Code Jam 2013 - Round 1A

Analysis: Good Luck

The luck factor

This problem is unusual in that you do not have full information and are forced to make guesses as part of the solution. We thought it would be a fun change from the usual deterministic setting.

Nevertheless, luck did not play a huge role in this problem. The first dataset was easy enough for many approaches to work. The second dataset was harder, but we estimated that an optimal solution would have very good chances: the probability of an optimal solution failing is only on the order of 1 in a million! This is because 8000 is a lot of independent guesses, and the limit **X** is rather conservative: about 5 standard deviations below the expected number of correct guesses.

The optimal strategy

One may be tempted to apply various heuristics to try to reason about what kind of hidden numbers are likely. In this case, it is best to approach the problem scientifically and simply always go for the highest probability of success!

In order to do that, we compute the probability of each of the 18564 possibilities (for the larger dataset) and pick the largest one.

Why 18564? There are 7 choices for each of the 12 hidden numbers. That seems to give 7^{12} = 13841287201 possibilities, which is a lot. But, the order of hidden numbers doesn't matter, which reduced the number of different possibilities to (12+7-1) choose (7-1) = 18564. Try to derive this formula! Or just generate them all and count.

A priori probabilities: K=0

What if K=0, so we have no information about the hidden numbers at all? It may seem like then it doesn't matter what we guess, since all possibilities are equally likely. Many contestants made this mistake. Some possibilities are more likely a priori than others, even without any additional information!

For example, for the small dataset: 333 is less likely than 234. 6 times less likely, to be exact. Why? Because 234 may have been generated in 6 different ways (234, 243, 324, 342, 423, 432) while 333 can be generated in only 1 way.

In general, if digit **d** appears $\mathbf{C_d}$ times among hidden cards, the probability of that set is $\mathbf{N}! / (\mathbf{C_2}! * ... * \mathbf{C_M}! * (\mathbf{M}-1)^{\mathbf{N}})$.

K=1 and Bayes' theorem

So we have computed the a priori probability of every set of hidden cards, but that does not use the crucial available information: the **K** products of random subsets. How do we use that information? Conditional probabilities are the right tool for the job.

Let's start with **K**=1. For each set of hidden numbers, **A**, we already know the probability of that set happening, Pr(**A**). We also know a product **p** of a random subset of these numbers. What we are trying to compute is the conditional probability that the hidden set is **A** given that the product of a random subset is **p**. Let's write that as Pr(**A** | **p**).

How to compute that? Use the definition of conditional probabilities:
Pr(**A** | **p**) = Pr(**A** ∩ **p**) / Pr(**p**) = Pr(**A**) * Pr(**p** | **A**) / Pr(**p**)
This derivation is called Bayes' theorem.

We already know Pr(**A**), so we only need to know Pr(**p** | **A**). We can pre-compute these values for every **A**. Simply try every possible subset of each possible set of hidden numbers, see what the products are in each case, and build a large table of all these probabilities. There are 18564 * 2^{12} ≈ 76 million such subsets.

Pr(**p**) can then be computed as the sum of Pr(**A**) * Pr(**p** | **A**) over all **A**.

The complete solution

K is greater than 1, but that's not a problem: we iterate the above reasoning for each of the **K** products, adjusting the probabilities of the hidden combinations in the process.

The full solution is then:

- Some precomputation:
  - Generate all possible combinations of hidden numbers, ignoring order.
  - Compute the initial probability of each of these hidden sets.
  - For each possible hidden set, find all possible products of subsets and compute Pr(**p** | **A**). Index these values by **p** for easy lookup later.
- For each hidden set:
  - Start with the pre-computed initial probability distribution over possible hidden sets.
  - Read one product **p** at a time, and adjust the probability distribution by using Bayes' theorem and the pre-computed conditional probabilities Pr(**p** | **A**).
  - Output the most probable possibility.