Subtask 1

1. Transform and Conquer — Problem Reduction

Bài toán được giải bằng kỹ thuật **Transform and Conquer**, cụ thể là **Problem Reduction**. Thay vì xử lý trực tiếp điều kiện:

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

ta **chuyển đổi bài toán ban đầu sang một bài toán tương đương** dễ giải hơn — cụ thể là bài toán trên **đồ thị có hướng**.

Trong đồ thị này, mỗi học sinh được xem là một đỉnh, và tồn tại cạnh $i \to j$ nếu học sinh j có thể gia nhập đội của học sinh i. Như vậy, hai học sinh thuộc cùng nhóm nếu và chỉ nếu tồn tại đường đi hai chiều giữa chúng, tức là chúng nằm trong cùng một **thành phần liên thông mạnh (SCC)** của đồ thị.

Do đó, việc chia nhóm học sinh đã được **rút gọn** thành việc **đếm số thành phần liên thông mạnh** trong đồ thị có hướng — bài toán này được giải bằng **thuật toán Tarjan**.

2. Nhận xét bài toán

Những bài toán có thể áp dụng phương pháp **Problem Reduction** thường là các bài toán mà:

- Điều kiện quan hệ giữa các phần tử phức tạp hoặc khó mô tả trực tiếp.
- Có thể **chuyển đổi tương đương** sang một mô hình toán học hoặc bài toán khác (ở đây là bài toán đồ thị) nhưng vẫn giữ nguyên kết quả đầu ra.

3. Từ công thức rút ra nhận xét

Từ điều kiện $|x_i - x_j| \leq r_i - r_j$, ta có:

- Tồn tại cạnh có hướng $i \to j$ nếu học sinh j có thể tham gia đội của i.
- Hai học sinh cùng nhóm nếu tồn tại đường đi hai chiều giữa chúng.
- Số nhóm tối thiểu tương ứng với số SCC trong đồ thị có hướng.

4. Mô tả thuật toán và cài đặt

- **Xây dựng đồ thị:**
  - Với mỗi cặp (i, j), nếu $|x_i - x_j| \leq r_i - r_j$, thêm cạnh $i \to j$.
- **Tìm thành phần liên thông mạnh (SCC):**
  - Dùng **thuật toán Tarjan** để đếm số SCC.

Listing 1: Cài đặt thuật toán Tarjan cho Subtask 1

```
import sys
sys.setrecursionlimit(10**6)
input = sys.stdin.readline

n = int(input())
x = [0]*n
r = [0]*n
for i in range(n):
    x[i], r[i] = map(int, input().split())

# X y  d ng        t h  c   h   ng
adj = [[] for _ in range(n)]
for i in range(n):
    for j in range(n):
        if i != j and abs(x[i] - x[j]) <= r[i] - r[j]:
            adj[i].append(j)

# Tarjan t m SCC
id = [0]*n
low = [0]*n
onstack = [False]*n
stack = []
timer = 0
scc_count = 0

def dfs(u):
    global timer, scc_count
    timer += 1
    id[u] = low[u] = timer
    stack.append(u)
    onstack[u] = True

    for v in adj[u]:
        if id[v] == 0:
            dfs(v)
            low[u] = min(low[u], low[v])
        elif onstack[v]:
            low[u] = min(low[u], id[v])

    if id[u] == low[u]:
        while True:
            v = stack.pop()
            onstack[v] = False
            if v == u:
                break
        scc_count += 1

for i in range(n):
    if id[i] == 0:
        dfs(i)
```

```
print(scc_count)
```

5. Phân tích thời gian

- Xây dựng đồ thị: $O(N^2)$.
- Tarjan: $O(N + E)$ với $E \leq N^2$.
- **Tổng độ phức tạp thời gian:** $O(N^2)$.

6. Phân tích bộ nhớ

- Danh sách kề: $O(N^2)$.
- Mảng hỗ trợ Tarjan: $O(N)$.
- **Tổng độ phức tạp bộ nhớ:** $O(N^2)$.

Subtask 2

1. Transform and Conquer — Representation Change

Kỹ thuật được sử dụng là **Transform and Conquer**, cụ thể là **Representation Change**. Thay vì trực tiếp so sánh điều kiện:

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

ta biến đổi biểu thức về dạng dễ so sánh hơn bằng hai đại lượng:

$$\text{left}_i = r_i - x_i, \quad \text{right}_i = r_i + x_i.$$

Điều này giúp biểu diễn mỗi học sinh bằng một cặp giá trị $(\text{left}_i, \text{right}_i)$, từ đó việc kiểm tra quan hệ giữa hai học sinh trở nên đơn giản hơn.

2. Nhận xét bài toán

Những bài toán áp dụng được **Representation Change** thường là các bài toán mà:

- Điều kiện ban đầu phức tạp (ở đây là dạng tuyệt đối).
- Có thể chuyển đổi cách biểu diễn dữ liệu sao cho điều kiện trở nên tuyến tính, giúp việc xử lý dễ hơn mà vẫn đảm bảo kết quả tương đương.

3. Từ công thức rút ra nhận xét

Từ phép biến đổi, ta có:

- Học sinh j có thể gia nhập đội của i nếu $r_i - x_i \leq r_j - x_j$ và $r_i + x_i \geq r_j + x_j$.
- Việc xét nhóm có thể thực hiện bằng cách duyệt các học sinh theo x và sử dụng cấu trúc dữ liệu hỗ trợ kiểm tra điều kiện này.

4. Mô tả thuật toán và cài đặt

Thuật toán:

- Tính $(r - x)$ và $(r + x)$ cho từng học sinh.
- Sắp xếp theo x tăng dần.
- Duyệt từ phải sang trái, duy trì các nhóm trong một stack dựa trên điều kiện bao phủ giữa các học sinh.

Listing 2: Cài đặt thuật toán cho Subtask 2

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

class Student:
    def __init__(self, x, r):
        self.x = x
```

```
        self.r = r
        self.left = r - x
        self.right = r + x

def comp(a: Student):
    return (a.x, a.r)

n = int(input())
st = [Student(*map(int, input().split())) for _ in range(n)]

st.sort(key=comp)

MAX = st[-1].left
ans = 1
stack = [st[-1].right]

for i in range(n - 2, -1, -1):
    if st[i].left > MAX:
        ans += 1
        while stack:
            if stack[-1] <= st[i].right:
                stack.pop()
                ans -= 1
            else:
                break
        stack.append(st[i].right)
    MAX = max(MAX, st[i].left)

print(ans)
```

5. Phân tích thời gian

- Sắp xếp: $O(N \log N)$.
- Duyệt và xử lý stack: $O(N)$.
- **Tổng độ phức tạp thời gian:** $O(N \log N)$.

6. Phân tích bộ nhớ

- Mảng lưu học sinh: $O(N)$.
- Stack phụ trợ: $O(N)$.
- **Tổng độ phức tạp bộ nhớ:** $O(N)$.