# Trận thư hùng ở Namek - Lời Giải

Nhóm 3

October 2025

## 1 Tóm tắt đề bài

Ta được cho $n$ chiến binh Z, mỗi chiến binh có ba chỉ số `(age, intelligence, power)`. Ta phải Chọn một tập hợp các chiến binh trong $n$ chiến binh ban đầu, sao cho 2 chiến binh bất kì trong số những chiến binh được chọn luôn có 1 người có `age` và intelligence nhỏ hơn người còn lại. Ta cần chọn ra tập hợp sao cho tích `power` của các chiến binh được chọn là lớn nhất.

**Input**:

- Dòng 1: số nguyên $n$ ($1 \leq n \leq 1000$).
- $n$ dòng tiếp theo: mỗi dòng là $a, b, c$ với $a, b \in [1, 10^5]$, $c \in [-200, 200]$.

**Output**: một số nguyên duy nhất là tích power lớn nhất của các chiến binh ta chọn.

## 2 Mô tả thuật toán

### Subtask 1 (40%): $n \leq 20$ – Duyệt toàn bộ (brute force)

**Ý tưởng** : Duyệt tất cả các tập con của $n$ chiến binh. Với mỗi tập:

1. Sắp xếp các phần tử trong tập theo khoá ($age, intelligence$) không giảm.
2. Kiểm tra tính hợp lệ: dãy sau sắp xếp phải đồng thời không giảm theo *age* và *intelligence*. Tương đương, với mọi cặp liên tiếp $(a_i, b_i)$ và $(a_{i+1}, b_{i+1})$ ta cần $a_i \leq a_{i+1}$ và $b_i \leq b_{i+1}$.
3. Nếu hợp lệ, tính tích power của cả tập và cập nhật kết quả lớn nhất.

**Vì sao dùng được** : Với $n \leq 20$, số tập con là $2^n$, đủ nhỏ để duyệt hết. Việc sắp xếp và kiểm tra cho mỗi tập con tốn thêm $O(k \log k)$ và $O(k)$ với $k$ là kích thước tập, nên vẫn thoải mải về giới hạn thời gian.

### Subtask 2 (60%): $n \leq 1000$ – Quy hoạch động

**Giải pháp từ sub 1 không hợp lí:** Vì giới hạn của giữ liệu bây giờ lớn hơn ($n \leq 1000$), nên ta không thể duyệt hết mọi tập con được nữa.

**Bước chuẩn hoá dữ liệu** Sắp xếp toàn bộ chiến binh theo khoá (*age*, *intelligence*) không giảm. Thêm một phần tử giả $(0, 0, 1)$ đứng đầu để khởi tạo tích = 1.

**Ý tưởng Quy Hoạch Động:** Ta sẽ sử dụng quy hoạch động tập con, gọi $dp[i]$ là tích các tập con có phần tử cuối cùng chính là i lớn nhất. Lúc đó, ta sẽ dễ dàng xây dựng từng tập con từ phần tử có (age, intelligence) thấp nhất đến lớn nhấtcon (tương đương với thứ tự trái qua phải sau khi sort). Để kiểm soát điều kiện (age, intelligence), ta đơn giản chỉ cần kiểm soát giữa 2 phần tử kề nhau trong quá trình xây dựng tập con

**Nhận xét:** Vì tích có thể âm/0/dương (do $c$ có thể âm), khi mở rộng tập con, giá trị lớn nhất có thể đến từ việc nhân một giá trị nhỏ nhất âm với một $c$ âm để thành dương. Do đó, tại mỗi vị trí $i$, ta cần lưu cả giá trị tích lớn nhất và nhỏ nhất có thể đạt được cho một chuỗi kết thúc tại phần tử đó.

**Trạng thái Quy Hoạch Động:**

- $dp\_\max[i]$: tích lớn nhất của một dãy hợp lệ kết thúc tại phần tử $i$.
- $dp\_\min[i]$: tích nhỏ nhất (có thể âm) của một dãy hợp lệ kết thúc tại phần tử $i$.

**Chuyển trạng thái:** Với mỗi $i$ từ 1 đến $n$ (sau khi đã sắp xếp), xét mọi $j < i$. Nếu ($a_j \leq a_i$ **và** $b_j \leq b_i$) thì phần tử $i$ có thể nối sau $j$. Khi đó:

$$dp\_\max[i] = \max\big(dp\_\max[i],\ dp\_\max[j] \cdot c_i,\ dp\_\min[j] \cdot c_i\big),$$

$$dp\_\min[i] = \min\big(dp\_\min[i],\ dp\_\max[j] \cdot c_i,\ dp\_\min[j] \cdot c_i\big).$$

Kết quả cuối cùng là giá trị $dp\_\max[i]$ lớn nhất ($1 \leq i \leq n$).

**Các bước của thuật toán:**

1. **Chuẩn hoá dữ liệu:** Sắp xếp toàn bộ các chiến binh theo khóa $(age, intelligence)$ không giảm. Thêm một phần tử giả $(0, 0, 1)$ đứng đầu để khởi tạo tích ban đầu bằng 1.
2. **Khởi tạo DP:** Với mỗi phần tử $i$, đặt:

$$dp_{\max}[i] = -INF$$

$$dp_{\min}[i] = INF.$$

3. **Duyệt theo thứ tự đã sắp xếp:** Với mỗi $i$ từ 1 đến $n$, xét tất cả $j < i$.
4. **Kiểm tra điều kiện nối chuỗi:** Nếu ($a_j \le a_i$ **và** $b_j \le b_i$) thì phần tử $i$ có thể nối sau $j$.
5. **Chuyển trạng thái DP:** Cập nhật:

$$dp_{\max}[i] = \max\left(dp_{\max}[i],\ dp_{\max}[j] \cdot c_i,\ dp_{\min}[j] \cdot c_i\right),$$

$$dp_{\min}[i] = \min\left(dp_{\min}[i],\ dp_{\max}[j] \cdot c_i,\ dp_{\min}[j] \cdot c_i\right).$$

6. **Kết quả:** Kết quả cần tìm là:

$$\max_{1 \le i \le n} dp_{\max}[i].$$

# 3 Phân tích độ phức tạp

### Subtask 1

**Thời gian**: $O\left(2^n \cdot n \log n\right)$ (trong đó mỗi tập con cỡ $k$ sắp xếp $O(k \log k)$ và kiểm tra $O(k)$).
**Không gian**: $O(n)$ cho lưu tập con tạm thời (bỏ qua input).

### Subtask 2

**Thời gian**: $O(n^2)$ do hai vòng lặp $i, j$ sau khi sắp xếp.
**Không gian**: $O(n)$ cho hai mảng $dp\_\max, dp\_\min$.

# 4 Cài đặt (Python) – Subtask 2

**Lưu ý**: Chỉ yêu cầu cài đặt cho Subtask 2. Mã nguồn dưới đây bám sát ý tưởng DP ở trên, có chú thích rõ ràng cho từng dòng (phần nhập không bắt buộc chú thích theo yêu cầu đề).

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

# Giới hạn lớn nhất của đáp án (200 ^ 1000)
INF = 200 ** 1000

if __name__ == "__main__":
    n = int(input())

    dp_min = [INF] * (n + 1)
```

```
    dp_max = [-INF] * (n + 1)
    dp_max[0] = dp_min[0] = 1

    v = []
    v.append((0, 0, 1))
    for i in range(1,  n + 1):
        a, b, c = map(int, input().split())
        v.append((a, b, c))

    # Ta sort lại input để dễ dàng kiểm soát điều kiện tăng dần
    v.sort()

    # Đặt đáp án ban đầu là thấp nhất có thể
    res = -INF

    # Duyệt qua các phần tử i
    # Tượng trưng cho các tập hợp kết thúc bằng phần tử i
    for i in range(1, n + 1):i
        # Lấy các giá trị (age, intelligence, power) của phần tử i
        a, b, c = v[i]

        # Duyệt các phần tử j trước đó
        # Tượng trưng cho các tập hợp kết thúc bằng phần tử j
        for j in range(0, i):
            # Các tập con có phần tử kết thúc ở j
            # có thể chọn i là phần tử kế tiếp
            if (v[j][0] <= v[i][0] and v[j][1] <= v[i][1]):
                # Dựng dp cho tích lớn nhất
                # Công thức gồm 2 trường hợp tích trước đó là âm hay dương
                dp_max[i] = max(dp_max[i], dp_max[j] * c)
                dp_max[i] = max(dp_max[i], dp_min[j] * c)

                # Dựng dp cho tích bé nhất
                # Công thức gồm 2 trường hợp tích trước đó là âm hay dương
                dp_min[i] = min(dp_min[i], dp_max[j] * c)
                dp_min[i] = min(dp_min[i], dp_min[j] * c)

        # Lấy dp_max[i] lớn nhất với mọi i làm đáp án
        res = max(res, dp_max[i])

    # In ra đáp án cuối cùng, tích lớn nhất
    print(res)
```