Find the Missing Number or Numbers: An Exposition

William Gasarch *

Univ. of MD at College Park

1 Introduction

The following is a classic problem in streaming algorithms and often the first one taught.

Assume n is large and k is constant. Alice is going to say all but k of the numbers in the set $\{1, 2, \ldots, n\}$ in some order. Bob will listen and try to discern what the k missing numbers are. If Bob's brain could easily store and access n bits then he would be able to store a bit vector and mark each number as it came in, then scan the bit vector for the k missing numbers. But what if Bob's brain can only store $m \ll n$ bits?

This can be presented as a fun math puzzle, and for $k = 1$ and even $k = 2$ the answer is fun. Is it fun for $k = 3$? $k \geq 4$? I leave that as an exercise for the reader. We present solutions for $k = 1$, $k = 2$, $k = 3$ and $k \geq 4$.

2 Find the Missing Number

Alice is going to say all but one of the numbers in the set $\{1, 2, \ldots, n\}$ in some order. Bob will listen and try to discern what the missing number is. Alice says the numbers $x_1, x_2, \ldots, x_{n-1}$. They are all distinct elements from $\{1, \ldots, n\}$ but one is missing. Let y_1 be the missing number.

*University of Maryland, College Park, MD 20742, gasarch@cs.umd.edu

Bob can do this problem storing just $O(\log n)$ bits. As Bob hears the numbers he maintains the SUM. This takes just $O(\log n)$ bits. At the end he has $\sum_{1\le i\le n-1} x_i$.

2.1 Solution Using the Sum

Note that

$$\sum_{1\le i\le n-1} x_i = (\sum_{1\le i\le n} i) - y = \frac{n(n+1)}{2} - y_1.$$

Bob finds the missing number is $y_1 = \frac{n(n+1)}{2} - \sum_{1\le i\le n-1} x_i$.

Note 2.1 $\frac{n(n+1)}{2}$ has size $\le \lceil 2\lg n \rceil$, hence this algorithm takes space $\le \lceil 2\lg n \rceil$. Can we do better? Yes! Realize that the final answer is between 1 and n. Hence if we did all calculations mod n we would get the same answer (equating 0 with n). If n is odd then $\frac{n(n+1)}{2} \equiv 0 \pmod{n}$. Hence y_1 is

$$\frac{n(n+1)}{2} - \sum_{1\le i\le n-1} x_i = \left(\frac{n(n+1)}{2} - \sum_{1\le i\le n-1} x_i\right) \pmod{n} = -\sum_{1\le i\le n-1} x_i \pmod{n}.$$

Hence Bob can compute $\sum_{1\le i\le n-1} x_i) \pmod{n}$ which takes $\lceil \lg n \rceil$ bits. He can then subtract it from n (can this by done in $\lg n$ bits?) and get the answer, only using $\lceil \lg n \rceil$ bits.

If n is even then use mod $n+1$.

2.2 Solution Using XOR

An alternative solution: View the numbers $x_1, \ldots, x_{n-1}$ as $\lceil \lg n \rceil$-bit strings. After seeing the $x_1, \ldots, x_L$ Bob maintains $x_1 \oplus x_2 \cdots \oplus x_L$. One can show that the final string Bob has, $x_1 \oplus \cdots \oplus x_{n-1}$, IS the missing number.

3 Find the Missing Two Numbers

Alice is going to say all but two of the numbers in the set $\{1, 2, \ldots, n\}$ in some order. Bob will listen and try to discern which two numbers are missing. We denote the missing numbers y_1, y_2. We give three solutions that use $O(\log n)$ bits.

3.1 Solution that Uses the Quadratic Formula

As Bob hears the numbers he maintains the SUM and SUM OF SQUARES. At the end Bob has

$$\sum_{1 \le i \le n-2} x_i$$
$$\sum_{1 \le i \le n-2} x_i^2$$

Since Bob can calculate and store

$\sum_{1 \le i \le n-2} x_i$ and $\sum_{1 \le i \le n-2} x_i^2$

he can deduce

$$\begin{aligned} s &= y_1 + y_2 = \sum_{1 \le i \le n} x_i - \sum_{1 \le i \le n} i \\ t &= y_1^2 + y_2^2 = \sum_{1 \le i \le n} x_i^2 - \sum_{1 \le i \le n} i^2 \end{aligned}$$

We derive y_1, y_2 from s, t as follows.

$y_2 = s - y_1$

$t = y_1^2 + (s - y_1)^2 = 2y_1^2 - 2sy_1 + s^2$

$2y_1^2 - 2sy_1 + s^2 - t = 0$

Now use the quadratic formula to find y_1 and then $y_2 = s - y_1$ to find y_2.

Note 3.1 The above solution takes $3 \lg n + 2 \lg n = 5 \lg n$ space since we need to store a sum of size $O(n^3)$ and a sum of size $O(n^2)$. We can do better! We can't use mod n since the quadratic might have more than 2 roots mod n. Let p be a prime such that $n \le p \le 2n$. Do all of the above mod p works. This will take $\le \lg(2n) + \lg(2n) \le 2\lg(n) + O(1)$.

3.2 Solution that Uses Sums of Powers

This solution is identical to the one in Section 3.1 up until we find s, t. We then find $y_1 y_2$ as follows:

$$\frac{s^2 - t}{2} = y_1 y_2.$$

Let $s = y_1 + y_2$ and $p = y_1 y_2$. Form the polynomial

$$X^2 - sX + p = X^2 - (y_1 + y_2)X + y_1 y_2 = (X - y_1)(X - y_2).$$

Find the roots of this polynomial.

We call this THE POLY-ROOTS TRICK throughout. Note that all we need is the symmetric functions y_1, y_2. More generally we will need the symmetric functions of $y_1, y_2, \ldots, y_k$.

Note 3.2 Similar to the solution in Section 3.1, we can do all of this mod p and hence space $2\lg(n) + O(1)$.

3.3 Solution that uses Symmetric Functions Throughout

As Bob hears the numbers he maintains the SUM and the SUM OF PRODUCTS OF PAIRS. After hearing the first L numbers he has in his head $\sum_{1 \le i \le L} x_i$ AND $\sum_{1 \le i < j \le L}^{L} x_i x_j$.

We need to show that he can actually do this. Let

$$\begin{aligned}
s_0^L(x_1, \ldots, x_L) &= 1 (\text{ We don't really need } s_0 \text{ but it will make the notation nice.}) \\
s_1^L(x_1, \ldots, x_L) &= \textstyle\sum_{1 \le i \le L} x_i \\
s_2^L(x_1, \ldots, x_L) &= \textstyle\sum_{1 \le i < j \le L} x_i x_j
\end{aligned}$$

For notational convenience we use s_i^L to mean $s_i^L(x_1, \ldots, x_L)$

Assume Bob has s_0^{L-1}, s_1^{L-1} and s_2^{L-1}. And then Bob sees x_L. Bob wants s_0^L, s_1^L, s_2^L.

We explain all of this by expanding everything out

$s_0^L = 1$. Thats easy.

$$s_1^L = (x_1 + \cdots + x_{L-1}) + x_L = s_1^{L-2} + x_L.$$

We rewrite this as

$$s_1^L = (x_1 + \cdots + x_{L-1}) + x_L = s_1^{L-2} + x_L s_0^{L-1}$$

since this way it will give all of he equations (and more so for the $k = 3$ and $k \geq 4$ cases) look the same.

$$s_2^L = \sum_{1 \leq i < j \leq L} x_i x_j$$

We break this sum up into two parts- those parts that use x_L and those parts that do not. If a product of two x_i terms uses x_L then it is of the form $x_i x_L$. hence

$$s_2^L = \sum_{1 \leq i < j \leq L-1} x_i x_j + x_L(x_1 + \cdots + x_{L-1}).$$

AH- note that $\sum_{1 \leq i < j \leq L-1} x_i x_j + x_L(x_1 + \cdots + x_{L-1})$ is s_2^{L-1} and $x_1 + \cdots + x_{L-1}$ is s_1^{L-1}. So we write this as

$$s_2^L = S_2^{L-1} + x_L s_1^{L-1}.$$

We now write all of the equations together:

$$
\begin{aligned}
s_0^L &= 1 \\
s_1^L &= s_1^{L-1} + x_L s_0^{L-1} \\
s_2^L &= s_2^{L-1} + x_L s_1^{L-1}
\end{aligned}
$$

Hence we can keep the counters s_1 and s_2 and update them easily, using only $O(\log n)$ space.

At the end we have s_1^{n-2} and s_2^{n-2}.

KEY: Bob can compute, before seeing any of the data:

$$
\begin{aligned}
s_1^n &= \sum_{1 \le i \le n} x_i = \sum_{1 \le i \le n} i \\
s_2^n &= \sum_{1 \le i < j \le n} x_i x_j = \sum_{1 \le i < j \le n} ij
\end{aligned}
$$

We want to derive $s_1^2(y_1, y_2) = y_1 + y_2$ and $s_2^2(y_1, y_2) = y_1 y_2$ and then finish up the proof as we did in Section 3.2. For notational convenience we denote $s_1^2(y_1, y_2)$ by s_1^2 and $s_2^2(y_1, y_2)$ by s_2^2. Note that

$$
\begin{aligned}
s_1^n &= s_1^{n-2} + s_1^2 \\
s_2^n &= s_2^{n-2} + s_1^{n-2} s_1^2 + s_2^2
\end{aligned}
$$

Note that s_1^n, s_2^n, s_1^{n-2}, s_2^{n-2} are known. Hence $s_1^2 = (y_1 + y_2)$ and $s_2^2 = y_1 y_2$ can be determined. Now do the poly-root trick.

4 Find the Missing Three Numbers

Alice is going to say all but three of the numbers in the set $\{1, 2, \ldots, n\}$ in some order. Bob will listen and try to discern which three numbers are missing. We denote the missing numbers y_1, y_2, y_3. We give two solutions.

4.1 Solution Using Sums of Powers

Bob keeps track of the sum of terms, squares of terms, and cubes of terms. By subtracting them from the known quantities $\sum_{i=1}^{n} i$, and $\sum_{i=1}^{n} i^2$, and $\sum_{i=1}^{n} i^3$ Bob obtains:

$y_1 + y_2 + y_3$

$y_1^2 + y_2^2 + y_3^2$

$y_1^3 + y_2^3 + y_3^3$

Can he use these to determine y_1, y_2, y_3?

We WANT to obtain:

$y_1 + y_2 + y_3$ (thats easy!)

$y_1y_2 + y_1y_3 + y_2y_3$

$y_1y_2y_3$

ONCE we have them we form the polynomial

$$X^3 - (y_1 + y_2 + y_3)X^2 + (y_1y_2 + y_1y_3 + y_2y_3)X - y_1y_2y_3 = (X - y_1)(X - y_2)(X - y_3).$$

Find its roots. Then you have $y_1, y_2.y_3$.

OKAY, now to find those functions of y_1, y_2, y_3.

We first try an intuitive thing:

$$(y_1 + y_2 + y_3)^2 - (y_1^2 + y_2^2 + y_3^2)$$

This is intuitive to try since we can already see that all of the square terms will drop out and might leave us with something simple.

$$(y_1+y_2+y_3)^2-(y_1^2+y_2^2+y_3^2)=2y_1y_2+2y_1y_3+2y_2y_3=2(y_1y_2+y_1y_3+y_2y_3).$$

GREAT!- that last term is twice what we want. In other words:

$$y_1y_2+y_1y_3+y_2y_3=\frac{(y_1+y_2+y_3)^2-(y_1^2+y_2^2+y_3^2)}{2}$$

NOW we want $y_1y_2y_3$.

The next equation is harder to motivate so I won't even try (though its a special case of Newton's identity which I will discuss when doing the general k case):

$$y_1y_2y_3=\frac{(y_1y_2+y_1y_3+y_2y_3)(y_1+y_2+y_3)-(y_1+y_2+y_3)(y_1^2+y_2^2+y_3^2)+(y_1^3+y_2^3+y_3^3)}{3}.$$

Great! Now that we have $y_1+y_2+y_3$, $y_1y_2+y_1y_3+y_2y_3$, $y_1y_2y_3$

4.2 Solution that uses Symmetric Functions Throughout

This solution is due to Y. Minsky, A. Trachtenberg, and R. Zippel [1].

The KEY to the solution in the last section was that we used the sums-of-powers to obtain the symmetric functions $y_1+y_2+y_3$, $y_1y_2+y_1y_3+y_2y_3$, $y_1y_2y_3$. In this solution we get the symmetric functions more directly.

As Bob hears the first L numbers $x_1,\ldots,x_L$ he maintains:

- $\sum_{1\le i\le L} x_i$.
- $\sum_{1\le i<j\le L}^{L} x_ix_j$.
- $\sum_{1\le i<j<k\le L}^{L} x_ix_jx_k$.

We need to show that he can actually do this. Let

$$
\begin{aligned}
s_0^L(x_1,\ldots,x_L) &= 1 \text{ (We have this just so the equations look nice.)}\\
s_1^L(x_1,\ldots,x_L) &= \textstyle\sum_{1\le i\le L} x_i\\
s_2^L(x_1,\ldots,x_L) &= \textstyle\sum_{1\le i<j\le L} x_i x_j\\
s_3^L(x_1,\ldots,x_L) &= \textstyle\sum_{1\le i<j<k\le L} x_i x_j x_k
\end{aligned}
$$

For notational convenience we use s_i^L to mean $s_i^L(x_1,\ldots,x_L)$

We need to show that Bob can easily get s_0^L, s_1^L, s_2^L, s_3^L from s_0^{L-1}, s_1^{L-1}, s_2^{L-1}, s_3^{L-1} and x_L.

$s_0^L = 1$ so thats easy.

$$s_1^L = (x_1 + \cdots + x_{L-1}) + x_L = s_1^{L-1} + x_L = s_1^{L-1} + x_L s_0^{L-1}.$$

SO we now have s_1^L in terms of stuff Bob knows.

Consider $s_2^L = \sum_{1\le i<j\le L} x_i x_j$ We separate out the pairs the involve x_L from the ones that don't. The ones that don't involve x_L are just $s_2^{L-1} = \sum_{1\le i<j\le L-1} x_i x_j$. The ones that DO involve x_L involve just x_L and some x_i with $i < L$. Thats just

$$x_1x_L + x_2x_L + \cdots + x_{L-1}x_L = x_L(x_1 + \cdots + x_{L-1}) = x_L s_1^{L-1}$$

Hence

$$s_2^L = s_2^{L-1} + x_L s_1^{L-1}$$

SO we now have s_2^L in terms of stuff Bob knows.

s_3^L is similar and we leave it to the reader. To summarize we have:

$$
\begin{aligned}
s_0^L &= 1 \\
s_1^L &= s_1^{L-1} + x_L s_0^{L-1} \\
s_2^L &= s_2^{L-1} + x_L s_1^{L-1} \\
s_3^L &= s_3^{L-1} + x_L s_2^{L-1}
\end{aligned}
$$

Hence we can keep the counters s_1, s_2, s_3 and update them easily, using only $O(\log n)$ space.

At the end we have s_1^{n-3} and s_2^{n-3} and s_3^{n-3}.

KEY: Bob can compute, before seeing any of the data:

$$
\begin{aligned}
s_0^n &= 1 \\
s_1^n &= \sum_{1\le i\le n} x_i = \sum_{1\le i\le n} i \\
s_2^n &= \sum_{1\le i<j\le n} x_i x_j = \sum_{1\le i<j\le n} ij \\
s_3^n &= \sum_{1\le i<j<k\le n} x_i x_j x_k = \sum_{1\le i<j<k\le n} ijk
\end{aligned}
$$

We want to derive $s_1^3(y_1, y_2, y_3) = y_1 + y_2 + y_3$ and $s_2^3(y_1, y_2, y_3) = y_1y_2 + y_1y_3 + y_2y_3$ and $s_3^3(y_1, y_2, y_3) = y_1y_2y_3$. We can then finish up the proof using the poly-roots trick. For notational convenience we denote $s_i^3(y_1, y_2)$ by s_i^3. Note that

For s_1 it is easy to relate s_1^n, s_1^{n-3}, and s_1^3 since

$$
s_1^n(x_1 \ldots, x_n) = x_1 + \cdots + x_n = (x_1 + \cdots + x_{n-3}) + (y_1 + y_2 + y_3) = s_1^{n-3} + s_1^3.
$$

Hence we can derive s_1^3 from s_1^n and s_1^{n-3}, both of which we know.

Consider $s_2^n = \sum_{1\le i<j\le n} x_i x_j$. We break this into pieces. Some pairs use NO elements of y_1, y_2, y_3. That would be $s_2^{n-3} = \sum_{1\le i<j\le n-3} x_i x_j$. Some pairs use y_1 and some element of $\{x_1, \ldots, x_{n-3}\}$. That would be

$$y_1(x_1 + \cdots + x_{n-3}) = y_1 s_1^{n-3}$$

Some pairs use y_2 and some element of $\{x_1, \ldots, x_{n-3}\}$. That would be

$$y_2(x_1 + \cdots + x_{n-3}) = y_2 s_1^{n-3}$$

Some pairs use y_3 and some element of $\{x_1, \ldots, x_{n-3}\}$. That would be

$$y_3(x_1 + \cdots + x_{n-3}) = y_3 s_1^{n-3}$$

The sum of the last three cases is

$$(y_1 + y_2 + y_3)(x_1 + \cdots + x_{n-3}) = (x_1 + \cdots + x_{n-3})(y_1 + y_2 + y_3) = s_1^{n-3} s_1^3$$

Some pairs use two elements from $\{y_1, y_2, y_3\}$. That would be

$$y_1 y_2 + y_1 y_3 + y_2 y_3 = s_2^3.$$

If you put this all together you get:

$$s_2^n = s_2^{n-3} s_0^3 + s_1^{n-3} s_1^3 + s_0^{n-3} s_2^3.$$

A similar equation for s_3^n can also be derived; however, we leave that for the reader.

To summarize we have in total:

$$\begin{aligned}
s_1^n &= s_1^{n-3} s_0^3 + s_0^{n-3} s_1^3 \\
s_2^n &= s_2^{n-3} s_0^3 + s_1^{n-3} s_1^3 + s_0^{n-3} s_2^3 \\
s_3^n &= s_3^{n-3} s_0^3 + s_2^{n-3} s_1^3 + s_1^{n-3} s_2^3 + s_3^{n-3} s_0^3
\end{aligned}$$

Note that s_1^n, s_2^n, s_3^n, s_1^{n-3}, s_2^{n-3}, s_2^{n-3}, s_0^{n-3}, s_0^3 are known. Hence s_1^3, s_2^3, s_3^3 can all be derived. Now we can do the poly-roots trick.

5 General k

We'll need the symmetric functions for both solutions.

Notation 5.1

$$
\begin{aligned}
s_0^L(x_1,\ldots,x_L) &= 1\\
s_1^L(x_1,\ldots,x_L) &= \sum_{1\le i\le L} x_i\\
s_2^L(x_1,\ldots,x_L) &= \sum_{1\le i_1<i_2\le L} x_{i_1}x_{i_2}\\
s_3^L(x_1,\ldots,x_L) &= \sum_{1\le i_1<i_2<i_3\le L} x_{i_1}x_{i_2}x_{i_3}\\
\vdots &= \vdots\\
s_k^L(x_1,\ldots,x_L) &= \sum_{1\le i_1<\cdots<i_k\le L} x_{i_1}\cdots x_{i_k}
\end{aligned}
$$

We often leave out the arguments for notational convenience.

We give two solutions. The arguments used here are similar to the ones used in the $k=3$ case so we omit them.

5.1 Solution Using Sums of Powers

Bob keeps track of the sums of powers, up to the kth power. At the end he has

- $\sum_{i=1}^{n-k} x_i$,
- $\sum_{i=1}^{n-k} x_i^2$,
- $\vdots$
- $\sum_{i=1}^{n-k} x_i^k$.

From these Bob can easily derive

- $\sum_{i=1}^{k} y_i$,
- $\sum_{i=1}^{k} y_i^2$,
- $\vdots$
- $\sum_{i=1}^{k} y_i^k$.

We find the symmetric functions FROM these. How? By using Newton's identities. We abbreviate $s_m^k(y_1, \ldots, y_k)$ by s_m^k. We abbreviate $\sum_{i=1}^{k} y_i^p$ by p_k.

$$ms_m^k(y_1, \ldots, y_k) = \sum_{i=1}^{m} (-1)^{i-1} s_{m-i}^n (y_1, \ldots, y_k) \sum_{j=1}^{k} y_j^i$$

$$ms_m^k = \sum_{i=1}^{m} (-1)^{i-1} s_{m-i}^n (y_1, \ldots, y_k) \sum_{j=1}^{k} y_j^i$$

5.2 Solution Using Symm Functions Throughout

This algorithm is essentially from a paper by Yaron Minksy, Ari Trachtenberg, Richard Zippel [1].

Using an argument similar to the one in Section 3.3 we can obtain:

Lemma 5.2

$$\begin{aligned}
s_0^L &= 1 \\
s_1^L &= s_1^{L-1} + x_L s_0^{L-1} \\
s_2^L &= s_2^{L-1} + x_L s_1^{L-1} \\
s_3^L &= s_3^{L-1} + x_L s_2^{L-1} \\
\vdots &= \vdots \\
s_k^L &= s_k^{L-1} + x_L s_k^{L-1}
\end{aligned}$$

Hence if Bob has s_1^{L-1}, ..., s_k^{L-1}, and then sees x_L, you will be able to calculate s_1^L, ..., s_k^L. If all of the x_i are in $\{1, \ldots, n\}$ then you can do all of this in space $O(k \log n)$. Therefore Bob can find s_1^{n-k}, ..., s_k^{n-k}.

KEY: Bob can compute the following independent of the data. He will do this after he knows $s_1^{n-k}, \ldots, s_k^{n-k}$, and use them one-at-a-time below to save space.

$$
\begin{aligned}
s_1^n &= \textstyle\sum_{1\le i\le n} i \\
s_2^n &= \textstyle\sum_{1\le i_1<i_2\le n} ij \\
s_3^n &= \textstyle\sum_{1\le i_1<i_2<i_3\le n} x_i x_j x_k = \sum_{1\le i_1<i_2<i_3\le n} ijk \\
\vdots &= \vdots \\
s_k^n &= \textstyle\sum_{1\le i_1<\cdots<i_k} x_{i_1}\cdots x_{i_k} = \sum_{1\le i_1<\cdots<i_k\le n} i_1 i_2\cdots i_k
\end{aligned}
$$

We seek, for all $1 \le i \le k$, $s_i^k(y_1,\ldots,y_k)$. We denote these s_i^k for notational convenience. The following are easily seen to be true:

$$
\begin{aligned}
s_1^n &= s_1^{n-k}s_0^k + s_1^k s_0^{n-k} \\
s_2^n &= s_2^{n-k}s_0^k + s_1^{n-k}s_1^k + s_0^{n-k}s_2^k \\
s_3^n &= s_3^{n-k}s_0^k + s_2^{n-k}s_1^k + s_1^{n-k}s_2^k + s_0^{n-k}s_3^k \\
\vdots &= \vdots \\
s_i^n &= s_i^{n-k}s_0^k + s_{i-1}^{n-k}s_1^k + s_{i-2}1^{n-k}s_2^k + \cdots s_0^{n-k}s_i^k \\
\vdots &= \vdots \\
s_k^n &= s_k^{n-k}s_0^k + s_{k-1}^{n-k}s_1^k + s_{k-2}1^{n-k}s_2^k + \cdots s_0^{n-k}s_k^k
\end{aligned}
$$

Note that Bob knows s_i^n, s_i^{n-k}, s_0^k, s_0^{n-k}. Hence Bob can use these equations to, one at a time (to save space) find s_1^k,s_2^k, …, s_k^k. Bob forms the polynomial

$$X^k - s_1^k X^{k-1} + s_2^k X^{k-2} + \cdots + (-1)^k s_k^k = (X-y_1)(X-y_2)\cdots(X-y_k).$$

Find the roots.

References

[1] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.