**Ví dụ**

**Input**

**7 9 3**

**1 2 50**

**1 3 60**

**2 4 120**

**2 5 90**

**3 6 50**

**4 6 80**

**4 7 70**

**5 7 40**

**6 7 140**

**1 7**

**2 6**

**6 2**

**7 6 3**

**1 2 50**

**1 3 60**

**2 4 120**

**3 6 50**

**4 6 80**

**5 7 40**

**7 5**

**1 7**

**2 4**

**0 0 0**

**Output**

**Case #1**

**80**

**60**

**60**

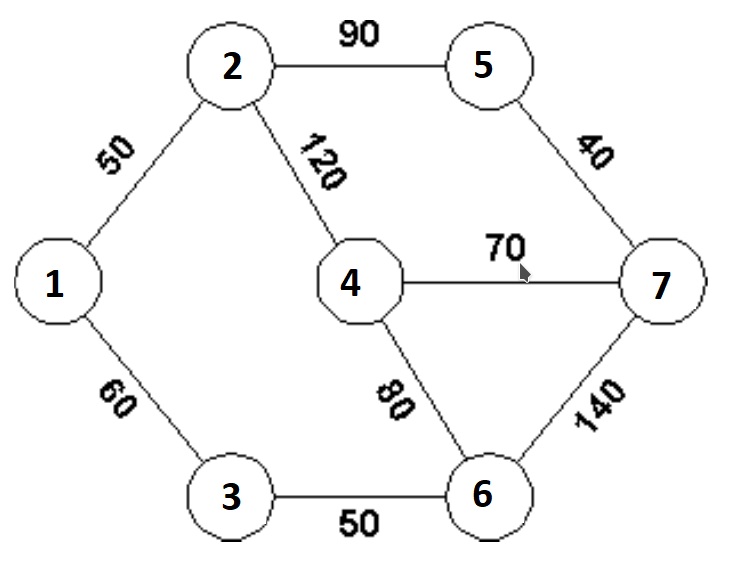
**Case #2**

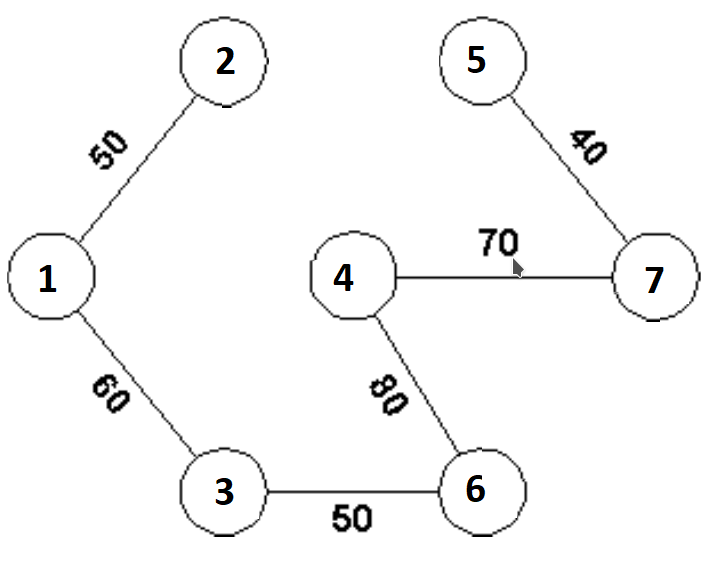
**40**

**no path**

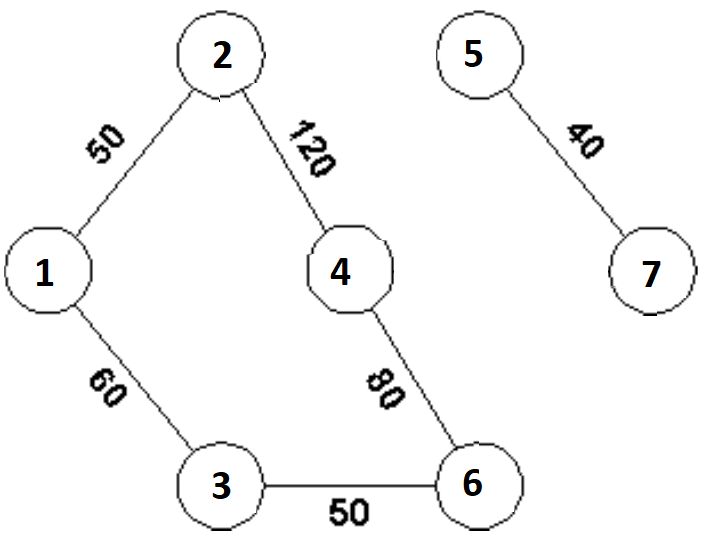
**80**

**Minh họa testcase 1:**

****

****

**Minh họa testcase 2:**

****

**Giải quyết bài toán:**

**Tìm 1 đường đi từ c1 → c2 sao cho cường độ âm thanh lớn nhất phải là nhỏ nhất**

**Tìm từ c1 → c2: Prim()**

**Với mỗi truy vấn, ta tìm cây khung nhỏ nhất bắt đầu từ đỉnh c1, rồi tìm cái đường tới c2 mà có max là min**

**Tìm cây khung nhỏ nhất, thì những cạnh được chọn là những cạnh được tối ưu**

**→ Độ phức tạp: O(T \* Q \* S \* log(C))**

**Q = 10^4**

**S = 10^3**

**C = 10^2 → log(C) ~= 6**

**→ Q \* S \* log(C) ~= 6 \* 10^7**

**Nếu T ~= 10 → 6 \* 10^8 → Quá thời gian**

**\*\*\* Nếu mỗi testcase, tìm 1 cây khung cho cái đồ thị đó.**

**Độ phức tạp tìm cây khung là O(S \* log(C))**

**Độ phức tạp thực hiện truy vấn: O(Q \* C)**

**→ O(T \* (S \* log(C) + Q \* C))**

**10^3 \* 6 + 10^4 \* 100**

**Giải thuật:**

**Bước 1:** Nhập dữ liệu và khởi tạo dữ liệu.

**Bước 2:** Dùng **Prim** tìm cây khung nhỏ nhất trong các đồ thị, tạo một đồ thị mới dựa trên cây khung nhỏ nhất.

**Bước 3:** Với mỗi truy vấn , dùng đồ thị mới tạo được, **DFS** tìm đường đi từ tới .

**Bước 4:** In kết quả.

**Mã giả**

**function Prim(s): {**

**pq = minHeap();**

**pq.push((0, s));**

**dist[s] = 0;**

**while pq.size() > 0: {**

**(du, u) = pq.top;**

**pq.pop();**

**visited[u] = true;**

**for (v, w) in graph[u]:**

**if visited[v] == false and dist[v] > w: {**

**dist[v] = w;**

**path[v] = u;**

**pq.push((dist[v], v));**

**}**

**}**

**}**

**function DFS(s, f, maxSound): {**

**if s == f:**

**return maxSound;**

**visited[s] = true;**

**for (v, w) in graph[s]:**

**if visited[v] == false {**

**tmp = DFS(v, f, max(maxSound, w));**

**if tmp != -1:**

**return tmp;**

**}**

**return -1;**

**}**

**main() {**

**test = 0;**

**while (true): {**

**read(C, S, Q);**

**if (C == 0 and S == 0 and Q == 0)**

**break;**

**test += 1;**

**graph = [];**

**path = [];**

**dist = [];**

**visited = [];**

**for i = 1 to S: {**

**read(c1, c2, d);**

**graph[c1].push((c2, d));**

**graph[c2].push((c1, d));**

**}**

**for i = 1 to C: {**

**dist[i] = INF;**

**path[i] = -1;**

**visited[i] = false;**

**}**

**for i = 1 to C:**

**if visited[i] == false:**

**prim(i);**

**mstGraph = []; // C + 1 cái list**

**for v = 1 to C:**

**if visited[v] == true: {**

**u = path[v];**

**w = dist[v];**

**mstGraph[u].push((v, w));**

**mstGraph[v].push((u, w));**

**}**

**print(‘Case #’, test);**

**for i = 1 to Q: {**

**read(c1, c2);**

**for j = 1 to C:**

**visited[j] = false;**

**result = DFS(c1, c2, 0);**

**if result == -1:**

**print(‘no path’);**

**else:**

**print(result);**

**}**

**print(‘\n’);**

**}**

**}**

**Độ phức tạp: O(T \* (S \* log(C) + Q \* C))**

**Trong đó:**

**T là số lượng testcase**

**S là số lượng cạnh**

**C là số lượng đỉnh**

**Q là số lượng truy vấn**