A greedy sorting algorithm

Sergi Elizalde    Peter Winkler

Dartmouth College

Rutgers Experimental Mathematics Seminar
The *homing* algorithm

Given a permutation $\pi$, repeat the following *placement* step:

- Choose an entry $\pi(i)$ such that $\pi(i) \neq i$.
- Place $\pi(i)$ in the correct position.
- Shift the other entries as necessary.

\[
\begin{array}{cccccccc}
3 & 5 & 6 & 1 & 8 & 4 & 7 & 2 \\
3 & 6 & 1 & 8 & 5 & 4 & 7 & 2 \\
3 & 2 & 6 & 1 & 8 & 5 & 4 & 7 \\
\end{array}
\]
Main questions

- Does the algorithm always finish?
Main questions

- Does the algorithm always finish? YES
Main questions

- Does the algorithm always finish? YES
- How many steps does it take in the worst case...
Main questions

- Does the algorithm always finish? \( \text{YES} \)
- How many steps does it take in the worst case...
  - with a good choice of placements?
  - with a random choice of placements?
  - with a bad choice of placements?
“Motivation”

- Makes sense when sorting physical objects, such as billiard balls.
“Motivation”

- Makes sense when sorting physical objects, such as billiard balls.
- In hand-sorting files, it is common to take the first file and move it to the front, then the second, and so on. This is a (fast) special case of homing.
“Motivation”

- Makes sense when sorting physical objects, such as billiard balls.
- In hand-sorting files, it is common to take the first file and move it to the front, then the second, and so on. This is a (fast) special case of homing.
- It is fun to analyze this algorithm.
“Motivation”

- Makes sense when sorting physical objects, such as billiard balls.
- In hand-sorting files, it is common to take the first file and move it to the front, then the second, and so on. This is a (fast) special case of homing.
- It is fun to analyze this algorithm.
- If you have to sort a list and you are paid by the hour, this is a great algorithm to use.
History

- Despite its simplicity, it seems not to have been considered in the literature.
History

- Despite its simplicity, it seems not to have been considered in the literature.

- Barry Cipra was looking at a variation of an algorithm of John H. Conway. In Cipra’s algorithm, after each placement, the intervening entries are reversed (instead of shifted). This algorithm does not necessarily terminate:

  \[
  71325684 \rightarrow 71348652 \rightarrow 56843172 \rightarrow 52713486 \rightarrow \\
  52317486 \rightarrow 71325486 \rightarrow 71325684
  \]
Despite its simplicity, it seems not to have been considered in the literature.

Barry Cipra was looking at a variation of an algorithm of John H. Conway. In Cipra’s algorithm, after each placement, the intervening entries are reversed (instead of shifted). This algorithm does not necessarily terminate:

$71325684 \rightarrow 71348652 \rightarrow 56843172 \rightarrow 52713486 \rightarrow 52317486 \rightarrow 71325486 \rightarrow 71325684$

Loren Larson misunderstood the definition of the algorithm, and thought the intervening numbers were shifted.
Noam Elkies gave a neat proof that homing always terminates:

- Suppose it doesn’t. Then there is a cycle, since there are only finitely many states.
Noam Elkies gave a neat proof that homing always terminates:

- Suppose it doesn’t. Then there is a cycle, since there are only finitely many states.
- Let $k$ be the largest number which is placed *upward* in the cycle.
Noam Elkies gave a neat proof that homing always terminates:

- Suppose it doesn’t. Then there is a cycle, since there are only finitely many states.
- Let $k$ be the largest number which is placed *upward* in the cycle.
- Once $k$ is placed, it can be dislodged upward and placed again downward, but nothing can ever push it below position $k$. 

Sergi Elizalde, Peter Winkler
A greedy sorting algorithm
Noam Elkies gave a neat proof that homing always terminates:

- Suppose it doesn’t. Then there is a cycle, since there are only finitely many states.
- Let $k$ be the largest number which is placed upward in the cycle.
- Once $k$ is placed, it can be dislodged upward and placed again downward, but nothing can ever push it below position $k$.
- Hence it can never again be placed upward, a contradiction.
Well-chosen placements

Theorem

- An algorithm that always places the smallest or largest available number will terminate in at most $n-1$ steps.
Well-chosen placements

Theorem

- An algorithm that always places the smallest or largest available number will terminate in at most $n-1$ steps.

- Let $k$ be the length of the longest increasing subsequence in $\pi$. Then no sequence of fewer than $n-k$ placements can sort $\pi$. 
Well-chosen placements

Theorem

- An algorithm that always places the smallest or largest available number will terminate in at most $n-1$ steps.

- Let $k$ be the length of the longest increasing subsequence in $\pi$. Then no sequence of fewer than $n-k$ placements can sort $\pi$.

- The permutation $n \ldots 21$ is the only one requiring $n-1$ steps.
Random placements

Theorem

The expected number of steps required by random homing from \( \pi \in S_n \) is at most \( \frac{n^2 + n - 2}{4} \).
Theorem

The expected number of steps required by random homing from \( \pi \in S_n \) is at most \( \frac{n^2 + n - 2}{4} \).

Proof.

- Suppose that we have a permutation where \( k \) of the extremal numbers are home:

  \[
  123746589
  \]
Theorem

The expected number of steps required by random homing from \( \pi \in S_n \) is at most \( \frac{n^2+n-2}{4} \).

Proof.

- Suppose that we have a permutation where \( k \) of the extremal numbers are home:
  \[
  \begin{array}{c}
  123746589 \\
  \end{array}
  \]

- With probability \( \geq \frac{2}{n-k} \), the next step will place an additional extremal number.
Random placements

Theorem

The expected number of steps required by random homing from \( \pi \in S_n \) is at most \( \frac{n^2 + n - 2}{4} \).

Proof.

- Suppose that we have a permutation where \( k \) of the extremal numbers are home:
  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 7 & 4 & 6 & 5 & 8 & 9
  \end{array}
  \]
- With probability \( \geq \frac{2}{n-k} \), the next step will place an additional extremal number.
- Total expected number of steps is \( \leq \sum_{k=0}^{n-2} \frac{n-k}{2} \).
Slow Homing: Example

Starting from

\[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1 \]

place always the leftmost possible entry:
Slow Homing: Example

Starting from

2 3 4 5 6 7 \ldots n 1

place always the leftmost possible entry:
Slow Homing: Example

Starting from

\[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots n \ 1 \]

place always the leftmost possible entry:

\[ 3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots n \ 1 \]
Slow Homing: Example

Starting from

\[ \begin{array}{cccccccc}
2 & 3 & 4 & 5 & 6 & 7 & \ldots & n & 1 \\
\end{array} \]

place always the leftmost possible entry:

\[ \begin{array}{cccccccc}
3 & 2 & 4 & 5 & 6 & 7 & \ldots & n & 1 \\
\end{array} \]
Slow Homing: Example

Starting from

\[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ \ldots \ \ n \ 1 \]

place always the leftmost possible entry:

\[ 3 \ 2 \ 4 \ 5 \ 6 \ 7 \ \ldots \ \ n \ 1 \]
\[ 2 \ 4 \ 3 \ 5 \ 6 \ 7 \ \ldots \ \ n \ 1 \]
Slow Homing: Example

Starting from

\[2 3 4 5 6 7 \ldots n 1\]

place always the leftmost possible entry:

\[3 2 4 5 6 7 \ldots n 1\]
\[2 4 3 5 6 7 \ldots n 1\]
Slow Homing: Example

Starting from

\[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]

place always the leftmost possible entry:

\[3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
Slow Homing: Example

Starting from

\[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]

place always the leftmost possible entry:

\[3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
Slow Homing: Example

Starting from

\[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]

place always the leftmost possible entry:

\[3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 3 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1\]
Slow Homing: Example

Starting from

\[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]

place always the leftmost possible entry:

\[3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 3 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1\]
Slow Homing: Example

Starting from

\[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1 \]

place always the leftmost possible entry:

\[ 3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1 \]

\[ 2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1 \]

\[ 4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1 \]

\[ 2 \ 3 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1 \]

\[ 3 \ 2 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1 \]
Slow Homing: Example

Starting from

$2\ 3\ 4\ 5\ 6\ 7\ldots\ n\ 1$

place always the leftmost possible entry:

$3\ 2\ 4\ 5\ 6\ 7\ldots\ n\ 1$
$2\ 4\ 3\ 5\ 6\ 7\ldots\ n\ 1$
$4\ 2\ 3\ 5\ 6\ 7\ldots\ n\ 1$
$2\ 3\ 5\ 4\ 6\ 7\ldots\ n\ 1$
$3\ 2\ 5\ 4\ 6\ 7\ldots\ n\ 1$
Starting from

$$2 \, 3 \, 4 \, 5 \, 6 \, 7 \ldots \, n \, 1$$

place always the leftmost possible entry:

$$3 \, 2 \, 4 \, 5 \, 6 \, 7 \ldots \, n \, 1$$
$$2 \, 4 \, 3 \, 5 \, 6 \, 7 \ldots \, n \, 1$$
$$4 \, 2 \, 3 \, 5 \, 6 \, 7 \ldots \, n \, 1$$
$$2 \, 3 \, 5 \, 4 \, 6 \, 7 \ldots \, n \, 1$$
$$3 \, 2 \, 5 \, 4 \, 6 \, 7 \ldots \, n \, 1$$
$$2 \, 5 \, 3 \, 4 \, 6 \, 7 \ldots \, n \, 1$$
Slow Homing: Example

Starting from

\[2 \ 3 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]

place always the leftmost possible entry:

\[3 \ 2 \ 4 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 4 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[4 \ 2 \ 3 \ 5 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 3 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1\]
\[3 \ 2 \ 5 \ 4 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 5 \ 3 \ 4 \ 6 \ 7 \ldots \ n \ 1\]
\[2 \ 5 \ 3 \ 4 \ 6 \ 7 \ldots \ n \ 1\]

It takes \(2^{n-1} - 1\) steps to sort this permutation.
Main result

Theorem

Homing always terminates in at most $2^{n-1} - 1$ steps.
Main result

Theorem

Homing always terminates in at most $2^{n-1} - 1$ steps.

To prove this, consider the reverse algorithm. We will show that, starting from the identity permutation, one can perform at most $2^{n-1} - 1$ displacements.
Main result

Theorem

Homing always terminates in at most $2^{n-1} - 1$ steps.

To prove this, consider the reverse algorithm. We will show that, starting from the identity permutation, one can perform at most $2^{n-1} - 1$ displacements.

$$2^{n-1} - 1 = \underbrace{2^{n-2}} + \underbrace{2^{n-2} - 1}$$

until 1 and $n$ are displaced after displacing 1 and $n$
Lemma

After $2^{n-2}$ displacements, both 1 and $n$ have been displaced and will never be displaced again.
Lemma

After $2^{n-2}$ displacements, both 1 and $n$ have been displaced and will never be displaced again.

Proof.

- Note that 1 and $n$ can each be displaced only once.
Lemma
After $2^{n-2}$ displacements, both 1 and $n$ have been displaced and will never be displaced again.

Proof.

- Note that 1 and $n$ can each be displaced only once.
- If after $2^{n-2}$ displacements one of these values hasn’t been displaced, then it played no role in the process.
Lemma

After $2^{n-2}$ displacements, both 1 and $n$ have been displaced and will never be displaced again.

Proof.

- Note that 1 and $n$ can each be displaced only once.
- If after $2^{n-2}$ displacements one of these values hasn’t been displaced, then it played no role in the process.
- Hence the remaining $n-1$ numbers allowed more than $2^{n-2}-1$ steps, contradicting the induction hypothesis.
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2}-1$ more displacements can occur.
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2} - 1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1}$, where

$$\alpha_i = \begin{cases} 0 & \text{if entry } i \text{ is exactly to the right of home.} \\ + & \text{if entry } i \text{ is to the right of home.} \\ - & \text{if entry } i \text{ is to the left of home.} \end{cases}$$
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2} - 1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1}$, where

$$\alpha_i = \begin{cases} 0 & \text{if entry } i \text{ is exactly to the right of home.} \\ + & \text{if entry } i \text{ is to the left of home.} \end{cases}$$

Example

$$\pi = 3\ 5\ 6\ 1\ 8\ 4\ 7\ 2 \quad \longrightarrow \quad \alpha(\pi) =$$
The code of a permutation

Assume now that 1 and \( n \) have both been displaced. We’ll show that only \( 2^{n-2} - 1 \) more displacements can occur.

Assign to each permutation \( \pi \) a code \( \alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1} \), where

\[
\alpha_i = \begin{cases} 
0 & \text{if entry } i \text{ is exactly to the right of home.} \\
+ & \text{if entry } i \text{ is to the left of } \text{home.} \\
- & \text{if entry } i \text{ is to the left of home.}
\end{cases}
\]

Example

\[
\pi = 3 \ 5 \ 6 \ 1 \ 8 \ 4 \ 7 \ 2 \quad \longrightarrow \quad \alpha(\pi) = +
\]
The code of a permutation

Assume now that 1 and \( n \) have both been displaced. We’ll show that only \( 2^{n-2}-1 \) more displacements can occur.

Assign to each permutation \( \pi \) a code \( \alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1} \), where

\[
\alpha_i = \begin{cases} 
0 & \text{if entry } i \text{ is exactly to the right of home.} \\
+ & \text{to the left of} \\
- & \text{home.}
\end{cases}
\]

Example

\[
\pi = 3 \ 5 \ 6 \ 1 \ 8 \ 4 \ 7 \ 2 \quad \rightarrow \quad \alpha(\pi) = + -
\]
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2} - 1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1}$, where

$$
\alpha_i = \begin{cases} 
0 & \text{if entry } i \text{ is exactly to the right of home.} \\
+ & \text{if entry } i \text{ is to the left of home.} \\
- & \text{if entry } i \text{ is exactly to the right of home.}
\end{cases}
$$

Example

$\pi = 3 \ 5 \ 6 \ 1 \ 8 \ 4 \ 7 \ 2 \quad \longrightarrow \quad \alpha(\pi) = + \ - \ +$
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2} - 1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1}$, where

$$\alpha_i = \begin{cases} 0 & \text{if entry } i \text{ is exactly to the right of home.} \\ + & \text{to the right of} \\ - & \text{to the left of} \end{cases}$$

Example

$$\pi = 3 5 6 1 8 4 7 2 \quad \rightarrow \quad \alpha(\pi) = + - + -$$
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2}-1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2\alpha_3 \ldots \alpha_{n-1}$, where

$$\alpha_i = \begin{cases} 0 & \text{if entry } i \text{ is exactly to the right of home.} \\ + & \text{to the left of} \\ - & \text{home.} \end{cases}$$

Example

$$\pi = 3 \ 5 \ 6 \ 1 \ 8 \ 4 \ 7 \ 2 \quad \longrightarrow \quad \alpha(\pi) = + \ - \ + \ - \ + \ - \ -$$
The code of a permutation

Assume now that 1 and $n$ have both been displaced. We’ll show that only $2^{n-2} - 1$ more displacements can occur.

Assign to each permutation $\pi$ a code $\alpha(\pi) = \alpha_2 \alpha_3 \ldots \alpha_{n-1}$, where

$$\alpha_i = \begin{cases} 0 & \text{if entry } i \text{ is exactly to the right of home.} \\ + & \text{if entry } i \text{ is to the right of home.} \\ - & \text{if entry } i \text{ is to the left of home.} \end{cases}$$

Example

$\pi = 3 \, 5 \, 6 \, 1 \, 8 \, 4 \, 7 \, 2 \quad \longrightarrow \quad \alpha(\pi) = + \, - \, + \, - \, - \, - \, 0$
The weight of a code

\[ \alpha = + - + - - 0 \]

Define the weight of a code \( \alpha \) recursively:
The weight of a code

\[ \alpha = + - + - - 0 \]
\[ 5 \quad 1 \quad 3 \quad 3 \quad 4 \]

Define the weight of a code \( \alpha \) recursively:

- For each \(-\), count the number of symbols to its left, and for each \(+\), count the number of symbols to its right.
The weight of a code

\[ \alpha = + - + - - 0 \]
\[ \hat{\alpha} = - + - - 0 \]

Define the weight of a code \( \alpha \) recursively:

- For each \( - \), count the number of symbols to its left, and for each \( + \), count the number of symbols to its right.

- Let \( d \) be the largest of these numbers, and let \( \hat{\alpha} \) be the code obtained by deleting the corresponding symbol.
The weight of a code

\[
\alpha = + - + - - 0 \\
5 1 3 3 4 \\
\hat{\alpha} = - + - - 0
\]

Define the weight of a code \( \alpha \) recursively:

- For each \(-\), count the number of symbols to its left, and for each \(+\), count the number of symbols to its right.
- Let \( d \) be the largest of these numbers, and let \( \hat{\alpha} \) be the code obtained by deleting the corresponding symbol.
- Define

\[
w(\alpha) = 2^d + w(\hat{\alpha}).
\]
The weight of a code: example

\[ w( + - + - - 0 ) \]
The weight of a code: example

\[ w \left( \begin{array}{cccccc} + & - & + & - & - & 0 \\ 5 & 1 & 3 & 3 & 4 \end{array} \right) \]
The weight of a code: example

\[ w\left( + - + - - 0 \right) \]
\[
\begin{array}{cccc}
5 & 1 & 3 & 3 & 4
\end{array}
\]

\[ = 2^5 + w\left( - + - - 0 \right) \]
The weight of a code: example

\[ w( + - + - - 0 ) = 2^5 + w( - + - - 0 ) \]
The weight of a code: example

\[ w \left( \begin{array}{cccccc} + & - & + & - & - & 0 \end{array} \right) \]

\[ \begin{array}{cccc} 5 & 1 & 3 & 3 & 4 \end{array} \]

\[ = 2^5 + w \left( \begin{array}{cccc} - & + & - & - & 0 \end{array} \right) \]

\[ \begin{array}{ccc} 0 & 3 & 2 & 3 \end{array} \]

\[ = 2^5 + 2^3 + w \left( \begin{array}{cccc} - & + & - & 0 \end{array} \right) \]

Sergi Elizalde, Peter Winkler

A greedy sorting algorithm
The weight of a code: example

\[ w( + - + - - 0 ) = 2^5 + w( - + - - 0 ) = 2^5 + 2^3 + w( - + - 0 ) = 5 + 1 + 3 + 3 + 4 = 15 \]
The weight of a code: example

\[
w( + - + - - 0 ) \\
5 1 3 3 4 \\
= 2^5 + w( - + - - 0 ) \\
0 3 2 3 \\
= 2^5 + 2^3 + w( - + - 0 ) \\
0 2 2 \\
= 2^5 + 2^3 + 2^2 + w( - + 0 )
\]
The weight of a code: example

\[
\begin{align*}
wr( + &- + - - 0 ) \\
5 &1 3 3 4 \\
= 2^5 + wr( &- + - - 0 ) \\
0 &3 2 3 \\
= 2^5 + 2^3 + wr( &- + - 0 ) \\
0 &2 2 \\
= 2^5 + 2^3 + 2^2 + wr( &- + 0 ) \\
0 &1
\end{align*}
\]
The weight of a code: example

\[ w( + - + - - 0 ) \]
\[
\begin{array}{cccc}
5 & 1 & 3 & 3 & 4 \\
\end{array}
\]
\[ = 2^5 + w( - + - - 0 ) \]
\[
\begin{array}{cccc}
0 & 3 & 2 & 3 \\
\end{array}
\]
\[ = 2^5 + 2^3 + w( - + - 0 ) \]
\[
\begin{array}{cccc}
0 & 2 & 2 \\
\end{array}
\]
\[ = 2^5 + 2^3 + 2^2 + w( - + 0 ) \]
\[
\begin{array}{cccc}
0 & 1 \\
\end{array}
\]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + w( - 0 ) \]
The weight of a code: example

\[ w( + - + - - 0 ) = 2^5 + w( - + - - 0 ) = 2^5 + 2^3 + w( - + - 0 ) = 2^5 + 2^3 + 2^2 + w( - + 0 ) = 2^5 + 2^3 + 2^2 + 2^1 + w( - 0 ) \]
The weight of a code: example

\[ w( + \quad - \quad + \quad - \quad - \quad 0 ) \]
\[ 5 \quad 1 \quad 3 \quad 3 \quad 4 \]
\[ = 2^5 + w( \quad - \quad + \quad - \quad - \quad 0 ) \]
\[ 0 \quad 3 \quad 2 \quad 3 \]
\[ = 2^5 + 2^3 + w( \quad - \quad + \quad - \quad 0 ) \]
\[ 0 \quad 2 \quad 2 \]
\[ = 2^5 + 2^3 + 2^2 + w( \quad - \quad + \quad 0 ) \]
\[ 0 \quad 1 \]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + w( \quad - \quad 0 ) \]
\[ 0 \]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + 2^0 + w( \quad 0 ) \]
The weight of a code: example

\[ w( + - + - - 0 ) \]
\[
\begin{array}{cccc}
5 & 1 & 3 & 3 & 4 \\
\end{array}
\]
\[ = 2^5 + w( - + - - 0 ) \]
\[
\begin{array}{cccc}
0 & 3 & 2 & 3 \\
\end{array}
\]
\[ = 2^5 + 2^3 + w( - + - 0 ) \]
\[
\begin{array}{cc}
0 & 2 \\
\end{array}
\]
\[ = 2^5 + 2^3 + 2^2 + w( - + 0 ) \]
\[
\begin{array}{c}
0 \\
\end{array}
\]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + w( - 0 ) \]
\[
\begin{array}{c}
0 \\
\end{array}
\]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + 2^0 + w( 0 ) \]
\[ = 2^5 + 2^3 + 2^2 + 2^1 + 2^0 = 47 \]
Bound on the weight

**Lemma**

The maximum of $w(\alpha)$ over codes $\alpha$ of length $k$ is $2^k - 1$, for codes of the form $++\cdots+−−\cdots−$. 
Bound on the weight

Lemma
The maximum of \( w(\alpha) \) over codes \( \alpha \) of length \( k \) is \( 2^k - 1 \), for codes of the form \( + + \cdots + - - \cdots - \).

Proof.
In the recursion,

\[
w(\alpha) \leq 2^{k-1} + w(\hat{\alpha}),
\]

with equality when a \( - \) is deleted from the right or a \( + \) from the left.
The weight increases at each displacement

**Lemma**

Let $\pi \in S_n$ with $\pi(1) \neq 1$ and $\pi(n) \neq n$, and let $\pi'$ be the result of applying some displacement to $\pi$. Let $\alpha = \alpha(\pi)$ and $\alpha' = \alpha(\pi')$. Then

$$w(\alpha') > w(\alpha).$$
The weight increases at each displacement

**Lemma**

Let $\pi \in S_n$ with $\pi(1) \neq 1$ and $\pi(n) \neq n$, and let $\pi'$ be the result of applying some displacement to $\pi$. Let $\alpha = \alpha(\pi)$ and $\alpha' = \alpha(\pi')$. Then

$$w(\alpha') > w(\alpha).$$

**Proof sketch.**

- A number $i$ can be displaced iff $\alpha_i = 0$ in the code.
The weight increases at each displacement

Lemma
Let \( \pi \in S_n \) with \( \pi(1) \neq 1 \) and \( \pi(n) \neq n \), and let \( \pi' \) be the result of applying some displacement to \( \pi \). Let \( \alpha = \alpha(\pi) \) and \( \alpha' = \alpha(\pi') \). Then

\[ w(\alpha') > w(\alpha). \]

Proof sketch.

- A number \( i \) can be displaced iff \( \alpha_i = 0 \) in the code.
- If it is displaced to the left, then \( \alpha_i \) becomes a \(-\), and some entries \( \alpha_j \) with \( j < i \) can change from \(-\) to 0 or from 0 to \(+\).
The weight increases at each displacement

Lemma

Let \( \pi \in S_n \) with \( \pi(1) \neq 1 \) and \( \pi(n) \neq n \), and let \( \pi' \) be the result of applying some displacement to \( \pi \). Let \( \alpha = \alpha(\pi) \) and \( \alpha' = \alpha(\pi') \). Then

\[
w(\alpha') > w(\alpha).
\]

Proof sketch.

- A number \( i \) can be displaced iff \( \alpha_i = 0 \) in the code.
- If it is displaced to the left, then \( \alpha_i \) becomes a \(-\), and some entries \( \alpha_j \) with \( j < i \) can change from \(-\) to \(0\) or from \(0\) to \(+\).
- It can be shown that this increases the weight of the code.
Finishing the proof

Combining these lemmas, the maximum number of displacements is

- at most $2^{n-2}$ until 1 and $n$ are displaced, plus
- at most $2^{n-2}-1$ after 1 and $n$ have been displaced.
Combining these lemmas, the maximum number of displacements is

- at most $2^{n-2}$ until 1 and $n$ are displaced, plus
- at most $2^{n-2}-1$ after 1 and $n$ have been displaced.

So at most $2^{n-1}-1$ in total.
The number of worst-case permutations

\[ h(\pi) = \max \text{ length of a seq. of placements from } \pi \text{ to } 12 \ldots n. \]
The number of worst-case permutations

\[ h(\pi) = \text{max. length of a seq. of placements from } \pi \text{ to } 12 \ldots n. \]
\[ M_n = \{ \pi \in S_n : h(\pi) = 2^{n-1} - 1 \}. \]
The number of worst-case permutations

\[ h(\pi) = \text{max. length of a seq. of placements from } \pi \text{ to } 12 \ldots n. \]

\[ M_n = \{ \pi \in S_n : h(\pi) = 2^{n-1} - 1 \}. \]

For example, \( 23 \ldots n1 \in M_n. \)
The number of worst-case permutations

\[ h(\pi) = \max \text{ length of a seq. of placements from } \pi \text{ to } 12\ldots n. \]

\[ M_n = \{ \pi \in S_n : h(\pi) = 2^{n-1} - 1 \}. \]

For example, \( 23\ldots n1 \in M_n. \)

**Theorem**

\[ B_{n-1} \leq |M_n| \leq (n - 1)!, \]

where \( B_n = n\text{-th Bell number} = \# \text{ partitions of } \{1, 2, \ldots, n\}. \)
The number of worst-case permutations

\[ h(\pi) = \text{max. length of a seq. of placements from } \pi \text{ to } 12 \ldots n. \]

\[ M_n = \{ \pi \in S_n : h(\pi) = 2^{n-1} - 1 \}. \]

For example, \( 23 \ldots n1 \in M_n \).

**Theorem**

\[ B_{n-1} \leq |M_n| \leq (n-1)!, \]

where \( B_n = n\text{-th Bell number} = \# \text{ partitions of } \{1, 2, \ldots, n\} \).

\( B_n \) grows super-exponentially:

\[ B_n \sim \frac{1}{\sqrt{n}} \lambda(n)^{n+1/2} e^{\lambda(n) - n - 1}, \]

where \( \lambda(n) = \frac{n}{W(n)} \), and \( W(n)e^{W(n)} = n \).
The number of worst-case permutations

\[ f_{i,j} = \left| \{ \pi \in M_{i+j} : \alpha(\pi) = \underbrace{+ + \cdots \cdot}^{i-1} + \underbrace{- - \cdots \cdot}_{j-1} \} \right| \]
The number of worst-case permutations

\[ f_{i,j} = |\{ \pi \in M_{i+j} : \alpha(\pi) = \underbrace{+ + \cdots +}_{i-1} - \underbrace{- \cdots -}_{j-1} \}| \]

\[ F(u, v) = \sum_{i,j \geq 1} f_{i,j} u^i v^j \]
The number of worst-case permutations

\[ f_{i,j} = |\{ \pi \in M_{i+j} : \alpha(\pi) = ++ \cdots + - - \cdots - \} | \]

\[ F(u, v) = \sum_{i,j \geq 1} f_{i,j} u^i v^j \]

\[ |M_n| = \sum_{i+j=n} f_{i,j} \text{ is the coefficient of } t^n \text{ in } F(t, t). \]
The number of worst-case permutations

\[ f_{i,j} = \left| \{ \pi \in M_{i+j} : \alpha(\pi) = \underbrace{+ + \cdots +}_{i-1} - \underbrace{- \cdots -}_{j-1} \} \right| \]

\[ F(u, v) = \sum_{i,j \geq 1} f_{i,j} u^i v^j \]

\[ |M_n| = \sum_{i+j=n} f_{i,j} \text{ is the coefficient of } t^n \text{ in } F(t, t). \]

**Theorem**

\[ F(u, v) = uv + uv \frac{\partial}{\partial u} F(u, v) + uv \frac{\partial}{\partial v} F(u, v) - u^2 v^2 \frac{\partial^2}{\partial u \partial v} F(u, v) \]
The problem
Fast Homing
Slow Homing
Counting bad cases

Sergi Elizalde, Peter Winkler
A greedy sorting algorithm
The problem
Fast Homing
Slow Homing
Counting bad cases

THANK YOU

Sergi Elizalde, Peter Winkler
A greedy sorting algorithm