Pensieve header: Non-Commutative Gaussian Elimination - Day 1.

Today. Commutative Gaussian elimination, then non-commutative Gaussian elimination (NCGE), then (unlikely) EIWL-10-12, then even less likely, **Patterns**.

Topics (in no particular order). Whatever you may suggest; whatever comes to my mind; ~~the Fibonacci numbers~~; ~~the Catalan numbers~~; ~~the Jones polynomial~~; ~~a more efficient Jones algorithm~~; ~~a riddle on spheres~~; Khovanov homology; Γ-calculus; the Hopf fibration; Hilbert's 13th problem; non-commutative Gaussian elimination; free Lie algebras; the Baker-Campbell-Hausdorff formula; wacky numbers; an order 4 torus; the Schwarz Lantern; knot colourings; the Temperley-Lieb pairing; the dodecahedral link; sound experiments; barycentric subdivisions; ~~some Peano curves~~; braid closures and Vogel's algorithm; ~~the insolubility of the quintic~~; phase portraits; the Mandelbrot set; shadows of the Cantor aerogel; quilt plots; some image transformations; De Bruijn graphs; the Riemann series theorem; finite type invariants and the Willerton fish; ~~the Towers of Hanoi~~; ~~Hochschild homology of (some) coalgebras~~; convolutions and image improvements.

An Image Manipulation Challenge

The image at http://drorbn.net/bbs/show?shot=17-1750-171016-111042.jpg is pathetic. Can you improve it? Whatever you do, should also work well with all other images at http://drorbn.net/bbs/show.php?prefix=17-1750.

A Graphics Challenge

The torus $S^1 \times S^1$ has an order 4 symmetry. Can you draw it in such a manner that it will manifest?

The Cow Problem (Leonard)

A farmer has 19 cows, and she wishes to give them to her daughters to that the first will get 1/2, the second 1/4, and the third 1/5. This is obviously impossible. A wise woman hears of the problem and suggests: "I'll add on one of my cows, so you'll have 20. The first daughter will get 10, the second 5, the third 4, I'll take back the remaining cow, and everyone is happy!".

Problem. Are there any other quadruples like (19, 2, 4, 5), for which the same trick will work? What are all of them?

Hint. It is sometimes better to analyze and generalize, first.

Reminders

- Mathematica receipts due Monday November 6, in class.
- Submissions are limited to 20Mb.

The Commutative Case

```
vs = Table[(i + j)^5, {i, 12}, {j, 8}];
vs // Column
```

```
{32, 243, 1024, 3125, 7776, 16 807, 32 768, 59 049}
{243, 1024, 3125, 7776, 16 807, 32 768, 59 049, 100 000}
{1024, 3125, 7776, 16 807, 32 768, 59 049, 100 000, 161 051}
{3125, 7776, 16 807, 32 768, 59 049, 100 000, 161 051, 248 832}
{7776, 16 807, 32 768, 59 049, 100 000, 161 051, 248 832, 371 293}
{16 807, 32 768, 59 049, 100 000, 161 051, 248 832, 371 293, 537 824}
{32 768, 59 049, 100 000, 161 051, 248 832, 371 293, 537 824, 759 375}
{59 049, 100 000, 161 051, 248 832, 371 293, 537 824, 759 375, 1 048 576}
{100 000, 161 051, 248 832, 371 293, 537 824, 759 375, 1 048 576, 1 419 857}
{161 051, 248 832, 371 293, 537 824, 759 375, 1 048 576, 1 419 857, 1 889 568}
{248 832, 371 293, 537 824, 759 375, 1 048 576, 1 419 857, 1 889 568, 2 476 099}
{371 293, 537 824, 759 375, 1 048 576, 1 419 857, 1 889 568, 2 476 099, 3 200 000}
```

```
MatrixRank[vs]
```

6

```
? Position
```

Position[*expr*, *pattern*] gives a list of the positions at which objects matching *pattern* appear in *expr*.
Position[*expr*, *pattern*, *levelspec*] finds only objects that appear on levels specified by *levelspec*.
Position[*expr*, *pattern*, *levelspec*, *n*] gives the positions of the first *n* objects found.
Position[*pattern*] represents an operator form of Position that can be applied to an expression. ≫

```
? FirstPosition
```

FirstPosition[*expr*, *pattern*] gives the position of the first
element in *expr* that matches *pattern*, or Missing["NotFound"] if no such element is found.
FirstPosition[*expr*, *pattern*, *default*] gives *default* if no element matching *pattern* is found.
FirstPosition[*expr*, *pattern*, *default*, *levelspec*] finds only objects that appear on levels specified by *levelspec*.
FirstPosition[*pattern*] represents an operator form of FirstPosition that can be applied to an expression. ≫

On to Rubik's Cube

```
? Cycles
```

Cycles[{cyc_1, cyc_2, …}] represents a permutation with disjoint cycles cyc_i. ≫

```
n = 54;
g1 = Cycles[
    {{1, 18, 45, 28}, {2, 27, 44, 19}, {3, 36, 43, 10}, {46, 52, 54, 48}, {47, 49, 53, 51}}];
g2 = Cycles[{{7, 16, 39, 30}, {8, 25, 38, 21}, {9, 34, 37, 12},
     {13, 15, 33, 31}, {14, 24, 32, 22}}];
g3 = Cycles[{{28, 31, 34, 48}, {29, 32, 35, 47}, {30, 33, 36, 46},
     {37, 39, 45, 43}, {38, 42, 44, 40}}];
g4 = Cycles[{{1, 3, 9, 7}, {2, 6, 8, 4}, {10, 54, 16, 13}, {11, 53, 17, 14}, {12, 52, 18, 15}}];
g5 = Cycles[
    {{1, 13, 37, 46}, {4, 22, 40, 49}, {7, 31, 43, 52}, {10, 12, 30, 28}, {11, 21, 29, 19}}];
g6 = Cycles[{{3, 48, 39, 15}, {6, 51, 42, 24}, {9, 54, 45, 33},
     {16, 18, 36, 34}, {17, 27, 35, 25}}];
```

```
? PermutationProduct
```

PermutationProduct[*a*, *b*, *c*] gives the product of permutations *a*, *b*, *c*. ≫

```
a_ ∘ b_ := PermutationProduct[a, b]
```

```
? InversePermutation
```

InversePermutation[*perm*] returns the inverse of permutation *perm*. ≫

```
? PermutationSupport
```

PermutationSupport[*perm*] returns the support of the permutation *perm*. ≫

```
? PermutationReplace
```

PermutationReplace[*expr*, *perm*] replaces each part in *expr* by its image under the permutation *perm*.

PermutationReplace[*expr*, *gr*] returns the list of images of *expr* under all elements of the permutation group *gr*. ≫