

CDM

Deterministic Pancakes

KLAUS SUTNER

CARNEGIE MELLON UNIVERSITY

FALL 2025



1 Flipping Pancakes

2 Transients, Potentials, Ordinals

3 Math Toolchain

Q11 size kernel = $|w \cdot T_{pk}^T|$

Vector k .

$$a = a_1, a_2, \dots, a_n \text{ perm. of } [n]$$

$$a' = \underbrace{a_k, a_{k-1}, \dots, a_2, a_1}_{k=a_1}, a_{k+1}, \dots, a_m$$

Claim: For all a : $a_i^{(t)} = 1$ for some t .

An old problem in combinatorics:

Given a stack of pancakes of different sizes, sort it by picking up some of the pancakes on top with a spatula, flipping the substack over, and placing it back on top of the rest.

More technically, one applies **prefix reversals** to sort a list.

Note that sorting is always possible: first, move the largest pancake on top, then to the bottom—done by induction.

The interesting question is: how many flips does sorting take at most?

Alas, it is quite difficult to come up with lower and upper bounds on the optimal number of flips m needed to sort any stack of n pancakes. The best result known today is

$$\frac{15}{14} n \leq m \leq \frac{18}{11} n$$

Bill Gates and Christos Papadimitriou published a paper with weaker bounds in 1979.

See [Pancake Sorting](#) for more background.

Here is a deterministic variant of the flipping operation.
Given a permutation π

Reverse exactly the first $\pi(1)$ elements.

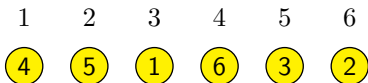
We will write $T(\pi)$ for this operation. So the permutation

$$\pi = a_1, a_2, \dots, a_{k-1}, a_k, a_{k+1}, \dots, a_n$$

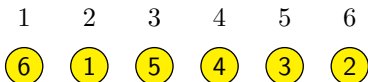
where $k = a_1 = \pi(1)$ turns into

$$T(\pi) = a_k, a_{k-1}, \dots, a_2, a_1, a_{k+1}, \dots, a_n$$

For intuition, think of a permutation as a sequence of numbered pebbles



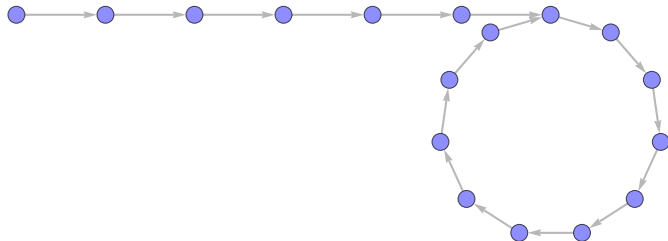
One application of T turns this into



What happens if we apply T repeatedly?

This is a beautiful example of a hopelessly open-ended question.

- Figure out if anything **interesting** is happening with T .
- Come up with an appropriate **conjecture**.
- Lastly, **prove** the conjecture.



Since there are only finitely many permutations of length n , each orbit must consist of a **transient** and a **period**.

Obvious Question:

What are the transients/periods for T , depending on n ?

6	10	4	8	3	9	2	1	7	5
9	3	8	4	10	6	2	1	7	5
7	1	2	6	10	4	8	3	9	5
8	4	10	6	2	1	7	3	9	5
3	7	1	2	6	10	4	8	9	5
1	7	3	2	6	10	4	8	9	5

6	10	1	4	8	5	2	9	7	3
5	8	4	1	10	6	2	9	7	3
10	1	4	8	5	6	2	9	7	3
3	7	9	2	6	5	8	4	1	10
9	7	3	2	6	5	8	4	1	10
1	4	8	5	6	2	3	7	9	10

Orbits of some permutations of length 10.

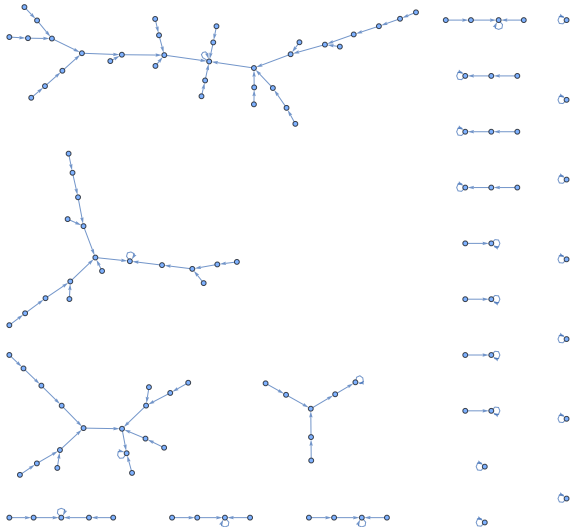
7	10	8	9	1	4	6	2	3	5
6	4	1	9	8	10	7	2	3	5
10	8	9	1	4	6	7	2	3	5
5	3	2	7	6	4	1	9	8	10
6	7	2	3	5	4	1	9	8	10
4	5	3	2	7	6	1	9	8	10
2	3	5	4	7	6	1	9	8	10
3	2	5	4	7	6	1	9	8	10
5	2	3	4	7	6	1	9	8	10
7	4	3	2	5	6	1	9	8	10
1	6	5	2	3	4	7	9	8	10

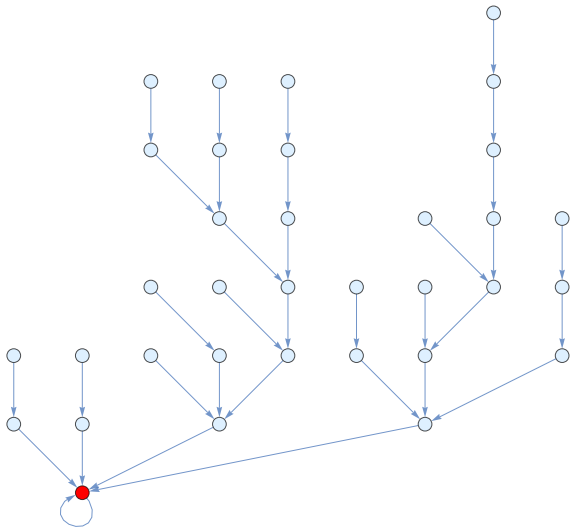
Large tables are difficult for humans to understand.

It is better to visualize orbits in terms of the **functional digraph** of T .

- vertices: permutations
- edges: $\pi \rightarrow T(\pi)$

Every node has outdegree 1, and each connected component contains exactly one cycle.





All orbits have period 1, they end in a fixed point.
Enshrine this in a definition:

A permutation π is **stable** iff $T(\pi) = \pi$.

Since reversal of a prefix of length greater than 1 always changes the permutation, there is a simple characterization:

$$\pi \text{ is stable} \quad \text{iff} \quad \pi(1) = 1$$

Conjecture:

All orbits end in a stable permutation.

Before one attempts a proof, it is probably a good idea to amass more computational evidence.

- All permutations of length $n \leq 14$.
- Millions of randomly chosen permutations for larger values of n .

$$13! = 6,227,020,800$$

$$14! = 87,178,291,200$$

$$15! = 1,307,674,368,000$$

Building the whole graph is not a good idea.

Instead we try to compute only the orbits generated by leaves.

Think before you hack.

Always.

Before one starts computing, it's a good idea to check if there is any low-hanging fruit as far as streamlining things is concerned.

With luck, this will guide the computations and help to avoid wasting time.

We should avoid computing orbits not starting at leaves.

Can we characterize the leaves?

If position $k \in [n]$ is a fixed point of $\pi = a_1, \dots, a_n$, (meaning $\pi(k) = a_k = k$), then $T(\sigma) = \pi$ where

$$\sigma = k, a_{k-1}, \dots, a_2, a_1, a_{k+1}, \dots, a_n$$

More generally, the number of preimages of π is exactly the number of fixed points in π .

A permutation has no preimages iff it has no fixed points; these are commonly referred to as **derangements**.

The number of **derangements** is sometimes written $!n$, we have

$$!n \approx n!/e$$

So roughly a third of all permutations will have no preimage under T .

Recurrences for $!n$:

$$!n = (n-1) (! (n-1) + ! (n-2))$$

$$!n = n \cdot ! (n-1) + (-1)^n$$

The values up to 10 are

$$1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961$$

Let's fix a bit of notation:

$$\pi_t = T^t(\pi)$$

$\pi_t(1)$ the **pivot** at time t

E.g., for $\pi = 7, 10, 8, 9, 1, 4, 6, 2, 3, 5$, the pivot sequence is

$$7, 6, 10, 5, 6, 4, 2, 3, 5, 7, 1$$

Note that there are repetitions.

We need to show that 1 always is a pivot.

One strongly suspects that induction on n might be useful.

Let m be the maximum pivot in the orbit of π .

Case 1: $m = n$

Then $\pi_t(n) = n$ for all sufficiently large t : we are essentially dealing with a permutation of length $n - 1$, done by induction.

Case 2: $m < n$

Then $k = \pi(n) = \pi_t(n)$ for all t . Neither k nor n can be pivots, so we may swap them without affecting the transient and period of the orbit—think about the pebble interpretation. Again done by induction.



1 Flipping Pancakes

2 **Transients, Potentials, Ordinals**

3 Math Toolchain

We know that the periods of all orbits are 1.

How about the transients?

The pictures indicate that things might be messy, so it's best to tackle a special case: what are the longest transients?

We would like to understand the function

$$S(n) = \text{maximum transient for length } n$$

Finding a closed form solutions is probably not so easy, so one should try to find lower and upper bounds for S .

My C++ code handles $n \leq 14$ in about 45 minutes.

0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101

Not exactly clear what sort of growth this is.

Here is another way to express our stability conjecture.
It may seem a bit overblown, but bear with me.

Define the map T^* on permutations by

$$T^*(\pi) = \begin{cases} T^*(T(\pi)) & \text{if } \pi \text{ is not stable,} \\ \pi & \text{otherwise.} \end{cases}$$

Then our conjecture is equivalent to T^* always terminating.

One fairly standard method to establish termination is to look for a **potential function**, a map $\Phi : \mathcal{P} \rightarrow \mathbb{N}$ such that

$$\Phi(T(\pi)) < \Phi(\pi) \quad \text{if } \pi \text{ is not stable}$$

$$\Phi(T(\pi)) = \Phi(\pi) \quad \text{otherwise.}$$

Claim: T^* always terminates iff it has a potential function.

\Rightarrow : The mother of all potential functions:

$$\Phi_0(\pi) = \min(t \geq 0 \mid \pi_t \text{ stable})$$

\Leftarrow : The natural numbers are well-ordered, so there are no infinite descending chains. Hence π_t is always stable for sufficiently large t .

Alas, our termination argument is non-constructive, it provides no bounds.

Of course, $n!$ is a trivial upper bound, but the computation of $S(n)$ up to $n = 14$ strongly suggests that it is way off.

Hope: Maybe we can find a nice, easily computable potential function that helps to bring the bound down.

More generally, one often encounters the following scenario:

- We are interested in a class \mathcal{C} of functions.
- We know that there is a computable function in \mathcal{C} .

Challenge:

Find a simple algorithm to compute some function in \mathcal{C} .

Find a simple algorithm to compute a “good” function in \mathcal{C} .

To be clear, Φ_0 is the perfect potential function.

But we don't know how to compute this potential without enumerating the whole orbit. What we need is something more “local,” a nice closed form expression would be ideal.

Making the idea of a *nice potential* precise is hard. One can try to argue in terms of complexity theory, but even that does not capture our intuition perfectly.

Or: Sometimes, knowledge is a dangerous thing.

Initially, I could not see an “easily computable” potential function[†].
So, I decided to look for a more general type of potential function first:

$$\Phi : \mathcal{P} \rightarrow On$$

where On is the class of **ordinals**, transfinite extensions of the naturals.

That may sound insane, but it's fairly straightforward for someone used to working with ordinals.

[†]No doubt a combinatorialist would find this to be quite obvious.

Think of the naturals as a total order $\langle \mathbb{N}; < \rangle$.

$$0, 1, 2, \dots, n, n+1, \dots$$

There are exactly two kinds of elements in this order:

- 0, the minimum element
- successors

Here y is the **successor** of x (and x the **predecessor** of y) if

$$x < y \wedge \neg \exists z (x < z < y)$$

Now think of the natural numbers as time, or as counting steps in a process, or stages in a construction.

In computability, we are mostly interested in finite processes, but it also makes sense to consider infinite ones. For example, an algorithm that enumerates all prime numbers obviously would never terminate.

This enumeration algorithm runs forever, but we can think of it as terminating after “infinitely many steps.” This is utterly wishy-washy, we need to make it precise to get any use out of this idea.

The infinitely many steps look like so:[†]

$$0, 1, 2, \dots, n, n+1, \dots \mid$$

We are using the notation $\dots \mid$ to indicate that we want to think of all the previous stages (here all finite stages) as being completed. All primes have been generated.

[†]The ellipsis is one of the most consistently abused symbols in all of math. Here the first one stands for finitely many steps, but the second one for infinitely many.

The next step is to assign a name to this stage at infinity:

$$\omega$$

When we append ω at the end, we now have a new order that looks like

$$0, 1, 2, \dots, n, n+1, \dots, \omega$$

Note that ω is neither 0 nor a successor, it is the first **limit ordinal**.

The ordinals $\alpha < \omega$ are exactly all the natural numbers.

Once we have reached level ω , we can take one more step and get to the successor $\omega + 1$. E.g., our prime enumeration algorithm could print the comforting message “All done.”

And then to $\omega + 2, \dots, \omega + n$, and so on. If we keep going, we finally wind up at

$$0, 1, 2, \dots, n, \dots \quad \omega, \omega+1, \omega+2, \dots, \omega+n, \dots \mid$$

The new limit ordinal is called $\omega + \omega = \omega \cdot 2$.
So we have two infinite blocks, one after the other[†].

[†]On the other hand, $2 \cdot \omega = \omega$ when one defines ordinal arithmetic formally.

You guessed it, we can also get $\omega + \omega + \omega = \omega \cdot 3$.

In fact, we can get an infinite sequence of increasing limit ordinals

$$\omega, \omega \cdot 2, \omega \cdot 3, \dots, \omega \cdot n, \dots \mid$$

We denote this level by $\omega \cdot \omega = \omega^2$: ω many blocks of size ω each.

And we can get ω^k for any natural number k^\dagger .

[†]In fact, this is just the very tip of the iceberg, but it's enough for our purposes.

Another simple way to produce the orders ω^k is to sort \mathbb{N}^k in the standard lexicographic way.

E.g., \mathbb{N}^2 produces the same order as ω^2 :

$$\begin{array}{ll} (u, x) < (u, y) & \text{for all } x < y \\ (u, x) < (v, y) & \text{for all } u < v \end{array}$$



So why bother with ordinals, why not just use lists of naturals?

Because ordinals allow us to express infinite stages purely in terms of **numbers**, albeit **transfinite** ones. These numbers come equipped with a full arithmetic that we can exploit.

There is an interesting model of computation called **infinite time Turing machines (ITTM)** that uses ordinals to make strict technical sense out of the intuitive notion of running a Turing machine for “infinitely many steps.”

There are lots of interesting results about ITTMs and the model is more appealing than many other approaches to generalized computability.

Full Disclosure: We will talk about ITTMs later.

Ordinals have a critical property that makes them important for recursion and induction:

Theorem

Ordinals are well-ordered.

In other words, there is no infinite descending chain of ordinals:

$$\alpha_0 > \alpha_1 > \dots > \alpha_n > \dots$$

Hence, if we can find a potential function $\Phi : \mathcal{P} \rightarrow On$, we still have a proof that all orbits must end in a stable permutation.

Why should it be easier to find $\Phi : \mathcal{P} \rightarrow On$ rather than $\Phi : \mathcal{P} \rightarrow \mathbb{N}$?

Because there is much more room in On , we can spread things out at infinite distances without worrying about details. Asserting that $\Phi(\pi) = \omega^2 + 123$ does not provide a concrete finite bound on the transient, it just forces it to be finite.

To be clear, this is mostly psychology, not logic.

For humans, psychology is critical.

For our purposes, we can think of a permutation π of length $n - 1$

$$\pi = a_1, a_2, \dots, a_{n-1}$$

as a permutation of length n

$$\pi' = a_1, a_2, \dots, a_{n-1}, n$$

This is part of the induction argument, once $\pi_t(n) = n$ we're done.

Wild Guess:

We should use a potential that drops radically when $\pi_t(n) = n$, less so for position $n - 1$, and so on.

With ordinals, this is fairly easy. We use base ω and consider numbers

$$\Phi(\pi) = \omega^{n-1} \cdot d_n + \omega^{n-2} \cdot d_{n-1} + \dots + \omega \cdot d_2 + d_1$$

Here the digits d_k are natural numbers and depend on whether k is a fixed point of π .

Probably the most natural choice for the digits is to measure the distance between k and $\pi(k)$:

$$d_k = |k - \pi(k)| < n$$

But, there is a simpler version using only binary digits:

$$d_k = \begin{cases} 1 & \text{if } \pi(k) \neq k \\ 0 & \text{otherwise.} \end{cases}$$

Let $\pi(1) = k > 1$, write d'_i for the digits of the potential of $T(\pi)$.

Then

$$\begin{array}{ll} \pi & d_k = 1 \\ T(\pi) & d'_k = 0 \quad d'_i = d_i \quad \text{for all } i > k \end{array}$$

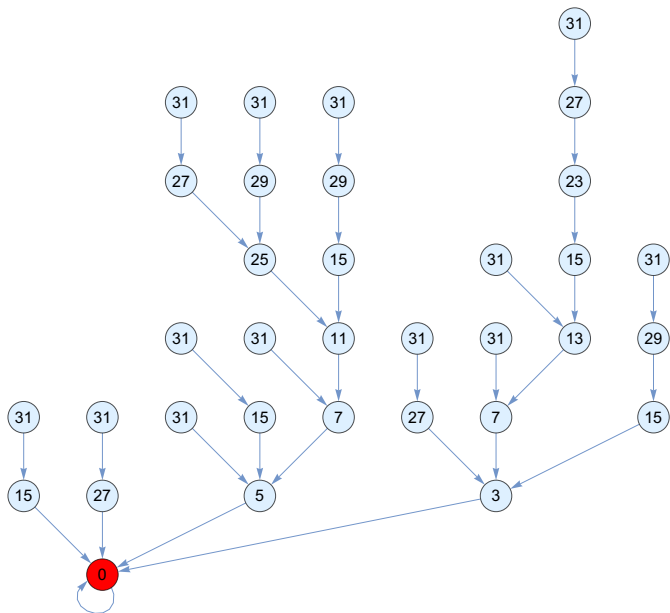
The most significant digits stay the same, digit k drops, and the remaining digits do not matter: they cannot compensate for the change in ω^{k-1} (even if all of them switch from 0 to 1).

This has nothing do with ordinals, instead of base ω we can simply use base 2, and the argument goes through unchanged.

$$\Phi(\pi) = d_n 2^{n-1} + d_{n-1} 2^{n-2} + \dots + d_2 2 + d_1$$

With this Φ , the derangements all have potential $2^n - 1$, and the identity has potential 0.

This is better than the factorial bound, but still way off.



The binary potential function arguably provides a better proof for our conjecture, and at least some non-trivial upper bound on maximum transients.

But: it does not help much with explaining the observed transients. Recall that $S(10) = 38$, but derangements have potential 1023. Also, it says nothing about lower bounds, not even linear ones.

Find better upper and lower bounds.

1 Flipping Pancakes

2 Transients, Potentials, Ordinals

3 **Math Toolchain**

So far, we have only used a bit of symbolic computation to provide evidence that the easy conjecture holds up.

Presumably this will help wasting time trying to find a proof for a wrong conjecture.

Warning:

Building confidence via computation is important, but tricky. What if the smallest counterexample were to appear for $n = 1000$? Maybe one needs that much space to build a “data-structure” that ruins stability.

Once we have the first 14 values of S , we can check Sloane's Online Encyclopedia of Integer Sequences.

It does: [A000375](#).

Turns out, $S(n)$ has been computed explicitly up to $n = 19$.

0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101, 113, 139, 159, 191, 221

Since $19! \approx 1.22 \times 10^{17}$ this is quite remarkable.

Given a description of the operation, GPT immediately recognized it as the **topswop** operation, due to John Conway in 1973; it also provided all (?) the relevant references.

It knew about the binary potential function (due to H. Wilf) and produced a corresponding proof.

When challenged to come up with a potential-free argument it produced the following slides.

Question: Is this argument correct?

Given a permutation $A = (A[1], \dots, A[n])$, repeat:

$$m := A[1], \quad A[1..m] \leftarrow \text{reverse}(A[1..m]).$$

Then the process halts with $A[1] = 1$.

Induction. Base $n = 1$ trivial. Assume true for all $< n$.

Let A_t be the deck at time t , $T_t := A_t[1]$. A time t is a **record** if T_t exceeds all previous tops. Let the distinct record tops be $r_1 < r_2 < \dots$.

Locking between records. When a record r occurs, the flip places card r at position r . Until a larger record appears, every flip has length $\leq r$, so only positions $1..r$ move; positions $r + 1..n$ (and card r) stay fixed.

Case 1: Records are bounded by $M < n$.

From the first time M appears, flips have length $\leq M$; positions $M+1..n$ are frozen. The run reduces to size M , so by IH it reaches top 1.

Case 2: n appears as a record. That flip puts card n at position n forever. The run reduces to size $n-1$, so by IH it reaches top 1.

In both cases we get a contradiction to “never reaches 1,” so the process halts with $A[1] = 1$. □

Paul Budnik implemented a calculator for ordinals.
It's quite sophisticated, but there is a learning curve.

- [Ordinal Calculator](#)
- [Ordcalc Manual](#)