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 6: Phương pháp thiết kế thuật toán: Divide, Decrease, Transform and Conquer

Nhóm 4:
Lê Văn Thức
Bảo Quý Định Tân

Giảng viên:
Nguyễn Thanh Sơn

1 Phân tích bài toán

Đề bài yêu cầu thành viên thứ j chỉ được tham gia nhóm của thành viên thứ i nếu :

$$|x_i - x_j| \leq r_i - r_j$$

Phá trị tuyệt đối, ta được:

$$-r_i + r_j \leq x_i - x_j \leq r_i - r_j$$

Chuyển vế, ta thu được hệ bất đẳng thức tương đương:

$$\begin{cases} r_i - x_i \geq r_j - x_j, \\ r_i + x_i \geq r_j + x_j. \end{cases}$$

Đặt:

$$a_i = r_i - x_i, \quad b_i = r_i + x_i.$$

Khi đó, điều kiện để học sinh j có thể vào đội của i trở thành:

$$a_i \geq a_j \quad \text{và} \quad b_i \geq b_j.$$

Ta nhận thấy đội trưởng i của mỗi đội sẽ là người thỏa $a_i \geq a_j \quad$ và $\quad b_i \geq b_j$ với mỗi thành viên j còn lại.

Khi đó bài toán trở thành bài toán tìm số người ít nhất để bao phủ toàn bộ tập điểm theo điều kiện: $a_i \geq a_j \quad$ và $\quad b_i \geq b_j$.

- Đây là một dạng **Representation Change**, tức là thay đổi cách biểu diễn bài toán, một hình thức của Transform and Conquer.
- Nhờ sự biến đổi này, điều kiện phức tạp trong bài toán gốc được biểu diễn thành dạng so sánh đơn giản giữa hai tọa độ.

2 Subtask 1

2.1 Kỹ thuật được sử dụng

Bài toán thuộc nhóm **Transform and Conquer**.

- Transform: Ta biến đổi dữ liệu đầu vào (x_i, r_i) thành một dạng mới (a_i, b_i) với:

  $$a_i = x_i - r_i, \quad b_i = x_i + r_i$$

  Khi đó, mỗi học sinh i sẽ tương ứng với một điểm (a_i, b_i) trên mặt phẳng tọa độ, thể hiện vùng mà bạn ấy có thể "bao phủ" (những người có thể tham gia cùng đội).

- Conquer: Sau khi biến đổi, bài toán trở thành:

  Tìm số lượng nhỏ nhất các điểm (a_i, b_i) sao cho không có điểm nào chứa trong khoảng của điểm

  Hay tương đương, ta cần đếm số lượng phần tử không bị "bao" bởi bất kỳ phần tử nào khác thỏa:

  $$a_i \geq a_j \text{ và } b_i \geq b_j$$

- Reinterpret: Từ lời giải trên tập (a_i, b_i), ta suy ra số đội tối thiểu ban đầu.

2.2 Thuật toán

Dựa trên nhận xét trên, ta chỉ cần đếm số học sinh độc lập, tức là không bị học sinh nào khác bao phủ trong cả hai chiều a và b. Riêng những học sinh có (a_i, b_i) trùng nhau, ta chỉ xét một trường hợp duy nhất vì chúng luôn có thể thuộc cùng một nhóm.

```
n = int(input())
x = []
for i in range(n):
    a, b = map(int, input().split())
    x.append((b-a,b+a))
x = list(set(x))
```

Đầu tiên, chúng ta nhập vào x_i và r_i của n học sinh, sau đó, chúng ta biến đổi thành dạng $(x - r, x + r)$ và khử đi những phần tử trùng bằng set.

```
ans = 0
for i in range(len(x)):
    Flag = True
    ai, bi = x[i]
    for j in range(len(x)):
        if i == j:
            continue
        aj, bj = x[j]
        if ai <= aj and bi <= bj:
            Flag = False
    ans = ans + Flag
print(ans)
```

Ta duyệt qua từng học sinh i, sau đó lại duyệt qua một lần nữa từng học sinh khác j, nếu $a_i \leq a_j$ và $b_i \leq b_j$ thì ta đánh dấu không tăng đáp án. Nếu không tồn tại học sinh j như trên thì tăng đáp án lên 1.

Khi hết vòng lặp ta sẽ in ra đáp án đã lưu.

2.3 Phân tích độ phức tạp

- **Thời gian:**
  - Loại trùng bằng set tốn $O(n \log n)$.
  - Hai vòng lặp lồng nhau tốn $O(n^2)$.

  $\Rightarrow$ Độ phức tạp tổng là $O(n^2)$.
- **Không gian:** Ta chỉ dùng một danh sách và một tập hợp, nên độ phức tạp không gian là $O(n)$.

3 Subtask 2

3.1 Kỹ thuật được sử dụng

Bài toán tiếp tục sử dụng kỹ thuật **Transform and Conquer**, nhưng được tối ưu ở giai đoạn **Conquer** bằng cách áp dụng sắp xếp và duyệt ngược để giảm độ phức tạp từ $O(n^2)$ xuống $O(n \log n)$.

- **Transform:** Giống như Subtask 1, ta biến đổi mỗi học sinh (x_i, r_i) thành điểm (a_i, b_i) với:

$$a_i = x_i - r_i, \quad b_i = x_i + r_i$$

Khi đó, một học sinh i có thể bao được học sinh j nếu:

$$a_j \leq a_i \quad \text{và} \quad b_j \leq b_i$$

- **Conquer:** Thay vì kiểm tra từng cặp (i, j) như Subtask 1, ta sắp xếp các điểm theo a_i tăng dần (và nếu trùng, theo b_i tăng dần). Sau đó, duyệt ngược từ phải sang trái, chỉ cần theo dõi giá trị b_i lớn nhất đã gặp, ta có thể xác định điểm nào là độc lập mà không bị bao bởi điểm khác.

- **Reinterpret:** Kết quả sau khi đếm số điểm độc lập chính là số đội tối thiểu cần tạo.

Cải tiến trên dựa trên nhận xét rằng, nếu ta sort a_i tăng dần (và nếu trùng, theo b_i tăng dần) thì nếu duyệt ngược, mọi điểm $i \leq j$ sẽ thỏa mãn $a_i \leq a_j$, tức là nếu $i < j$ thì ta chỉ cần $b_i \leq b_j$ là điểm i sẽ bị điểm j bao.

Như vậy, nếu điểm i có $b_i > b_j$ với mọi $i < j$ thì sẽ không tồn tại điểm có $b_i \leq b_j$ mà $a_i \leq a_j$, tức điểm đó độc lập.

3.2 Thuật toán

Bắt đầu bằng việc nhập dữ liệu và biến đổi về dạng (a_i, b_i).

```
n = int(input())
x = []
for i in range(n):
    a, b = map(int, input().split())
    x.append((b - a, b + a))
```

Tiếp theo, ta sắp xếp các cặp (a_i, b_i) theo thứ tự tăng dần của a_i. Điều này giúp ta dễ dàng xác định quan hệ bao phủ giữa các điểm trong một lượt duyệt.

```
x.sort()
```

Sau khi sắp xếp, ta duyệt ngược từ cuối danh sách, đồng thời theo dõi biến mxy là giá trị b_i lớn nhất đã thấy. Nếu $b_i > mxy$, đoạn này không bị bao bởi các điểm đã gặp $\rightarrow$ ta tăng biến đếm.

```
ans = 0
mxy = 0
for i in reversed(range(n)):
    xi, yi = x[i]
    if i > 0:
        xj, yj = x[i - 1]
        if xi == xj and yi == yj:
            continue
    if yi > mxy:
        ans += 1
        mxy = yi
print(ans)
```

Ở đây:

- Biến mxy giữ giá trị b_i lớn nhất hiện tại.

- Việc bỏ qua các phần tử trùng nhau như subtask 1 vì chúng luôn là 1 nhóm.
- Mỗi khi $b_i > mxy$, nghĩa là điểm này vượt ra ngoài tất cả các điểm sau → tương ứng với một đội trưởng độc lập.

3.3 Phân tích độ phức tạp

- **Thời gian:**
  - Bước sắp xếp: $O(n \log n)$
  - Bước duyệt ngược: $O(n)$

  $\Rightarrow$ Tổng độ phức tạp: $O(n \log n)$
- **Không gian:** Chỉ dùng một danh sách và vài biến phụ $\Rightarrow O(n)$