VOL. 88, NO. 1, FEBRUARY 2015

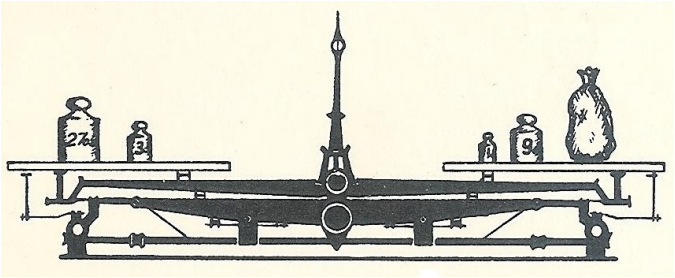

Figure 1 Steinhaus weighs in

Balance weighing — variations on a theme

Ethan D. Bolker
UMass Boston
ebolker@gmail.com
Samuel A. Feuer
Wesleyan University
sfeuer45@gmail.com
Catalin Zara
UMass Boston
Catalin.Zara@umb.edu

Summary We explore weighing problems when you may use at most one of each of an increasing sequence of weights but may put some on the balance along with the unknown object. Our solutions depend on analyzing arithmetic when you expand integers in the mixed base defined by the weights. We begin with elementary school exercises and end with conjectures.

A group of fourth graders at the Heath School in Brookline, MA, works to discover the optimal sequence of weights they'd need to balance an object of unknown integral weight. They quickly discover the greedy algorithm for extending the sequence $(1, 2)$, realize that the next weight is one more than the sum of the ones found so far, and easily guess that it is double the last one found. Second graders can guess this too. Fourth graders can understand that knowing how to place the weights is equivalent to expanding integers in base 2.

Suppose you allow weights on either side of the balance. Then the optimal sequence begins $(1, 3)$ because $2 = 3 - 1$, so you can weigh a 2 pound object by putting it on the balance along with the 1 weight opposite the 3. The largest integer you can weigh is $4 = 1 + 3$. The next weight might be 6 instead of $5 = 4 + 1$ since $5 = 6 - 1$. With time and patience the fourth graders discover that 9 and then 27 are the right next weights. Since $1 + 3 + 9 + 27 = 40 = (3^4 - 1)/2$ they have solved the weight problem of Bachet de Méziriac:

> *A merchant has a forty-pound measuring weight that broke into four pieces as the result of a fall. When the pieces were subsequently weighed, it was found that the weight of each piece was a whole number of pounds and that the four pieces could be used to weigh every integral weight between* 1 *and* 40 *pounds.*
>
> *What were the weights of the pieces?* [**1**]

That problem reappears often. Figure 1 is from an early edition of Hugo Steinhaus's

Mathematical Snapshots [**10**]. It was a Car Talk puzzler in 2011, reposted online in 2017 [**6**] [**7**]. It's a frequent visitor on StackExchange and other web question and answer sites [**3**] [**9**] [**11**]. These references show that the following theorem isn't new. This proof is implicit in the literature too.

Theorem 1. *When you may place weights on either side of the balance you can weigh any integer uniquely using weights* $(3^k) = (1, 3, 9, \ldots)$.

Proof. To weigh n, write its base 3 representation. Then rewrite that representation using the digits $-1, 0, 1$ instead of the usual $0, 1, 2$ by repeating one of the following transformations as long as that's possible:

- Replace an instance of $x2y$ with $(x + 1)(-1)y$.
- Replace an instance of $x3y$ with $(x + 1)0y$.

Each of these replacements leaves the value of n invariant, the first because $2 \times 3^{k-1} = 3^k - 3^{k-1}$, the second because $3 \times 3^{k-1} = 3^k$.

The algorithm must terminate since the sum of the digits is strictly decreasing. The result tells you which weights to put on the balance along with the unknown n.

To convert back to the the usual base 3 representation

- Replace an instance of $x(-1)y$ with $(x - 1)2y$.
- Replace an instance of $x(-2)y$ with $(x - 1)1y$.

These changes do not change n, increase the digit sum, and end when the only digits are 0, 1 and 2. The existence of this inverse map shows that the balanced ternary representation and hence the disposition of weights is unique. □

Using digits $0, \pm 1$ to represent integers in base 3 is called *balanced ternary* notation. It has a long history [**2**][**5**].

We posed a generalization. What if you are allowed *at most one* weight along with the unknown object? We didn't know the answer when we asked — intentionally, so kids could know that mathematicians regularly invented and attacked new questions that didn't yet have answers. It's easy to see that the sequence of weights begins $(1, 3, 8)$; to weigh 5 you write $5 = 8 - 3$ rather than $5 = 9 - 3 - 1$ since the latter requires two negatives. The kids ran out of time before they could get any further. But we remained intrigued.

The start looks like every other Fibonacci number. We fantasized that the sequence might continue with 21. It doesn't: that would require two negative weights for $15 = 21 - 8 + 3 - 1$. After correcting several arithmetic errors we found that the maximum next weight is 18. Watch the mystery weight sequence $X = (1, 3, 8, 18, \ldots)$ reappear later in this paper.

We hoped the Fibonacci pattern could be rescued by changing the problem — mathematicians do that all the time. Perhaps "using at most one negative weight" is the wrong hypothesis. So we tried "using at most half negative weights". Sadly, the weight sequence then starts $(1, 3, 8, 24, \ldots)$. The good news is that asking the more general questions led to some nice mathematics.

A framework for weighing problems

We'll start by restricting ourselves to weights on just one side of the balance.

Let $W = (w_0 < w_1 < w_2 < \cdots)$ be an increasing sequence of positive integers; we call such a sequence a *weight sequence*, and refer to the terms w_k as *weights*. We want to study weighing problems that allow for at most one weight of each kind. Since the theorems and proofs work just as well for a while with a bounded number $c \geq 1$ of each weight we'll state them in that generality.

Definition 1. *The weight sequence W is* c-complete *if every positive integer n is a sum of weights in W using each weight at most c times.*

Write $s_k = w_0 + w_1 + \cdots + w_{k-1}$ for the sum of the first k weights. Set $s_0 = 0$.

Theorem 2. *The weight sequence W is c-complete if and only if*

$$w_k \leq 1 + cs_k \tag{1}$$

for each $k \geq 0$.

Proof. The largest n you can weigh with the first k weights is cs_k. If the next weight is greater than $1 + cs_k$ then you can't weigh $1 + cs_k$.

To prove the converse, suppose inequality (1) is true for all k.

For each nonnegative integer n let $r(n)$ be the base $c + 1$ representation of n: a string $d_k d_{k-1} \ldots d_0$ of digits between 0 and c. Then let $f(n)$ be the value of $r(n)$ in base W:

$$f(n) = d_k w_k + d_{k-1} w_{k-1} + \cdots + d_0 w_0.$$

Since $r((c+1)^k)$ is the string with a 1 followed by k 0's, $f((c+1)^k) = w_k$. In particular that says f is unbounded.

If $r(n)$ ends in anything other than c then $r(n+1)$ agrees with $r(n)$ except at the units digit, which increases by 1, so $f(n+1) = f(n) + 1$.

If $r(n)$ ends in a string of c's of length $j > 0$ then calculating $r(n+1)$ from $r(n)$ involves a carry to place $j+1$, essentially replacing cs_j by w_j. Then inequality (1) implies $f(n+1) = f(n) + (w_j - cs_j) \leq f(n) + 1$.

Thus f is an unbounded function with $f(1) = w_0 = 1$ and at each step f either increases by 1, or is unchanged, or decreases. Hence f is surjective. □

The proof of Theorem 2 can be easily tweaked and generalized to prove slightly stronger results. Rewriting the argument in each of the following corollaries would obscure the bones of the argument, so we will leave some of the tweaks to the reader.

Corollary 1. *A weight sequence W is c-complete if and only if for every index $k \geq 0$, every positive integer $n \leq cs_{k+1}$ can be written as a sum of weights using only the weights $w_0, w_1, \ldots, w_k$ at most c times each.*

Proof. Note that $r((c+1)^k - 1) = ccc \ldots cc$, hence $f(((c+1)^k - 1)) = cs_k$. □

Corollary 2. *You need not allow the same maximum number of each kind of weight. The theorem has an obvious generalization when allowing c_k instances of weight k. The proof works with a mixed base representation using digits $0, 1, \ldots, c_k$ in column k.*

Corollary 3. *The weight sequence W is c-complete if $w_{k+1}/w_k \leq c + 1$ for all k.*

Proof. Clearly $w_0 \leq 1 + cs_0$. Then induction works:

$$w_{k+1} \leq (c+1)w_k \leq cw_k + w_k \leq cw_k + 1 + cs_k = 1 + cs_{k+1}.$$ □

The condition $w_{k+1}/w_k \le c+1$ is sufficient, but not necessary. For example, the sequence $(1, 2, 3, 7, 14, 28, \ldots)$ is 1-complete even though $7/3 > 2$.

Corollary 4. *As long as the weak inequality in (1) is an equality (which may be true for all k) the weight sequence is the powers of $c+1$ and each n has a unique representation corresponding to its base $c+1$ expansion. Once there is a strict inequality there is occasional nonuniqueness.*

Using weights on both sides of the balance

In Theorem 1 we explored the relation between ternary and balanced ternary representations of integers: the digit sets $[0, 2] = \{0, 1, 2\}$ and $[-1, 1] = \{-1, 0, 1\}$ are essentially equivalent. That kind of equivalence persists when we allow multiple copies of each weight.

Lemma 1. *Suppose the weight sequence W represents n using $v(d)$ instances of coefficient digit d, for d in the digit set $[0, \ldots, 2c]$. Suppose further that $n < cs_k$. Then W represents $cs_k - n$ using digit $c - d$ in the digit set $[-c, \ldots, c]$ just $v(d)$ times. Conversely, that assertion is true when you swap the roles of the two digit sets.*

Proof. Use the known representation of n to represent the complement:

$$\begin{array}{rcccccc} cs_k - n & = & cw_0 & +cw_1 & \cdots & +cw_k \\ & & -d_0w_0 & -d_1w_1 & \cdots & -d_kw_k \\ & = & (c-d_0)w_0 & +(c-d_1)w_1 & \cdots & +(c-d_k)w_k. \end{array}$$

The argument works the other way since the function $x \mapsto a - x$ is its own inverse; we use it with $a = cs_k$ for n and $a = c$ for the digit sets. □

For example, $s_3 = 30$ for the mystery weight sequence $X = (1, 3, 8, 18, \ldots)$. The pair $(6, 24)$ exhibits these three sets of paired representations:

$$\begin{array}{rrrr} 6 = & 8-3+1 = & 18-8-3-1 = & 2\times 3 \\ 24 = & 18+2\times 3 = & 2\times 8+2\times 3+2\times 1 = & 18+8-3+1 \end{array}$$

This lemma provides an alternative algorithm for calculating the balanced ternary representation of n when you can use as many negative weights as you wish. For the powers of 3, $s_k = (3^k - 1)/2$. Find the smallest k such that $s_k > n$, find the ordinary base 3 representation of $s_k - n$ using $k+1$ digits and swap: interchange 0 and 1 and change 2 to -1.

Definition 2. *The weight sequence W is* c-balanced *if for every index $k \ge 0$, every positive integer $n \le cs_{k+1}$ can be written as a sum*

$$n = d_kw_k + \cdots + d_0w_0$$

with digits d_i in $[-c, c]$.

We first proved the next theorem by modifying the proof of Theorem 2. Then we discovered Lemma 1, which makes for a much cleaner argument.

Theorem 3. *A weight sequence is c-balanced if and only if it is $2c$-complete.*

Proof. Let W be a weight sequence. Suppose $n > 0$. Choose k large enough so that $cs_k > n$. If W is $2c$-complete then $m = cs_k - n$ is a sum of weights using each at most $2c$ times. Then Lemma 1 implies n is a sum of weights using digits from $[-c, c]$. Conversely, if W is c-balanced, the lemma shows how to use the representation of m with those digits to represent n using $[0, 2c]$. □

Restricting the number of negative weights

Now we're ready to return to open problems like the one we asked fourth graders. Since we have just one weight of each kind to play with we'll restrict our discussion to the special case $c = 1$. Then we know that powers of 2 form the 1-complete weight sequence that leads to unique solutions while the sequence of powers of 3 does the same for the 2-complete, hence 1-balanced sequences.

We will continue to call weights that share the balance pan with the unknown "negative weights". Theorem 3 says we can isolate the object and use some "double weights" on the other pan instead when that is more convenient

We want to allow the number of negative/double weights to depend on the number of weights we have available at any moment.

Definition 3. *A* bounding sequence *is a nondecreasing sequence of non-negative integers that increases by at most* 1 *at each step. Formally, it's a sequence* $\mu = (\mu_0, \mu_1, \mu_2 \cdots)$ *where* $\mu_0 = 0$ *and* $\mu_{k+1} - \mu_k \in \{0, 1\}$ *for all* $k > 0$.

A bounding sequence tells us that when using k weights we may put at most μ_k of them on the balance along with the unknown. The bounding sequence $\mu_{\text{min}} = (0, 0, \ldots)$ allows no negative weights; the bounding sequence $\mu_{\text{max}} = (0, 1, 2, \ldots)$ allows as many as you wish. The bounding sequence $(0, 1, 1, \ldots)$ allows just one negative weight.

Definition 4. *Let* μ *be a bounding sequence. A weight sequence* W *is* μ-balanced *if every* n *between* 1 *and* w_k *can be weighed using at most* μ_k *negative weights, that is, can be represented as*

$$n = d_k w_k + d_{k-1} w_{k-1} + \cdots + d_1 w_1 + d_0 w_0, \tag{2}$$

with digits in $[-1, 1]$ *at most* μ_k *of which are* -1.

Definition 5. *Let* μ *be a bounding sequence. A weight sequence* W *is* μ-complete *if every* n *between* 1 *and* $w_k - 1$ *can be weighed using at most* μ_k *double weights, that is, can be represented as*

$$n = b_{k-1} w_{k-1} + \cdots + b_1 w_1 + b_0 w_0,$$

with digits in $[0, 2]$ *at most* μ_k *of which are* 2.

Lemma 2. *A weight sequence* W *is* μ*-balanced if every* n *between* 1 *and* s_{k+1} *can be weighed using at most* μ_k *negative weights, that is, can be represented as*

$$n = d_k w_k + d_{k-1} w_{k-1} + \cdots + d_1 w_1 + d_0 w_0, \tag{3}$$

with digits in $[-1, 1]$ *at most* μ_k *of which are* -1.

Proof. The only difference between this lemma and the definition of μ-balanced is the change from $n \le w_k - 1$ to $n \le s_{k+1} - 1$.

One implication is clear, since $w_k < s_{k+1}$ for all k.

If $w_k \le n < s_k$ then $n = w_k + m$ with $m \le s_{k-1}$. Then by induction m is a sum of weights at most μ_k of which are negative; hence that's true for n as well. □

Theorem 4. *Let* μ *be a bounding sequence. A weight sequence* W *is* μ*-balanced if and only if it is* μ*-complete.*

Proof. Suppose W is μ-balanced and $n < w_k$. Then $1 \le n < s_{k+1}$, so $s_{k+1} - n$ can be represented using at most μ_k negative weights. Then Lemma 1 shows n can be represented using at most μ_k double weights.

Conversely, suppose W is μ-complete and $n \le s_{k+1}$. If $n = s_j$ for some $j \le k+1$ then n is a sum of weights none of which is negative. If not, let j be the unique index such that $s_j < n < s_{j+1}$. Then apply Lemma 1 to the pair $(n, s_{j+1} - n)$. □

Optimal weight sequences

Definition 6. *Let μ be a bounding sequence. A weight sequence W is* μ-optimal *if for every index $k \ge 0$, the value of w_k is the largest possible among μ-balanced/μ-complete weight sequences.*

Theorem 5. *For every μ there is a unique μ-optimal weight sequence W, constructed recursively as follows:*

- $w_0 = 1$*;*
- *Suppose $w_0, w_1, \ldots, w_{k-1}$ known. Then w_k is the smallest integer that can't be represented as a sum of smaller weights, allowing at most μ_k double weights.*

Then

$$2w_k \le w_{k+1} \le 3w_k$$

with equality on the left if and only if $\mu_k = 0$ and equality on the right if and only if $\mu_{k+1} = 1 + \mu_k$.

Proof. With only the first term of the sequence, all we can represent is w_0, so we must start with $w_0 = 1$. Suppose that we have determined optimal values for w_0, w_1, w_2, $\ldots$, w_k and want to find the optimal w_{k+1}.

Since W is μ-complete, all integers from 0 to $w_k - 1$ can be represented using weights up to w_{k-1} with the number of double weights bounded by μ. Then simply adding w_k to each representation weighs everything up to $2w_k - 1$ without using another double weight. Therefore $w_{k+1} \ge 2w_k$. That inequality can be strict only if we can weigh something larger. That's only possible if some earlier representation has a double weight we could make single so as to allow doubling the new next weight. That would require $\mu_k > 0$.

If $\mu_{k+1} = 1 + \mu_k$ then we may use another double weight. By adding $2w_k$ to the representation using weights up to w_{k-1} we can represent everything up to $3w_k - 1$, so $3w_k \le w_{k+1}$.

Suppose we were able to represent $3w_k$ as well:

$$3w_k = d_k w_k + d_{k-1} w_{k-1} + \cdots + d_1 w_1 + d_0 w_0.$$

Since W is optimal, $w_k > s_k$, hence $3w_k > w_k + 2s_k$. Therefore $d_k = 2$ and then $w_k = d_{k-1} w_{k-1} + \cdots + d_0 w_0$, using at most $\mu_{k+1} - 1 = \mu_k$ double weights. That contradicts the optimality of w_k.

Therefore $3w_k$ is the smallest positive integer that cannot be represented by W using at most μ_k double weights, hence $w_{k+1} = 3w_k$. □

The optimal weight sequence for μ_{min} (no negative/double weights) is the powers of 2. The optimal sequence for μ_{max} (as many negative/double weights as you wish) is the powers of 3. The problem we set fourth graders was to find the μ-optimal

weight sequence for $\mu = (0, 1, 1, 1, \ldots)$: one negative/double weight allowed. That is the mystery sequence $X = (1, 3, 8, 18, \ldots)$. We wrote a Python program to explore further. It told us the next two weights are 41 and 84. We found no pattern in X but did find X in OEIS at A066425 [**8**]:

$$W = (1, 3, 8, 18, 41, 84, 181, 364, 751, 1512, 3037, 6107, 12216, 24547, \ldots)$$

The next theorem shows that the OEIS definition for W is equivalent to ours for X.

Theorem 6. *For every integer $k \geq 0$, let w_k be the smallest positive integer M such that M minus any sum of distinct earlier terms is not already in the sequence. Then $w_k = x_k$ for all k.*

Proof. Use strong induction on k. The assertion is true for $w_0 = x_0 = 1$. Suppose it true up to $k - 1$.

Call the weights up to $k - 1$ "small".

Consider $m = w_k - 1$. The minimality of w_k implies there is some set S of small weights and a small weight $w_j = x_j$ such that

$$m - \Sigma S = w_j$$

where ΣS is the sum of the weights in S. Then

$$m = w_j + \Sigma S$$

represents m as a sum of small weights with at most one double weight. Therefore $m = w_k - 1 < x_k$ so $w_k \leq x_k$.

Consider $m = x_k - 1$. Then the optimality of x_k in X implies some set S of small weights represents m using at most one of them twice. If there is such a one, let it be w_j, else pick any element $w_j \in S$. Remove (one copy of) w_j from S (which may then be empty, but that's OK). Then

$$m = w_j + \Sigma S.$$

Then

$$m - \Sigma S = w_j$$

so the minimality of w_k implies $m = x_k - 1 < w_k$ so $x_k \leq w_k$. □

The OEIS entry asks for an efficient algorithm to calculate X. We haven't found one. Our Python program is essentially brute force, speeded up some by the restrictions proved in Theorem 5.

Efficient algorithms are an unsolved problem. We don't have one to find any of the optimal sequences other than the powers of 2 and the powers of 3. Moreover, given an optimal sequence we don't have a good algorithm that tells us how to weigh things. With Lemma 1 we can switch back and forth between negative and double weights, but we don't know how to get started with either.

For the powers of 3 the greedy algorithm finds the ordinary base 3 representation: subtract as many as possible of the largest weight you can (1 or 2 of them), then represent the difference with smaller weights.

Since $w_{k+1} > 2w_k$ for every optimal weight sequence other than the powers of 2, the greedy algorithm will always call for 1 or 2 of the largest possible weight. But it may not succeed when 2 will fit. It fails to discover how X represents 38 since $38 = 2 \times 18 + 2 \times 1$ uses two double weights. The correct representation is

$$38 = 18 + 2 \times 8 + 3 + 1.$$

We hoped that the greedy algorithm would work at least for $w_k - 1$, the last n you can represent without needing a new weight. But no: it fails for X at

$$w_{11} - 1 = 6106 = 3037 + 2 \times 1512 + 41 + 3 + 1$$

even though $2 \times 3037 < 6107$.

Conjecture 1. *For every optimal weight sequence other than the powers of 2 and powers of 3, using the greedy algorithm to find representations will always fail somewhere. The first failure will be for an n for which the largest possible weight fits twice.*

We can modify the greedy algorithm by adding a little backtracking. If at some point subtracting a double weight leads to failure, try a single instead. This is essentially the well known (in computer science) recursive solution to the knapsack problem [**4**]. Starting with the largest weight is just a way to organize the brute force search through all the possibilities (which must succeed) in hopes that reducing the problem recursively by subtracting large weights will find a solution as quickly as possible.

The tree of optimal weight sequences

When we computed optimal weight sequences for several bounding sequences μ we discovered that although we could not easily understand any particular sequence, studied together the set of weight sequences displays surprising structure.

Given a bounding sequence μ the sequence of differences $\mu_{k+1} - \mu_k \in \{0, 1\}$ determines μ, and any infinite sequence of 0's and 1's determines a bounding sequence. We will use the sequence of differences to identify bounding sequences and hence optimal weight sequences.

Definition 7. *For the finite bit string Δ of length k let $w(\Delta)$ be the weight w_k for any of the optimal weight sequences whose bounding sequence differences begin with Δ. Theorem 5 guarantees that $w(\Delta)$ is independent of the choice of bounding sequence. Set $W($empty string$) = 1$.*

Then for example $w(\mathtt{0}) = 2$, $w(\mathtt{1}) = 3$ and $w(\mathtt{1000}) = 41$.

It's natural to display data determined by finite bit sequences at the nodes of a binary tree: `0` moves to the left child, `1` to the right.

Figure 2 shows part of the tree that in its infinite entirety would display all the optimal weight sequences. Each path from the root corresponds to a sequence of `0`'s and `1`'s, hence to an infinite bit string Δ, a μ and the μ-optimal weight sequence recorded in the nodes along the path. The mystery weight sequence X is the leftmost branch in the right half of the tree, corresponding to $\Delta = \mathtt{1000}\ldots$.

Visible patterns in the tree lead at once to the following.

Theorem 7. *Let Δ be a finite bit string. Write $\mathtt{b}\Delta$ and $\Delta\mathtt{b}$ for pre- and post-concatenation with bit* `b`*. Then*

1. *The left half of the tree below $w(\mathtt{0}) = 2$ is double the whole tree: $w(\mathtt{0}\Delta) = 2w(\Delta)$.*
2. *Following a right branch triples the weight: $w(\Delta\mathtt{1}) = 3w(\Delta)$.*
3. *Following a left branch at least doubles but less than triples the weight: $2w(\Delta) \leq w(\Delta\mathtt{0}) < 3w(\Delta)$.*

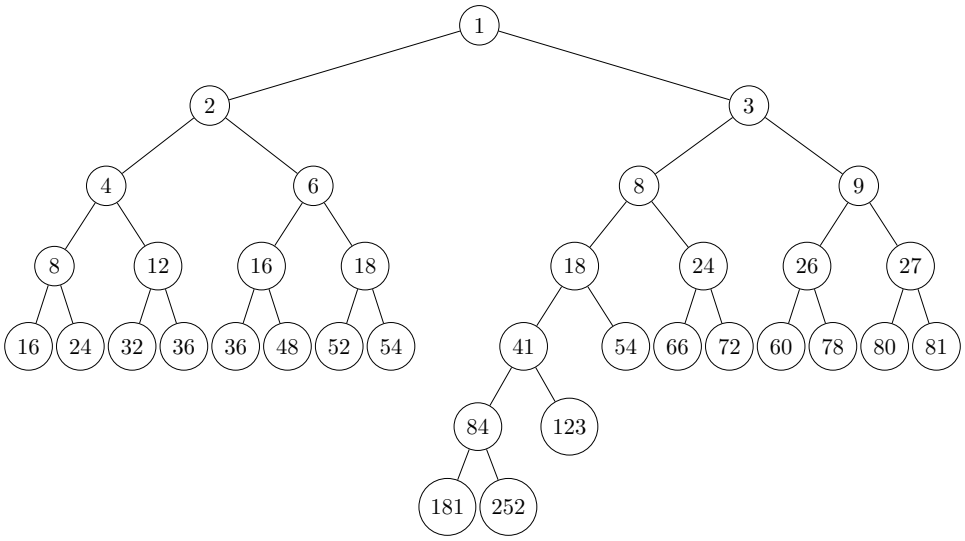

Figure 2 The tree of optimal weight sequences

Proof. Suppose Δ is of length k, so determines the weights up to $w_k = w(\Delta)$ in one of the optimal weight sequences W that start on the path determined by Δ.

To prove the first assertion, imagine constructing the tree starting with $w_0 = 2$ rather than 1. It's clear that you will then be able to weigh all even integers optimally up to $2w(\Delta) - 2$ while respecting the number of double or negative weights specified by Δ. If you then view $\mathtt{0}\Delta$ as a path in the full tree you're allowed one more weight, but no more double weights. Use that extra weight to add the 1 at the root to the even numbers you already know how to represent in order to represent all the integers up to any $2w(\Delta) - 1$.

The second and third assertions follow immediately from Theorem 5 . □

It's natural to analyze data in a tree by following paths from the root. It's less natural to look for patterns between paths, but there are many in this tree. The first we noticed concerns the differences between the weights of siblings. (We will often shorten expressions like "difference between weights of siblings" by the slightly less precise "difference between siblings".)

Theorem 8 (Siblings)**.** *When you follow a right branch in the tree the difference between siblings is constant. Formally, for each finite bit string* Δ

$$w(\Delta\mathtt{11}) - w(\Delta\mathtt{10}) = w(\Delta\mathtt{1}) - w(\Delta\mathtt{0}). \tag{4}$$

Proof. Theorem 7 (2) says $w(\Delta\mathtt{11}) = 3w(\Delta\mathtt{1})$ so Equation (4) is equivalent to

$$w(\Delta\mathtt{10}) = 2w(\Delta\mathtt{1}) + w(\Delta\mathtt{0}).$$

Let $w_0, w_1, \ldots, w_k, w_{k+1}$ be the weight list corresponding to $\Delta\mathtt{1}$ and let $m = \mu(\Delta)$ be the sum of the entries of Δ; then $w_k = w(\Delta)$ is the smallest integer that can not be weighed with $w_0, \ldots, w_{k-1}$ with at most m double weights.

Let $N = 2w(\Delta\mathtt{1}) + w(\Delta\mathtt{0})$ and suppose N can be represented as

$$N = d_0w_0 + \cdots + d_kw_k + d_{k+1}w_{k+1} \tag{5}$$

with at most $\mu(\Delta 10) = m + 1$ coefficients equal to 2. Since

$$w(\Delta 0) > w_0 + \cdots + w_k$$
$$\text{and } w(\Delta 1) > w_0 + \cdots + w_k$$

it follows that

$$N = w(\Delta 0) + w(\Delta 1) + w(\Delta 1) > 2(w_0 + \cdots + w_k) + w_{k+1}.$$

That implies $d_{k+1} = 2$ and therefore

$$w(\Delta 0) = N - 2w(\Delta 1) = d_0 w_0 + \cdots + d_k w_k,$$

with at most $m = \mu(\Delta 0)$ coefficients equal to 2, contradicting the definition of $w(\Delta 0)$. Hence $N = 2w(\Delta 1) + w(\Delta 0)$ does not have a representation as in Equation (5).

Next we show that every positive integer $n < N$ has such a representation. We consider two cases:

Case 1: $n \leq 2w(\Delta 1)$. Then $n < w(\Delta 10)$, hence n can be represented as

$$n = d_0 w_0 + \cdots + d_k w_k + d_{k+1} w_{k+1}$$

with at most $m + 1 = \mu(\Delta 10)$ coefficients equal to 2.

Case 2: $2w(\Delta 1) < n < 2w(\Delta 1) + w(\Delta 0)$. Then $n - 2w(\Delta 1) < w(\Delta 0)$, hence it can be represented as

$$n - 2w_{k+1} = d_0 w_0 + \cdots + d_k w_k,$$

with at most $\mu(\Delta 0) = \mu(\Delta) = m$ coefficients equal to 2. Therefore

$$n = d_0 w_0 + \cdots + d_k w_k + 2w_{k+1},$$

with at most $1 + \mu(\Delta 0) = \mu(\Delta 10)$ coefficients equal to 2.

Therefore $N = 2w(\Delta 1) + w(\Delta 0)$ is the smallest positive integer that does not have a representation (5), hence $N = w(\Delta 10)$. □

First cousins are nodes with a different parent but a common grandparent. For each node, the right child of the left child and the left child of the right child are neighboring first cousins.

Corollary 5 (Cousins)**.** *The difference between neighboring first cousins is twice the difference between their sibling parents. Formally,*

$$w(\Delta 10) - w(\Delta 01) = 2(w(\Delta 1) - w(\Delta 0).$$

Proof. Let

$$a = w(\Delta 1) - w(\Delta 0).$$

Then

$$\begin{aligned} w(\Delta 10) - w(\Delta 01) &= w(\Delta 10) - 3w(\Delta 0) \\ &= w(\Delta 11) - a - 3w(\Delta 0) \\ &= 3w(\Delta 1) - a - 3w(\Delta 0) \\ &= 3a - a = 2a. \end{aligned}$$

□

With the results in this section we can begin to fill in the tree using only elementary arithmetic and the weights in the mystery sequence.

- At the root, we know the value is 1.
- Theorem 7 fills the second row with $w(\mathtt{0}) = 2$ and $w(\mathtt{1}) = 3$. That tells us the left half of the third row too: $w(\mathtt{00}) = 4$ and $w(\mathtt{01}) = 6$. The right half of the third row is $w(11) = 9$ (Theorem 7) and $w(10) = 8$ (Siblings).
- The right half of the fourth row starts with $w(\mathtt{100}) = 18$ from the mystery sequence. Theorem 7 tells us $w(\mathtt{101}) = 24$ and $w(\mathtt{111}) = 27$. Then the Sibling Theorem or the Cousin Corollary finishes with $w(\mathtt{110}) = 26$.
- Similar reasoning fills the fourth row except for $w(1000) = 41$, from the sequence X, and $w(\mathtt{1100}) = 60$, which seems to require an actual search for the optimal weight.

We hoped that the mystery sequence X might tell us $w(\mathtt{1100}) = 60$, in fact everything, if only we we could generalize Corollary 5 to more distant cousins.

The rest of this paper shows how that hope played out.

Kissing cousins

In a binary tree a pair of nodes in a row are kth cousins for their nearest common ancestor node N if they live $k + 1$ rows below N. 0th cousins are siblings. There are 2^{2k} pairs of of kth cousins with common ancestor N since you form such a pair by choosing one cousin from the left half of the tree below N and the other from the right half.

Among those pairs we single out 2^{k-1} pairs of *kissing cousins*. If you number the nodes in row $k + 1$ below N as $1, 2, \ldots, 2^k$ in the left and right halves then the kissing kth cousins are the pairs

$$(2, 1), (4, 2), (6, 3), \ldots, (2^k, 2^{k-1}) .$$

Figure 3 marks the pairs of kissing cousins below the root with solid lines and those below node 1 (weight 3) with dashed lines.

This figure suggests the conjecture that kissing kth cousin differences propagate when you travel down to the right just as first cousin differences do. For example

$$w(\mathtt{100}) - w(\mathtt{001}) = 18 - 12 = 60 - 54 = w(\mathtt{1100}) - w(\mathtt{1001}).$$

We have much more data supporting that conjecture If it is true we can use it to extend what's possible with just simple arithmetic and the weights in X. In particular, we can fill in the missing fourth row value $w(\mathtt{1100}) = 60$.

Unfortunately we will still be stuck trying to find $w(\mathtt{10100})$ in the fifth row. The two terminal moves left stymie us and there is no pair of kissing cousins to help out.

Since the conjecture doesn't tell us everything, we content ourselves with stating it formally and drawing some consequences, and leave the proof to the reader.

Definition 8. *For each bit string Δ we define the kissing cousin pairs below node Δ recursively.*

The pair $(\Delta\mathtt{01}, \Delta\mathtt{10})$ *are* kissing first cousins.

For each pair $(\Delta\mathtt{0X1}, \Delta\mathtt{1Y})$ *of kissing kth cousins there are two pairs of kissing* $(k + 1)$*st cousins:*

$$\textit{left:}\ (\Delta\mathtt{0X01}, \Delta\mathtt{1Y0})$$

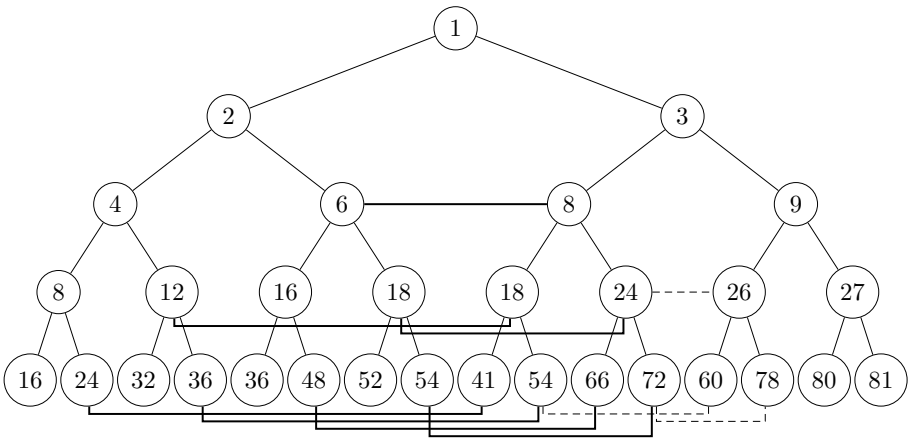

Figure 3 Kissing cousins

and

$$\textit{right: } (\Delta\texttt{0X11}, \Delta\texttt{1Y1}).$$

Note that $\texttt{Y}$ *is one bit longer than* $\texttt{X}$*. For the kissing first cousins,* $\texttt{X}$ *is empty and* $\texttt{Y} = \texttt{0}$*. For* k*th cousins,* $|Y| = k$*.*

Conjecture 2 (Kissing cousins)**.** *The difference between kissing cousins is invariant when you move their common great-parent down a right branch. Formally, for each bit string* Δ*, for each pair of kissing cousins*

$$w(\Delta\texttt{11Y0}) - w(\Delta\texttt{10X01}) = w(\Delta\texttt{1Y0}) - w(\Delta\texttt{0X01}). \tag{6}$$

Definition 8 shows that for each Δ the pairs of kissing cousins naturally arrange themselves in a binary tree with root $(\Delta\texttt{01}, \Delta\texttt{10})$. In that tree we define the value at each node to be the difference between the weights of the pair of cousins at that node. When we wrote out the first few levels of that tree for several fixed ancestors Δ we discovered that it too seemed to satisfy Conjecture 2. That's not an accident.

Theorem 9. *Let* T *be a binary tree in which following a right branch triples the value at a node and the kissing cousin conjecture holds. Then the same is true for the trees* $K(\Delta)$ *of differences between kissing cousin pairs below* Δ *in* T*.*

Proof. Write $t(N)$ for the value at node N in T. Following a right branch in $K(\Delta)$ triples because for each X and Y defining a kissing cousin pair

$$t(\Delta\texttt{1Y1}) - t(\Delta\texttt{0X11}) = 3t(\Delta\texttt{1Y}) - 3t(\Delta\texttt{0X1}) = 3(t(\Delta\texttt{1Y}) - t(\Delta\texttt{0X1})).$$

If $(\Delta\texttt{A}, \Delta\texttt{B})$ and $(\Delta\texttt{C}, \Delta\texttt{D})$ are kissing cousins in $K(T)$ then

$$\begin{aligned} t(\Delta\texttt{1B}) - t(\Delta\texttt{1A}) - (t(\Delta\texttt{1D}) - t(\Delta\texttt{1C})) \\ = t(\Delta\texttt{1C}) - t(\Delta\texttt{1A}) - (t(\Delta\texttt{1D}) - t(\Delta\texttt{1B})) \\ = t(\Delta\texttt{C}) - t(\Delta\texttt{A}) - (t(\Delta\texttt{D}) - t(\Delta\texttt{B})). \end{aligned}$$

The last equality is true because the terms on the right are kissing cousin differences in T, equal because they come from following the right branch appending $\texttt{1}$ to Δ. □

The tree of kissing cousin differences depends on the chosen common ancestor Δ, which you can think of as the head of a family, so taken together those trees are a forest of (overlapping) family trees. Examining the tree headed by $w(10) = 8$ shows that we can't conjecture some further resemblances with the weight tree. In particular, there are places when the left kissing cousin difference is greater than three times the parent cousin difference. There is much more to be explored in these forests.

What next?

We started with an unanswered fourth grade balance weighing question: what happens when you allow just one weight on the pan with the unknown?

Explorations at that elementary level revealed the power of positional notation as a tool and a metaphor.

We showed how answers to generalizations of that question live in an interesting, richly structured tree of weights.

There is more nice mathematics waiting to be discovered (or invented, depending on your philosophy of mathematics):

- Find good algorithms for calculating optimal weight sequences, or prove that these calculations are intrinsically difficult, essentially calling for brute force search. In particular, understand the mystery sequence $X = (1, 3, 8, 18, 41, \ldots)$ that answers the fourth grade open question.
- Given an optimal weight sequence, find a good algorithm for how to weigh with it, or prove that these calculations are intrinsically difficult.
- Prove the kissing cousin conjecture.
- Think about which weight sequences are optimal for some bounding sequence.
- Generalize to allow multiple (but bounded in number) weights of each size.

Ethan Bolker is Professor of Mathematics, Emeritus, at the University of Massachusetts, Boston. His favorite research topics come from elementary mathematics and ideas he first encountered in Hugo Steinhaus' *Mathematical Snapshots*.

Sam Feuer is a freshman at Wesleyan University. When he was six, he discovered that he loved mathematics and music. They are still his greatest passions today.

Catalin Zara is Professor of Mathematics at the University of Massachusetts, Boston. His ongoing passion for solving math problems traces back to early years of school in Romania. He credits this passion to his parents for starting it and his mentors for nurturing it.

REFERENCES

1. Bachet de Méziriac. (1624) Problèmes plaisants et délectable qui se font par les nombres. In: Dörrie, H. (1958), *Triumph der Mathematik*. (Antin, D. (trans). (1965). *100 Great Problems of Elementary Mathematics*. Dover Publications.
2. Hayes, Brian. (2001) *Third base* American Scientist, 89 (6): 490–494, doi:10.1511/2001.40.3268. Reprinted in Hayes, B. 2008. *Group Theory in the Bedroom, and Other Mathematical Diversions*. Farrar, Straus and Giroux. pp. 179–200. See: *Balanced ternary*. Wikipedia, The Free Encyclopedia. `en.wikipedia.org/wiki/Balanced_ternary`. (accessed December 14, 2019).
3. Joe Z. What's the fewest weights you need to balance any weight from 1 to 40 pounds?. (2014). `puzzling.stackexchange.com/questions/186/whats-the-fewest-weights-you-need-to-balance-any-weight-from-1-to-40-pounds`. (accessed November 10, 2019).

4. Kellerer, Hans; Pferschy, Ulrich; Pisinger, David. (2004). *Knapsack Problems*. Springer. See: *Knapsack problem*. Wikipedia, The Free Encyclopedia, `en.wikipedia.org/wiki/Knapsack_problem` (accessed December 14, 2019).
5. Knuth, Donald E. (1981) *The Art of Computer Programming. Vol. 2: Seminumerical Algorithms*. Second edition. Reading,Mass: Addison-Wesley, pp. 190–193.
6. Magliozzi, T., Magliozzi, R. Stone Temple Farmers. (2011). `www.cartalk.com/content/stone-temple-farmers-0`. (accessed November 10, 2019).
7. Magliozzi, T., Magliozzi, R. The Farmer's Stone. (2017). `www.cartalk.com/puzzler/farmers-stone`. (accessed November 10, 2019).
8. The On-Line Encyclopedia of Integer Sequences, published electronically `oeis.org,2010`. Smallest increasing sequence (with $a(1) = 1$) such that $a(n)$ minus any sum of distinct earlier terms is not already in the sequence. `oeis.org/A066425`. (accessed November 10, 2019).
9. Quora. (2016). There is a 40 kg stone that broke into 4 pieces such that you can measure any weight between 1kg to 40kg using a balance. What are the weights of the broken pieces? `www.quora.com/There-is-a-40-kg-stone-that-broke-into-4-pieces-such-that-you-can-measure-any-weight-between-1kg-to-40-kg-using-a-balance-What-are-the-weights-of-the-broken-pieces`. (accessed November 10, 2019).
10. Steinhaus, H. (1938). *Mathematical Snapshots*. New York, Leipzig: G.F. Stechert & Co.
11. Swapnil, Least number of weights required to weigh integer weights. (2014). `math.stackexchange.com/q/1039528`. (accessed November 10, 2019).