Bài toán: Trận đối đầu trên hành tinh Namek - Lời giải và Phân tích

Phân tích và chứng minh thuật toán

(Độ phức tạp thời gian: $O(n^2)$, độ phức tạp không gian: $O(n)$)

1. Phân tích đề bài

Cho n chiến binh, mỗi chiến binh có ba thông số:

$$(\text{age}_i, \text{intelligence}_i, \text{power}_i), \quad \text{với power}_i \in [-200, 200].$$

Yêu cầu chọn một tập hợp chiến binh sao cho:

- Mọi cặp chiến binh (u, v) trong tập thỏa:

$$(\text{age}_u \le \text{age}_v \text{ và intelligence}_u \le \text{intelligence}_v) \text{ hoặc ngược lại.}$$

- Tích Power của các chiến binh được chọn là lớn nhất.

Nhận xét: Đây là bài toán biến thể của *Longest Increasing Subsequence 2D*, nhưng thay vì đếm số lượng, cần tối đa hóa tích Power. Vì power có thể âm, cần xử lý cả min/max product.

2. Phân tích hướng giải

1. Sắp xếp các chiến binh theo $(\text{age}, \text{intelligence})$ tăng dần. Sau khi sắp xếp, mọi chuỗi con đều thỏa điều kiện không giảm.
2. Dùng DP: với mỗi chiến binh i, lưu tích Power lớn nhất $dp_{\max}[i]$ và nhỏ nhất $dp_{\min}[i]$ kết thúc tại i.
3. Duyệt từ trái sang phải, với mỗi chiến binh i:
   - Lặp qua tất cả chiến binh $j < i$ sao cho $(\text{age}_j \le \text{age}_i$ và $\text{intelligence}_j \le \text{intelligence}_i)$.
   - Cập nhật $dp_{\max}[i]$ và $dp_{\min}[i]$ theo công thức phụ thuộc vào dấu của power_i.
4. Kết quả là $\max_i \text{dp\_max}[i]$.

3. Phân tích thuật toán

3.1. Khởi tạo DP

$$\text{dp\_max}[i] = \text{dp\_min}[i] = \text{power}_i$$

3.2. Cập nhật DP

$$\text{for } j \text{ từ } 0 \text{ đến } i-1: \quad \text{nếu } \text{age}_j \leq \text{age}_i \text{ và } \text{intelligence}_j \leq \text{intelligence}_i:$$
$$\begin{cases} \text{nếu power}_i \geq 0: & \text{dp\_max[i]} = \max(\text{dp\_max[i]}, \text{dp\_max[j]} \cdot \text{power}_i) \\ & \text{dp\_min[i]} = \min(\text{dp\_min[i]}, \text{dp\_min[j]} \cdot \text{power}_i) \\ \text{nếu power}_i < 0: & \text{dp\_max[i]} = \max(\text{dp\_max[i]}, \text{dp\_min[j]} \cdot \text{power}_i) \\ & \text{dp\_min[i]} = \min(\text{dp\_min[i]}, \text{dp\_max[j]} \cdot \text{power}_i) \end{cases}$$

3.3. Kết quả

$$\text{ans} = \max_i \text{dp\_max}[i]$$

4. Chứng minh thuật toán

1. **Tính đúng đắn:**
   - Sau khi sắp xếp, mọi chuỗi con đều thỏa điều kiện tăng không giảm.
   - Với mỗi chiến binh i, ta theo dõi:

     $\text{dp}_{\text{max}}[i] = \text{tích Power lớn nhất kết thúc tại } i, \quad \text{dp}_{\text{min}}[i] = \text{tích Power nhỏ nhất để xử lý số â}$
   - Lặp qua tất cả $i \to$ đảm bảo xét tất cả chuỗi hợp lệ.
2. **Tính tối ưu:**
   - Với mọi chuỗi hợp lệ, phần tử cuối cùng là một chiến binh i. DP đã xét tất cả khả năng kết thúc tại i.
   - Cập nhật DP:

     $$\text{for } j < i, \text{ nếu } \text{age}_j \leq \text{age}_i \text{ và } \text{intelligence}_j \leq \text{intelligence}_i:$$
     $$\begin{cases} \text{nếu power}_i \geq 0: & \text{dp}_{\text{max}}[i] = \max(\text{dp}_{\text{max}}[i], \text{dp}_{\text{max}}[j] \cdot \text{power}_i) \\ & \text{dp}_{\text{min}}[i] = \min(\text{dp}_{\text{min}}[i], \text{dp}_{\text{min}}[j] \cdot \text{power}_i) \\ \text{nếu power}_i < 0: & \text{dp}_{\text{max}}[i] = \max(\text{dp}_{\text{max}}[i], \text{dp}_{\text{min}}[j] \cdot \text{power}_i) \\ & \text{dp}_{\text{min}}[i] = \min(\text{dp}_{\text{min}}[i], \text{dp}_{\text{max}}[j] \cdot \text{power}_i) \end{cases}$$
   - Kết quả tối ưu toàn cục là:

     $$\text{ans} = \max_i \text{dp}_{\text{max}}[i].$$

5. Phân tích độ phức tạp

5.1. Thời gian

- Sắp xếp: $O(n \log n)$
- DP: $O(n^2)$ do mỗi chiến binh i lặp qua tất cả $j < i$

Tổng: $O(n^2)$

5.2. Không gian

- Mảng dp_max, dp_min: $O(n)$
- Lưu chiến binh: $O(n)$

6. Code mẫu Python

```
import sys
input = sys.stdin.readline

n = int(input())
fighters = [tuple(map(int, input().split())) for _ in range(n)]
fighters.sort()

dp_max = [0] * n
dp_min = [0] * n
ans = -float('inf')

for i in range(n):
    age_i, int_i, power_i = fighters[i]
    dp_max[i] = power_i
    dp_min[i] = power_i
    for j in range(i):
        age_j, int_j, power_j = fighters[j]
        if age_j <= age_i and int_j <= int_i:
            if power_i >= 0:
                dp_max[i] = max(dp_max[i], dp_max[j] * power_i)
                dp_min[i] = min(dp_min[i], dp_min[j] * power_i)
            else:
                dp_max[i] = max(dp_max[i], dp_min[j] * power_i)
                dp_min[i] = min(dp_min[i], dp_max[j] * power_i)
    ans = max(ans, dp_max[i])

print(ans)
```

7. Kết luận

- Thuật toán dùng sắp xếp + DP 2D max/min product.
- Đảm bảo mọi chuỗi con thỏa điều kiện tăng không giảm.
- Độ phức tạp thời gian $O(n^2)$, không gian $O(n)$.
- Đúng tối ưu toàn cục và xử lý tốt các giá trị âm của power.