Cats or Dogs (Solution)

Let us restate the problem in terms of graphs. Then the garden becomes a tree structure, and each hut becomes a vertex. For simplicity, a hut with a cat is called a "red" vertex, and a hut with a dog is called a "blue" vertex.

Subtask 1

The tree has small number of vertices only. When we calculate the danger level, we can try all possible ways to cut the edges. Once we decide which edges we will cut, it is easy to calculate the danger level. For each query, the time complexity is $O(N \cdot 2^{N-1})$.

Subtask 2

For each query, it is enough to solve the problem in linear time. In the above solution, we did not use the tree structure. Let us consider the solution using the tree structure.

Let us consider the solution using Dynamic Programming on the tree. We choose a vertex as a root, and consider the graph as a rooted tree. For each vertex v, let R_v be the minimum number of edges we have to cut if v belongs to the "red connected component." Similarly, let B_v be the minimum number of edges we have to cut if v belongs to the "blue connected component." We shall update these values from the leaves of the tree. Here, "red connected component" is a connected component without blue vertices after we cut the edges. The definition of "blue connected component" is similar.

We have the following recurrence formula:

- $B_v = \sum_{c\,:\,\text{a child of } v} \min(B_c, R_c + 1)$
- $R_v = \sum_{c\,:\,\text{a child of } v} \min(R_c, B_c + 1)$

We can calculate them using Depth-First Search. For each query, we can solve the problem in $O(N)$ time. The total time complexity is $O(NQ)$.

Subtask 3 (Full Score)

What are the properties we did not use in the above solutions? We do not use the fact that the status of at most one vertex will change for each day.

Let us try to optimize the above recurrence formula. We may consider using Heavy Light Decomposition. But, we will not explain this technique in detail because many documents are available on the web. In the following, when we use HLD, we shall explain what kind of data structures we need to consider.

The problem may be reduced to the case of the straight line. But, if we consider the original problem when the graph is the straight line, we can solve it easily. So, let us consider the situation where the values of B_v, R_v are fixed for each vertex. Then the problem may be reduced to this case appropriately.

By this reduction, the problem becomes as follows.

- In a sequence, there are N cells. Each cell will be colored red or blue.
- For each cell, the cost is R_i if it is colored red, and the cost is B_i if it is colored blue.
- However, if there are two adjacent cells with different colors, the cost is 1 for each of such pairs.
- Minimize the total cost.

Reformulating the problem in this way, you may think whether it can be solved using Segment Tree. In fact, for each interval, it is enough to keep the minimum of the total cost if we fix the colors (red or blue) for the two extreme vertices.

By this method, we can update each interval in $O(\log N)$ time. In total, for each query, we can solve the problem in $O(\log^2 N)$ time.