Riddle

David Keisar Schmidt

April 22, 2022

Abstract

In this article we will discuss an interesting riddle.

Imagine you are a chef at a kingdom's palace, and assume you have 1000 bottles of wine. However, the secret service has managed to reveal a vicious treason against the king, and told you that one of the bottles contains some poison. You, don't know which one has the poison and you need to give the king an answer in 24 hours. Fortunately, there are 10 majestic tasters who wish to help you find the bottle, by tasting the wine by themselves (what a sacrifice!). It is given to you that when someones drinks the poison, he dies exactly 24 hours after, that also implies that all of the tasters need to drink from the bottles at the same time. Assume you can preprocess the bottles, glasses and whatever you want so they can drink at the same time, and also assume there is enough wine **and poison** for everyone. How would you manage to find the bottle?

Contents

1 The Standard Solution

Every whole number can be written in different numeric base. We usually use base 10, which means we look at a number as a powers of 10 :

$$\sum_{i=0}^{k} a_i 10^i$$

for $a_i \in \{0, \ldots, 9\}$. We can also look at base 2, and have

$$\sum_{i=0}^{k} a_i 2^i$$

for $a_i \in \{0, 1\}$. This base yields number such as

$$101 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 = 2 + 4 = 6$$

$$000 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 = 0 + 0 + 0 = 0$$

$$11111 = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 = 31$$

and many others. This type of number encoding is called the binary representation of a number. What is good here is that we can represent each digit in a very simple way, by looking at it as on and off buttons. On means 1 and off means 0. Dead from poison means 1 and didn't die, means 0.

Thus, we can number the bottles from 1 to 1000 and look at their binary representation. For each number, we will look at the indices that has 1 in the binary representation. For example, 1000000000 becomes $\{1\}$, and 1010000000 becomes $\{1, 3\}$. After 24 hours, only the people that drank from the poisoned bottle would die, but those people are exactly the indices that has 1 in the binary representation of the poisoned bottle's number, denote it as D, so we can restore the number by just computing the sum $\sum_{i \in D} 2^i$.

Remark 1. Since indices start from 0 and not from 1, we need to number the people from 0 to 9 instead from 1 to 10.

2 Ruby's solution

Ruby offered the following algorithm.

First Ruby preprocessed the bottles and the people.

We denote $S + \text{idx}$ as the sequence $\{s + idx \mid s \text{ in } S\}$.

Algorithm 1 Preprocess bottles

```
number the bottles from 1 to 1000, and the people from 1 to 10
Let S be a sequence of 10
for i = 0, ..., 9 do do
    initialize S_i = [1, ..., i]
    idx = 2 · i
    while idx < 1024 do
        concatenate S_i and S_i + idx and store the result in S_i
        update idx = idx + 2 · i
Return S
```

After that, Ruby was able to find the Poison using the following algorithm:

Algorithm 2 Find Poison

```
Let S=Preprocess Data()
for i=0,...,9 do
    Let the i taster taste from all the bottles in S_i
Wait 24 hours
init amount = 0
for i=0,...,9 do
    if the i^th taster died then
        amount += 2^i
Return 1024 − amount
```

So, why does it even work?

First, checking all the possibilities yields that it works! So yeah, that works for sure. (see the appendix)

Now, lets prove it formally

Claim 1. let S be the preprocessed set, so for all $i = 1, \ldots, 1023$ it holds that for $D_i = \left\{2^j \mid i \in S_j\right\}$,

$$1024 - \text{sum}\left(D_i\right) = i$$

Proof. We want to find which groups in S contains i. Let $0 \leq j \leq 9$, examine when i is in S_j.

It holds that $a \in S_j$ if and only if there exist $q \in \mathbb{N} \cup \{0\}, 1 \leq r \leq 2^j$ such that $a = (2 \cdot q) \cdot 2^j + r$, this could be written as $a = q \cdot \left(2 \cdot 2^j\right) + r$, thus, $1 \leq a \mod \left(2^{j+1}\right) \leq 2^j$. Hence, we deduce that

$$S_j = \bigcup_{q=0}^{\lfloor \frac{1024}{2^j} \rfloor} \left\{ q \cdot \left(2^{j+1}\right) + r < 1024 \mid 1 \leq r \leq j \right\}$$

how many j exists such that i could be written in the form $i = q \cdot \left(2^{j+1}\right) + r$? Lets look at i's binary representation, that is, there exists an ordered sequence $(n_i)_{i=1}^k$ such that $i = \sum_{j=1}^{k} 2^{n_i}$. assume j is in the sequence, that means,

$$\begin{aligned} i \mod 2^{j+1} &= \sum_{n_i \leq j} 2^{n_i} \mod \left(2^{j+1}\right) \\ \overset{\sum_{n=0}^{k} 2^n = 2^{k+1} - 1 < 2^{k+1}}{=} & \sum_{n_i \leq j} 2^{n_i} \geq 2^j \end{aligned}$$

since j is in the sequence. if there is no smaller value than j in the sequence, that is $i \mod 2^{j+1} = 2^j$ it holds that $i \in S_j$. Otherwise, $i \mod 2^{j+1} > 2^j$ and thus, $i \notin S_j$. This also implies that when $j = 0$ is in the sequence, then $i \in S_j$.

So far we have found that $i \in S_j$ when j is in i's sequence $\iff$ there is no smaller element than j in the sequence.

So we have examined the case when j is in $(n_i)_{i=1}^k$. But, if it isn't in the sequence, then there are two sub cases, if there is a smaller element than j that is in the sequence, and if there isn't. Lets assume there is. Thus

$$\begin{aligned} i \mod 2^{j+1} = \sum_{n_i \leq j} 2^{n_i} \mod \left(2^{j+1}\right) &= \sum_{n_i \leq j} 2^{n_i} \\ \Rightarrow 1 = 2^0 \leq \sum_{n_i \leq j-1} 2^{n_i} &\leq \sum_{i=0}^{j-1} 2^i = 2^j - 1 < 2^j \end{aligned}$$

And, we get that $1 \leq i \mod 2^{j+1} < 2^j$ and thus, $i \in S_j$. Otherwise, if there isn't such element then $i \mod 2^{j+1} = 0$ and thus, $i \notin S_j$.

Before we continue, lets summarize we have found so far:

- if j is in the sequence $(n_i)_{i=1}^{k}$ then $i \in S_j$ if and only if $i \mod 2^{j+1} = 2^j$, that is, there is no element in the sequence that is smaller than j.
- if j isn't in sequence $(n_i)_{i=1}^{k}$ then $i \in S_j$ if and only if $1 \leq i \mod 2^{j+1} < 2^j$ which means, there exists an element in the sequence that is smaller than j.

Now it remains to prove the above statement. We define 6 groups:

1. let I_1 be the indices j such that $i \mod 2^{j+1} = 2^j$, that is, when j is in the sequence and $i \in S_j$.
2. let I_2 be the indices j such that $i \mod 2^{j+1} > 2^j$, that is, when j is in the sequence and $i \notin S_j$.
3. let J_1 be the set of indices j such that $1 \leq i \mod 2^{j+1} < 2^j$, that is, when j isn't in the sequence and $i \in S_j$.
4. let J_2 be the set of indices j such that $i \mod 2^{j+1} = 0$, that is, when j isn't in the sequence and $i \notin S_j$.
5. let I be the indices $n_1, \ldots, n_k$, that is, the binary representation of i.
6. let J be I^c, all the indices that are not in the sequence.

it holds that $I_1 \cup I_2 = I$ and also $J_1 \cup J_2 = J$, and besides $I \cup J = \{0, \ldots, 9\}$.

It is enough to prove the following equality:

$$\sum_{i \in I_1 \cup J_1} 2^i = 1024 - i$$

note that

$$\sum_{i \in I_1 \cup J_1} 2^i = \sum_{i \in I_1} 2^i + \sum_{i \in J_1} 2^i$$

and also note that

$$\sum_{i \in J} 2^i + \sum_{i \in I} 2^i = 1024 - 1 = 1023$$

$$\Rightarrow i = \sum_{i \in I} 2^i = 1023 - \sum_{i \in J} 2^i$$

lets examine exactly how many elements can be in I_1. let i^* be the index of the first bit of i that has the value 1. it follows that only for i^* it holds that $i \mod 2^{i^*+1} = 2^{i^*}$. thus, since $i \neq 0$ it holds that $I_1 = \{i^*\}$ and thus, $|I_1| = 1$. Let us also examine how many elements there are in J_2. lets look again at i^* it holds that $i \mod 2^{j+1} = 0$ if and only if $j + 1 \leq i^*$, that is, $j \leq i^* - 1$ and thus $J_2 = \{0, \ldots, i^* - 1\}$ which implies $|J_2| = i^*$. Thus:

$$\begin{aligned}
\sum_{i \in I_1} 2^i + \sum_{i \in J_1} 2^i &= 2^{i^*} + \sum_{i \in J \setminus J_2} 2^i \\
&= 2^{i^*} + \sum_{i \in J} 2^i - \sum_{i \in J_2} 2^i \\
&= 2^{i^*} + \sum_{i \in J} 2^i - \sum_{i=0}^{i^*-1} 2^i \\
&= 2^{i^*} - \left(2^{i^*} - 1\right) + \sum_{i \in J} 2^i \\
&= 1 + \sum_{i \in J} 2^i
\end{aligned}$$

and then we get

$$1024 - \left(\sum_{i \in I_1} 2^i + \sum_{i \in J_1} 2^i\right) = 1023 - \sum_{i \in J} 2^i = \sum_{i \in I} 2^i = i$$

And that's exactly what we wanted to show. □

3 Appendix

3.1 General Approach

Some of you might have noticed that though the binary representation makes this very simple, it isn't the only possible way.

The only thing we need to find is a function $f : \{0,1\}^{10} \to \{1,\ldots,1024\}$ that is injective, and save it in our memory.

If we do so, for all the bottles, only the people with indices that has the value 1, will drink from the bottle, and when a group dies, we look at its indices, form the sequence of zeros and ones S and return $f(S)$.

3.2 Testing Ruby's solution empirically

The following code checks Ruby's solution, you can run it by creating a file "main.py" copy paste the following content to it, and run the following command in your terminal "python3 main.py".

Here is the code:

```
import numpy as np
def generate_data():
    seqs = dict()
    idx = 1
    while idx < 1024:
        base = np.arange(1, idx+1)
        seqs[idx] = np.concatenate([base+idx · n for n in range(0, int(1024/idx),
            2)])
        idx += idx
    return seqs
def check_solution(seqs):
    for i in range(1, 1024):
        amount = np.sum([2^j for j in range(10) if i in seqs[2^j]])
        if 1024 - amount != i:
            print("Error", i)

if __name__ == '__main__':
    seqs = generate_data()
    check_solution(seqs)
```