Chuyên đề QHĐ

Nhóm 17

Bùi Huỳnh Tây - 24521589

Phạm Ngọc Thọ - 24520032

Tóm tắt đề bài

Có **N vật thể**, mỗi vật thể có **3 trọng số là a, b, c**.

Một vật thể **X được coi là nhỏ hơn vật thể Y** khi $X_a \leq Y_a$ và $Y_a <= Y_b$

Yêu cầu:

Chọn ra **một tập vật thể** sao cho **tích c của chúng là lớn nhất** và với mỗi **cặp bất kỳ trong tập**, phải có **một vật thể nhỏ hơn vật thể còn lại.**

Subtask 1: N ≤ 20

Đề yêu cầu tạo ra **một tập con** chứa các vật thể sao cho **tích c là lớn nhất**, mà **N rất nhỏ (N ≤ 20)**

→ Ta có thể **sinh ra dãy nhị phân** với ý nghĩa:

- **bin(N, i) = 1** → chọn vật thể *i* vào tập.
- **bin(N, i) = 0** → không chọn vật thể *i* vào tập.

Sau đấy, **tính tích của các vật thể đã chọn** và **so sánh để tìm max với biến** `res`

Trong đó:

- **bin(N, i)**: giá trị biểu diễn thứ *i* của số *N* bằng mã nhị phân.
- **ans**: tích của các vật thể được chọn vào tập con.

Từ đó `res` **chính là đáp án của bài toán.**

Phân tích

- **Độ phức tạp thời gian:**

- Nhập mảng input: $O(N)$
- Sinh nhị phân: $O(2^N)$

⇒ **Tổng thời gian ≈** $O(2^N)$

- **Độ phức tạp không gian:**
  - Mảng input: $O(N)$
  - Mảng dãy nhị phân: $O(N)$

⇒ **Tổng không gian ≈** $O(N)$

Subtask 2: N ≤ 1000

Với **N ≤ 1000**, việc **sinh nhị phân là không thể**, nên phải **suy nghĩ theo hướng khác.**

Nhận thấy: Khi vật thể **X < Y** và **Y < Z**, thì có thể nói **X < Z**.

Vì vậy việc **tính kết quả của Z dựa vào Y mà Y lại dựa vào X** hoàn toàn **không vi phạm ràng buộc** của đề bài. Do đó, các bài toán này **gối nhau**, và ta có thể **tiếp cận bằng Quy hoạch động (Dynamic Programming)**.

Ý tưởng

- **Gọi:**
  - `f[0][i]` : **giá trị max** khi xét tới vị trí *i*
  - `f[1][i]` : **giá trị min** khi xét tới vị trí *i*

Nhận thấy tại vị trí *i* có **giá trị c[i] âm**, thì nên **update f[1][i] (min)** với giá trị min đạt được trước đó.

Tuy nhiên, nếu **giá trị max trước đó × c[i]** lại tạo ra **giá trị nhỏ hơn**, thì ta cũng cần xét đến trường hợp đó.

Công thức cập nhật **min** với mỗi **j < i** như sau:

$$f[1][i] = min\left(f[1][i], min\left(f[1][j] \times c[i], f[0][j] \times c[i]\right)\right)$$

Tương tự, công thức cập nhật **max** là:

$$f[0][i] \;=\; max\left(f[0][i],\, max\left(f[1][j] \;\times\; c[i],\, f[0][j] \;\times\; c[i]\right)\right)$$

Phân tích

- **Độ phức tạp thời gian:**
  - Nhập mảng input: $O(N)$
  - Tính kết quả DP: $O(N^2)$

  ⇒ Tổng thời gian ≈ $O(N^2)$
- **Độ phức tạp không gian:**
  - Mảng input: $O(N)$
  - Mảng DP (f[0], f[1]): $O(2N)$

  ⇒ **Tổng không gian ≈** $O(N)$

Code hoàn chỉnh

```python
n = int(input())

vector = []
vector.append((0, 0, 1))
# Tạo vật thể giả để index bắt đầu từ 1
# và vì giá trị (a, b) ≥ 1 nên tự động cập nhật c vào chính kết quả của nó

#Nhập input đầu vào thành 1 tuple(a, b, c)
for i in range(n):
    inp = list(map(int, input().split()))
    vector.append((inp[0], inp[1], inp[2]))

# Sort giá trị theo key (a, b)
vector.sort(key = lambda val : (val[0], val[1]))

# Hàm kiểm tra xem i có nhỏ hơn j hay không
def valid(i, j):
    return (vector[i][0] ≤ vector[j][0] and vector[i][1] ≤ vector[j][1])

inf = int(1e9)
# Khởi tạo mảng f với ý nghĩa f[0][i]: kết quả tốt nhất đạt được, f[1][i] là kết quả xấu nhất đạt được
f = [[-inf] * 1005, [inf] * 1005]

# Mặc định vật thể giả có giá trị bằng 1 để tính được giá trị của chính nó
f[0][0], f[1][0] = 1, 1

for i in range(1, n + 1): # for từ 1 tới n
    for j in range(i - 1, -1, -1): # for từ i - 1 đi lùi về 0
        if(valid(j, i)):
            # giá trị lớn nhất tại i sẽ được cập nhật bởi giá trị lớn nhất tại j hoặc giá trị nhỏ nhất tại j
            f[0][i] = max(f[0][i], max(f[0][j] * vector[i][2], f[1][j] * vector[i][2]))
            # giá trị nhỏ nhất tại i sẽ được cập nhật bởi giá trị lớn nhất tại j hoặc giá trị nhỏ nhất tại j
            f[1][i] = min(f[1][i], min(f[0][j] * vector[i][2], f[1][j] * vector[i][2]))

res = -inf # Tạo biến kết quả nhỏ vô cùng
for i in range(1, n + 1): # chỉ lấy kết quả từ 1 --> n
    res = max(res, f[0][i]) # Tìm max của những giá trị lớn nhất

print(res)
```