CS112 - Quy Hoạch Động

Hồ Hữu Tây - 24520029
Dương Hoàng Việt - 24520036

October 29, 2025

1 Bài toán:

Son Goku đang chuẩn bị lên đường đến hành tinh Namek để tìm ngọc rồng hồi sinh những người bạn của anh ta. Tuy nhiên, trên hành tinh đó có một phản diện rất mạnh là Frieza. Goku không thể đấu lại hắn nên quyết định tập hợp các chiến binh mạnh nhất của Trái Đất cùng đến hành tinh Namek để hợp sức tiêu diệt Frieza.

Mỗi chiến binh của Trái Đất có 3 chỉ số đại diện lần lượt là `Age`, `Intelligence` và `Power`. Goku chỉ có thể chọn những chiến binh sao cho 2 chiến binh bất kỳ trong số những chiến binh được chọn thì 1 người phải có cả `age` và `intelligence` nhỏ hơn hoặc bằng người còn lại. Nói cách khác, trong những chiến binh được chọn thì tất cả đều phải được sắp xếp thành dãy không giảm cả về `Age` lẫn `Intelligence`.

Bạn hãy giúp Goku chọn ra những chiến binh phù hợp sao cho tích `Power` của họ là lớn nhất.

Input

- Dòng thứ nhất gồm 1 số nguyên n ($1 \leq n \leq 1000$) là n chiến binh Z.
- n dòng tiếp theo lần lượt là 3 số nguyên a, b, c cách nhau bởi khoảng trắng, lần lượt biểu thị `age`, `intelligence` và `power` của từng chiến binh.
- Các số nguyên a, b trong phạm vi $[1, 100000]$, $-200 \leq c \leq 200$.

Output

Một số nguyên duy nhất là tích `power` lớn nhất của các chiến binh được chọn.

Subtasks

- **Subtask 1:** 40% test có $n \leq 20$.
- **Subtask 2:** 60% test còn lại không có ràng buộc gì thêm.

Ví dụ

INPUT	OUTPUT
5 1 7 -10 2 8 10 3 11 -7 4 12 -4 5 15 0	700
7 23 7 10 20 13 30 28 13 25 21 11 7 25 6 9 26 9 15 20 17 36	3750

2 Subtask1:

2.1 Mô tả thuật toán:

Mục tiêu:

Từ danh sách n chiến binh (mỗi chiến binh có 3 thuộc tính: `age`, `intelligence`, `power`), ta muốn chọn một tập con (không rỗng) sao cho:

- Khi sắp xếp theo thứ tự xuất hiện trong dãy đã chọn (giữ nguyên thứ tự theo sắp xếp `age` rồi `intelligence` nếu cần), dãy `age` không giảm; đồng thời `intelligence` cũng không giảm.
- Tích `power` của các chiến binh trong tập con là lớn nhất có thể (có thể âm/dương/0).

Ý tưởng Brute Force (duyệt toàn bộ):

- Vì $n \leq 20$, số tập con khả dĩ là 2^n (tối đa $\approx 1,048,576$) — hợp lý để thử toàn bộ.

- Duyệt mọi bitmask từ 1 đến $(1 \ll n) - 1$ (bỏ qua mask = 0 nếu đề yêu cầu chọn ít nhất 1 người).
- Với mỗi mask:
  - Lấy danh sách các chiến binh được chọn theo thứ tự chỉ số (tương ứng với thứ tự input).
  - Kiểm tra dãy được chọn có thỏa 2 điều kiện không giảm hay không: `age` không giảm (tức là `age[i] <= age[i+1]`) cho mọi cặp kề trong dãy được chọn), `intelligence` không giảm (tương tự).
  - Nếu hợp lệ, tính tích `power` (nhân từng `power` lại). Lưu ý: vì `power` có thể âm hoặc 0, ta phải tính tích chính xác (sử dụng `int` lớn).
  - Cập nhật `res = max(res, current_ans)`.
- Sau khi duyệt tất cả mask, in `res` ra.

2.2 Phân tích độ phức tạp:

- **Độ phức tạp thời gian:**
  Có $2^n - 1$ tập con không rỗng. Với mỗi tập con ta có thể phải quét tới $O(n)$ phần tử (để kiểm tra điều kiện và tính tích).
  Vậy thời gian là:
  $$T(n) = O(2^n \times n)$$
  Với $n \leq 20$, giá trị này khả thi trong thời gian thực (khoảng vài chục triệu phép toán đơn giản).
- **Độ phức tạp không gian:**
  Ta chỉ lưu mảng `warriors` kích thước n và vài biến tạm; do đó:
  $$S(n) = O(n)$$

2.3 Có thể áp dụng thuật toán subtask1 cho subtask2?

Không hiệu quả khi $n \geq 21$ thì độ phức tạp thuật toán quá lớn để có thể **ACC** bài này.

3 Subtask2: Quy hoạch động

3.1 Ý tưởng chính

Bài toán này có thể được quy về một biến thể của bài toán **Tìm dãy con tăng dài nhất (LIS)**. Sau một bước tiền xử lý sắp xếp, bài toán trở thành tìm một dãy con có chỉ số `Intelligence` không giảm và có tích `Power` lớn nhất.

Do `Power` có thể là số âm, một tích lớn nhất có thể được tạo ra bằng cách nhân một tích âm nhỏ nhất với một số âm khác. Vì vậy, tại mỗi bước, ta cần theo dõi cả **tích dương lớn nhất** và **tích âm nhỏ nhất**.

3.2 Các bước thực hiện

Bước 1: Tiền xử lý - Sắp xếp

Đây là bước đột phá. Ta sắp xếp toàn bộ danh sách các chiến binh theo tiêu chí:

1. **Ưu tiên 1:** `Age` tăng dần.
2. **Ưu tiên 2 (nếu `Age` bằng nhau):** `Intelligence` tăng dần.

Sau khi sắp xếp, với hai chiến binh bất kỳ j và i ($j < i$), ta luôn có `warriors[j].age <= warriors[i].age`. Điều kiện để xây dựng dãy hợp lệ giờ đây chỉ còn là `warriors[i].intelligence >= warriors[j].intelligence`.

Bước 2: Định nghĩa Trạng thái DP

Ta sử dụng mảng hai chiều $dp[i][\text{state}]$:

- $dp[i][1]$: Lưu trữ tích `Power` **LỚN NHẤT** của một dãy con hợp lệ kết thúc tại chiến binh i.
- $dp[i][0]$: Lưu trữ tích `Power` **NHỎ NHẤT** (có thể là số âm) của một dãy con hợp lệ kết thúc tại chiến binh i.

Bước 3: Công thức Truy hồi

Ta duyệt qua mỗi chiến binh i từ 1 đến n. Với mỗi i, ta tìm các chiến binh $j < i$ mà i có thể nối tiếp vào.

Khởi tạo (Trường hợp cơ sở): Mỗi chiến binh i có thể tự tạo thành một dãy.

$$dp[i][0] = dp[i][1] = \text{warriors}[i].\text{power}$$

Cập nhật: (Bottom-up) Với i từ 1 đến n:

```
for i in range(1, n + 1):
        dp[i][0] = dp[i][1] = warriors[i].get_power()
        for j in range(i - 1, 0, -1):
            if warriors[i].get_intelligence() >= warriors[j].get_intelligence():
                dp[i][0] = min(
                    dp[i][0],
                    min(dp[j][0] * warriors[i].get_power(), dp[j][1] * warriors[i].get_power())
                )
                dp[i][1] = max(
                    dp[i][1],
                    max(dp[j][0] * warriors[i].get_power(), dp[j][1] * warriors[i].get_power())
                )
```

Bước 4: Tìm kết quả cuối cùng

Lời giải là giá trị lớn nhất trong tất cả các $dp[i][1]$ với i chạy từ 1 đến n.

$$\text{Result} = \max_{1 \le i \le n} (dp[i][1])$$

3.3 Phân tích độ phức tạp

- **Độ phức tạp thời gian:** Bước sắp xếp tốn $O(N \log N)$. Phần quy hoạch động có hai vòng lặp lồng nhau, tốn $O(N^2)$. Vậy độ phức tạp tổng thể là $O(N^2)$.
- **Độ phức tạp không gian:** Mảng lưu các chiến binh và bảng dp đều tốn $O(N)$ không gian.

4 Code minh họa (Python)

```python
class Warrior:
    def __init__(self, age=0, power=0, intelligence=0):
        self.age = age
        self.power = power
        self.intelligence = intelligence

    def get_power(self):
        return self.power

    def get_intelligence(self):
        return self.intelligence

    def __lt__(self, other):
        if self.age != other.age:
            return self.age < other.age
        return self.intelligence < other.intelligence

    def __str__(self):
        return f"{self.age} {self.intelligence} {self.power}"


def solve():
    n = int(input())
    warriors = [None]
    for _ in range(n):
        age, intel, power = map(int, input().split())
        warriors.append(Warrior(age, power, intel))

    warriors[1:] = sorted(warriors[1:])
```

```
    dp = [[1, 1] for _ in range(n + 1)]
    res = -10**18

    for i in range(1, n + 1):
        dp[i][0] = dp[i][1] = warriors[i].get_power()
        for j in range(i - 1, 0, -1):
            if warriors[i].get_intelligence() >= warriors[j].get_intelligence():
                dp[i][0] = min(
                    dp[i][0],
                    min(dp[j][0] * warriors[i].get_power(), dp[j][1] * warriors[i].get_power())
                )
                dp[i][1] = max(
                    dp[i][1],
                    max(dp[j][0] * warriors[i].get_power(), dp[j][1] * warriors[i].get_power())
                )

        res = max(res, dp[i][1])

    print(res)


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