - 24520755

Hà Bùi Trọng Nghĩa - 24520020

Lời giải bài toán Pokémon

I. Tóm tắt bài toán

Bạn có n Pokémon, mỗi Pokémon có m thuộc tính và chi phí thuê c_i. Ban đầu Pokémon số 1 đứng trong đấu trường và mục tiêu là đưa Pokémon n vào đấu trường với chi phí nhỏ nhất.

Có hai thao tác:

- Tăng thuộc tính $a_{i,j}$ thêm k, chi phí là k.
- Thuê Pokémon i đấu dựa trên thuộc tính j, Pokémon i thắng nếu $a_{i,j} \geq a_{cur,j}$, chi phí là c_i.

Tìm chi phí tối thiểu để Pokémon n cuối cùng đứng trong đấu trường.

II. Nhận xét

1. Việc tăng thuộc tính chỉ có ích đối với Pokémon mà ta chuẩn bị đưa vào đấu trường.
2. Để Pokémon i đánh bại Pokémon cur, chi phí buff nhỏ nhất là:

$$\min_{j=1..m} \max(0,\ a_{cur,j} - a_{i,j})$$

3. Tổng chi phí khi chuyển từ $cur \to i$ là:

$$\text{cost}(cur \to i) = c_i + \min_j (a_{cur,j} - a_{i,j})^+$$

4. Đây là một bài toán đường đi ngắn nhất trên đồ thị với chi phí không âm.

III. Phương pháp thiết kế thuật toán

- Sử dụng thuật toán Dijkstra để tìm đường đi chi phí nhỏ nhất.
- Không thể xây dựng đồ thị đầy đủ n^2 cạnh.
- Với mỗi thuộc tính j, sắp xếp Pokémon theo $a[i][j]$.
- Khi đang ở Pokémon *cur*, chỉ cần xét các Pokémon có thuộc tính không nhỏ hơn theo từng chiều.
- Nhờ đó mỗi Pokémon trên mỗi chiều chỉ được xét một lần, giảm độ phức tạp xuống mức chấp nhận được.

IV. Lời giải

Ý tưởng chính:

- Mỗi Pokémon là một đỉnh trong đồ thị.
- Chi phí cạnh từ *cur* đến i là:

$$buff = \max(0,\ a_{cur,j} - a_{i,j}), \quad cost = buff + c_i$$

- Duyệt cạnh theo từng thuộc tính đã sắp xếp.

Thuật toán

```
dist[1] = 0, các đỉnh còn lại = INF
PQ.push((0, 1))

for j = 1..m:
    sorted_list[j] = danh sách Pokemon sắp theo thuộc tính j
    pointer[j] = vị trí của Pokemon 1

while PQ không rỗng:
    lấy (cost, cur)

    nếu cost != dist[cur]: continue

    for j = 1..m:
        di chuyển pointer[j] đến các Pokemon chưa xét
```

```
        for mỗi Pokemon i sau pointer:
            buff = max(0, a[cur][j] - a[i][j])
            new = dist[cur] + buff + c[i]
            nếu new < dist[i]:
                dist[i] = new
                PQ.push((new, i))

kết quả: dist[n]
```

V. Lý do chọn phương pháp

- Chi phí chuyển luôn không âm nên Dijkstra là phù hợp nhất.
- Không thể duyệt toàn bộ n^2 cặp Pokémon, nhưng có thể tận dụng cấu trúc theo từng thuộc tính.
- Sắp xếp theo từng chiều cho phép tối ưu truy vấn $\min_j$ một cách hiệu quả.
- Tổng số thao tác thực tế chỉ $O(nm)$ do mỗi Pokémon được xử lý 1 lần trên mỗi chiều.

VI. Phân tích độ phức tạp

Thời gian:

$$O(n \log n) \text{ để sắp xếp}$$

$$O((n \cdot m) \log n) \text{ cho Dijkstra}$$

Vì $n \cdot m \leq 4 \cdot 10^5$, điều này hoàn toàn đảm bảo.

Không gian:

$$O(nm) \text{ để lưu thuộc tính}$$

$$O(n) \text{ cho dist và PQ}$$

$$O(nm) \text{ cho m danh sách đã sắp}$$

VII. Kết luận

Bài toán được mô hình thành bài toán đường đi ngắn nhất trong đồ thị n đỉnh, sử dụng Dijkstra kết hợp duyệt theo từng thuộc tính đã sắp xếp để tối ưu hoá. Độ phức tạp đảm bảo và lời giải đạt tối ưu.