Village

Subtask 1

We can try each permutation of villagers, claculate the total distance and find both the minimum and maximum answers.

Complexity: $O(N!)$.

Subtask 2

See the Subtask 3 solution; here we can store the tree in slower data structures and work with the tree slower.

Complexity: $O(N^2)$.

Subtask 3

Minimal total distance. Each villager needs to move to a new place so we can process the tree from leaves (greedy): if a villager that currently is in a leaf has been there from the very beginning (still needs to move to other place), change places with its (only) neighbour node villager, add 2 to the answer and mark the leaf node as processed (or remove it from the tree so that new nodes can become leaves). If the last villager is not moved then it can change with any of it neighbors.

Complexity: $O(N)$.

Maximal total distance. In the beginning let's think about each edge independently — how many villagers can go through it in each direction? If the edge is between nodes a and b then the maximal number of such villagers is $\min(\text{subtreeSize}(a), \text{subtreeSize}(b))$. Calculate this value for each edge and add them up — this way we get the theoretically maximal achievable total distance. Now we find a node in the tree with a property that the biggest neighbour subtree is at most $N/2$. Such vertex is called a tree centroid and can be found in linear time. Now we just need to arrange all nodes from the neighbour subtrees and the centroid node itself so that no nodes stay in the same subtree where they were before. This is possible because no subtree is bigger than the sum of all other subtrees and it guarantees that the maximal possible number of villagers will pass through each of the edges.

Complexity: $O(N)$.