Giải bài tập: Trận Thư Hùng ở Namek

1 Tóm tắt bài toán

Goku chuẩn bị chiến đấu với Frieza tại hành tinh Namek. Để tăng sức mạnh chiến đấu, Goku muốn lựa chọn một nhóm chiến binh từ tổng số n chiến binh Trái Đất ($1 \leq n \leq 1000$). Mỗi chiến binh được mô tả bởi bộ ba số nguyên (a_i, b_i, c_i), trong đó a_i là tuổi (Age), b_i là độ thông minh (Intelligence) và c_i là sức mạnh chiến đấu (Power). Ta có:

$$1 \leq a_i, b_i \leq 100000, \qquad -200 \leq c_i \leq 200.$$

2 Ràng buộc lựa chọn

Đội hình được chọn phải đảm bảo rằng giữa mọi cặp chiến binh đều có thể sắp xếp sao cho giá trị tuổi và trí tuệ đều không giảm. Cụ thể, với mọi i, j trong đội hình:

$$(a_i \leq a_j \text{ và } b_i \leq b_j) \quad \text{hoặc} \quad (a_j \leq a_i \text{ và } b_j \leq b_i).$$

Do đó, nếu sắp xếp các chiến binh theo tuổi tăng dần, dãy độ thông minh tương ứng cũng phải không giảm. Nói cách khác, tập chiến binh được chọn tạo thành một chuỗi đồng tăng theo hai thuộc tính Age và Intelligence.

3 Mục tiêu tối ưu

Mục tiêu là tối đa hóa tích sức mạnh chiến đấu của các chiến binh được chọn:

$$\max_S \prod_{i \in S} c_i.$$

Vì c_i có thể mang giá trị âm, số lượng phần tử có $c_i < 0$ ảnh hưởng trực tiếp đến dấu và độ lớn của tích:

- Nếu số lượng $c_i < 0$ là chẵn, tích có thể mang giá trị dương lớn.
- Nếu số lượng đó là lẻ, tích có thể trở thành âm và làm suy giảm hiệu quả.

Do đó, cần lựa chọn tập chiến binh thỏa điều kiện sắp xếp và cho giá trị tích lớn nhất.

4 Định dạng vào/ra

4.1 Input

- Dòng đầu tiên chứa một số nguyên n — số lượng chiến binh.
- Mỗi trong số n dòng tiếp theo chứa ba số nguyên a_i, b_i, c_i tương ứng với tuổi (Age), trí tuệ (Intelligence) và sức mạnh chiến đấu (Power) của chiến binh thứ i.

4.2 Output

- In ra một số nguyên duy nhất là giá trị lớn nhất của tích sức mạnh các chiến binh được chọn sao cho tập chiến binh đó thỏa mãn điều kiện sắp xếp.

5 Ghi chú quan trọng

- Các chiến binh được chọn có thể có cùng tuổi hoặc cùng trí tuệ; yêu cầu duy nhất là có thể sắp xếp sao cho dãy tuổi và dãy trí tuệ tương ứng đều không giảm.
- Không bắt buộc phải chọn toàn bộ n chiến binh. Có thể chọn bất kỳ số lượng nào, kể cả chỉ một chiến binh nếu phù hợp.
- Vì giá trị c_i có thể âm, việc chọn hoặc loại một số chiến binh có thể ảnh hưởng đáng kể đến kết quả tích cuối cùng. Trong nhiều trường hợp, bỏ bớt một chiến binh có power âm có thể giúp làm tăng giá trị tích thu được.

6 Hướng giải quyết

6.1 Tiền xử lý và sắp xếp

Mỗi chiến binh được biểu diễn dưới dạng bộ (a_i, b_i, c_i) tương ứng với tuổi, trí tuệ và sức mạnh chiến đấu. Ta tiến hành sắp xếp toàn bộ danh sách theo bộ khóa (a_i, b_i) theo thứ tự tăng dần. Nhờ đó, khi duyệt từ trái sang phải, mọi phần tử phía sau đều có giá trị *age* không nhỏ hơn phần tử phía trước và ta chỉ cần đảm bảo rằng dãy *intelligence* không giảm.
Khi đó, yêu cầu bài toán trở thành việc tìm một dãy con sao cho giá trị b_i không giảm và tích các giá trị c_i trong dãy là lớn nhất.

6.2 Áp dụng cho Subtask 1 ($n \leq 20$)

Với kích thước nhỏ, ta có thể duyệt toàn bộ các cách chọn chiến binh. Sau khi sắp xếp theo *age*, ta sử dụng phương pháp vét cạn hoặc quay lui để xây dựng các dãy có thể chọn. Khi xét thêm một chiến binh mới, ta chỉ chấp nhận trường hợp b_i của chiến binh này không nhỏ hơn b_i của chiến binh cuối cùng đã chọn, nhằm duy trì tính không giảm của dãy.
Với mỗi dãy hợp lệ thu được, tính tích các giá trị c_i và cập nhật đáp án lớn nhất. Có thể cắt tỉa nhánh khi tích trung gian bằng 0 hoặc không thể vượt qua kết quả tốt nhất đã ghi nhận.

- **Độ phức tạp thời gian:** $O(n \cdot 2^n)$.
- **Độ phức tạp không gian:** $O(n)$, do chỉ cần lưu trữ dãy đang xét trong quá trình đệ quy.

6.3 Bàn về subtask 2 ($n \leq 1000$)

Khi n tăng lên đến 1000, việc duyệt toàn bộ các tập con là không khả thi do số lượng lựa chọn lên tới 2^n. Sau khi sắp xếp danh sách chiến binh theo bộ khóa (a_i, b_i) theo thứ tự không giảm, ta thu được dãy:

$$S = (a_1, b_1, c_1), (a_2, b_2, c_2), \ldots, (a_n, b_n, c_n),$$

trong đó $a_1 \leq a_2 \leq \cdots \leq a_n$ và nếu $a_i = a_{i+1}$ thì $b_i \leq b_{i+1}$. Khi đó, bài toán trở thành tìm một dãy con không giảm theo b_i sao cho tích các giá trị c_i là lớn nhất. Đây là một biến thể có trọng số (theo phép nhân) của bài toán dãy con không giảm truyền thống.

6.4 Ý tưởng DP hai nhánh

Tại mỗi vị trí i, ta muốn xác định giá trị tích tốt nhất của một dãy các chiến binh hợp lệ kết thúc tại i. Do c_i có thể mang giá trị âm, việc chỉ theo dõi "tích lớn nhất" là chưa đủ: một tích âm có giá trị tuyệt đối lớn có thể trở thành kết quả tối ưu nếu sau đó được nhân với một hệ số âm khác. Vì vậy, ta lưu đồng thời hai đại lượng:

$$\mathrm{dpmax}_i = \max\left\{ \prod_{k=1}^{\ell} c_{t_k} \;\middle|\; t_1 < \cdots < t_\ell = i,\;\; b_{t_k} \le b_{t_{k+1}} \text{ với mọi } k \right\},$$

$$\mathrm{dpmin}_i = \min\left\{ \prod_{k=1}^{\ell} c_{t_k} \;\middle|\; t_1 < \cdots < t_\ell = i,\;\; b_{t_k} \le b_{t_{k+1}} \text{ với mọi } k \right\}.$$

6.5 Công thức truy hồi

Ban đầu, dãy chỉ gồm riêng chiến binh i luôn hợp lệ, do đó:

$$\mathrm{dpmax}_i = \mathrm{dpmin}_i = c_i.$$

Khi xét một vị trí i, với mỗi $j < i$ thoả $b_j \le b_i$, ta có thể nối chiến binh i vào một dãy kết thúc tại j. Khi đó phát sinh hai giá trị ứng viên:

$$\widehat{P}_{ij} = \mathrm{dpmax}_j \cdot c_i, \qquad \widehat{Q}_{ij} = \mathrm{dpmin}_j \cdot c_i.$$

Giá trị tốt nhất và tệ nhất tại i được cập nhật như sau:

$$\mathrm{dpmax}_i = \max\left(\mathrm{dpmax}_i, \widehat{P}_{ij}, \widehat{Q}_{ij}\right), \qquad \mathrm{dpmin}_i = \min\left(\mathrm{dpmin}_i, \widehat{P}_{ij}, \widehat{Q}_{ij}\right).$$

Nhờ lưu cả dpmin, khi $c_i < 0$ ta có thể chuyển một tích âm lớn (theo trị tuyệt đối) thành tích dương vượt trội tại bước i.

6.4 Giải thích thuật toán theo cài đặt

Gọi mỗi chiến binh là một bộ (a_i, b_i, c_i) tương ứng lần lượt với *age*, *intelligence* và *power*. Để đơn giản hoá xử lý, ta thêm một phần tử giả ở đầu danh sách:

$$(0, 0, 1)$$

Phần tử này luôn có thể đứng trước mọi chiến binh khác và không làm thay đổi tích do giá trị $c = 1$.

Tiếp theo, ta sắp xếp toàn bộ danh sách chiến binh theo thứ tự từ điển của cặp (a_i, b_i). Việc này đảm bảo rằng khi duyệt theo chỉ số tăng dần, chỉ số tuổi a luôn không giảm; từ đó ta chỉ cần kiểm tra thêm điều kiện b để đảm bảo dãy chọn ra là hợp lệ.

Để lưu trữ kết quả tốt nhất khi kết thúc tại mỗi chiến binh i, ta sử dụng hai mảng:

$$\text{dp\_max}[i] = \text{tích lớn nhất có thể đạt được khi chọn dãy kết thúc tại } i,$$

$$\text{dp\_min}[i] = \text{tích nhỏ nhất (âm nhất theo trị tuyệt đối) khi kết thúc tại } i.$$

Việc duy trì đồng thời hai giá trị trên là cần thiết vì c_i có thể âm; khi nhân thêm một số âm, một tích âm lớn (theo trị tuyệt đối) có thể trở thành tích dương rất lớn.

Khởi tạo:

$$\text{dp\_max}[0] = \text{dp\_min}[0] = 1.$$

Công thức chuyển trạng thái diễn ra như sau:

$$\text{khi xét } j < i \text{ thỏa } a_j \le a_i \text{ và } b_j \le b_i :$$

$$P_{ij} = \text{dp\_max}[j] \cdot c_i, \qquad Q_{ij} = \text{dp\_min}[j] \cdot c_i.$$

$$\text{dp\_max}[i] = \max(\text{dp\_max}[i], P_{ij}, Q_{ij}),$$

$$\text{dp\_min}[i] = \min(\text{dp\_min}[i], P_{ij}, Q_{ij}).$$

Sau khi duyệt hết tất cả i, đáp án cuối cùng là:

$$\max_{1 \le i \le n} \text{dp\_max}[i].$$

Độ phức tạp:

- Thời gian: $O(n^2)$ do duyệt cặp (i, j) với $j < i$.
- Không gian: $O(n)$ cho hai mảng dp_max và dp_min.

6.5 Cài đặt python cho subtask 2:

```python
n = int(input())

a = [(0, 0, 1)]
for _ in range(n):
    x, y, z = map(int, input().split())
    a.append((x, y, z))

a.sort(key=lambda v: (v[0], v[1]))

def ok(i, j):
    return a[i][0] <= a[j][0] and a[i][1] <= a[j][1]

INF = 10**18
dp_max = [-INF] * (n + 1)
dp_min = [INF] * (n + 1)

dp_max[0] = dp_min[0] = 1

for i in range(1, n + 1):
    for j in range(i - 1, -1, -1):
        if ok(j, i):
            v = a[i][2]
            dp_max[i] = max(dp_max[i], dp_max[j] * v,
               dp_min[j] * v)
            dp_min[i] = min(dp_min[i], dp_max[j] * v,
               dp_min[j] * v)

ans = max(dp_max[1:])
print(ans)
```