

# Computational Discrete Mathematics

Klaus Sutner  
Carnegie Mellon University

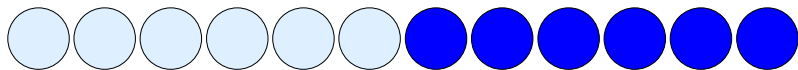
Fall 2009

# Outline

- 1 Not So Frivolous Example
- 2 CDM Philosophy (Ideology?)
- 3 Administrivia

## Flipping Pebbles

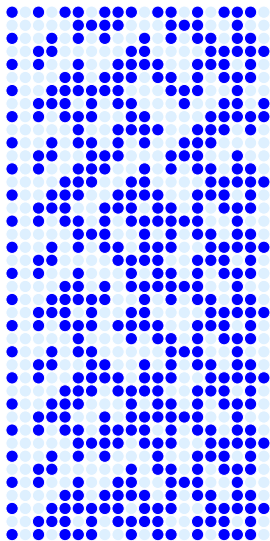
Take a row of tokens, white on one side, blue on the other.



Starting at the left, flip the current token. If it is now white, skip the next token. Otherwise, skip the next two tokens. Repeat till you fall off the end.



And Repeat . . .



## Pebbles, Schmebbles

- We are really talking about the set  $\mathbf{2}^*$  of all (finite) binary sequences, aka binary words.
- The flipping rule defines an operation  $\tau : \mathbf{2}^* \rightarrow \mathbf{2}^*$ .
- We want to “understand” this operation.

## Easy Observation

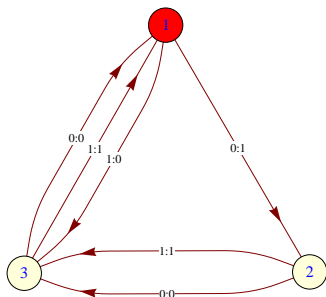
First,  $\tau$  is trivially **length-preserving**.

Second,  $\tau$  is **reversible**: we can reconstruct  $x$  from  $\tau(x)$ .

By General Abstract Nonsense (aka basic math facts)  $\tau$  is just a permutation of  $2^*$ . The operation produces cycles of words, each cycle containing only words of the same length.

## Full Disclosure: Transducers

Our pebble-flipping operation is really a transduction on binary words.  
The corresponding 3-state **transducer**:



We are really interested in understanding the behavior of transducers of this type.

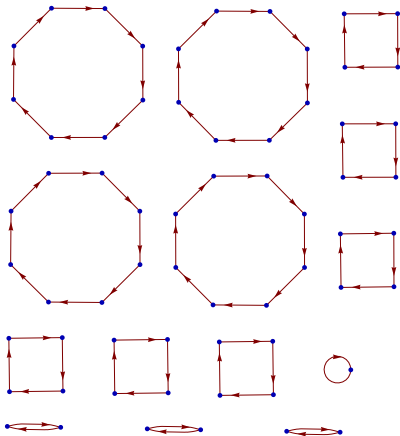
## A Small Cycle

A small example:

000 → 100 → 001 → 101 → 000

So 000 produces a cycle of length 4.

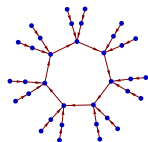
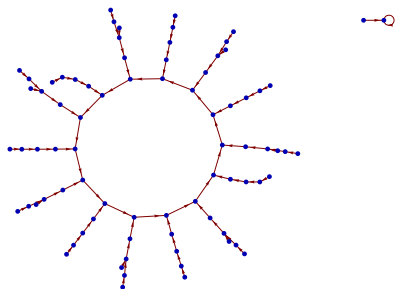
## Some More Cycles



## Pop Quiz

- Beware beautiful pictures. Does this one make any sense?
- For example, explain the fixed point.
- Then explain the three 2-cycles.
- Overall, what might this be a picture of?

## Without Reversibility



## The Right Questions

- How many cycles are there?
- How long are these cycles?

More precisely, we would like answers for each word length  $n$ .

We are really talking about the **functional digraph** of  $\tau$ , also called a **permutation structure** in model theory.

## The Right Questions, 2

- How hard is it to test membership in a cycle?
- How hard is it to compute the least element?

These are **computational complexity** problems; note that it is important here that points are not anonymous but have internal structure.

We want elegant algorithmic answers as well as upper and lower bounds.

## Need More Data

One could sit down with paper and pencil and compute a lot more cycles such as

0000 → 1001 → 0011 → 1010 → 0000

This is an excellent way to waste time and go mad in the process.

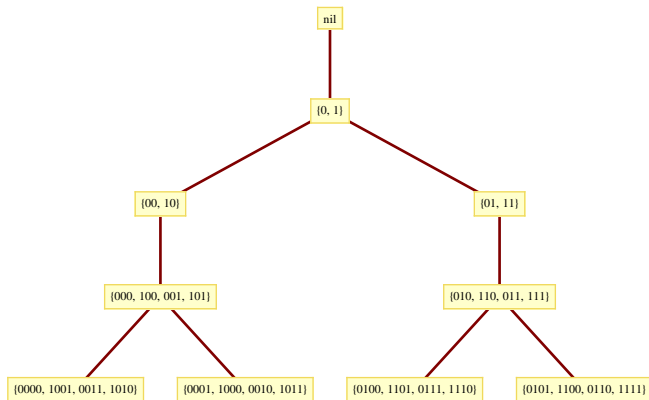
Much better is to write a little program and compute some cycles by brute force.

## Cycle Count/Length Table

	1	2	4	8	16	32
0	1					
1		1				
2		2				
3			2			
4			4			
5				4		
6				8		
7					8	
8					16	
9						16
10						32

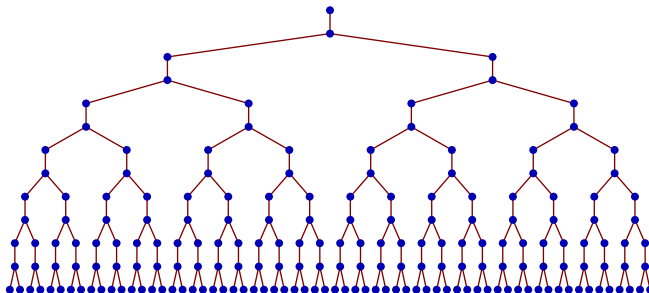
→: cycle length, ↓: word length, missing entries are 0

# Cycle Tree



The root corresponds to the fixed point  $\varepsilon$  (the empty word).

## Cycle Tree, Schematic



Note: the tree seems to be **regular**: there are only 2 subtrees.

## Counting Cycles

### Lemma

*There are  $2^{\lfloor k/2 \rfloor}$  cycles on words of length  $k$ .  
The length of each cycle is  $2^{\lceil k/2 \rceil}$ .*

Proof of lemma is trivial if the picture is correct.

Proof for the picture is quite hard.

**Challenge:** Find a better proof.

## A Harder Question

How hard is it to test if  $u$  and  $v$  lie on the same cycle?

An obvious upper bound for a decision algorithm on words of length  $2k$  is

$$O(k2^k)$$

It takes  $O(k)$  steps to compute  $\tau(x)$  and there are at most  $2^k$  words to consider.

Can we do better than this? Is there a computational shortcut?

# Shuffle

Suppose  $a = a_1a_2 \dots a_k$  and  $b = b_1b_2 \dots b_k$  are two words of length  $k$ . The **shuffle**  $a \parallel b$  of  $a$  and  $b$  is defined by

$$a \parallel b = a_1b_1a_2b_2 \dots a_kb_k$$

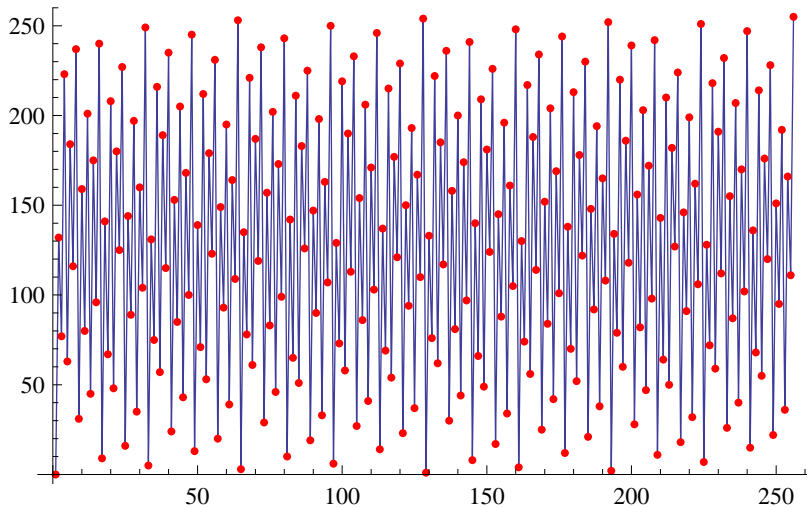
E.g.,  $0010 \parallel 1101 = 01011001$ .

## Intrinsic Coordinates

Suppose  $u$  has length  $2k$ .

Then for every word  $a = a_1 a_2 \dots a_k$  there is a unique word  $b = b_1 b_2 \dots b_k$  such that  $a \parallel b$  lies on the cycle of  $u$ .

## Bouncing Around



## Cycle Membership

### Theorem

*Given  $x, y \in 2^k$  one can test in linear time whether  $x$  and  $y$  lie on the same cycle (actually, a finite state machine can do it).*

The proof is short and a complete fraud.

I would never have found the proof without being able to compute some wild approximations to certain linear time algorithms, plotting pictures of the algorithms and noticing that there was a kind of convergence.

- Not So Frivolous Example

- ② CDM Philosophy (Ideology?)

- Administrivia

## Mathematics, the Last Two Centuries

Beginning in the 1800's, mathematics has become increasingly precise, abstract and formal.

This has led to a great many success stories: Fermat's theorem, Poincare Conjecture, classification of finite simple groups; very solid foundations.

The price is an often almost impenetrable structure.

*If one cannot explain a mathematical problem to a man in the streets in 10 minutes it is not interesting.*

*D. Hilbert*

## The 21<sup>st</sup> Century

We now have a golden opportunity to bring things back into alignment: computation.

The key point here is not the abstract theory of computation (which was developed in the middle of the 20<sup>th</sup> century) but the universal availability of practical computation.

Computation has already taken over the sciences and many other areas of human intellectual pursuits. Somewhat surprisingly, mathematics has been lagging behind a bit.

## Donald Knuth

*“For three years I taught a sophomore course in abstract algebra for mathematics majors at Caltech, and the most difficult topic was always the study of Jordan canonical forms for matrices. The third year I tried a new approach, by looking at the subject algorithmically, and suddenly it became quite clear. The same thing happened with the discussion of finite groups defined by generators and relations, and in another course with the reduction theory of binary quadratic forms. By presenting the subject in terms of algorithms, the purpose and meaning of the mathematical theorems became transparent.”*

## Leslaw Szczerba

*“There occurred a substantial change in my role as a teacher. Earlier, when assigning homework tests, I was treated as an enemy who has to be forced to accept the solution, sometimes in not an exactly honest way. Now the enemy to be defeated was the computer and I was turned into an ally helping to fight this horrible device. This small fact has seriously influenced my contacts with the students. They were much more eager to approach me with their problems, to report on their difficulties, and ask for help.”*

## Alan Turing

*"I expect that digital computing machines will eventually stimulate a considerable interest in symbolic logic . . . The language in which one communicates with these machines . . . forms a sort of symbolic logic."*

## William Thurston

*“The standard of correctness and completeness necessary to get a computer program to work at all is a couple of orders of magnitude higher than the mathematical community’s standard of valid proofs.”*

## More

*“The Algorithm’s coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics.”*

*Bernard Chazelle*

*“If you want to understand the 21<sup>st</sup> Century, you must first understand computation.”*

*Alan Bundy*

## Uses of Computing in Math

- **Number Crunching**

E.g., to solve complicated differential equations. Historically the first major application.

- **Symbolic Computation**

Directly manipulate symbolically presented entities (computer algebra).

- **Example Generation**

Examples are critical to understand a concept, producing them by hand can be too time consuming.

- **Counterexample Generation**

Can help in avoiding dead-end arguments in a proof.

## Less Developed

- **Theorem Proving**

On rare occasions, proof assistants and automatic theorem provers are helpful in finding (parts of) a proof. Better at verification (proof checkers) than search, not easy to use.

- **Knowledge Management**

A global mathematical library would be some  $10^8$  pages. Some small but growing part of this is available in digital form on the web. Some even smaller part is indexed and searchable (semantic markup). Some yet smaller part is validated.

## The Knowledge Soup

There is an ever increasing amount of math material on the web: MathWorld, PlanetMath, wikis, journals, research websites, . . .

Be careful, though – not all the stuff out there is reliable. Sometimes the information is misleading, and sometimes it's simply false.

Try to stick to reputable sources. And, always do a sanity check.

## Computational Discrete Math

The central axiom of CDM is very simple:

- Computation helps to understand mathematics and TCS.
- Mathematics and TCS help to compute.

Duh, is a Bluebird blue?

## The CDM Loop

- Specify/Formalize

We start with a question, often vague and imprecise. With some effort the question is turned into a concise and precise problem in (discrete) mathematics.

- Experiment/Compute

To help develop basic understanding we use computation to generate data (examples and counterexamples). Setting up the programs may well require a bit of work.

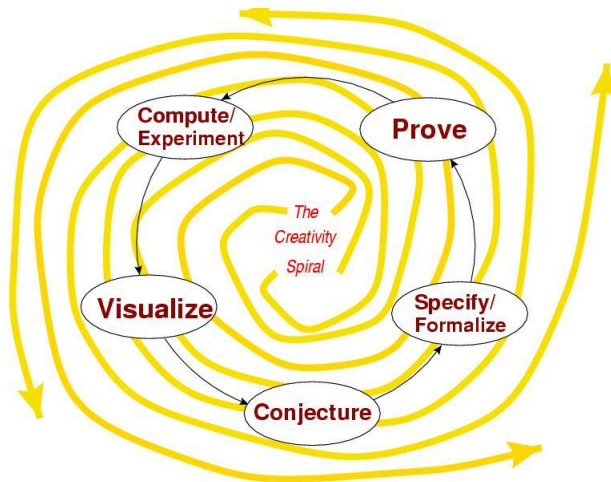
- Analyze/Visualize

Analyze and interpret the data, find patterns and structure.

## The CDM Loop, 2

- Specify/Formalize
- Experiment/Compute
- Analyze/Visualize
- Conjecture/Prove  
Ultimately formulate a conjecture and proceed to prove it. Wrong conjectures and dead-ends in proof attempts can sometimes be eliminated by more computation.
- Apply/Generalize  
Use the new and proven theorem to improve the power of computation; discover new questions.

# The Magic Spiral



## Harsh Reality

*In theory there is no difference between theory and practice.*

*In practice there is.*

*Yogi Berra*

## The Catch

In practice, there is a little issue, a resource allocation problem. You usually have a choice, you can either

- think and reason, or
- program and compute.

Sadly, this is often an exclusive or.

This is not to say that programming is mindless (quite the opposite), but it is very different from classical, paper&pencil based reasoning.

- Not So Frivolous Example
  - CDM Philosophy (Ideology?)
- 3 Administrivia

## Web and Communication

- Course web site:

<http://www.cs.cmu.edu/~cdm>

- No Bleeping Blackboard

## Course Staff

- Prof:  
Klaus Sutner, [sutner@cs.cmu.edu](mailto:sutner@cs.cmu.edu)
- TA:  
Sam Tetrushvili, [samt@cmu.edu](mailto:samt@cmu.edu)
- Course secretary:  
Rosie Battenfelder, [rosemary@cs.cmu.edu](mailto:rosemary@cs.cmu.edu)

## Course Material: Good News

- There is no text book.
- A typical Discrete Mathematics textbook is 1000 pages nowadays and mostly useful for weight lifting.
- Some references are posted on the web. Alas, none of these texts are entirely appropriate.
- I am writing a book, but only bits and pieces exist at this point.

## Course Material: Bad News

- There will be a handout for each lecture.
- A lot of material will be posted on the web.
- It is a good idea to read **all this stuff**.
- The web is often helpful. Again, use with care.
- Lastly, a revolutionary suggestion: go to the library.
- And: Turn up at office hours regularly, not just at times of major crisis.

## Learning Style

- The topics covered in this course translate into quite a bit of material.
- All the most important items will be presented in class, but there will necessarily be things that are left out. It is your responsibility to acquire this additional material.
- Again, read the notes, search the net, go to the library, talk to each other, talk to us.
- One of the desired outcomes of this course is that you know where to find more information should you ever need it.

## Assessment

- The usual testing:
  - homeworks 50%
  - midterm (in-house) 20%
  - final 30%
- The homework is crucial in this course; solving actual problems is the only way to really get a grip on the material.
- The midterm is constrained to 80 minutes; think of it as a little reality check.
- Final will mean one of the following: take-home final, project or oral exam. You will probably have a choice of one out of two. Let's talk after the midterm.

## Preserving TA Sanity

- The homework will involve quite a bit of math (surprise, surprise), so it is essential that it is typeset, not hand-written.
- Submit hard-copy at the beginning of class on the due date.
- Do not even think about sending email.
- If you use a program in your homework make sure to reference it properly (but do not hand in 50 pages of code).

# Typesetting

- There are several ways for you to generate the files:
  - $\text{\LaTeX}$  has a bit of a learning curve, but since it is the publication standard for CS and math it's a good idea to learn how to use the system now.
  - Mathematica has a reasonable WYSIWYG editor. It's not as good as  $\text{\LaTeX}$ , but easier to use initially.
  - Anything else that can produce math.
- Make sure you choose a good, solid editor (i.e. emacs) and become an expert in your system of choice.

## Cooperation

- Lectures will be warm and friendly. Make sure to be an active participant – CDM is not a spectator sport.
- Sam has a conflict for the recitation slot; we'll do figure out some convenient other time to meet, more like office hours.
- You are strongly encouraged to talk about the course material to each other, the course staff and other students.
- This includes discussions of homework problems.

## Limits to Cooperation

- However, even after ample consultation, the work you submit must be written entirely by yourself.
- List all your “consultants” on the first page of your homework.
- To avoid problems with originality, **do not take notes** when discussing homework problems.
- If you write on a board, erase everything in the end.

## More on Limits

- Think of this as being able to copyright your solution: you may have had conversations about it with other during the problem solving phase, but the actual work is solely yours. No one should be in the room when you start writing things up.
- Needless to say, you have to be able to explain all the details of your solution at any time.
- Don't even think about copy & paste (from each other or the web), file sharing, clairvoyance, telepathy, ...
- If you have any questions about policy issues talk to me or Sam, preferably some time before you get into trouble.