Trận thư hùng ở Namek – Solution

Nhóm 8

Ngày 29 tháng 10 năm 2025

1. Tóm tắt đề (summary)

Có n chiến binh, mỗi chiến binh i có ba chỉ số:

- `age[i]` (tuổi),
- `intelligence[i]` (trí tuệ),
- `power[i]` (sức mạnh — có thể âm, 0 hoặc dương).

Ta cần chọn một dãy con (subset nhưng duy trì thứ tự sau khi sắp xếp theo `age`) sao cho tất cả chiến binh được chọn có thể sắp xếp không giảm theo cả `age` và `intelligence`.

(Một cách hiểu: tồn tại cách sắp xếp các chiến binh được chọn sao cho cả `age` và `intelligence` đều là không giảm.)

Mục tiêu: tối đa hóa tích các `power` của những chiến binh được chọn. In ra một số nguyên duy nhất là tích lớn nhất có thể.

Lưu ý: `power` có thể âm → bài toán tối ưu tích có tính chất phức tạp hơn đơn thuần là cộng/gộp.

2. Nhận xét & quan sát quan trọng

Nếu ta sắp xếp tất cả chiến binh theo `age` tăng (và thứ tự thứ hai là `intelligence` tăng khi `age` bằng nhau), thì mọi dãy con (subsequence) của dãy đã sắp xếp sẽ tự động thỏa điều kiện `age` không giảm. Do đó ta chỉ cần đảm bảo phần `intelligence` của dãy con là không giảm.

Bài toán trở thành: trong dãy đã sắp xếp theo `age`, chọn một subsequence mà `intelligence` không giảm, để tích `power` tối đa.

Vì `power` có thể âm, zero, dương, khi nhân liên tiếp các phần tử ta cần theo dõi cả giá trị lớn nhất và giá trị nhỏ nhất (để tận dụng âm $\times$ âm = dương). Do đó một DP cần lưu hai giá trị: `best_max` và `best_min` cho mỗi vị trí kết thúc.

3. Giải pháp brute-force (subtask 1, $n \leq 20$)

Ý tưởng

Vì $n \leq 20$, có thể duyệt toàn bộ 2^n tập con (khoảng 1 triệu tập hợp). Duyệt mọi tập con (2^n), kiểm tra điều kiện `age` & `intelligence` có thể sắp xếp không giảm (hoặc đơn giản: sau khi lấy tập con, sắp xếp theo `age` rồi kiểm tra `intelligence` không giảm), rồi tính tích `power` và ghi nhận max.

Độ phức tạp

- **Thời gian:** $O(2^n \cdot n \log n)$ (hoặc $O(2^n \cdot n)$ nếu kiểm tra thông minh).
- **Không gian:** $O(n)$.

4. Giải pháp hiệu quả (subtask 2, $n \leq 1000$)

4.1. Chuẩn hóa dữ liệu

- Sắp xếp mảng chiến binh theo (`age` ↑, `intelligence` ↑) (tức nếu `age` bằng nhau, sắp xếp theo `intelligence` tăng).
- Với mảng đã sắp xếp, chỉ cần chọn subsequence chỉ số tăng $i_1 < i_2 < \cdots < i_k$ sao cho `intelligence`$[i_1] \leq$ `intelligence`$[i_2] \leq \cdots \leq$ `intelligence`$[i_k]$.

4.2. Quy hoạch động (DP) — ý tưởng cốt lõi

Vì tích có thể âm/ dương, cho mỗi vị trí i ta giữ hai giá trị:

- `dp_max[i]` = giá trị lớn nhất (max product) có thể đạt được từ một subsequence hợp lệ kết thúc tại i (tức chọn i là phần tử cuối).
- `dp_min[i]` = giá trị nhỏ nhất (min product) có thể đạt được từ một subsequence hợp lệ kết thúc tại i.

Chuyển tiếp:

- **Khởi tạo:** `dp_max[i] = dp_min[i] = power[i]` (chỉ chọn chính i).
- Với mỗi i duyệt mọi $j < i$ sao cho `intelligence[j]` $\leq$ `intelligence[i]` (vì chúng ta cần không giảm `intelligence`):
  - Ta có hai khả năng nhân:
    - `dp_max_candidate = dp_max[j] * power[i]`
    - `dp_min_candidate = dp_min[j] * power[i]`
  - Cập nhật:
    - `dp_max[i] = max(dp_max[i], dp_max_candidate, dp_min_candidate)`

* `dp_min[i] = min(dp_min[i], dp_max_candidate, dp_min_candidate)`

- Cuối cùng kết quả là `ans = max_i dp_max[i]`.

4.3. Chứng minh đúng ngắn gọn

Mỗi subsequence hợp lệ có phần tử cuối tại một vị trí i. Giá trị tích của subsequence là `power[i]` nhân với tích các phần tử trước i (từ một subsequence hợp lệ kết thúc tại $j < i$ với `intelligence[j]` $\leq$ `intelligence[i]`). Việc lưu đồng thời max/min tại j đảm bảo ta không bỏ sót trường hợp tận dụng số âm để thu được tích dương lớn hơn. Vì ta duyệt tất cả j trước i, công thức truy hồi xét mọi khả năng hợp lệ và do đó `dp_max[i]` chứa giá trị lớn nhất có thể cho mọi subsequence kết thúc tại i. Kết quả global là max trên tất cả các i.

4.4. Độ phức tạp

- **Thời gian:** $O(n^2)$ (vì với mỗi i duyệt $j < i$).
- **Bộ nhớ:** $O(n)$.

5. Tối ưu ($n \log n$) — ý tưởng nâng cao (tùy chọn)

Ta cần cho mỗi i truy vấn nhanh các `dp_max[j]`, `dp_min[j]` cho mọi j có `intelligence[j]` $\leq$ `intelligence[i]`. Đây là bài toán kiểu prefix query trên trục `intelligence` nếu ta nén tọa độ `intelligence`.

Dùng Fenwick tree / Segment tree trên chỉ số `intel` (sau coordinate compression), mỗi node lưu hai giá trị: max và min cho `dp_max` (hoặc lưu cặp cho `dp_max` & `dp_min`). Khi xử lý i theo thứ tự `age` tăng (và `intelligence` tăng khi `age` bằng), ta:

- Query range `[1..pos_intel_i]` lấy `max(dp_max)`, `min(dp_max)`, `max(dp_min)`, `min(dp_min)`.
- Từ đó tính các ứng viên (các phép nhân) và lấy max/min.
- Cập nhật tree tại `pos_intel_i` với `dp_max[i]` / `dp_min[i]` (phải hợp nhất nếu nhiều phần tử cùng pos).

Độ phức tạp: $O(n \log n)$ cập nhật + query.

Thực hiện với Big Integer phức tạp hơn (segment tree nodes giữ big-int), nhưng khả thi.

6. Code

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

def giai_bai_toan():
    try:
```

```
# Doc so luong chien binh n
n_str = input().strip()
if not n_str:
    print(0)
    return

n = int(n_str)
if n == 0:
    print(0) # Truong hop n = 0
    return

# Khoi tao danh sach de luu thong tin cac chien binh
danh_sach_chien_binh = []
for _ in range(n):
    age, intelligence, power = map(int, input().split())
    danh_sach_chien_binh.append((age, intelligence, power))

# Buoc 1: Sap xep (Chuan hoa du lieu)
danh_sach_chien_binh.sort(key=lambda chien_binh: (chien_binh[0], ch

# Buoc 2: Quy hoach dong (DP)

# dp_max[i]: Tich SUC MANH LON NHAT cua day con hop le ket thuc tai
# dp_min[i]: Tich SUC MANH NHO NHAT cua day con hop le ket thuc tai
# (Luu min la de xu ly truong hop am * am = duong)
dp_max = [0] * n
dp_min = [0] * n

# Khoi tao ket qua cuoi cung
ket_qua_lon_nhat = -float('inf')

for i in range(n):
    # Lay thong tin cua chien binh hien tai (i)
    # Khong can lay tuoi (age) vi da duoc dam bao boi sap xep
    current_intel = danh_sach_chien_binh[i][1]
    current_power = danh_sach_chien_binh[i][2]

    # Khoi tao DP: Day con chi chua 1 phan tu la chinh no
    dp_max[i] = current_power
    dp_min[i] = current_power

    # Duyet qua tat ca cac chien binh j dung truoc i (j < i)
    for j in range(i):
        prev_intel = danh_sach_chien_binh[j][1]

        # --- Dieu kien chuyen trang thai DP ---
        # Vi mang da sap xep theo (Tuoi, Tri tue),
        # Tuoi[j] <= Tuoi[i] da duoc dam bao.
```

```
                    # Ta chi can kiem tra Tri tue[j] <= Tri tue[i].
                    # (Code C++ goc co kiem tra ca 2, nhung 'Age' la khong can
                    if prev_intel <= current_intel:

                        # Tinh 2 ung cu vien khi nhan power[i] voi day con ket
                        # Ung cu vien 1: max[j] * power[i]
                        candidate_1 = dp_max[j] * current_power
                        # Ung cu vien 2: min[j] * power[i] (quan trong khi powe
                        candidate_2 = dp_min[j] * current_power

                        # Cap nhat dp_max[i]:
                        # Lay gia tri lon nhat trong 3 kha nang:
                        # 1. dp_max[i] (hien tai, tuc chi chon minh i)
                        # 2. candidate_1 (ghep i vao day max ket thuc tai j)
                        # 3. candidate_2 (ghep i vao day min ket thuc tai j)
                        dp_max[i] = max(dp_max[i], candidate_1, candidate_2)

                        # Cap nhat dp_min[i]: (Tuong tu)
                        dp_min[i] = min(dp_min[i], candidate_1, candidate_2)

                # Sau khi duyet het cac 'j', ta da co dp_max[i] cuoi cung
                # Cap nhat ket qua lon nhat toan cuc
                ket_qua_lon_nhat = max(ket_qua_lon_nhat, dp_max[i])

            # In ra ket qua cuoi cung
            print(ket_qua_lon_nhat)

    except (IOError, ValueError, EOFError) as e:
        pass

if __name__ == "__main__":
    giai_bai_toan()
```