FrogShopping Solution

In order to solve the problem, I decided to use the greedy BFS approach “Pick and cover”.
The reasons are that:

- It is fast enough for large graphs: it uses only adjacency lists and standard BFS. In the worst case, it might be needed to do one BFS per node, which is O(N(N+M)). But in practice, each BFS often covers a large region, so the number of BFS calls is much smaller than N.
- Easy to implement: just a single loop and BFS for coverage. The solution does not have any complicated data structures, only adjacency lists, a queue, and some arrays.
- It guarantees that every node is covered, so no risk of incomplete solution.
- Though it does not guarantee the best minimal solution set, the answer is close to the desirable one.

The implementation is based on a greedy BFS approach. The goal is to find a set of nodes where to place a FrogShop and to ensure that every node in a graph is within distance D of at least one of the shops.
The method keeps a Boolean array (covered[]) that tracks which nodes are already covered. It iterates through all nodes: if a node is not yet covered, we place a shop there and run a breadth-first search (BFS) up to distance D to mark all nodes within that radius as covered. This ensures that, by the end of the process, every node is covered without needing excessively complex data structures or algorithms. The BFS uses a dist[] array (initialized to −1) to mark unvisited nodes, and updates the distance for neighbors incrementally until distance D is reached.

The solution was submitted by **dashashevchuk**, it beats swats2 and scored 98.22 points.

Done by Dariia Shevchuk 161200