



Creator



David X. Cohen
BA Physics (Harvard)
MSc Comp Sci (Berkeley)

Producer



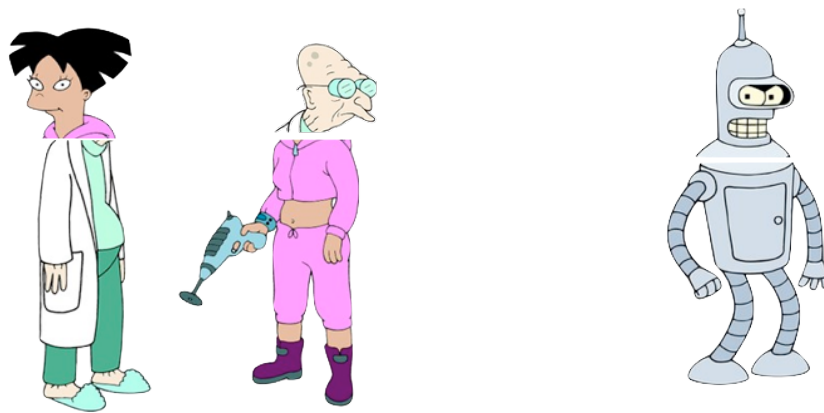
Ken Keeler
PhD Applied Math
(Harvard)

"We wanted to get in as much science as possible where it didn't clog up the gears of the story... Our hope is that, although such material will fly by most people unnoticed, it might make die-hard fans of the people who do appreciate it."

Dave X. Cohen

Episode Up Next: Prisoner of Benda...

Let's try to fix Professor & Amy using Bender :



bodies	1	2	x	
minds	2	1	x	← starting configuration
	x	1	2	(prof, bender) swap → (1 x)
	x	2 ✓	1	(amy, bender) swap → (2 x)

↑ can't swap these since we already used (1 x)

Observe : Using Bender we have just two transpositions we can use : $(1 x)$, $(2 x)$.
 We can't form the odd permutation $(1 2)$ as a product of 2 odd permutations. Therefore we can't fix the mind swap using Bender alone .

How about if we bring a 4th body into the mix?

bodies	1	2	x	y	
mind	2	1	x	y	
	x	1	2	y	$(1 x)$
	x	y	2	1	$(2 y)$
	1	y	2	x	$(1 y)$
	1	2	y	x	$(2 x)$
	1	2	x	y	$(x y)$

}

$$\overbrace{(12)(1x)(2y)(1y)(2x)(xy)}^{(12)}$$

$$= \varepsilon$$

Sequence of switches that occurred during the episode :

switch 1



switch 2



Switch 3



Switch 4



Switch 5



Switch 6



Switch 7

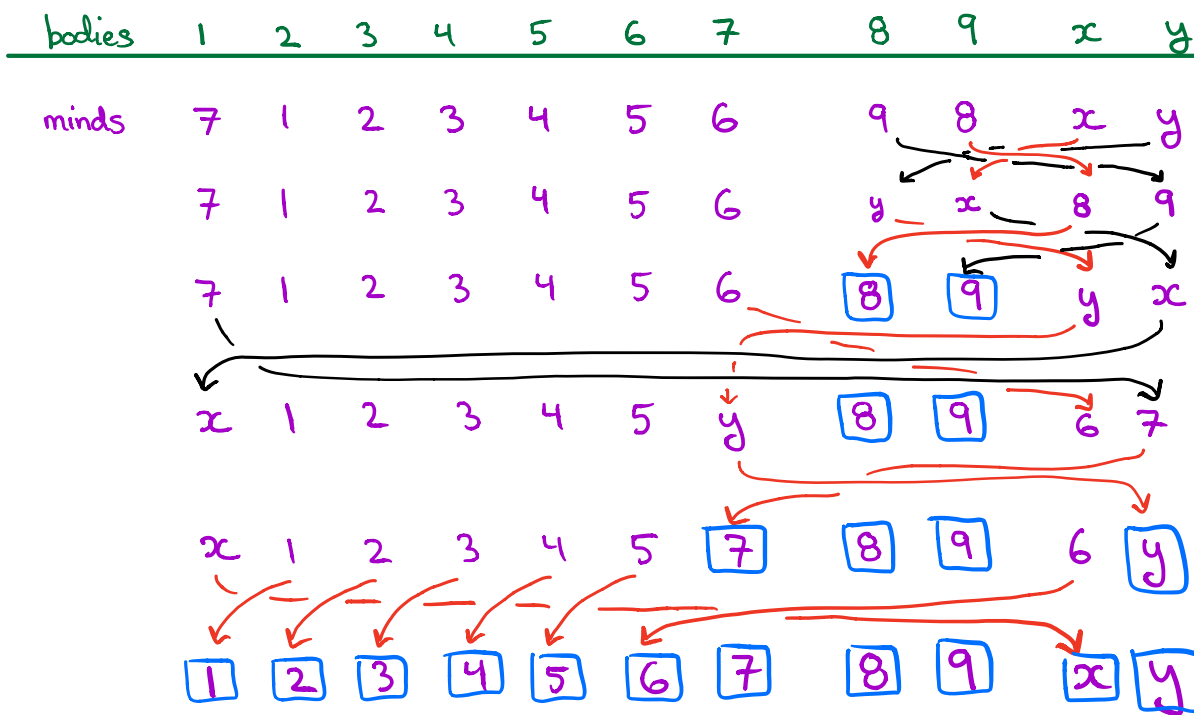


Resulting permutation : (prof bender emperor bucket amy hermes leela) (fry zoidberg)



Bibbeldrum Tate

Sweet Clyde



← original configuration

$(8y)(9x)$

$(8x)(9y)$

$(1y)(7x)$

$(7y)$

$(6x)(5x)(4x)(3x)(2x)(1x)$

$$\therefore (1234567)(89) (8y)(9x)(8x)(9y)(1y)(7x)(7y)(6x)(5x)(4x)(3x)(2x)(1x) \\ = \varepsilon$$

Swap back step 1,2



Swap back step 3,4



Swap back step 5,6



Swap back step 7,8



Swap back step 9,10



Swap back step 11,12



Swap back step 13,14



Sweet Clyde's Inversion Theorem:

Basically, no matter how permuted up your minds are they can be restored using at most two extra players.

Known in popculture as Keeler's Theorem

translation ...

For any $\rho \in S_n$ there exist 2-cycles $\tau_1, \tau_2, \dots, \tau_\ell \in S_{n+2}$ such that

$$\rho \tau_1 \tau_2 \dots \tau_\ell = \varepsilon$$

where

- the τ_i 's are distinct
- each τ_i involves at least one of $n+1$ or $n+2$.

First, let π be some k -cycle on $[n] = \{1 \dots n\}$: WLOG write

$$\pi = \begin{pmatrix} 1 & 2 & \dots & k & k+1 & \dots & n \\ 2 & 3 & \dots & 1 & k+1 & \dots & n \end{pmatrix}$$

Let $\langle a, b \rangle$ represent the transposition that switches the contents of a and b .
By hypothesis π is generated by DISTINCT switches on $[n]$.

Introduce two "new bodies" $\{x, y\}$ and write $\pi^* = \begin{pmatrix} 1 & 2 & \dots & k & k+1 & \dots & n & x & y \\ 2 & 3 & \dots & 1 & k+1 & \dots & n & x & y \end{pmatrix}$

For any $i = 1, \dots, k$ let σ be the (L-to-R) series of switches

$$\sigma = \langle x, i \rangle \langle x, i+1 \rangle \dots \langle x, k \rangle \langle y, i+1 \rangle \langle y, i+2 \rangle \dots \langle y, k \rangle \langle x, i+1 \rangle \langle y, i \rangle.$$

Note each switch exchanges an element of $[n]$ with one of $\{x, y\}$, so they're all distinct from the switches within $[n]$ that generated π , and also from $\langle x, y \rangle$. By routine verification,

$$\pi^* \sigma = \begin{pmatrix} 1 & 2 & \dots & n & x & y \\ 1 & 2 & \dots & n & y & x \end{pmatrix} \text{ i.e., } \sigma \text{ inverts the } k\text{-cycle and leaves } x \text{ and } y \text{ switched (without performing } \langle x, y \rangle \text{)}.$$

NOW let π be an ARBITRARY permutation on $[n]$: it consists of disjoint (nontrivial) cycles, and each can be inverted as above in sequence, after which x and y can be switched if necessary via $\langle x, y \rangle$, as was desired.



Keele's Method for constructing a product $\sigma \in S_{n+2}$ which undoes $P = C_1 \dots C_r$.

Let $C_1 = (a_1 a_2 \dots a_k)$ and let x, y be two extra bodies.

Notice that

$$\begin{aligned}\sigma_1 &= (y a_1)(x a_k)(y a_k)(x a_{k-1}) \dots (x a_2)(x a_1) \\ &= (a_k a_{k-1} \dots a_2 a_1)(x y)\end{aligned}$$

satisfies $C_1 \sigma_1 = (x y)$.

Construct a σ_i for each C_i , and let $\sigma = \left(\prod_{i=1}^r \sigma_i \right) (x y)^e$ where $e \equiv r \pmod{2}$. Then

$$P\sigma = C_1 \dots C_r \sigma_1 \dots \sigma_r (x y)^e = (x y)^{r+e} = \varepsilon$$

So we can undo P with $n+2r+1$ (if r odd) or $n+2r$ (if r even) transpositions.

Improved: Can find a product of $n+r+2$ distinct transpositions to undo $P = C_1 \dots C_r$.

For a k -tuple $C_i = (a_1 \dots a_k)$ define

$$G_i(x) = (x a_k) \dots (x a_2)(x a_1) \quad \& \quad F_i(x) = (x a_1).$$

Now set

$$\lambda = F_r(y) \dots F_2(y) F_1(x) G_1(y) \underset{\substack{\uparrow \\ \text{from } C_1}}{(x a_k)} G_2(x) \dots G_r(x) (x y)$$

Then $P\lambda = \varepsilon$ and λ uses $n+r+1$ 2-cycles.

Playing with arrangements :

Ad Hoc :

	1	2	3	...	k-1	k	x	y
$C_1 \rightarrow$	k	1	2	...	k-2	k-1	x	y
	x	1	2	...	k-2	k-1	k	y
	x	1	2	...	k-2	(k)	k-1	y
	x	1	2	...	(k-1)	(k)	k-1	y
				...				
	x	(2)	(3)	...	(k-1)	(k)	1	y
	x	(2)	(3)	...	(k-1)	(k)	y	1
	(1)	(2)	(3)	...	(k-1)	(k)	y	x

use x as a revolving door to restore $2 \rightarrow k$.

can't use (1x) again so now involve y.

but these are left swapped and we cannot apply (xy) again.

Doesn't work. We must involve y earlier.

Why? At the 3rd last line we have 2 people switched (x & 1) and we cannot use (1x) since this was already used. Therefore we need to restore two people but can't swap them directly, therefore two extra players must be needed: y and another, say z. So we are in trouble.

Keeler's Method :

	1	2	3	...	k-1	k	x	y
$C_1 \rightarrow$	k	1	2	...	k-2	k-1	x	y
	y	1	2	...	k-2	x	k-1	k
	y	1	2	...	k-2	(k)	k-1	x
	(1)	(2)	(3)	...	(k-1)	(k)	y	x
	(1)	(2)	(3)	...	(k-1)	(k)	(x)	(y)

original scramble

restore

$(y1)(xk)$

(yk)

$(xk-1) \dots (x1)$

(xy)

Keeler's Method on cycle structure 3,3 :

1	2	3	4	5	6	x	y
3	1	2	6	4	5	x	y
y	1	x	6	4	5	2	3
y	1	(3)	6	4	5	2	x
(1)	(2)	(3)	6	4	5	y	x
(1)	(2)	(3)	x	4	y	5	6
(1)	(2)	(3)	x	4	(6)	5	y
(1)	(2)	(3)	(4)	(5)	(6)	(x)	(y)

original scramble

restore 1st cycle

restore 2nd cycle

$(y1)(x3)$

$(y3)$

$(x2)(x1)$

$(y4)(x6)$

$(y6)$

$(x5)(x4)$

An example using the generalization (optimization):

