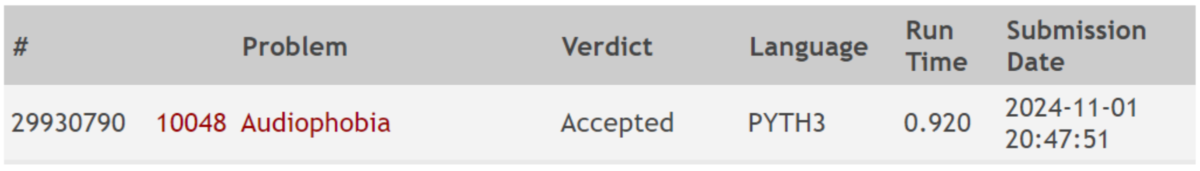
MST: Minimax

```
import sys

class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n

    def find_set(self, u):
        if self.parent[u] != u:
            self.parent[u] = self.find_set(self.parent[u])
        return self.parent[u]

    def same_set(self, u, v):
        return (self.find_set(u) == self.find_set(v))

    def union_set(self, u, v):
        if not self.same_set(u, v):
            root_u = self.find_set(u)
            root_v = self.find_set(v)

            if self.rank[root_u] > self.rank[root_v]:
                self.parent[root_v] = root_u
            elif self.rank[root_u] < self.rank[root_v]:
                self.parent[root_u] = root_v
            else:
                self.parent[root_v] = root_u
                self.rank[root_u] += 1


def kruskal_mst(nro_vertices, aristas):
    uf = UnionFind(nro_vertices + 1)
    mst_graph = [[] for _ in range(nro_vertices + 1)]
    aristas.sort(key=lambda x: x[2])

    for origin, destination, cost in aristas:
        if not uf.same_set(origin, destination):
            uf.union_set(origin, destination)
            mst_graph[origin].append((destination, cost))
            mst_graph[destination].append((origin, cost))

    return mst_graph

def max_intensity_path(crossing_start, crossing_end, mst_graph):
    visited_crossings = [False] * len(mst_graph)
    stack = [(crossing_start, 0)]

    while stack:
        crossing, current_max_intensity = stack.pop()
        if crossing == crossing_end:
            return current_max_intensity

        visited_crossings[crossing] = True
        for neighbor_crossing, intensity in mst_graph[crossing]:
            if not visited_crossings[neighbor_crossing]:
                stack.append((neighbor_crossing, max(current_max_intensity, intensity)))

    return None

def main():
    data = []
    while True:
        line = input().strip()
        data.append(line)
        if line == "0 0 0":
            break

    index = 0
    case_number = 1
    results = []

    while True:
        total_crossings, total_streets, total_queries = map(int, data[index].split())
        index += 1
        if total_crossings == 0 and total_streets == 0 and total_queries == 0:
            break

        streets = []
        for _ in range(total_streets):
            crossing1, crossing2, decibels_intensity = map(int, data[index].split())
            streets.append((crossing1, crossing2, decibels_intensity))
            index += 1

        mst_graph = kruskal_mst(total_crossings, streets)
        results.append(f"Case #{case_number}")

        for _ in range(total_queries):
            crossing_start, crossing_end = map(int, data[index].split())
            index += 1
            min_tolerable_noise = max_intensity_path(crossing_start, crossing_end, mst_graph)
            if min_tolerable_noise is None:
                results.append("no path")
            else:
```

```
            results.append(str(min_tolerable_noise))

        case_number += 1
        results.append("")

    print("\n".join(results).strip())

main()
```



#	Problem	Verdict	Language	Run Time	Submission Date
29930790	10048 Audiophobia	Accepted	PYTH3	0.920	2024-11-01 20:47:51