Pittsburgh Interdisciplinary Mathematics Review **5** (2026), 120–130

Beach Math

Problems to do on an unknown beach

Jason Wein

What is Beach Math?

You wake up on a beach. You don't know how you got there, perhaps you were marooned or perhaps something more nefarious. All you know is that, as with most things in life, the only way to survive is solving math problems.

Contents

1. Problems

1.1. Calm Waves

A menagerie of folks (mathematicians, no doubt) pile out of a Peel P50 near some calm waves, introducing themselves as $P_1, P_2, \ldots$ and eventually, P_{n-1}. You could run away, but you're somehow drawn in. Seeing your interest, everyone holds a seashell of colors $C_1, C_2, \ldots, C_{n-1}$ and they hand you a seashell C_n. You get distracted by a creative plot device. Now, they are each holding a seashell of colors $C_2, C_3, \ldots, C_n$ respectively and you hold C_1. Everybody wants their corresponding seashell back (i.e. for P_1 to have C_1, P_2 to have C_2, and so on). No two people may swap more than once. They have a couple of questions that you must answer (for plot reasons).

1. What is the minimum number of swaps possible?
2. Prove that $\binom{n}{2}$ is an upper bound on the number of swaps when $n \equiv 1, 2 \pmod 4$ and $\binom{n}{2} - 1$ is an upper bound on the number of swaps when $n \equiv 0, 3 \pmod 4$.
3. (From Futurama, Keeler's Theorem) Now, nobody is allowed to make swaps with people they have made swaps before (due to plot reasons, of course). Since you got distracted by plot convenience, you do not know what swaps were made to get to the current configuration where everyone now has random shells. Prove that with the 2 extra mathematicians from the trunk of the car who have their trusty seashells C_{n+1} and C_{n+2}, everybody can get their seashells back.

1.2. **Rough Waters**

After some rather daring event, you're now somewhere in the middle of the ocean, or as one may put it, in rough waters. Now you're *really* stuck, good job. And still with those strange mathematicians. They have a collection of pretty seashells of colors $C_1, C_2, \ldots C_n$. Each person P_i (including yourself, having been promoted to P_n) has a certain preference list $V_i = \{v_{1i}, v_{2i}, \ldots, v_{ni}\}$ for the seashells with $v_{ji} \geq v_{j+1,i}$ for all $j \in \{1, 2, \ldots, n-1\}$ and $\sum_j v_{ji} = 1$. You know the relative ordering of everyone's list, but not the exact values people have for each color seashell.

1. You must design an algorithm that gives people seashells (which they have 1 each of) in such a way that no 2 people would both want to swap seashells with each other.

2. Instead, there exists seashells of colors $C_1, C_2, \ldots, C_k$ (so P_i will have the preference list $V_i = \{v_{1i}, v_{2i}, \ldots, v_{ki}\}$) such that $0 \leq v_{ji} \leq \frac{1}{n}$ for all j. They want an algorithm that makes everyone "kinda happy," i.e., so everybody gets seashells that give them greater than or equal to $\frac{1}{n}$ of their preference list and gets no duplicate seashells, or can say when it is not guaranteed that everyone can be "kinda happy."

1.3. **Tsunami**

Just then, a rumble undistract you from these ridiculous questions. You look off into the distance. There's now a tsunami approaching (themes and such).

You will be playing against the $n-1$ other people in a game as follows.[1] There are k piles with $s_1, s_2, \ldots, s_k$ seashells each. All players will take turns removing any positive number of seashells from a nonempty pile. The game ends when no legal moves are left, i.e., when all piles are empty. The player whose move it is when the game ends loses. Each player plays to obtain the highest possible position in the final ranking. The loser is ranked last while everyone else is ranked in order of how recently they moved. You do not want to lose.

1. Derive a way to find out what turns you can do to guarantee you do not lose.

Instead, the other $n-1$ people are ganging up on you (you can think of them as having merged into one enemy). They have also made all of the seashells colorless (sad). You will take turns either placing or removing a seashell from one of k piles. Each pile can have at most m shells. Initially, none of the piles have any shells. After a certain configuration of seashells has been made, it cannot be repeated. The game ends when somebody does not have a legal move to make. This person is designated the loser and bad things happen to them.

2. What turn should you play to guarantee you do not lose (dependent on m)?[2]

[1] Adapted from Shuo-Yen Robert Li.
[2] Adapted from Putnam 2025, $A3$.

2. Solutions

2.1. Calm Waves

The problem is actually analyzing an n-cycle, in particular $(1\ 2 \dots n)$. Let s be the number of total swaps.

Problem 1. We can view each swap as a transposition. As such, we need at least $n-1$ transpositions to invert our n-cycle. An explicit construction to get this lower bound would be $(n\ 1)(n\ 2)\dots(n\ n-1)$.

Problem 2. We write $s \leq \binom{n}{2}$ when $n \equiv 1, 2 \pmod 4$ and $s \leq \binom{n}{2} - 1$ when $n \equiv 0, 3 \pmod 4$.

We have $s \leq \binom{n}{2}$ due to the condition that no two people may swap more than once. Though, this may not be necessarily possible for all values of n. Viewing the initial state as $\sigma = (1\ 2\ \dots\ n)$, we have that $\operatorname{sgn}(\sigma) = (-1)^{n-1}$. We want to find σ^{-1}, a composition of s transpositions where s is maximal and $\sigma\sigma^{-1} = e$, where e is the identity. We know $\operatorname{sgn}(e) = 1$, $\operatorname{sgn}(\sigma) = (-1)^{n-1}$, and $\operatorname{sgn}(\sigma^{-1}) = (-1)^s$. As such, we need $s + (n-1)$ to be even, or $s \equiv n - 1 \pmod 2$.

For $s = \binom{n}{2}$, we then need

$$\binom{n}{2} = \frac{n(n-1)}{2} \equiv n - 1 \pmod 2 \implies \frac{(n-1)(n-2)}{2} \equiv 0 \pmod 2.$$

This is equivalent to

$$(n-1)(n-2) \equiv 0 \pmod 4,$$

implying we need $n \equiv 1, 2 \pmod 4$ for $s = \binom{n}{2}$ to be possible. Otherwise, we can take $s = \binom{n}{2} - 1$ as, going through the same process as above, we get

$$(n-1)(n-2) \equiv 2 \pmod 4$$

implies $n \equiv 0, 3 \pmod 4$.

Problem 3. Consider an arbitrary cycle of length $k \geq 2$. Without loss of generality (by relabeling the people and shells), with the 2 additional mathematicians we can represent this as a cycle as follows:

$$\begin{pmatrix} 1 & 2 & \dots & k-1 & k & k+1 & \dots & n & n+1 & n+2 \\ 2 & 3 & \dots & k & 1 & k+1 & \dots & n & n+1 & n+2 \end{pmatrix}.$$

Then, we can make the swaps $(n+1\ 1)$, $(n+1\ 2), \dots, (n+1\ k-1)$. Essentially,

we are using P_{n+1} to give $P_2, \ldots, P_{k-1}$ their correct shells. Everyone except P_1, P_k, and P_{n+1} have their corresponding shells. P_k is still holding C_1. They can give C_1 to P_{n+2}. P_1 then can swap their held shell, C_{n+1}, with P_{n+2} to get their shell C_1. Now, P_k can swap with P_{n+1} to get C_k. Repeat this process for all nontrivial cycles. Finally, swap P_{n+1} and P_{n+2} (if necessary) and we are done.

2.2. **Rough Waters**

Problem 1. We will design a serial dictatorship algorithm. Order each list from the person's highest to lowest preferences. Next, arbitrarily order the lists $V_1, \ldots, V_n$ relative to each other. Starting from the first list in the ordering down to the last, give the corresponding person their highest preference shell still available. Each person will get a shell as there are n people and n shells.

We now want to show that nobody would want to swap shells with each other. Assume for the sake of contradiction that there existed two people who would want to swap shells, say P_i and P_j with shells C_a and C_b respectively such that V_i came before V_j in the ordering. If P_i had preferred C_b, then the algorithm would have assigned them C_b as it must have been still available, which would be a contradiction to the algorithm.

Problem 2. We first describe the intuition. The algorithm will output that it cannot guarantee everyone can be "kinda happy" when k is not a multiple of n. As such, we assume $k = ni$ for some $i \in \mathbb{N}$. For a person P_j, we want to assign them i shells with the condition that their t^{th} shell must be in their top $(t-1)n + 1$ shells.

Claim 1. *k is a multiple of n.*

Proof. To show that k must be a multiple of n, consider for the sake of contradiction that $k = n \cdot i + r$ for some integer i and $0 < r < n$. If everyone equally preferred all shells with a value $\frac{1}{k}$, a person would need at least $i + 1$ shells to have a partition of value greater than or equal to $\frac{1}{n}$. But to give everyone at least $i + 1$ shells we would need at least $n(i+1)$ shells. However, $n(i+1) = n \cdot i + n > n \cdot i + r = k$ which means not everyone can be "kinda happy." □

Fix a person P_ℓ. Let $f_\ell(p)$ be the number of shells in the partition assigned to P_ℓ that are in their top p most preferred shells.

Claim 2. *Let $C_\ell \subseteq \{1, \ldots, k\}$ be arbitrary with $i = |C_\ell|$ the i shells assigned to P_ℓ. Then*

$$\min_v \sum_{j \in C_\ell} v_{j\ell} \geq \frac{1}{n} \iff f_\ell(p) \geq \frac{p}{n}, \ \forall p = 1, \ldots, k.$$

This essentially connects our discrete count of the shells to the continuous values of P_ℓ's preferences and means our second condition is necessary.

Proof. For the backwards implication, write $u_p = v_{p\ell} - v_{p+1,\ell}$. Then, we have that

$v_{j\ell} = \sum_{p=j}^{k} u_p$ (letting $v_{k+1,\ell} = 0$). We note that

$$1 = \sum_{j=1}^{k} v_{j\ell} = \sum_{j=1}^{k} \sum_{p=j}^{k} u_p = \sum_{p=1}^{k} \sum_{j=1}^{p} u_p = \sum_{p=1}^{k} p u_p.$$

As such, we can write

$$\sum_{j \in C_\ell} v_{j\ell} = \sum_{j \in C_\ell} \sum_{p=j}^{k} u_p = \sum_{p=1}^{k} f_\ell(p) u_p \geq \frac{1}{n} \sum_{p=1}^{k} p u_p = \frac{1}{n}.$$

For the forwards implication, we use contraposition and counterexamples to show that the condition is necessary. Suppose $f_\ell(p_0) < \frac{p_0}{n}$. Then, if $p_0 \geq n$, we can let $v_{j\ell} = \frac{1}{p_0}$ for $j \leq p_0$ and $v_{j\ell} = 0$ otherwise, which will give

$$\sum_{j \in C_\ell} v_{j\ell} = \frac{f_\ell(p_0)}{p_0} < \frac{1}{n}.$$

If instead $p_0 < n$, then $0 \leq f_\ell(p_0) < \frac{p_0}{n}$ and f_ℓ being an integer implies $f_\ell(p_0) = 0$. We then can use

$$v_{1\ell} = \frac{1}{n},\ v_{2\ell} = \ldots = v_{k\ell} = \frac{1 - \frac{1}{n}}{k-1} = \frac{n-1}{n(k-1)}.$$

Since $f_\ell(p_0) = 0$, then in particular $1 \notin C_\ell$, so with $i > 1$

$$\sum_{j \in C_\ell} v_j = \frac{i(n-1)}{n(k-1)} = \frac{k-i}{n(k-1)} < \frac{1}{n}.$$

Having $i = 1$ is trivial since then $k = n$ and $v_{i\ell} = \frac{1}{n}$ for all $i, \ell \in \{1, \ldots, n\}$, so it would be the same as assigning 1 arbitrary shell to everyone. □

We can finally write our algorithm:

1. If $k \nmid n$, then output: *Not everyone can be "kinda happy."*
2. For each person P_j for $j \in \{1, \ldots, n\}$, give them i shells such that their t^{th} shell is in their top $(t-1)n + 1$ preferred shells.[3]
3. If step 2 is possible, give people the shells the algorithm assigned. Otherwise, output: *Not everyone can be "kinda happy."*

The first step is necessary due to Claim 1. The second step is necessary due to Claim 2.

[3]This step is possible since the problem did not stipulate any time complexity requirements... Though, this also allows a brute force search from the start. Less trivially, a bipartite matching algorithm can be used in this step.

2.3. **Tsunami**

Problem 1. If the m-th player to move would lose, we call the game an $(m-1)$-position. Since everybody is playing for the highest possible position in the final ranking, all players will try moving into a 0-position, or if that is not available, the 1-position, or so on.

Write each pile in binary, i.e.,

$$s_i = \sum_{j \geq 0} s_{ij} 2^j, \qquad s_{ij} \in \{0, 1\}.$$

Let

$$a_j = \sum_{i=1}^{k} s_{ij},$$

be the sum of each column j. A single move reduces a pile from c to some $c' < c$. Let ℓ be the greatest bit such that c and c' differ. Then we define the n-ary number for a position g as

$$\Delta(g) = \sum_{j \geq 0} (a_j \bmod n) n^j$$

which captures the residues of the columns. Then, if $a_j \equiv 0 \pmod{n}$ for all j at any point, any move can only cause a_ℓ to decrease by 1. As such, at least n moves are needed to get back to having $a_j \equiv 0 \pmod{n}$ for all j.

Define $\delta(g)$ to be the left-most nonzero digit of $\Delta(g)$ if $\Delta(g) \neq 0$ and 0 otherwise.

We make some claims about the game.

Claim 1. *Suppose from a state g, one move brings the game to a state h. Let ℓ be the highest bit that is different between the states. If $\delta(g) = 0$, then $\delta(h) = n - 1$. If $\delta(g) \neq 0$, then $\delta(h) \geq \delta(g) - 1$.*

Proof. If $\delta(j) = 0$, all columns are 0 mod n so $a_\ell \equiv 0 \bmod n$ changes to $a_\ell - 1 \equiv n - 1 \bmod n$.

Otherwise, when $\delta(j) \neq 0$, first let L be the column number of $\delta(g)$.

- If $\ell > L$, then since $a_L \equiv 0 \bmod n$ before, $\delta(h) = n - 1 \geq \delta(g) - 1$.
- If $\ell = L$, then column L becomes $\delta(h) - 1$. If this is 0, $\delta(h)$ becomes the highest positioned nonzero digit, so is greater than $0 = \delta(g) - 1$ and otherwise $\delta(h) = \delta(g) - 1$.
- If $\ell < L$, column L is unchanged so $\delta(h) = \delta(g) \geq \delta(g) - 1$. □

Claim 2. *If g is a 0-position, then another state with $\Delta = 0$ cannot be reached in less than n moves.*

Proof. Let g_r be a position after r moves from g. With Claim 1 and induction on r, we can see $\delta(g_r) \geq n - r$ for $1 \leq r \leq n-1$. Thus $\delta(g_r) > 0$ and so $\Delta(g_r) \neq 0$. □

Claim 3. *If $\Delta(g) \neq 0$, then a state with $\Delta = 0$ can be reached from g in less than n moves.*

Intuitively, the proof is that using only up to $n-1$ heaps, we can remove shells in such a way to make $\Delta = 0$.

Proof. We will construct such a state.

Let L be the column of $\delta(g)$. We will keep a collection C of heaps to remove shells from. We have $a_L \geq \delta(g)$, which implies at least $\delta(g)$ have $s_{i,L} = 1$. Choose exactly $\delta(g)$ of these heaps, and we will remove shells from them so that their bit L will be 0. Then column L will have sum $a_L - \delta(f) \equiv 0 \pmod{n}$.

We now need to make the columns $j < L$ also equivalent to 0. Let $F_j = \sum_{i \neq C} s_{ij}$ be the number of heaps with a 1 in their j^{th} bit that are not already in C. Let $|C| = p$. We can remove shells from heaps in C in such a way that a_j is equivalent mod n to

$$\{F_j, F_j + 1, \ldots, F_j + p\}.$$

If $p + 1 \geq n$, one of these is equivalent to 0 mod n by the Pigeonhole Principle. Otherwise, let $r = F_j \bmod n$. Take r more heaps into the collection C. We can now make all the j^{th} column bits in our collection 0. Now, $|C| = p + r \leq n - 1$ as $r < n - p$ (otherwise one of the $F_j + t$ would have been 0 mod n). We can do this for all of the columns and $|C|$ will never exceed $n - 1$ due to how we added heaps to it. □

Claim 4. *$\Delta(g) = 0$ if and only if the game is a 0-position. More generally, let $S(g) \in \{0, \ldots, n-1\}$ be how many moves it takes to reach a 0-position from a state g. Then g is a q-position if and only if $q = S(g)$.*

Proof. We proceed by strong induction on q. In the base case, all heaps are empty and vacuously every move is a 0-position with $\Delta = 0$ and $S = 0$.

Assume the claim holds for every position h reachable in one move from g. If $\Delta(g) = 0$, then by Claim 2, $S(h) \geq n - 1$. By Claim 3, $S(h) \leq n - 1$, so $S(h) = n - 1$, which by the inductive hypothesis means h is an $(n-1)$-position. This in turn implies g is a 0-position.

In the case that $\Delta(g) \neq 0$, take a sequence of $q = S(g)$ moves to a state where $\Delta = 0$. Call the first position in this sequence h_1. Then, $S(h_1) = q - 1$ as if it were less, it would contradict the definition of $S(g)$. Therefore, by the inductive hypothesis, h_1 is a $(q-1)$-position. We need to show this holds for any move from g. Assume for the sake of contradiction that there exists h_2 that g has a move to with $q_2 \leq q - 2$.

This would imply $S(g) = S(h_2) + 1 \leq q - 1 < q$ which is a contradiction. Thus, by definition, g is a q-position. □

Let g_0 be the starting position. Then, the loser will be $S(g_0) + 1$. Any other position will be safe.

Problem 2. You should play first if m is odd and second otherwise. Note that $(m+1)^k - 1 = m + 1 - 1 = m \pmod 2$. We construct a graph G with vertices $\{0, 1, \ldots, m\}^k \setminus \{(0, \ldots, 0)\}$ that represent each state the game can be in. The edges will then represent adding or subtracting 1 from a pile to get to another position. We want to partition G into pairs of neighboring states (x, y) such that if your enemies go to either x or y, you can go to the other. Then, you will always have a response to your enemies and thus can win. This is known as a *perfect matching*.

We first assume m is even. Consider if your enemies went to a state in the form $0 \ldots 0 a_1 a_2 \ldots a_\ell$ with $a_1 \neq 0$. Then, if m is even you can go to the state $0 \ldots 0 b_1 a_2 \ldots a_\ell$ where $b_1 = a_1 - 1$ if a_1 is even or $b_1 = a_1 + 1$ if a_1 is odd.

Otherwise, if m is odd, we can do a pairing such that if the enemy goes to $a_1 a_2 \ldots a_k$ then you can go to $a_1 a_2 \ldots b_k$ where $b_k = a_k - 1$ if a_k is odd and $b_k = a_k + 1$ if a_k is even.

Jason Wein
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA
Email: JSW116@pitt.edu