Bài 3: Trận thư hùng ở Namek

Báo cáo bài tập về nhà

Mô tả bài toán

Son Goku chuẩn bị lên đường đến hành tinh Namek để tìm ngọc rồng hồi sinh những người bạn của anh ta. Trên hành tinh đó có phản diện rất mạnh là Frieza.

Mỗi chiến binh của Trái Đất có 3 chỉ số: Age, Intelligence và Power.

Goku chỉ chọn những chiến binh sao cho với mọi cặp chiến binh trong tập hợp, một người phải có cả Age và Intelligence nhỏ hơn hoặc bằng người còn lại. Nói cách khác, các chiến binh được chọn phải sắp xếp thành dãy không giảm cả về Age lẫn Intelligence.

Mục tiêu: Tìm tích Power lớn nhất của các chiến binh được chọn.

Input/Output

- Input: Dòng đầu tiên n ($1 \leq n \leq 1000$), tiếp theo n dòng mỗi dòng 3 số nguyên a, b, c (Age, Intelligence, Power)
- Output: Một số nguyên duy nhất: tích Power lớn nhất

Subtask 1: n $\leq$ 20

Thuật toán

- Sử dụng **brute force**: sinh tất cả tập con chiến binh.
- Kiểm tra với mỗi tập con: Age và Intelligence không giảm.
- Nếu hợp lệ, tính tích Power, lưu giá trị lớn nhất.
- Lý do: Với n nhỏ, số tập con 2^n vẫn chấp nhận được.

Code Python Subtask 1

```
from itertools import combinations  # import de sinh tat
    ca tap con
import math
```

```
n = int(input())  # nhap so luong chien binh
fighters = [tuple(map(int, input().split())) for _ in
   range(n)]  # nhap danh sach chien binh

max_product = -float('inf')  # khoi tao tich lon nhat

for size in range(1, n+1):  # duyet kich thuoc tap con
    for subset in combinations(fighters, size):  # sinh
       tat ca tap con
        subset = sorted(subset, key=lambda x: x[0])  #
           sap xep theo age
        valid = all(subset[i][1] <= subset[i+1][1] for i
            in range(len(subset)-1))  # kiem tra
           intelligence khong giam
        if valid:
            product = math.prod(x[2] for x in subset)  #
                tinh tich power
            max_product = max(max_product, product)  #
               cap nhat tich lon nhat

print(max_product)  # in ket qua
```

Phân tích độ phức tạp Subtask 1

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

Subtask 2: n ≤ 1000

Thuật toán

- Brute force không khả thi do 2^{1000} quá lớn.
- Sử dụng **Dynamic Programming** (DP):
  - Sắp xếp chiến binh theo Age, nếu bằng nhau thì Intelligence.
  - dp[i] = tích Power lớn nhất kết thúc tại chiến binh i.
  - Với mỗi i, duyệt j $<$ i: nếu j có thể đứng trước i, cập nhật dp[i].
  - Kết quả = max(dp)

Code Python Subtask 2

```
n = int(input())  # nhap so luong chien binh
fighters = [tuple(map(int, input().split())) for _ in
   range(n)]  # nhap danh sach chien binh

fighters.sort(key=lambda x: (x[0], x[1]))  # sap xep
   theo age, neu bang nhau theo intelligence

dp = [f[2] for f in fighters]  # khoi tao dp[i] = power
   cua chien binh i

for i in range(n):  # duyet tung chien binh i
    for j in range(i):  # duyet cac chien binh j truoc i
        if fighters[j][0] <= fighters[i][0] and fighters
           [j][1] <= fighters[i][1]:  # kiem tra dieu
           kien age va intelligence
            dp[i] = max(dp[i], dp[j] * fighters[i][2])
               # cap nhat dp[i] voi tich lon nhat

print(max(dp))  # in ket qua tich power lon nhat
```

Phân tích độ phức tạp Subtask 2

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