

# CDM

## Permutations and Groups

Klaus Sutner  
Carnegie Mellon University

Fall 2009

### Outline

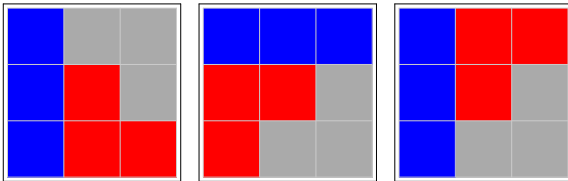
- 1 Counting, with a Twist
- 2 Groups and Such Like
- 3 Symmetric Groups
- 4 Decompositions

### Problem 1: Tic-Tac-Toe

How many different ways are there to place 3 crosses and 3 noughts on a (standard  $3 \times 3$ ) Tic-Tac-Toe board?

No problem: there are  $\binom{9}{3,3,3} = 1680$  possible placements.

True, but what if we wanted to identify boards that can be obtained from rotation, or looking through a mirror?



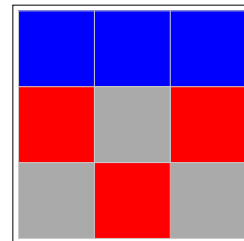
### Eyeballing It

There are 4 rotations, and 4 reflections for the Tic-Tac-Toe board.

In all, a single placement of marks can have as many as 8 variants.

So we should expect some  $1680/8 = 210$  patterns – placements “up to equivalence”.

But that is not quite right: some placements have less than 8 variants, so we are under-counting the number of patterns.



### Problem 2: Boolean Circuits

Here is a similar but somewhat less frivolous problem. Suppose we want to implement Boolean functions  $f: \mathbb{B}^n \rightarrow \mathbb{B}$  as circuits.

There are  $2^{2^n}$  such functions, but we don't need as many circuits. For example, we may have  $f(x, y, z) = g(y, z, x)$  so it suffices to implement either  $f$  or  $g$ .

In general, we can permute the variables arbitrarily: two functions  $f$  and  $g$  are equivalent if

$$f(\vec{x}) = g(\pi(\vec{x}))$$

for some permutation  $\pi$ .

How many Boolean functions are there modulo input permutations?

### Another Version

Permuting the inputs is an obvious modification of a circuit, but there are other possibilities.

For example, it is straightforward to negate input bits and/or the output bit. Thus, there is another equivalence. In general, we can permute the variables arbitrarily: two functions  $f$  and  $g$  are equivalent if

$$f(\vec{x}) = g(\vec{x} \oplus \vec{c}) \oplus d$$

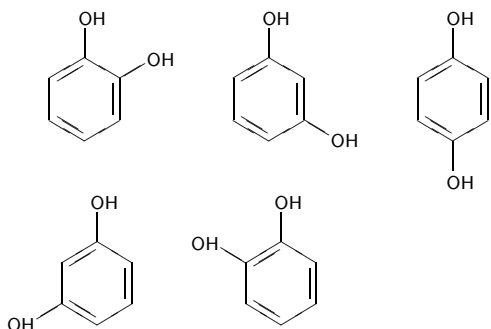
where  $\vec{c} \in \mathbb{B}^n$ ,  $d \in \mathbb{B}$  and  $\oplus$  denotes exclusive or.

And, of course, we can combine permutations and negations.

So how many functions  $\mathbb{B}^n \rightarrow \mathbb{B}$  are there in all these cases?

## Problem 3: Chemistry

Combinatorial (or computational) chemistry leads to many counting problems. For example, suppose we want to enumerate carbocycles like benzene, where some H atoms have been replaced by OH groups.



## Burnside, Pólya and Redfield

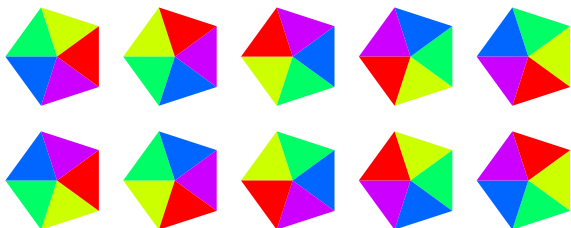
To tackle this type of problem in style one can use some ideas from classical algebra:

- groups
- subgroups
- homomorphisms
- actions

These algebraic ideas are all 19th century, but the applications are 20th century.

## Central Idea

Groups describe symmetries, so if we want to count modulo symmetry it is natural to use groups to do so.



## ■ Counting, with a Twist

## ● Groups and Such Like

## ■ Symmetric Groups

## ■ Decompositions

## Groupoids

## Definition

A **groupoid** is a structure with a single binary operation  $*$ :

$$\mathcal{G} = \langle G, * \rangle$$

where  $*$  :  $G \times G \rightarrow G$ .

There are no further restrictions on the operation. It is quite difficult to describe these structures in general; to obtain a good theory one needs to impose further restrictions on the properties of  $*$ .

Note that the terms over a groupoid can be construed as full binary trees: the interior nodes correspond to "multiplications" and the leaves are variables or constants.

## Some Groupoids

## Example

The natural numbers with exponentiation form a groupoid.

## Example

The integers with subtraction form a groupoid.

## Example

The positive rationals with division form a groupoid.

## More Groupoids

## Example

Binary trees can be considered as groupoid

$$\langle \mathcal{T}, * \rangle$$

where  $\mathcal{T}$  is the collection of all binary trees and  $*$  denotes the operation of attaching two trees to a new root. Note that this operation is highly non-associative:

$$r * (s * t) \neq (r * s) * t$$

no matter what  $r$ ,  $s$  and  $t$  are.

## Example

Likewise we could consider lists over some groundset  $A$  as a groupoid

$$\langle \text{List}(A), * \rangle$$

where  $*$  is interpreted as join (or concatenation).

## Shallon's Graph Algebras

It is often helpful to translate combinatorial structures into algebraic ones. For example, suppose  $\mathcal{G} = \langle V, E \rangle$  is a digraph. We can translate the graph into a groupoid

$$\mathcal{A}(\mathcal{G}) = \langle V_{\perp}, * \rangle$$

by setting  $V_{\perp} = V \cup \{\perp\}$  where  $\perp \notin V$  is a new point and

$$u * v = \begin{cases} u & \text{if } (u, v) \in E, \\ \perp & \text{otherwise.} \end{cases}$$

This operation is not associative in general.

## Exercise

Figure out what left (or right) parenthesized products mean in  $\mathcal{A}(\mathcal{G})$ . Is such a graph algebra commutative?

## Groupoids, II

Groupoids are also (mildly) helpful when dealing with more complicated structures: it is always a good idea to try to understand if a result (or even a definition) also works over groupoids or whether it really requires the additional assumptions.

All the fundamental notion of sub-structure, congruence, homomorphism and so on already make sense for groupoids and are perhaps a bit easier to understand there since there are no other properties lying around that can obscure the view.

## Exercise

Rewrite all the definitions below in the context of groupoids.

## Semigroups

## Definition

A **semigroup** is a structure with a single associative operation  $*$ :  $\mathcal{G} = \langle G, * \rangle$ .

Thus, for all  $x, y, z$  in  $G$  we have associativity:

$$x * (y * z) = (x * y) * z.$$

Many natural algebraic operations have this property, but not all:

- Exponentiation is not associative.
- Subtraction is not associative.
- Graph algebras are generally not associative.

## Idempotents

## Definition

An **idempotent** in a semigroup is an element  $e$  such that  $e * e = e$ .

An idempotent is a bit weaker than an identity.

## Lemma

Let  $S$  be a finite semigroup. Then  $S$  contains an idempotent.

## Exercise

Prove the idempotent lemma. Think lasso.

## Monoids

## Definition

A **monoid** is a semigroup with an **identity element**  $e$ :  $e * x = x * e = x$ .

One usually writes monoids in the form

$$\mathcal{A} = \langle A, *, e \rangle$$

to indicate the neutral element. Of course, the neutral element is idempotent.

## Proposition

The neutral element in a monoid is unique.

*Proof.*  $e = e * e' = e'$ . □

## Classical Examples

## Example

The set of all words over a fixed alphabet forms a monoid with concatenation as operation. The neutral element is the empty word.

## Example

The set of all lists over some fixed ground set forms a monoid with join as operation. The neutral element is the empty list.

## Example

The set of all functions  $f : A \rightarrow A$  for some arbitrary set  $A$  forms a monoid with functional composition as operation. The neutral element is the identity function.

## Example

The set of all binary relations on  $A$ , for some arbitrary ground set  $A$ , forms a monoid with relational composition as operation. The neutral element is the identity relation.

## Classical Examples, contd.

## Example

The set of natural numbers with addition forms a monoid; the neutral element is 0.

Ditto for integers, rationals, algebraic numbers, reals, complex numbers.

## Example

The set of positive natural numbers with multiplication forms a monoid; the neutral element is 1.

## Example

The set of all  $n$  by  $n$  matrices of, say, integers, with matrix multiplications forms a monoid; the neutral element is the identity matrix.

## Strange Examples

## Example

A **band** is a semigroup defined on the Cartesian product  $A \times B$  where  $A$  and  $B$  are two arbitrary sets (non-empty). The operation is

$$(a, b) * (c, d) = (a, d)$$

It is obvious that this operation is associative. Note that a band is idempotent:  $x * x = x$  for all  $x$ .

## Example

The **bicyclic semigroup** is defined on  $\mathbb{N} \times \mathbb{N}$  by the operation

$$(a, b) * (c, d) = (a - b + \max(b, c), d - c + \max(b, c))$$

Associativity requires a little argument here. This may look strange, but it is just the free semigroup on two generators  $r$  and  $s$  subject to  $sr = 1$ .

The idempotents of this semigroup are exactly the elements  $(a, a)$ .

## Solving Equations

Monoids appear quite frequently, but have one crucial flaw from the point of view of solving equations: in general we cannot solve the equation

$$a * x = b.$$

To make sure solutions always exist we need more assumptions. One step in the right direction is the **(left) cancellation property**:

$$a * x = a * y \quad \text{implies} \quad x = y$$

This guarantees that a solution, if it exist, is unique.

## Exercise

Check which of the monoids from above have the cancellation property. What restrictions on the left multiplier  $a$  are necessary to guarantee cancellation?

## Generalized Inverses

To solve equations, one needs some kind of inverse element. There are several classes of semigroups that are closer to groups than general ones.

## Definition

An element  $a$  of a semigroup is **regular** if there is an element  $b$  such that  $aba = a$ . A semigroup is **regular** if all its elements are.

Element  $a$  is said to have a **generalized inverse** if there is a  $b$  such that  $aba = a$  and  $bab = b$ . A semigroup is **inverse** if every element has exactly one generalized inverse.

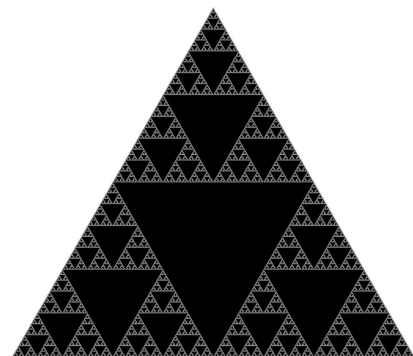
If we write  $a'$  for the inverse in an inverse semigroup we have

$$a'' = a \quad \text{and} \quad (ab)' = b'a'$$

## Lemma

A semigroup is inverse iff it is regular and all its idempotents commute.

## Sierpinski Triangle



## Symmetries in Geometry

We can rotate and reflect the Sierpinski triangle, the underlying group being the dihedral groups  $D_3$  (symmetries of an equilateral triangle).

But that's not really the whole story. For example, we could shift one of the component triangles to another one.

Or we could map the whole figure into the upper triangle (the left or right triangle).

Or into one of the smaller triangles.

Or one of the smaller triangles into yet another one.

## Partial Symmetries

One reason inverse semigroups are very important is that they generalize symmetry groups.

For any set  $X$  consider the **symmetric monoid**

$$\mathcal{I}(X) = \{ f : X \rightarrow_p X \mid f \text{ partial, injective} \}$$

Then  $\mathcal{I}(X)$  is an inverse semigroup.

For the Sierpinski triangle  $T$  the symmetric monoid of  $T$  has many maps that are absent in the corresponding group.

## Exercise

Explain why  $\mathcal{I}(X)$  fails to be a group (for  $X$  non-empty).

## Groups

At last, here is the kind of structure that guarantees existence and uniqueness of solutions.

## Definition

A **group** is a monoid  $G = \langle G, \cdot, e \rangle$  where

$$\forall x \exists y (x \cdot y = y \cdot x = e)$$

The  $y$  above is uniquely determined by  $x$  and called the **inverse** of  $x$ .

## Abelian Groups

## Definition

A group is **commutative** or **Abelian** if  $x \cdot y = y \cdot x$  for all  $x$  and  $y$ .

## Notation:

It is customary to write Abelian groups additively as  $\langle G, +, 0 \rangle$  or  $\langle G, + \rangle$ ,

General groups are written multiplicatively as  $\langle G, \cdot, 1 \rangle$  or  $\langle G, \cdot \rangle$ . As usual, the multiplication operator is often omitted.

The inverse is correspondingly written  $-x$  in additive notation and  $x^{-1}$  in multiplicative notation.

## Classical Examples

## Example

The set of integers (rationals, reals, complexes) with addition forms a group; the neutral element is 0.

## Example

The set of modular numbers relatively prime to modulus  $m$  with multiplication forms a group; the neutral element is 1.

## Example

The set of non-zero rationals (reals, complexes) with multiplication forms a group; the neutral element is 1.

## Classical Examples

## Example

The set of all regular  $n$  by  $n$  matrices of reals, with matrix multiplications forms a group; the neutral element is the identity matrix.

## Example

The set of all permutations  $f : A \rightarrow A$  for some arbitrary set  $A$  forms a group with functional composition as operation. The neutral element is the identity function.

## Solving Equations

In a group, the equation

$$a \cdot x = b$$

always has the unique solution

$$x = a^{-1} \cdot b$$

Note that this is easier than standard arithmetic: there is no need to worry about the case  $a = 0$ .

The systematic study of abstract groups was one of the central accomplishments of 19th century mathematics.

They appear in many, many places and some understanding of their basic properties is crucial.

## Why Abstract Algebra?

**Laziness:** any property, of, say, groups derived from only the axioms holds in all groups, automatically. You only check three simple properties, and all results apply.

**Psychology:** it is sometimes easier to argue abstractly than in a concrete situation (you can't see the forest because of all the trees).

This is a bit hard to believe, but true. E.g., consider non-singular matrices of reals. Show that

$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$$

One could try to use the properties of matrix multiplication and, say, Gaussian elimination, to prove this. Any such argument would be very hard and technically difficult.

## Exercise

Give a simple, abstract proof of this equation in any group whatsoever.

- Counting, with a Twist
- Groups and Such Like
- Symmetric Groups
- Decompositions

## Permutations

For our purposes the most important examples of groups are those comprised of permutations.

## Definition

A **permutation** is a bijection  $f : A \rightarrow A$ , in particular when  $A$  is a finite set.

The collection of all permutations on  $A$ , an  $n$ -element set, under functional composition is the **symmetric group (on  $n$  letters or points)**.

Notation:  $\mathbb{S}_n$

As we will see shortly, in most cases the full symmetric group is too large; we need to focus on subgroups of  $\mathbb{S}_n$ .

## Permutations

We will focus on the carrier set  $A = [n]$ . In this case, we can represent  $f$  by a  $2 \times n$  matrix of the form

$$\begin{pmatrix} 1 & 2 & 3 & \dots & n-1 & n \\ f(1) & f(2) & f(3) & \dots & f(n-1) & f(n) \end{pmatrix}$$

This is the so-called **two-line** representation of  $f$ . Needless to say, the first row in this matrix is really redundant, but this redundancy makes it a bit easier to read off specific values. Alternatively, we can use **one-line** representation:

$$(f(1), f(2), \dots, f(n-1), f(n))$$

## Rearrangements

The one-line representation is interesting since it suggests a different interpretation: a permutation is a **rearrangement** of the values  $1, 2, \dots, n$  (or some other ordered carrier set).

$$(f(1), f(2), \dots, f(n-1), f(n))$$

To explain precisely how the two notions are related we have to talk about **actions** (here, permutations acting on lists). More about this later, for the time being we just think of rearrangements informally.

## Notation:

Note that this notation is a bit dangerous: it might be confused with all kinds of other lists. On occasion we may write  $T(a_1, \dots, a_n)$  (where the  $T$  stands for transformation) for clarity.

## Too Many

No yawning, please.

### Lemma

There are  $n!$  permutations on  $[n]$ .

*Proof.* Induction on  $n$ . The base case  $n = 1$  is obvious.

For the step from  $n - 1$  to  $n$  note that we can insert the new element  $n$  in  $n$  places (indicated by  $|$ ) into any permutation of  $n - 1$ :

$$| a_1 | a_2 | a_3 | \dots | a_{n-2} | a_{n-1} |$$

By IH, this produces  $(n - 1)! \cdot n = n!$  permutations. □

So this is worse than exponential:  $2^n = o(n!)$ .

## Generating All Permutations

The counting argument in the last proof also produces an algorithm to construct all permutations on  $[n]$ :

```
gen_perm( n )
  if( n == 1 )
    return ((1));

P = gen_perm( n-1 );
res = nil;
forall p in P do
  insert n in all places in p,
  append all these perms to res

return res;
```

This is correct but atrocious as far as memory requirements go. Can we generate all permutations of  $[n]$  using less memory? More later.

- Counting, with a Twist
- Groups and Such Like
- Symmetric Groups
- 4 Decompositions

## Cycle Decomposition

Suppose  $f$  is a permutation. The functional digraph of  $f$  is particularly simple: it consists only of cycles (all transients are 0).

### Example

Here is a permutation on  $n = 12$  in two-line notation.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 10 & 12 & 8 & 3 & 7 & 9 & 5 & 6 & 11 & 2 & 4 & 1 \end{pmatrix}$$

It produces the cycles

$$(1, 10, 2, 12), (3, 8, 6, 9, 11, 4), (5, 7)$$

There are 3 cycles of lengths 4, 6, 2, respectively.

Note that there could be just a single cycle of length  $n$ :  $(1, 2, \dots, n)$  in cycle notation stands for the cyclic shift.

## Omitting Fixed Points

It is customary (and often very useful) to omit fixed points from the list of cycles.

### Example

The cycle decomposition of

$$T(4, 7, 1, 6, 8, 9, 11, 5, 2, 10, 3, 12, 14, 13)$$

would be written as

$$(1, 4, 6, 9, 2, 7, 11, 3), (5, 8), (13, 14),$$

leaving out the fixed points 10 and 12.

In standard mathematics texts you should expect to find the more compact notation used a lot.

## Computing the Cycle Decomposition

### Lemma

We can compute the cycle decomposition of a permutation  $f : [n] \rightarrow [n]$  in time and space linear in  $n$ .

Here we tacitly assume that  $f(x)$  can be computed in time  $O(1)$  (which is safe since ordinarily  $f$  will be given by an explicit array).

There are at least two ways to think about this:

- Compute the strongly connected components in the functional digraph of  $f$ .
- Compute the orbits of the function  $f$ , exploiting the fact that they are all periodic.

Note that generating the cycle decomposition seems to require linear space (as opposed to Floyd's algorithm for transients and period).

## Canonical Cycle Decomposition

Note that we can rearrange the cycles arbitrarily, and we can rotate each individual cycle without changing the underlying permutation.

For example, the following two decompositions describe the same permutation on  $n = 14$ .

$$((7, 5), (11, 4, 3, 8, 6, 9), (12, 1, 10, 2))$$

$$((1, 10, 2, 12), (3, 8, 6, 9, 11, 4), (5, 7))$$

The second representation may seem more natural from the implementer's point of view, but it is the first that has better combinatorial properties.

### Definition

The **canonical cycle decomposition (CCD)** of a permutation is obtained by rotating all cycles so that the largest element is up front and the cycles are ordered by first element. If the least element is in the first position we speak of the **reverse canonical cycle decomposition (RCCD)**

## More Natural?

Here is the prototype algorithm that almost everybody would write when asked to implement cycle decomposition.

```
for x = 1, ..., n do
  if( x unmarked )
    mark x;
    res = (x);
    while( f(x) unmarked )
      x = f(x);
      mark x;
      append( res, x );
    output res;
```

This program places the least element first in each cycle and returns the cycles sorted by first element.

## CCD for Length 4

Here are the CCDs for all elements of  $S_4$ , enumerated in lex order.

((1), (2), (3), (4))	((1), (2), (4, 3))	((1), (3, 2), (4))	((1), (4, 2, 3))
((1), (4, 3, 2))	((1), (3), (4, 2))	((2, 1), (3), (4))	((2, 1), (4, 3))
((3, 1, 2), (4))	((4, 1, 2, 3))	((4, 3, 1, 2))	((3), (4, 1, 2))
((3, 2, 1), (4))	((4, 2, 1, 3))	((2), (3, 1), (4))	((2), (4, 1, 3))
((3, 1), (4, 2))	((4, 1, 3, 2))	((4, 3, 2, 1))	((3), (4, 2, 1))
((2), (4, 3, 1))	((2), (3), (4, 1))	((4, 2, 3, 1))	((3, 2), (4, 1))

### Exercise

Find a good algorithm to compute the CCD of a given permutation. What is the running time of your algorithm?

## Flattening CCDs

Here are these CCDs flattened out.

1, 2, 3, 4	1, 2, 4, 3	1, 3, 2, 4	1, 4, 2, 3
1, 4, 3, 2	1, 3, 4, 2	2, 1, 3, 4	2, 1, 4, 3
3, 1, 2, 4	4, 1, 2, 3	4, 3, 1, 2	3, 4, 1, 2
3, 2, 1, 4	4, 2, 1, 3	2, 3, 1, 4	2, 4, 1, 3
3, 1, 4, 2	4, 1, 3, 2	4, 3, 2, 1	3, 4, 2, 1
2, 4, 3, 1	2, 3, 4, 1	4, 2, 3, 1	3, 2, 4, 1

We get all permutations. Could this be coincidence?

## A Bijection

From the data structure point of view, the cycle decomposition is a list of lists of integers. Hence we can flatten it to obtain a plain list of integers:

$$\text{flat} : \text{List}(\text{List}(\mathbb{N})) \rightarrow \text{List}(\mathbb{N})$$

If we start with the full cycle decomposition (including fixed points) we obtain a permutation (in one-line representation) this way. For arbitrary decompositions this is of little interest, but if we start with the CCD we get the following proposition, which is helpful in enumeration problems related to permutations.

### Proposition

The map  $\text{CCD} \circ \text{flat}$  is a bijection on  $S_n$ .

## Questions

### Exercise

Prove that  $\text{CCD} \circ \text{flat}$  is indeed a bijection.

### Exercise

What are the fixed points of this bijection?

### Exercise

How about  $\text{RCCD} \circ \text{flat}$ ?

## Algebraic Decomposition

Since permutations are functions we can compose them by ordinary functional composition  $f \circ g$ . Recall that we write composition in diagrammatic form:

$$(f \circ g)(x) = g(f(x))$$

Some (misguided) texts use the opposite convention.

**Basis Problem:**

Find a small and/or simple set of permutations so that all permutations can be written as a product of these.

**Decomposition Problem:**

Given such a basis  $B$ , find a way to decompose a given permutations into a product of permutations in  $B$ .

## Transpositions

## Definition

A **transposition** is a permutation that consists of a single 2-cycle.

In cycle notation, transpositions are exactly the permutations of the form  $(a, b)$  for  $a \neq b$ .

## Example

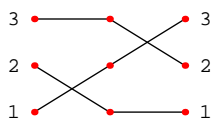
Consider the following transpositions over  $[3]$ , given in cycle notation.

$$(1, 2) \circ (2, 3) = (3, 1, 2)$$

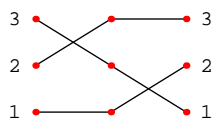
$$(2, 3) \circ (1, 2) = (2, 3, 1)$$

Thus, composition of permutations is not commutative (it is associative, though, since composition of functions is so associative).

## In Pictures



$$(1, 2) \circ (2, 3) = (3, 1, 2)$$



$$(2, 3) \circ (1, 2) = (2, 3, 1)$$

## Basis theorem

## Lemma

Every permutation can be written as a product of transpositions.

*Proof.* (sketch)

Since every permutation is composed of disjoint cycles, it suffices to show that every cycle  $(a_1, \dots, a_m)$  is a product of transpositions.

Show this by induction on  $m \geq 2$ . The crucial step is

$$(a_m, b) \circ (a_1, \dots, a_m) = (a_1, a_2, \dots, a_{m-1}, a_m, b)$$

□

## Exercise

Fill in all the gaps in this argument.

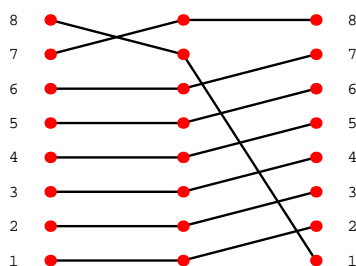
## Exercise

Find a direct decomposition

$$(a_1, b_1) \circ (a_2, b_2) \circ \dots \circ (a_m, b_m) = (c_1, c_2, \dots, c_m, c_{m+1}).$$

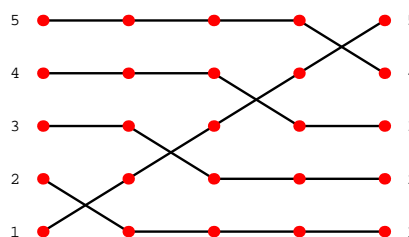
## In Pictures

For a simple cycle this is easy to see in the composition picture.



## Another Easy Case

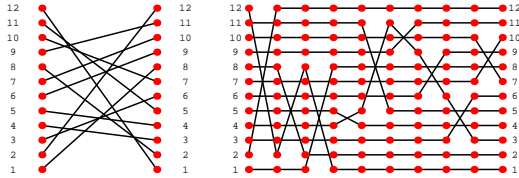
Here is another simple decomposition of a cycle into transpositions.



$$\text{So } (1, 2)(2, 3)(3, 4)(4, 5) = (5, 4, 3, 2, 1).$$

## More Pictures

A more complicated permutation on  $n = 12$ , and its decomposition into transpositions.



## Exercise

Find an algorithm to generate the picture on the right.

## More Identities

For the next proposition, we abuse notation and use exponents for permutations given in cycle notation.

## Proposition

$$(a, b) \circ (b, c) \circ (a, b) = (a, c)$$

$$(1, \dots, n)^i \circ (1, 2) \circ (n, \dots, 1)^i = (i + 1, i + 2)$$

where  $0 \leq i \leq n - 2$ .

## Exercise

Prove these identities.

## Even and Odd Permutations

Needless to say, the decomposition into transpositions is not unique.

## Definition

A permutation is **even** if it can be written as the product of an even number of transpositions, and **odd** if it can be written as the product of an odd number of transpositions.

Note the cautious wording: this does not say that every permutation is either even or odd. It leaves open the possibility that some permutation could be both even and odd. However, one can show that any permutation is either even or odd, never both.

## Lemma

No permutation is even and odd.

## Proof Sketch

*Proof.* Let  $\sigma$  be a permutation of  $[n]$ . Consider the polynomials

$$P(x_1, \dots, x_n) = \prod_{i < j} (x_i - x_j)$$

$$P_\sigma(x_1, \dots, x_n) = \prod_{i < j} (x_{\sigma(i)} - x_{\sigma(j)})$$

Then necessarily  $P = \pm P_\sigma$ . One can show that  $P = +P_\sigma$  iff  $\sigma$  is even, and  $P = -P_\sigma$  iff  $\sigma$  is odd.  $\square$

Note that  $\sigma$  is operating on the variables here.

## Exercise

Fill in the details of this argument.

## Alternating Groups

The composition of even permutations is again even, so we can assemble them into a new group.

## Definition

The collection of all even permutations of  $A$ , an  $n$ -element set, is the **alternating group** on  $n$  points.

Notation:  $A_n \subseteq S_n$ .

As we will see,  $A_n$  is indeed a subgroup of  $S_n$  and has size  $n!/2$ .

Part of the importance of alternating groups comes from the fact that for  $n \geq 5$  each alternating group  $A_n$  is simple: it has only trivial normal subgroups.

## Computing the Inverse

## Definition

The **order** of a permutation  $f$  is the least  $m > 0$  such that  $f^m = I$ .

## Lemma

Let the cycles of permutation  $f$  have lengths  $l_1, \dots, l_k$  and let  $m$  be the LCM of  $l_1, \dots, l_k$ . Then  $m$  is the order of  $f$ .

This has the consequence that

$$f^{-1} = f^{m-1}.$$

Hence we can compute the inverse by iteration when the carrier set is finite. Of course, this does not work in the infinite case.

Needless to say, no one would compute the inverse this way.

## Really Computing the Inverse

Note that computing  $m$  and iterating  $f$  to get  $f^{m-1}$  is not a very good idea. Here is a computationally better way to get at the inverse. Define  $g$  by

$$g(f(i)) = i \text{ for } i = 1, \dots, n.$$

Then  $g \circ f = f \circ g = I$  and thus  $g = f^{-1}$ . This takes linear time.

Here is another way: sort the list of pairs

$$((f(1), 1), (f(2), 2), \dots, (f(n), n))$$

in the usual lexicographic order. Then throw away the first components. The resulting permutation is  $f^{-1}$ .

### Exercise

*Explain how the last method works. What is the running time?*

## Upcoming Attractions

We need a few more concepts.

- Subgroups: groups contained in groups.
- Homomorphisms: maps between groups.
- Representations: how to compute in groups.

The first two notions are perfectly general: sub-objects and structure preserving maps are essential in all of algebra, not just groups.

Group representations are a bit special, there is a large and well-developed machinery to express and compute in groups. We will only touch the surface here, check the literature for more information.