BÁO CÁO BÀI TẬP: TRẬN THƯ HÙNG Ở NAMEK

Nhóm 9
Môn học: Cấu trúc dữ liệu và giải thuật
Ngôn ngữ lập trình: Python

October 28, 2025

1. Mô tả bài toán

Son Goku cần chọn ra các chiến binh sao cho nếu có hai người bất kỳ trong nhóm được chọn, thì người có `Age` nhỏ hơn hoặc bằng phải đồng thời có `Intelligence` nhỏ hơn hoặc bằng người còn lại.

Mục tiêu: tìm **tích Power lớn nhất** của tập hợp chiến binh hợp lệ

Input:

- Số nguyên n – số chiến binh ($1 \leq n \leq 1000$).
- n dòng, mỗi dòng gồm 3 số nguyên a, b, c – lần lượt là Age, Intelligence, Power.

Output: Một số nguyên duy nhất – tích Power lớn nhất có thể đạt được.

Test ví dụ:

```
Input:
5
1 7 -10
2 8 10
3 11 -7
4 12 -4
5 15 0

Output:
700
```

—

2. Mô tả thuật toán

Subtask 1 (n $\leq$ 20): Duyệt toàn bộ tập con – Brute Force

- Sắp xếp danh sách chiến binh theo Age, nếu trùng thì theo Intelligence.
- Liệt kê tất cả các tập con chiến binh.
- Với mỗi tập con, kiểm tra điều kiện "không giảm theo Age và Intelligence".
- Nếu hợp lệ, tính tích Power và cập nhật giá trị lớn nhất.

Độ phức tạp thời gian:

$$O(2^n \cdot n)$$

Độ phức tạp không gian:

$$O(n)$$

—

Subtask 2 (n $\leq$ 1000): Quy hoạch động

Do n lớn nên không thể dùng brute force. Ta cần dùng **Dynamic Programming (DP)**:

- Sắp xếp danh sách chiến binh theo Age, nếu trùng thì theo Intelligence.
- Gọi:
  - $max\_dp[i]$: tích Power lớn nhất kết thúc tại chiến binh i.
  - $min\_dp[i]$: tích Power nhỏ nhất kết thúc tại chiến binh i (để xử lý số âm).
- Với mỗi i, xét tất cả $j < i$, nếu $Age[j] \leq Age[i]$ và $Int[j] \leq Int[i]$, ta cập nhật:

$$max\_dp[i] = \max(max\_dp[i], power_i \times max\_dp[j], power_i \times min\_dp[j])$$

$$min\_dp[i] = \min(min\_dp[i], power_i \times max\_dp[j], power_i \times min\_dp[j])$$

- Kết quả cuối cùng là $\max(max\_dp[i])$ cho mọi i.

  Độ phức tạp thời gian:

  $$O(n^2)$$

  Độ phức tạp không gian:

  $$O(n)$$

—

3. Cài đặt thuật toán (Python – có chú thích chi tiết)

```
n = int(input())

# Khai báo danh sách lưu thông tin từng chiến binh
# Mỗi phần tử là bộ (Age, Intelligence, Power)
warriors = []
for _ in range(n):
    a, b, c = map(int, input().split())
    warriors.append((a, b, c))

# Sắp xếp chiến binh theo Age tăng dần,
# Nếu Age trùng thì sắp xếp tiếp theo Intelligence tăng dần.
warriors.sort(key=lambda x: (x[0], x[1]))

# Bước 2: Khởi tạo hai mảng DP:
# max_dp[i] = tích Power lớn nhất kết thúc tại chiến binh i
# min_dp[i] = tích Power nhỏ nhất kết thúc tại chiến binh i
max_dp = [0] * n
min_dp = [0] * n

# Duyệt qua từng chiến binh
for i in range(n):
    power_i = warriors[i][2]   # Power của chiến binh i
    max_dp[i] = min_dp[i] = power_i  # Ban đầu, dãy chỉ gồm chính chiến binh i

    # So sánh với các chiến binh j đứng trước i
    for j in range(i):
        age_j, intel_j, power_j = warriors[j]

        # Điều kiện để có thể nối j vào trước i:
        # cả Age và Intelligence của j đều <= i
        if age_j <= warriors[i][0] and intel_j <= warriors[i][1]:

            # Ba trường hợp có thể xảy ra khi nhân power_i với dãy trước đó:
            # 1. power_i * max_dp[j]  -> khi cả hai dương hoặc cùng âm
            # 2. power_i * min_dp[j]  -> khi power_i âm, đảo dấu kết quả
            # 3. chỉ chọn riêng power_i (bắt đầu dãy mới)
            candidate_products = [
                power_i * max_dp[j],
                power_i * min_dp[j],
                power_i
            ]

            # Cập nhật giá trị tốt nhất và tệ nhất có thể tại i
            max_dp[i] = max(max_dp[i], max(candidate_products))
            min_dp[i] = min(min_dp[i], min(candidate_products))
```

```
# Bước 3: Kết quả là giá trị lớn nhất trong toàn bộ mảng max_dp
print(max(max_dp))
```