# Task Trick

## Abridged Problem Statement

You are not given a permutation of size $N$, $(A_1, A_2, \ldots, A_N)$.

You are allowed to query a permutation of size $N$, $(X_1, X_2, \ldots, X_N)$. A black box splits $X$ into $B$ sets of $K$ integers. The order of integers within each set is randomized, then the order of sets is randomized.

The black box applies permutation $A$ to each element in each set of $X$ to obtain $Y$ ($Y_i = A_{X_i}$), and the result $Y$ is returned.

Recover $A$ with the least number of queries.

## Subtask 1

In this subtask, $B = 2$ and $K = 3$.

There are $\binom{6}{3} \div 2! = 10$ possible unique queries that can be made, well under the limit of $Q = 100$.

Additionally, $N$ is small and there are $N! = 720$ possible arrangements of $A$. All 10 possible queries can be generated and queried to the grader. Then iterate through all possible $A$ and simulate the 10 queries, comparing results between the grader and the simulation. Once all results match, $A$ is obtained.

## Subtask 2

In this subtask, $B = 3$ and $K = 2$.

There are $\binom{6}{2} \times \binom{4}{2} \div 3! = 15$ possible unique queries that can be made. $A$ can once again be determined using the brute force method presented in Subtask 1.

## Subtask 3

In this subtask, $Q = 12$ and the grader is prevented from randomizing the order among sets.

For each query, the $n$th group of $K$ integers in a query will in some way map to the $n$th group of $K$ integers in its result. If some integer $i$ appears in the $n_1$, $n_2$, $n_3$, ..., $n_q$ sets across $q$ queries, the integer $A_i$ will also appear in the same $n_1$, $n_2$, $n_3$, ..., $n_q$ sets across $q$ results.

The goal of this subtask is to find some sequence of queries such that each integer from 1 to $N$ has a unique 'fingerprint' sequence of $n_1, n_2, \ldots, n_q$ of sets that it appears in. $A_i$ is determined by finding the integer which across all queries has the same 'fingerprint'.

Considering the limitations imposed by the black box, this problem can be reduced to:

**Generate $N$ unique base-$B$ integer IDs, such that each digit from 0 to $B-1$ appears in each place value $K$ times.**

With each query we recover one digit of all IDs. An optimal construction can be guaranteed to generate $N$ IDs with a maximum ID length (and thus optimal number of queries) of $\lceil \log_B(N) \rceil$.

## Subtask 4

In this subtask, $K = 2$ and $Q = 4$.

A key observation is that this problem can be bijected to a graph problem. Each input integer can be a node. Placing two integers in a box of size $K = 2$ is to create an undirected edge between them.

By querying the black box the node IDs are reassigned but the overall structure of the graph is still preserved: if a group $[i, j]$ is in the query, the black box will return a group with $(A_i, A_j)$ in some order, indicating an edge between them. Multiple queries can be likened to creating edges of a different 'color'.

The final product is two isomorphic graphs with colored, undirected edges: one collected from data in queries and the other from the results of the black box.

To recover $A_i$, node $i$ in the query graph must have the same connectivity as another node $A_i$ in the result graph. To recover all of $A$, all nodes must be uniquely identifiable by their connectivity, i.e. the graph must not be automorphic.

An intuitive step would be to create a single connected component in as few queries as needed. A cyclic graph can be made in 2 queries:

- Query 1 (Red): $[1, 2], [3, 4], [5, 6], \ldots, [N - 3, N - 2], [N - 1, N]$
- Query 2 (Blue): $[2, 3], [4, 5], [6, 7], \ldots, [N - 2, N - 1], [N, 1]$

The result is a cyclic graph with edges alternating in red and blue color, but this graph is still automorphic. To remove the symmetry, two more queries can be used:

- Query 3 (Green): mix first two groups of Query 1, e.g. $[1, 3], [2, 4], \ldots$
- Query 4 (Yellow): mix first two groups of Query 2, e.g. $[2, 4], [3, 5], \ldots$

Node 1 in the query graph (and thus $A_1$ in the result graph) will have a green edge but no yellow edge connected. The graph is no longer automorphic. The remainder of $A$ can then be recovered by traversing red and blue edges in the result graph, starting from $A_1$.

## Subtask 5

In this subtask, $B = 2$ and $Q = 12$.

Using three queries, it is possible to identify a single value of $A$, such as $A_1$. For subsequent queries, always place this 'root' element in the first set. Then the set containing $A_1$ in the result must correspond to the first set in the query.

This allows us to undo the randomized order among sets, reducing the problem to the case in Subtask 3.

This subtask can thus be solved in $3 + \lceil \log_B(N) \rceil$ queries. Further optimizations may reduce this value to 12 in the worst case.

## Subtask 6

In the general case with arbitrary $K$, a set of $K$ integers can be treated as a densely connected $K$-component.

We can reuse the strategy in Subtask 4 by treating a set of $K$ integers as a supernode with one root (ID 1) and the remaining as connected nodes (IDs 2, 3, ..., $K$). Now each set contains only two 'effective' nodes and the Subtask 4 method applies.

Knowing the values of $B$ root elements in $A$, use the strategy in Subtask 5 by placing each root in a different set. The resulting mapping of root elements recovers the order among sets, again reducing to Subtask 3.

This subtask can thus be solved in $3 + \lceil \log_B(N) \rceil$ queries. With optimizations, this can be reduced to 9 in the worst case.