Trận thư hùng ở Namek

Nhóm 2:
Nguyễn Ngọc Hưng – Nguyễn Trường Giang

29/10/2025

Mục lục

1 Phát biểu bài toán

Có n chiến binh. Mỗi chiến binh được mô tả bằng bộ ba số nguyên:

$$(a, b, c) = (\texttt{age},\ \texttt{intelligence},\ \texttt{power}),$$

trong đó `power` có thể âm, dương hoặc bằng 0.
Ta muốn chọn ra một đội hình hợp lệ (không rỗng) sao cho:

- Nếu sắp xếp các thành viên trong đội theo `age` không giảm, thì dãy `intelligence` tương ứng cũng không giảm.

Nói cách khác: ta chỉ được chọn một dãy chiến binh sao cho cả `age` **không giảm** và `intelligence` **không giảm** theo thứ tự đã chọn.
Giá trị của một đội là **tích tất cả các** `power` trong đội.
Mục tiêu: **tối đa hoá tích này**.
Lưu ý:

- Ta có thể chọn đúng một người (khi đó tích chỉ là `power` của người đó).
- Tích có thể âm hoặc dương. Hai số âm nhân nhau thành số dương rất lớn, nên trạng thái "âm cực to" vẫn có thể trở thành lời giải tối ưu sau khi nhân thêm một số âm khác.

Ràng buộc:

- Subtask 1: $n \leq 20$.
- Subtask 2: $n \leq 1000$.
- `power` nằm trong khoảng $[-200, 200]$.

2 Subtask 1 ($n \leq 20$)

2.1 Ý tưởng thuật toán

Với $n \leq 20$, số lượng giá trị `age` và `intelligence` phân biệt cũng nhỏ. Ta làm như sau:

1. Nén toạ độ:

   - Biến tất cả giá trị `age` khác nhau thành chỉ số $0, 1, 2, \ldots, A-1$.
   - Biến tất cả giá trị `intelligence` khác nhau thành chỉ số $0, 1, 2, \ldots, B-1$.

   Việc nén này giúp ta có thể dùng một bảng quy hoạch động kích thước $A \times B$ mà $A, B \leq n \leq 20$.

2. Sắp xếp danh sách chiến binh theo (a, b) tăng dần. Như vậy, nếu ta duyệt tuần tự qua danh sách này thì `age` không giảm; với các phần tử có cùng `age` thì `intelligence` cũng không giảm.

3. Duy trì hai mảng 2 chiều:

   - `dp_max[x][y]`: tích lớn nhất (dương) của một dãy hợp lệ kết thúc ở một chiến binh có chỉ số nén ($\texttt{age\_idx} = x,\ \texttt{int\_idx} = y$).

- `dp_min[x][y]`: tích nhỏ nhất (âm) của một dãy hợp lệ kết thúc ở một chiến binh có chỉ số nén (`age_idx` $= x$, `int_idx` $= y$).

4. Xét từng chiến binh hiện tại (a, b, c) với chỉ số nén (x, y). Ta muốn biết "nếu kết thúc bằng người này" thì tích lớn nhất và nhỏ nhất là bao nhiêu. Gọi hai biến tạm:

$$\texttt{best\_max} = c, \quad \texttt{best\_min} = c$$

(trường hợp chọn đúng người này, tức dãy độ dài 1).

5. Ta thử nối người này vào **mọi trạng thái trước đó** (i, j) sao cho $i \le x$ và $j \le y$. Điều kiện $i \le x$, $j \le y$ tương ứng với việc `age` và `intelligence` không giảm khi ta thêm chiến binh hiện tại vào sau.

   Nếu đã từng có một dãy kết thúc tại (i, j) với tích T, thì khi thêm người hiện tại (power c), ta có tích mới $T \cdot c$. Ta so sánh các $T \cdot c$ đó để cập nhật lại `best_max` (lớn nhất) và `best_min` (nhỏ nhất). Lưu ý: phải xét cả `dp_max[i][j]` và `dp_min[i][j]` vì nếu $c < 0$ thì "nhỏ nhất" $\times c$ có thể thành "lớn nhất" mới.

6. Cuối cùng, cập nhật:

   - `dp_max[x][y] = max(dp_max[x][y], best_max)`
   - `dp_min[x][y] = min(dp_min[x][y], best_min)`

   Đồng thời cập nhật đáp án toàn cục = giá trị lớn nhất từng thấy của `best_max`.

2.2 Độ phức tạp

Gọi A là số giá trị `age` phân biệt, B là số giá trị `intelligence` phân biệt. Rõ ràng $A \le n$, $B \le n$.

- **Độ phức tạp thời gian:** Với mỗi chiến binh ta quét qua toàn bộ ô (i, j) với $i \le x$, $j \le y$, nên xấp xỉ $O(A \cdot B)$ cho mỗi chiến binh. Do đó tổng thời gian cỡ $O(n \cdot A \cdot B)$, và trong trường hợp xấu nhất $A, B \approx n$, tức $O(n^3)$. Với $n \le 20$ thì vẫn chạy tốt.
- **Độ phức tạp không gian:** Ta lưu hai bảng $A \times B$, nên bộ nhớ là $O(A \cdot B)$, tối đa khoảng $O(n^2)$.

2.3 Mã nguồn Python (Subtask 1)

```
import sys
readline = sys.stdin.readline

n = int(readline())

warriors = []
ages = []
ints = []

for _ in range(n):
    a, b, c = map(int, readline().split())
    warriors.append((a, b, c))
    ages.append(a)
    ints.append(b)
```

```
ages_sorted = sorted(set(ages))
ints_sorted = sorted(set(ints))

pos_age = {v: i for i, v in enumerate(ages_sorted)}
pos_int = {v: i for i, v in enumerate(ints_sorted)}

A = len(ages_sorted)
B = len(ints_sorted)

NEG = -(10**1000)
INF = 10**1000

dp_max = [[NEG for _ in range(B)] for __ in range(A)]
dp_min = [[INF for _ in range(B)] for __ in range(A)]

warriors.sort(key=lambda w: (w[0], w[1]))

answer = NEG

for (a, b, pwr) in warriors:
    x = pos_age[a]
    y = pos_int[b]

    best_max = pwr
    best_min = pwr

    for i in range(x + 1):
        for j in range(y + 1):
            if dp_max[i][j] != NEG:
                cand = dp_max[i][j] * pwr
                if cand > best_max:
                    best_max = cand
                if cand < best_min:
                    best_min = cand
            if dp_min[i][j] != INF:
                cand = dp_min[i][j] * pwr
                if cand > best_max:
                    best_max = cand
                if cand < best_min:
                    best_min = cand

    if best_max > dp_max[x][y]:
        dp_max[x][y] = best_max
    if best_min < dp_min[x][y]:
        dp_min[x][y] = best_min

    # Cp nht p n ton cc
    if best_max > answer:
        answer = best_max

print(answer)
```

3 Subtask 2 ($n \leq 1000$)

3.1 Ý tưởng thuật toán

Khi n tăng lên đến 1000, ta muốn lời giải $O(n^2)$.

Ta coi dãy chiến binh sau khi sắp xếp theo (`age`, `intelligence`) là:

$$(a_1, b_1, c_1),\ (a_2, b_2, c_2),\ \ldots,\ (a_n, b_n, c_n)$$

(sau khi sort theo (`age`, `intelligence`)).

Ý tưởng là duyệt theo thứ tự đó và xây dựng dần dãy hợp lệ kết thúc ở từng chiến binh.

Ta dùng hai mảng:

- `get_max[i]`: tích **lớn nhất** có thể đạt được của một dãy hợp lệ kết thúc đúng tại chiến binh i.
- `get_min[i]`: tích **nhỏ nhất** (âm sâu nhất) có thể đạt được của một dãy hợp lệ kết thúc đúng tại chiến binh i.

Giải thích:

- Khi xét chiến binh i có chỉ số sức mạnh z, ta khởi tạo trạng thái riêng lẻ: dãy chỉ gồm người này. Nếu $z \geq 0$ thì đây là một ứng viên tốt cho `get_max[i]`, nếu $z \leq 0$ thì đây là một ứng viên tốt cho `get_min[i]`.
- Sau đó, ta thử nối chiến binh i vào sau mỗi chiến binh $j < i$ miễn là $\texttt{intel}_j \leq \texttt{intel}_i$ để đảm bảo dãy `intelligence` không giảm.
- Nếu nối được, ta xét các trường hợp nhân:
  - `get_max[j]` $\times z$ có thể ra ứng viên mới cho `get_max[i]` nếu cùng dấu dương.
  - `get_min[j]` $\times z$ có thể ra ứng viên mới cho `get_max[i]` nếu cả hai đều âm.
  - Tương tự, khi z âm nhân với `get_max[j]` dương có thể cho ra giá trị âm mới nhỏ hơn, cập nhật `get_min[i]`; và khi z dương nhân với `get_min[j]` âm cũng cho ra âm mới nhỏ hơn.
- Cuối cùng, đáp án toàn cục là `max` của tất cả `get_max[i]` và cả từng `power` đơn lẻ.

Mấu chốt vẫn như Subtask 1: ta luôn phải giữ cả "tích tốt nhất" và "tích tệ nhất", vì số âm có thể lật dấu.

3.2 Độ phức tạp

- **Độ phức tạp thời gian:** Với mỗi chiến binh i ta duyệt tất cả $j < i$, nên tổng số cặp (j, i) là $\frac{n(n-1)}{2} = O(n^2)$. Mỗi lần duyệt chỉ làm vài phép nhân và so sánh hằng số. Kết luận: $O(n^2)$ thời gian.
- **Độ phức tạp không gian:** Ta dùng hai mảng `get_max` và `get_min` độ dài $n + 1$. Bộ nhớ phụ: $O(n)$.

3.3 Mã nguồn Python (Subtask 2)

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

n = int(readline())

arr = []
arr.append((0,0,0))

for _ in range(n):
    x,y,z = map(int,readline().split())
    arr.append((x,y,z))

INF = 200**1000
NEG = -200**1000

arr.sort(key=lambda p:(p[0],p[1]))

get_max = (n+1)*[NEG]
get_min = (n+1)*[INF]

get_max[1] = get_min[1] = arr[1][2]

res = NEG

for i in range(2,n+1):
    x,y,z = arr[i]

    res = max(res,z)
    if z<=0:
        get_min[i] = z
    if z>=0:
        get_max[i] = z

    for j in range(1,i):
        pre_x,pre_y,pre_z = arr[j]

        if y>=pre_y:
            if get_max[j]>=0 and z>=0:
                get_max[i] = max(get_max[i],get_max[j]*z)

            if get_min[j]<=0 and z<=0:
                get_max[i] = max(get_max[i],get_min[j]*z)

            if get_max[j]>=0 and z<=0:
                get_min[i] = min(get_min[i],get_max[j]*z)

            if get_min[j]<=0 and z>=0:
                get_min[i] = min(get_min[i],get_min[j]*z)

for val in get_max:
    res = max(res,val)

print(res)
```

4 Kết luận

Cả hai subtask đều dựa trên cùng một tư tưởng:

- Sắp xếp chiến binh theo (`age`, `intelligence`) để đảm bảo nếu chỉ đi từ trái qua phải, `age` sẽ không giảm.
- Chỉ được "nối" chiến binh i sau chiến binh j nếu `intelligence` không giảm: $b_j \leq b_i$.
- Phải theo dõi đồng thời hai đại lượng: tích lớn nhất và tích nhỏ nhất, vì số âm có thể lật dấu và biến "kết quả tệ nhất" thành "kết quả tốt nhất".

Khác biệt chủ yếu giữa hai subtask:

- **Subtask 1 ($n \leq 20$)**: dùng bảng hai chiều theo toạ độ nén (`age_idx`, `int_idx`), độ phức tạp xấp xỉ $O(n^3)$ thời gian và $O(n^2)$ không gian.
- **Subtask 2 ($n \leq 1000$)**: dùng quy hoạch động kiểu "LIS mở rộng" theo chỉ số sau khi sort, duyệt cặp (j, i) và cập nhật `get_max`, `get_min`, độ phức tạp $O(n^2)$ thời gian và $O(n)$ không gian.