Solution for Counting Mushrooms

10 points

For each $1 \leq i \leq n-1$, query `use_machine([0,i])`. We can thus determine the species of every mushroom.

25 points

Same idea as above, except that we query `use_machine([i, 0, i+1])` for all odd values of i. As we only query odd values, the number of queries required is halved.

Beyond 25 points

The general strategy is as follows:

- Collect a set of k mushrooms which are known to be of the same species, let these mushrooms be $x_1, x_2, \ldots x_k$.
- We can now make a query of the form $[y_1, x_1, y_2, x_2, \ldots, y_k, x_k]$. The return value to this query reveals the number of type A mushrooms among $y_1, y_2, \ldots y_k$.

A possible strategy is to query $[0, i]$ for each $1 \leq i \leq 2 \cdot k - 2$, thus determining the species of the first $2k-1$ mushrooms. We can then find k mushrooms of the same type. The total number of queries is given by $2k - 2 + \lceil (n - 2k - 1)/k \rceil$. Optimizing over possible values of k gives 397 queries, achieved when $94 \leq k \leq 107$.

It is possible to further reduce the number of queries to $k + \lceil (n - 2k - 1)/k \rceil$. We first query $[0, 1]$ and $[0, 2]$. This gives us two types of mushrooms of the same species, call it x, y.

We can proceed by querying $[3, x, 4, y]$. The return value of this function will identify the species of both mushroom 3 and 4.

Using this approach, we can now find the species of the first $2k-2$ mushrooms with only k queries. By setting k to any value between 137 and 146, we use at most 281 queries.

Finally, observe that when we query $[y_1, x_1, y_2, x_2, \ldots, y_k, x_k]$, we gain information about the species of y_1 by looking at the parity of the return value.

Thus, we can repeatedly expand our set of known mushrooms and increase the value of k appropriately, using at most 245 queries when $73 \leq k \leq 94$.

100 points

Full solution identifies about 5 mushrooms in 2 queries during the first phase. There are constructive appoaches, but we can also try to brute-force all possible strategies that can achieve this. Suppose that we need to identify mushrooms A, B, C, D, E, and we have already identified x and y mushrooms of types 0 and 1 respectively. We can try all possible strings of characters $A, B, C, D, E, 0 (x \text{ times}), 1 (y \text{ times})$ to ask as a first query q_1. Depending on the answer to q_1, the second query q_2 should be able to unambiguously determine each type of A, B, C, D, E, so we can brute-force it as well (note that q_2 may depend on the result of q_1). The number of possible queries is small enough so that all decision trees can be processed within reasonable time.

To avoid overhead for identifying the first few mushrooms, we can actually have two decision trees:

- The first tree uses the only additional mushroom of type 0, and will either figure out that A, B, C, D, E are all type 0 in 1 query, or identify them in 3 queries.
- The second tree uses, when available, 3 mushrooms of type 0 and 1 mushroom of type 1 (or vice versa). It always identifies A, B, C, D, E in 5 queries.

We will employ the second tree whenever there are enough extra mushrooms, and the first one otherwise (that only happens when all identified mushrooms are type 0). This approach can unconditionally identify $5k$ mushrooms in $2k + 1$ queries. Combining this with ideas above and choosing k optimally, we can solve the problem in 226 queries.

Note that there is a much simpler randomized solution that also identifies ~ 2.5 mushrooms per query, but it can't be used because of adaptive grading (unless you can somehow confuse the grader...).

Beyond 100 points

Since the end of the contest a much better solution has been discovered by the community: https://codeforces.com/blog/entry/82924