Dynamic Programming

Báo cáo nhóm 14

Phân tích và thiết kế thuật toán(CS112)

Nhóm: Nhóm 14

Thành viên:
Đặng Phú Duy 24520010

Mai Quốc Anh 24520002

Ngày 28 tháng 10 năm 2025

Lời giản

Mô tả thuật toán :

Subtask 1:

- Ta thấy được n ở subtask 1 khá nhỏ (n <= 20), gợi cho ta ý tưởng để sử dụng hướng tiếp cận brute force để giải quyết.

- Cụ thể, ta sẽ duyệt qua $2^n - 1$ trạng thái chọn chiến binh trồi tìm ra trạng thái có tích power là lớn nhất.

- Với mỗi trạng thái, ta có thể kiểm tra tính valid của trạng thái bằng cách sắp xếp những chiến binh được chọn tăng dần theo chỉ số age. Sau đó chỉ cần xét chỉ số intelligence theo thứ tự đã sắp xếp ở trên. Nếu ta có được một dãy không giảm intelligence thì trạng thái đó hợp lệ, không thì không hợp lệ.

Subtask 2:

- Ta thấy được ở subtask này thì n khá to (n <= 1000), không thể nào duyệt được tất cả các trạng thái của cách chọn nên ta tìm một cách khác tối ưu.

- Ở subtask này, ta sẽ dùng cách tiếp cận quy hoạch động để giải quyết vì nó có tính chất con tối ưu và bài toán có tính chồng lập.

- Cụ thể ta giải quyết bài toán như sau :

+ Ta thực hiện sắp xếp lại các trạng thái tăng dần theo chỉ số age, nếu cùng age thì tăng dần theo chỉ số intelligence.

+ Gọi dpMin[i] là cách chọn trạng thái sao cho tích power các trạng thái là nhỏ nhất khi xét các phần tử từ 1 đến i.

+ Gọi dpMax[i] là cách chọn trạng thái sao cho tích power các trạng thái là lớn nhất khi xét các phần tử từ 1 đến i.

+ Ta thực hiện xét các j với j < i, vì ta đã sắp xếp các phần tử tăng dần theo age nên muốn xét xem i và j có thể chung nhóm không thì chỉ cần xét chỉ số intelligence của j <= intelligence của i.

+ Từ đó, nếu ta có thể ghép i vào nhóm j, ta có thể xây dựng một công thức để chuyển trạng thái từ j sang i như sau :

+ dpMin[i] = min(dpMin[i], min(dpMax[j] * power[i], dpMin[j] * power[i]))

+ dpMax[i] = max(dpMax[i], max(dpMax[j] * power[i], dpMin[j] * power[i]))

+ Ta cần phải duy trì cả min lẫn max vì các giá trị power có âm, khi đó để được tích lớn nhất thì ta có thể phải tối ưu từ 2 trạng thái là lớn nhất hoặc nhỏ nhất từ các phần tử trước đó.

+ Trạng thái cơ sở của công thức này có thể xác định bằng cách với mỗi i, ta sẽ có thể không ghép với bất kỳ ai ở nhóm trước đó mà tự mình thành lập nhóm, khi đó ta sẽ có công thức :

+ dpMin[i] = dpMax[i] = power[i]

+ Kết quả cuối cùng sẽ là max của dpMax[i] với i chạy từ 0 -> n-1.

Mô tả cài đặt trong code :

Code :

```
import sys

input = sys.stdin.readline

n = int(input())
arr = []
for i in range(n):
    a, b, w = map(int, input().split())
    arr.append((a, b, w))

arr.sort() # sort mảng tăng dần theo age, nếu cùng age thì tăng dần theo intelligenc

dpMin = [1e18] * n # Khởi tạo mảng để lưu
dpMax = [-1e18] * n # Khởi tạo mảng để lưu
for i in range(n): # Duyệt qua các trạng thái
    dpMin[i] = min(dpMin[i], arr[i][2]) # Thực hiện lưu trạng thái cơ sở
    dpMax[i] = max(dpMax[i], arr[i][2]) # Thực hiện lưu trạng thái cơ sở
    for j in range(i): # Duyệt qua các trạng thái j trước i
        if arr[j][1] <= arr[i][1]: # Điều kiện kiểm tra xem có thể kết hợp nhóm
            dpMin[i] = min(dpMin[i], dpMin[j] * arr[i][2]) # Thực hiện chuyển trạng
            dpMin[i] = min(dpMin[i], dpMax[j] * arr[i][2]) # Thực hiện chuyển trạng

            dpMax[i] = max(dpMax[i], dpMin[j] * arr[i][2]) # Thực hiện chuyển trạng
            dpMax[i] = max(dpMax[i], dpMax[j] * arr[i][2]) # Thực hiện chuyển trạng

print(max(dpMax)) # In kết quả
```

Độ phức tạp :

Độ phức tạp thời gian :

Subtask 1 :

- Ta thực hiện sắp xếp lại mảng các phần tử. Độ phức tạp $O(NlogN)$.

- Ta thực hiện duyệt $2^n - 1$ trạng thái. Độ phức tạp $O(2^n)$.

- Tổng độ phức tạp $O(2^n)$.

Subtask 2 :

- Ta thực hiện sắp xếp lại mảng các phần tử. Độ phức tạp $O(NlogN)$.

- Ta thực hiện duyệt N lần các phần tử, với mỗi phần tử, ta cần duyệt qua hết các phần tử trước đó và cố gắng ghép với phần tử hiện tại. Độ phức tạp $O(N^2)$

- Tổng độ phức tạp $O(N^2)$

Độ phức tạp không gian :

Subtask 1 :

- Ta chỉ cần dùng mảng để lưu lại các chỉ số của các phần tử. Độ phức tạp $O(N)$

- Tổng độ phức tạp $O(N)$

Subtask 2 :

- Ta dùng mảng để lưu lại chỉ số các phần tử. Độ phức tạp $O(N)$

- Ta dùng mảng dpMax và dpMin để duy trì trạng thái. Độ phức tạp $O(N)$

- Tổng độ phức tạp $O(N)$