# Trận thư hùng ở Namek

Báo cáo bài lập trình Quy hoạch động (Dynamic Programming)

Trần Xuân Đạt - Nguyễn Quốc Phú - Nhóm 13 – UIT

## 1. Mô tả thuật toán

### Subtask 1: Brute Force (Vét cạn)

Thông tin thuật toán

**Phương pháp:** Duyệt tất cả tập con (subset brute force)
**Áp dụng khi:** $n \leq 20$
**Lý do:** Với $n$ nhỏ, $2^n$ tập con vẫn trong giới hạn thời gian

**Các bước thực hiện:**

1. **Sinh tất cả tập con:** Duyệt qua $2^n$ tập con của $n$ chiến binh
2. **Kiểm tra tính hợp lệ:** Với mỗi tập con:
   - Sắp xếp theo `Age` tăng dần (nếu trùng thì theo `Intelligence`)
   - Kiểm tra `Intelligence` có tạo thành dãy không giảm
3. **Tính toán kết quả:**
   - Tính tích `Power` của tất cả chiến binh trong tập con hợp lệ
   - Cập nhật giá trị lớn nhất tìm được

**Độ phức tạp:**

- **Thời gian:** $O(2^n \cdot n \log n)$
- **Không gian:** $O(n)$

### Subtask 2: Dynamic Programming (Quy hoạch động)

Thông tin thuật toán

**Phương pháp:** Quy hoạch động kết hợp sắp xếp
**Áp dụng khi:** $n = 1000$
**Lý do:** $2^{1000}$ quá lớn, cần thuật toán hiệu quả hơn

**Ý tưởng chính:**

- Sau khi sắp xếp theo (`Age`, `Intelligence`), bài toán trở thành tìm dãy con tăng với tích lớn nhất
- Do `Power` có thể âm, cần theo dõi cả giá trị lớn nhất và nhỏ nhất

**Các bước thực hiện:**

1. **Sắp xếp:** Sắp xếp chiến binh theo (`Age`, `Intelligence`) tăng dần
2. **Khởi tạo DP:**
   - `dp_max[i]` = tích lớn nhất kết thúc tại chiến binh i
   - `dp_min[i]` = tích nhỏ nhất kết thúc tại chiến binh i
   - Khởi tạo: `dp_max[i] = dp_min[i] = power[i]`
3. **Cập nhật DP:** Với mỗi cặp ($j < i$) thỏa điều kiện:
   - **Nếu power[i]  0:**

$$\text{dp\_max[i]} = \max(\text{dp\_max[i]}, \text{dp\_max[j]} \times \text{power[i]})$$
$$\text{dp\_min[i]} = \min(\text{dp\_min[i]}, \text{dp\_min[j]} \times \text{power[i]})$$

   - **Nếu power[i] $<$ 0:**

$$\text{dp\_max[i]} = \max(\text{dp\_max[i]}, \text{dp\_min[j]} \times \text{power[i]})$$
$$\text{dp\_min[i]} = \min(\text{dp\_min[i]}, \text{dp\_max[j]} \times \text{power[i]})$$

4. **Kết quả:** Giá trị lớn nhất trong mảng `dp_max`

**Độ phức tạp:**

- **Thời gian:** $O(n^2)$
- **Không gian:** $O(n)$

## 2. Cài đặt thuật toán

Mã nguồn Python

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

def solve():
    n = int(input())
    fighters = []

    for _ in range(n):
        a, i, p = map(int, input().split())
        fighters.append((a, i, p))

    # Sort by Age then Intelligence
    fighters.sort()

    # Initialize DP array
    dp_max = [0] * n
    dp_min = [0] * n
    result = -10**18

    # Browse each warrior
    for i in range(n):
        age_i, intel_i, power_i = fighters[i]
        dp_max[i] = power_i
        dp_min[i] = power_i

        # Browse previous warriors to update DP
        for j in range(i):
            age_j, intel_j, power_j = fighters[j]

            # Check condition: Age and Intelligence do not
                decrease
            if age_j <= age_i and intel_j <= intel_i:
                if power_i >= 0:
                    # Positive power: multiply by max to get max
                        , multiply by min to get min
                    dp_max[i] = max(dp_max[i], dp_max[j] *
                        power_i)
                    dp_min[i] = min(dp_min[i], dp_min[j] *
                        power_i)
                else:
```

```
                    # Negative power: multiply by min to get max
                        , multiply by max to get min
                    dp_max[i] = max(dp_max[i], dp_min[j] *
                        power_i)
                    dp_min[i] = min(dp_min[i], dp_max[j] *
                        power_i)

        # Update best result
        result = max(result, dp_max[i])

    print(result)

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

# 3. Phân tích độ phức tạp

## Subtask 1: Brute Force

- **Thời gian:** $O(2^n \cdot n \log n)$ - Chấp nhận được với $n \leq 20$
- **Không gian:** $O(n)$ - Lưu trữ danh sách chiến binh

## Subtask 2: Dynamic Programming

- **Thời gian:** $O(n^2)$ - Hiệu quả với $n = 1000$
- **Không gian:** $O(n)$ - Hai mảng DP kích thước $n$

**Kết luận**

Thuật toán sử dụng quy hoạch động với việc quản lý cả giá trị lớn nhất và nhỏ nhất đã giải quyết hiệu quả bài toán, xử lý được trường hợp `Power` có thể âm. Việc sắp xếp trước giúp đơn giản hóa việc kiểm tra điều kiện về `Age` và `Intelligence`.

---

**– Hết –**