ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Lớp: CS112.Q11.KHTN

Môn học: Phân Tích và Thiết Kế Thuật Toán

Chủ đề 7: Phương pháp thiết kế thuật toán: Dynamic programming

Nhóm 4:
Lê Văn Thức
Bảo Quý Định Tân

Giảng viên:
Nguyễn Thanh Sơn

1 Mô tả thuật toán

1.1 Subtask 1

Với $n \leq 20$, ta có thể duyệt toàn bộ các cách chọn tập chiến binh bằng phương pháp **sinh nhị phân (brute force)**. Lý do chọn thuật toán này là vì khi n nhỏ, ta có thể kiểm tra toàn bộ các tổ hợp khả thi để đảm bảo tìm được nghiệm tối ưu tuyệt đối.

- Duyệt qua từng chiến binh, tại mỗi bước có hai lựa chọn: *chọn* hoặc *không chọn* chiến binh đó.
- Khi chọn, cần kiểm tra xem chiến binh có thể thêm vào tập hiện tại mà vẫn thỏa mãn điều kiện **Age** và **Intelligence** không giảm dần hay không.
- Nếu hợp lệ, nhân chỉ số **Power** của chiến binh vào tích hiện tại và tiếp tục xét chiến binh tiếp theo.
- Sau khi duyệt hết, cập nhật **tích Power lớn nhất** trong tất cả các tổ hợp hợp lệ.

Độ phức tạp thời gian:

- Xét toàn bộ tổ hợp chiến binh: $O(2^n)$.
- Kiểm tra tính hợp lệ mỗi tập: $O(n)$.
- Tổng độ phức tạp: $O(n \times 2^n)$.

Độ phức tạp không gian: $O(n)$, do chỉ cần lưu danh sách chiến binh hiện tại.

—

1.2 Subtask 2

Với Subtask 2, ta không thể áp dụng thuật toán của Subtask 1, vì với độ phức tạp $O(n \times 2^n)$, chương trình sẽ không thể chạy được khi n lên đến 1000.

Ta nhận thấy rằng, nếu sắp xếp các chiến binh theo **Age** tăng dần, thì với mỗi cặp chiến binh i, j $(i < j)$, chỉ cần điều kiện $Intelligence_i \leq Intelligence_j$ là đủ để hai chiến binh này có thể cùng thuộc một nhóm hợp lệ.

Xét ba chiến binh i, j, k thỏa mãn:

- $i < j < k$
- $Intelligence_i \leq Intelligence_j$
- $Intelligence_j \leq Intelligence_k$

Ta suy ra rằng $Intelligence_i \leq Intelligence_k$ và đồng thời $Age_i \leq Age_j \leq Age_k$. Điều này có nghĩa là ba chiến binh i, j, k đều có thể cùng thuộc một nhóm nếu hai cặp (i, j) và (j, k) đều hợp lệ.

Do đó, bài toán có thể được quy về bài toán **tìm dãy con tăng theo chỉ số Intelligence** (sau khi đã sắp xếp theo Age) sao cho **tích Power là lớn nhất**.

Xét Power âm

Vì giá trị Power có thể âm, ta cần theo dõi đồng thời cả giá trị tích lớn nhất và nhỏ nhất tại mỗi bước. Khi nhân với một giá trị âm, hai giá trị này có thể hoán đổi vai trò cho nhau.

- $maxdp[i]$: tích Power lớn nhất của dãy con kết thúc tại chiến binh i.
- $mindp[i]$: tích Power nhỏ nhất (âm nhất) của dãy con kết thúc tại chiến binh i.

Công thức quy hoạch động

$$maxdp[i] = \max\left(Power_i, \max_{j<i,\, Intelligence_j \le Intelligence_i} \{\max(maxdp[j] \times Power_i,\, mindp[j] \times Power_i)\}\right)$$

$$mindp[i] = \min\left(Power_i, \min_{j<i,\, Intelligence_j \le Intelligence_i} \{\min(maxdp[j] \times Power_i,\, mindp[j] \times Power_i)\}\right)$$

Giá trị kết quả cuối cùng là:

$$\max_{1 \le i \le n}(maxdp[i])$$

Các bước thực hiện thuật toán

1. Sắp xếp danh sách chiến binh theo **Age** tăng dần.
2. Khởi tạo hai mảng $maxdp$ và $mindp$, ban đầu đều bằng giá trị $Power_i$ tương ứng.
3. Với mỗi chiến binh i từ 1 đến n:
   - Xét tất cả các chiến binh $j < i$ thỏa $Intelligence_j \le Intelligence_i$.
   - Cập nhật $maxdp[i]$ và $mindp[i]$ theo công thức trên.
4. Sau khi xử lý xong, kết quả là giá trị lớn nhất trong mảng $maxdp$.

Độ phức tạp thời gian: $O(n^2)$ do phải xét mọi cặp (i, j) với $j < i$.
Độ phức tạp không gian: $O(n)$ để lưu hai mảng $maxdp$ và $mindp$.

1.2.1 Cài đặt

```
import sys
input = sys.stdin.readline
arr = []   # khoi tao mang chua cac thanh vien
n = int(input())
maxdp = [0] * n # khoi tao mang luu gia tri dp lon nhat
mindp = [0] * n # khoi tao mang luu gia tri dp nho nhat
for i in range(n):
    age, intelligent, power = map(int, input().split())
    arr.append((age, intelligent, power))
arr.sort() # sap xep mang theo thu tu tang dan
for i in range(n):
    maxdp[i] = arr[i][2] # gan gia tri dp lon nhat bang gia tri z cua thanh vien i (gia tri mac dinh)
    mindp[i] = arr[i][2] # gan gia tri dp nho nhat bang gia tri z cua thanh vien i (gia tri mac dinh)
    for j in range(i): # duyet qua cac thanh vien truoc i
```

```
        if arr[i][1] >= arr[j][1]:    # kiem tra dieu kien y
            maxdp[i] = max(maxdp[i], maxdp[j] * arr[i][2]) # cap nhat gia tri dp lon nhat
            mindp[i] = min(mindp[i], mindp[j] * arr[i][2]) # cap nhat gia tri dp nho nhat
            maxdp[i] = max(maxdp[i], mindp[j] * arr[i][2]) # cap nhat gia tri dp lon nhat voi gia tri dp nho nhat (truong hop so am)
            mindp[i] = min(mindp[i], maxdp[j] * arr[i][2]) # cap nhat gia tri dp nho nhat voi gia tri dp lon nhat (truong hop so am)
print(max(maxdp)) # in ra ket qua lon nhat trong mang dp
```