Inside information (servers)
BY JAKUB RADOSZEWSKI (POLAND)

Subtask 1. \( N \leq 4\,000 \)

This subtask can be solved by a brute force.

Subtask 2. Server 1 is directly connected to servers 2, 3, ..., \( N \).

Let \( t_i \) be the first point of time when servers 1 and \( i \) were connected. We set \( t_1 = 0 \). To check whether server \( a \) stores data chunk \( d \) at time \( i \), you have to check that \( t_d < t_a < i \) holds. To count the number of servers that store data chunk \( d \) at time \( i \), you have to consider the following two cases:

- \( d = 1 \): In this case the answer is simply 1 + the number of Share operations which happened so far.
- \( d \neq 1 \): In this case the answer is 2 + the number of Share operations which happened after \( t_d \) or 1 if \( i < t_d \).

Subtask 3. Servers \( A \) and \( B \) are directly connected to each other if and only if \( |A - B| = 1 \).

In the solution, we refer to servers as nodes and to connections as edges. Let the label of an edge \( s_1 s_2 \) be the moment of time when a Share operation is called for this edge - a value in 1, ..., \( N - 1 \). The label of an edge for which no Share operation was called yet is undefined. The solution is based on the following key observation: Node \( s \) has a chunk of data \( d \) if and only if the sequence of labels of edges on the path from node \( d \) to node \( s \) is increasing.

Let us proceed to subtask 3. To each maximal path of edges with increasing or decreasing labels, we will assign a different colour. We define an \( i \)-label / \( d \)-label of a node as a pair consisting of the colour of a maximal increasing/decreasing path containing this node and the distance from this node to the beginning of this path. A node can have between 0 and 2 \( i \)-labels and \( d \)-labels. As an example, if \( N = 7 \), the nodes are numbered 1..7 and the labels of subsequent edges are 4, 1, 3, 2, -,-, 5, where '-' means undefined, then the maximal increasing paths are 4 (colour A), 13 (B), 2 (C), 5 (D), the maximal decreasing paths are 41 (E), 32 (F), 5 (G). Node 3 has an \( i \)-label B1 and \( d \)-labels E2 and F0. The colours of max paths are completely arbitrary, they just need to be different.

If \( s_1 \) and \( s_2 \) Share all their data, only the \( i/d \)-labels of nodes \( s_1 \), \( s_2 \) change. They can be easily updated based on the \( i/d \)-labels of their neighbours. For a Query it is enough to compare the \( i/d \)-labels of nodes \( s \) and \( d \). To Count the number of nodes that store data chunk \( d \) it suffices to inspect the lengths of maximal increasing/decreasing paths on which node \( d \) is located and its \( i/d \)-labels. Thus each operation works in \( O(1) \) time.

This approach leads to a solution that can answer all Queries in \( O(\log N) \)-time using heavy-light decomposition. Each such query can be answered by decomposing the path from \( s \) to \( d \) via their LCA into \( O(\log N) \) fragments of heavy paths. For each heavy path, a data structure from subtask 3 is stored. (Here \( i/d \)-labels do not need to store the distance of the node.)

Subtask 4. Servers \( A \) and \( B \), with \( A < B \), are directly connected to each other if and only if \( 2A = B \) or \( 2A + 1 = B \).

For each node \( v \), let \( T_v \) be its subtree. For each edge \( e \) outgoing from \( v \) we will store, as \( C(e) \), the number of nodes from \( T_v \) that can be reached through a path with increasing edge labels starting...
from edge $e$.

To **Count** the number of nodes that store data chunk $d$, we consider each of the $O(\log N)$ parents of $d$. For each such parent $v$, with a $Q \ d \ v$ query we check if the path from $d$ to $v$ is increasing. If so, we compute the number of nodes on increasing paths going from $d$ via $v$ based on the label of edge used to enter node $v$. The complexity used per parent is $O(\log N)$ (one $Q$ query), which gives $O(\log^2 N)$ time in total. Similarly, a data query can be used to update $C(e)$ upon a **Share** operation. Since there are only 2 outgoing edges this part also works in $O(\log^2 N)$.

**Subtask 5.** Any server is directly connected to at most 5 other servers.

In this case, we make use of a centroid decomposition. For each node $v$, let $T_v$ be the subtree from the decomposition whose centroid is $v$. Since the maximal degree is very small you can apply the same solution as in the previous subtask.

**Subtask 6.** No further constraints.

In the general case, we will store for every node $v$ a static segment tree $ST(v)$ allowing us to compute suffix sums of values $C(e)$, stored according to increasing labels of edges $e$. 