Inflatable mathematics

David Vogan

Sophus Lie Days, Cornell, April 27, 2008

Outline

The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.

The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.

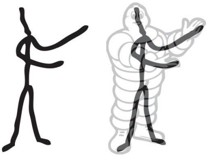
The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.



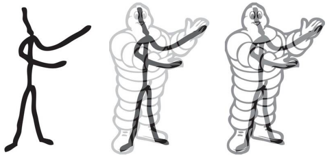
The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.



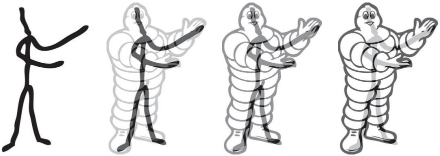
The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.



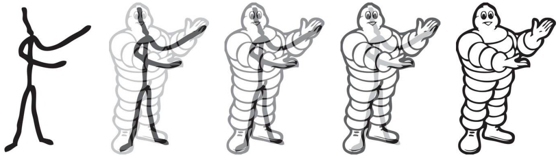
The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.



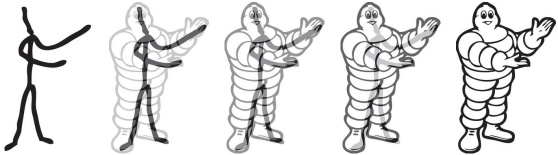
The main idea

Begin with linear algebra: solving systems of linear equations by Gaussian elimination.

Idea: reduce number of coordinates by one.

Relate to geometry: arranging lines and planes.

Idea: reduce to geometry of one dimension less.



Use same idea for more complicated geometry.

Gaussian elimination: easy cases

System of three equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & c_1 \\ a_{21}\,x_1 & + & a_{22}x_2 & + & a_{23}x_3 & = & c_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & c_3 \end{array}$$

I'll assume always the system has just one solution.

Gaussian elimination: easy cases

System of three equations in three unknowns is

$$\begin{aligned} a_{11}x_1 \qquad\qquad\qquad\qquad &= c_1 \\ a_{22}x_2 \qquad\qquad &= c_2 \\ a_{33}x_3 &= c_3 \end{aligned}$$

I'll assume always the system has just one solution.

Easiest case is diagonal system: divide each equation by a constant to solve.

Gaussian elimination: easy cases

System of three equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & & & & & = & c_1 \\ a_{21}\,x_1 & + & a_{22}x_2 & & & = & c_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & c_3 \end{array}$$

I'll assume always the system has just one solution.

Easiest case is diagonal system: divide each equation by a constant to solve.

Next easiest is lower triangular: add multiples of some eqns to later ones to make diagonal.

Gaussian elimination: easy cases

System of three equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & & & = & c_1 \\ a_{21}\,x_1 & + & a_{22}x_2 & & & = & c_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & c_3 \end{array}$$

I'll assume always the system has just one solution.

Easiest case is diagonal system: divide each equation by a constant to solve.

Next easiest is lower triangular: add multiples of some eqns to later ones to make diagonal.

Suppose lower triangular EXCEPT one coefficient $a_{12} \neq 0$. Add multiple of 1st eqn to second to get. . .

Gaussian elimination: easy cases

System of three equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & & & = & c_1 \\ a_{21}'x_1 & & & & & = & c_2' \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & c_3 \end{array}$$

I'll assume always the system has just one solution.

Easiest case is diagonal system: divide each equation by a constant to solve.

Next easiest is lower triangular: add multiples of some eqns to later ones to make diagonal.

Suppose lower triangular EXCEPT one coefficient $a_{12} \neq 0$. Add multiple of 1st eqn to second to get. . .

This system is nearly lower triangular, except that the first two equations are interchanged.

Gaussian elimination: typical case

"Typical" system of equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & c_1 \\ a_{21}\,x_1 & + & a_{22}\,x_2 & + & a_{23}x_3 & = & c_2 \\ a_{31}\,\,x_1 & + & a_{32}\,x_2 & + & a_{33}x_3 & = & c_3 \end{array}$$

where "typically" $a_{13} \neq 0$. Add multiple of 1st equation to each later eqn to get...

Gaussian elimination: typical case

"Typical" system of equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & c_1 \\ {a_{21}}'x_1 & + & {a_{22}}'x_2 & & & = & {c_2}' \\ {a_{31}}'\,x_1 & + & {a_{32}}'x_2 & & & = & {c_3}' \end{array}$$

where "typically" $a_{13} \neq 0$. Add multiple of 1st equation to each later eqn to get...

Now "typically" $a'_{22} \neq 0$. Add multiple of 2nd eqn to last to get...

Gaussian elimination: typical case

"Typical" system of equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & c_1 \\ a_{21}{}'x_1 & + & a_{22}{}'x_2 & & & = & c_2{}' \\ a_{31}{}''x_1 & & & & & = & c_3{}'' \end{array}$$

where "typically" $a_{13} \neq 0$. Add multiple of 1st equation to each later eqn to get...

Now "typically" $a'_{22} \neq 0$. Add multiple of 2nd eqn to last to get...

Again this last system is nearly lower triangular, except that order of the three eqns is reversed.

Gaussian elimination: typical case

"Typical" system of equations in three unknowns is

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & c_1 \\ a_{21}{}'x_1 & + & a_{22}{}'x_2 & & & = & c_2{}' \\ a_{31}{}''x_1 & & & & & = & c_3{}'' \end{array}$$

where "typically" $a_{13} \neq 0$. Add multiple of 1st equation to each later eqn to get...

Now "typically" $a'_{22} \neq 0$. Add multiple of 2nd eqn to last to get...

Again this last system is nearly lower triangular, except that order of the three eqns is reversed.

To say what happens in general, use matrix notation $A\mathbf{x} = \mathbf{c}$. Here $A = (a_{ij})$ is $n \times n$ coeff matrix, and $\mathbf{x} = (x_j)$ is the column vector of n unknowns.

Theorem for Gaussian elimination

Theorem

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ is an n-tuple of constants. Consider the system of n equations in n unknowns

$$A\mathbf{x} = \mathbf{c}.$$

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ is an n-tuple of constants. Consider the system of n equations in n unknowns

$$A\mathbf{x} = \mathbf{c}.$$

Using the two operations

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ *is an n-tuple of constants. Consider the system of n equations in n unknowns*

$$A\mathbf{x} = \mathbf{c}.$$

Using the two operations

1. *dividing an equation by a non-zero constant, and*

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ is an n-tuple of constants. Consider the system of n equations in n unknowns

$$A\mathbf{x} = \mathbf{c}.$$

Using the two operations

1. *dividing an equation by a non-zero constant, and*
2. *adding a multiple of one equation to a later one,*

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ is an n-tuple of constants. Consider the system of n equations in n unknowns

$$A\mathbf{x} = \mathbf{c}.$$

Using the two operations

1. *dividing an equation by a non-zero constant, and*
2. *adding a multiple of one equation to a later one,*

we can transform this system into a new one

$$A'\mathbf{x} = \mathbf{c}'.$$

Theorem for Gaussian elimination

Theorem

Suppose A is an invertible $n \times n$ matrix, and $\mathbf{c}$ *is an n-tuple of constants. Consider the system of n equations in n unknowns*

$$A\mathbf{x} = \mathbf{c}.$$

Using the two operations

1. *dividing an equation by a non-zero constant, and*
2. *adding a multiple of one equation to a later one,*

we can transform this system into a new one

$$A'\mathbf{x} = \mathbf{c}'.$$

The new system, after reordering the equations, is lower triangular.

Possibilities for three unknowns

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$a_{12}=a_{23}=a_{13}=0$$

Possibilities for three unknowns

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (123)$$

$$a_{12} = a_{23} = a_{13} = 0$$

Possibilities for three unknowns

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213)$$

$a_{13}{=}a_{23}{=}0,\ a_{12}{\neq}0$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$a_{12}{=}a_{13}{=}0,\ a_{23}{\neq}0$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$a_{12}{=}a_{23}{=}a_{13}{=}0$

Possibilities for three unknowns

$\uparrow$ $\uparrow$

$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (213) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix}$ (132)

$a_{13}{=}a_{23}{=}0,\ a_{12}{\neq}0$ $a_{12}{=}a_{13}{=}0,\ a_{23}{\neq}0$

$\nwarrow$ $\nearrow$

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (123)

$a_{12}{=}a_{23}{=}a_{13}{=}0$

Possibilities for three unknowns

$$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (312)$$

$$a_{13}=0,\ a_{12}\neq 0,\ a_{23}\neq 0$$

$$\begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (231)$$

$$\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}=0,\ a_{13}\neq 0$$

$$\uparrow \qquad\qquad \uparrow$$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213)$$

$$a_{13}=a_{23}=0,\ a_{12}\neq 0$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$a_{12}=a_{13}=0,\ a_{23}\neq 0$$

$$\nwarrow \qquad\qquad \nearrow$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$a_{12}=a_{23}=a_{13}=0$$

Possibilities for three unknowns

$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ (312)

$a_{13}=0,\ a_{12}\neq 0,\ a_{23}\neq 0$

$\begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ (231)

$\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}=0,\ a_{13}\neq 0$

$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (213)

$a_{13}=a_{23}=0,\ a_{12}\neq 0$

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix}$ (132)

$a_{12}=a_{13}=0,\ a_{23}\neq 0$

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (123)

$a_{12}=a_{23}=a_{13}=0$

Possibilities for three unknowns

$$\begin{pmatrix} * & * & 1 \\ * & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad (321)$$

$$a_{13} \neq 0, \quad \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \neq 0$$

$$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (312)$$

$$a_{13} = 0, \quad a_{12} \neq 0, \quad a_{23} \neq 0$$

$$\begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (231)$$

$$\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} = 0, \quad a_{13} \neq 0$$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213)$$

$$a_{13} = a_{23} = 0, \quad a_{12} \neq 0$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$a_{12} = a_{13} = 0, \quad a_{23} \neq 0$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

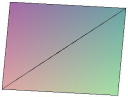
$$a_{12} = a_{23} = a_{13} = 0$$

From algebra to geometry

A flag in 3 dimensions is a (straight) line through the origin, contained inside a plane through the origin:

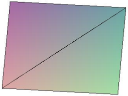
From algebra to geometry

A flag in 3 dimensions is a (straight) line through the origin, contained inside a plane through the origin:



From algebra to geometry

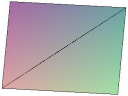
A flag in 3 dimensions is a (straight) line through the origin, contained inside a plane through the origin:



One flag not so interesting. What's interesting is how many different flags there are, and how they're related.

From algebra to geometry

A flag in 3 dimensions is a (straight) line through the origin, contained inside a plane through the origin:

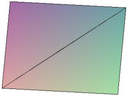


One flag not so interesting. What's interesting is how many different flags there are, and how they're related.

System of equations $=$ 3×3 matrix $\rightsquigarrow$ flag:
line $=$ multiples of first row, plane $=$ span of first two rows.

From algebra to geometry

A flag in 3 dimensions is a (straight) line through the origin, contained inside a plane through the origin:



One flag not so interesting. What's interesting is how many different flags there are, and how they're related.

System of equations $=$ 3×3 matrix $\rightsquigarrow$ flag:
line $=$ multiples of first row, plane $=$ span of first two rows.

Two matrices give same flag if and only if differ by

- *multiply row by constant*
- *add multiple of one row to later row.*

Possible flags $L \subset P$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$L_x \subset P_{xy}$

Possible flags $L \subset P$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$L_x \subset P_{xy}$$

Possible flags $L \subset P$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213) \qquad\qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$L_x \neq L \subset P_{xy} \qquad\qquad L_x \subset P \neq P_{xy}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$L_x \subset P_{xy}$$

Possible flags $L \subset P$

$\uparrow$ $\qquad\qquad\qquad$ $\uparrow$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213) \qquad\qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$L_x \neq L \subset P_{xy}$ $\qquad\qquad$ $L_x \subset P \neq P_{xy}$

$\nwarrow$ $\qquad\qquad$ $\nearrow$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$L_x \subset P_{xy}$

Possible flags $L \subset P$

$$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (312) \qquad\qquad \begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (231)$$

$$P_{xy} \supset L \subset P' \not\supset L_x \qquad\qquad P_{xy} \not\supset L' \subset P \supset L_x$$

$$\uparrow \qquad\qquad \uparrow$$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213) \qquad\qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$L_x \neq L \subset P_{xy} \qquad\qquad L_x \subset P \neq P_{xy}$$

$$\nwarrow \qquad\qquad \nearrow$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$L_x \subset P_{xy}$$

Possible flags $L \subset P$

$$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (312)$$

$$P_{xy} \supset L \subset P' \not\supset L_x$$

$$\begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (231)$$

$$P_{xy} \not\supset L' \subset P \supset L_x$$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213)$$

$$L_x \neq L \subset P_{xy}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$L_x \subset P \neq P_{xy}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$L_x \subset P_{xy}$$

Possible flags $L \subset P$

$$\begin{pmatrix} * & * & 1 \\ * & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad (321)$$

$$P_{xy} \not\supset L' \subset P' \not\supset L_x$$

$$\begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (312)$$

$$P_{xy} \supset L \subset P' \not\supset L_x$$

$$\begin{pmatrix} * & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (231)$$

$$P_{xy} \not\supset L' \subset P \supset L_x$$

$$\begin{pmatrix} * & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (213)$$

$$L_x \neq L \subset P_{xy}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (132)$$

$$L_x \subset P \neq P_{xy}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (123)$$

$$L_x \subset P_{xy}$$

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

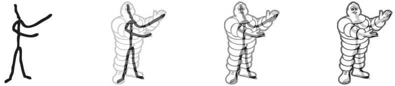
Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.



Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.

$L_x \subset P_{xy}$

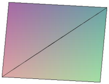
Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L$ $\rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.



$L_x \subset P_{xy}$

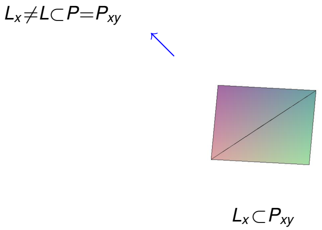
Geometric picture

Moving up $\rightsquigarrow$ more complicated geometry.

up one blue step: fixed line $\rightsquigarrow$ variable line in a plane.

up one red step: fixed plane $\supset L \rightsquigarrow$ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.



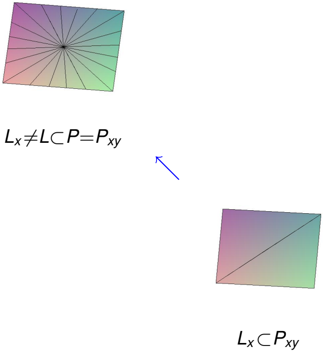
Geometric picture

Moving up ⇝ more complicated geometry.

up one blue step: fixed line ⇝ variable line in a plane.

up one red step: fixed plane ⊃ L ⇝ variable plane ⊃ L.

Inflatable mathematics: replace points by circles.



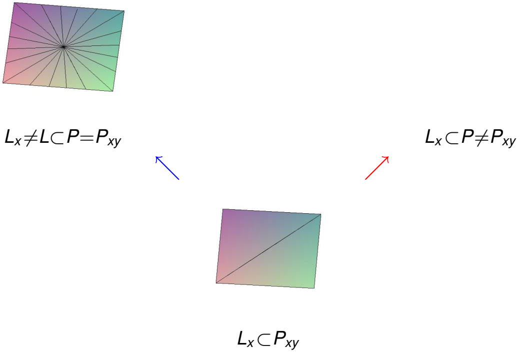
Geometric picture

Moving up ⇝ more complicated geometry.

up one blue step: fixed line ⇝ variable line in a plane.

up one red step: fixed plane ⊃ L ⇝ variable plane ⊃ L.

Inflatable mathematics: replace points by circles.



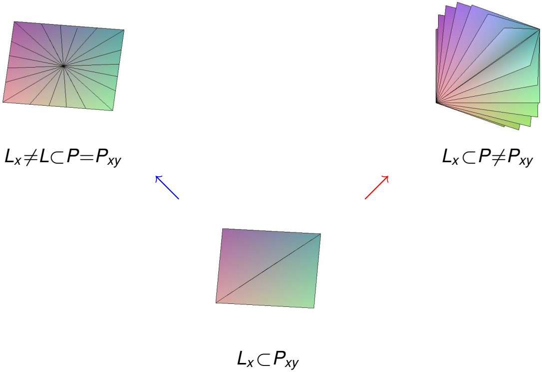
Geometric picture

Moving up ⇝ more complicated geometry.

up one blue step: fixed line ⇝ variable line in a plane.

up one red step: fixed plane $\supset L$ ⇝ variable plane $\supset L$.

Inflatable mathematics: replace points by circles.



What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:

What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top. . .

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:

$L' \subset P', \; L \subset P' \cap P_{xy}$ $\qquad$ $L' \subset P', \; P \supset L' + L_x$

$P_{xy} \not\supset L' \subset P' \not\supset L_x$

$\uparrow L \rightsquigarrow L' \subset P'$ $\qquad$ $\uparrow P \rightsquigarrow P' \supset L'$

$P_{xy} \supset L \subset P' \not\supset L_x$ $\qquad$ $P_{xy} \not\supset L' \subset P \supset L_x$

$\uparrow P_{xy} \rightsquigarrow P' \supset L$ $\qquad$ $L \rightsquigarrow L' \subset P \uparrow$

$L_x \neq L \subset P_{xy}$ $\qquad$ $L_x \subset P \neq P_{xy}$

$\nwarrow L_x \rightsquigarrow L \subset P_{xy}$ $\qquad$ $\nearrow P_{xy} \rightsquigarrow P \supset L_x$

$L_x \subset P_{xy}$

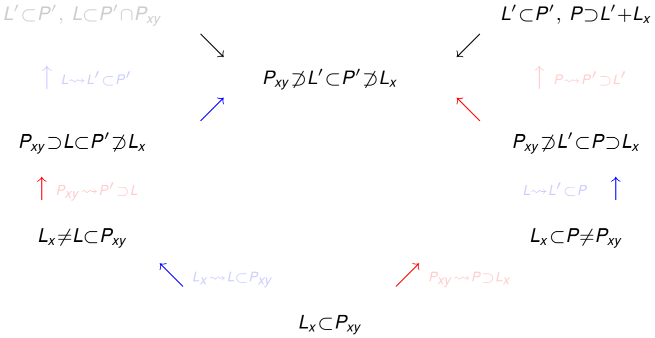
What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:



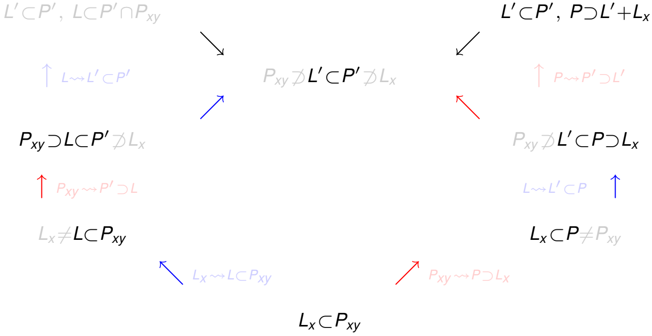
What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:



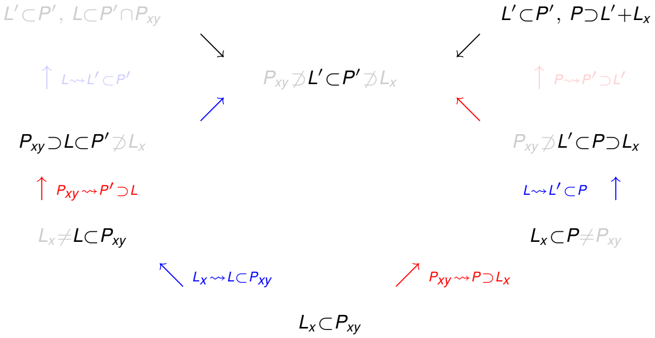
What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:



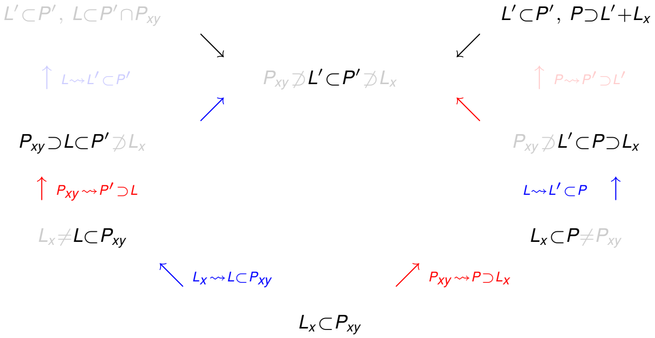
What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:



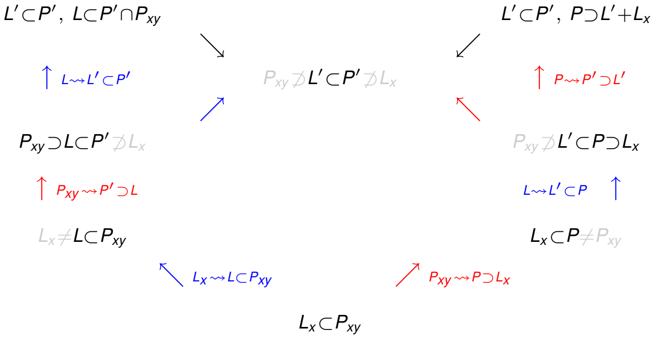
What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

What's a Schubert variety?

Divided flags (in three dimensions) into six "Bruhat cells" by relation with standard flag $L_x \subset P_{xy}$.

Schubert variety is one cell and everything below it:



What's almost true: each Schubert variety "inflated" from a smaller one, replacing each point by a circle.

Fails only at the top...

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:

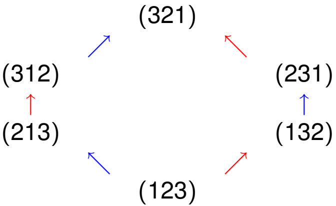
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \dots 2$: blue arrow up.
3. Exchange $2 \dots 3$: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



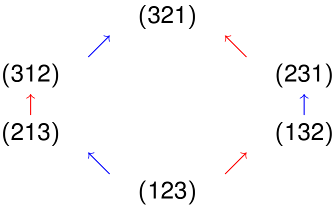
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \dots 2$: blue arrow up.
3. Exchange $2 \dots 3$: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



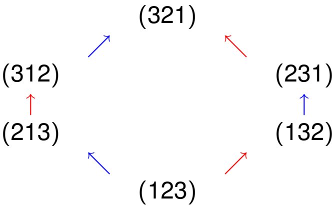
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \ldots 2$: blue arrow up.
3. Exchange $2 \ldots 3$: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



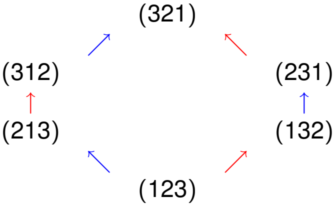
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \dots 2$: blue arrow up.
3. Exchange $2 \dots 3$: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



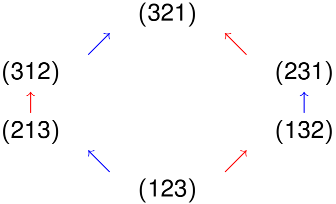
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange 1 . . . 2: blue arrow up.
3. Exchange 2 . . . 3: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



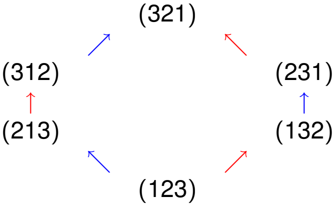
Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \dots 2$: blue arrow up.
3. Exchange $2 \dots 3$: red arrow up.

Mathematics on a need-to-know basis

To compute with Schubert varieties, need only arrangement of blue and red arrows, describing how small Schubert varieties are inflated:



Permutations recorded which rows had pivots in Gaussian elimination. Now they're just symbols.

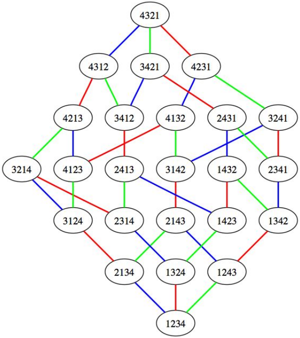
Rules for making diagram:

1. One entry for each permutation of $\{1, 2, 3\}$.
2. Exchange $1 \dots 2$: blue arrow up.
3. Exchange $2 \dots 3$: red arrow up.

As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.



Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play…

height of a permutation $= \#\{$ pairs (i,j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$$

Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

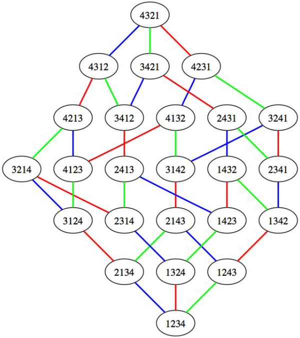
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play…



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

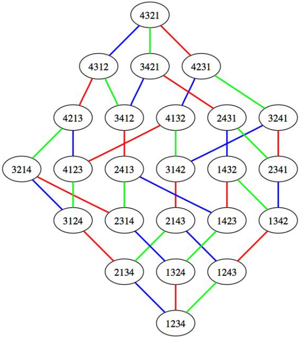
$$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.



Counting problems in this picture ⇝ geometry of Schubert varieties.

There are lots of counting games to play…

height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

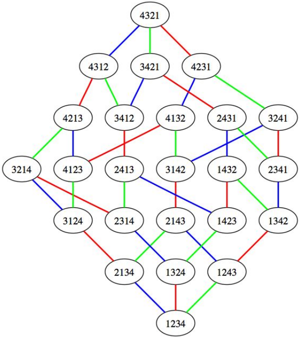
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture ⇝ geometry of Schubert varieties.

There are lots of counting games to play. . .



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

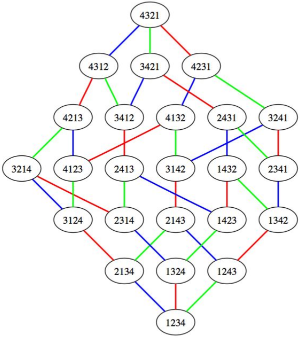
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play...



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

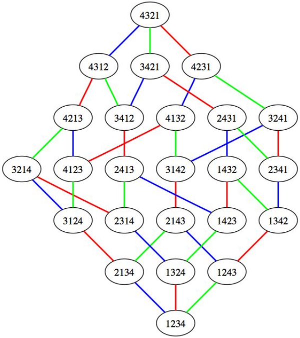
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play...



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

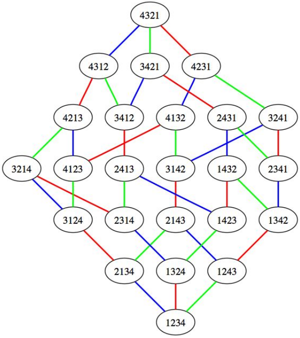
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play...



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\} =$ coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\} =$

$$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$$ Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

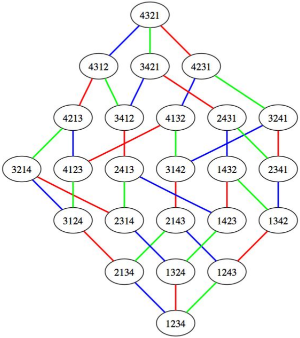
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play...



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\}$ = coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\}$ =

$$\binom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$$

Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

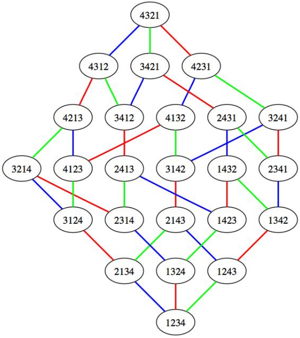
As many dimensions as you want

Rules in n dimensions:

1. One entry for each permutation of $\{1, 2, \ldots, n\}$.
2. Exchange $i \ldots i+1$: arrow up of color i.

Counting problems in this picture $\leftrightsquigarrow$ geometry of Schubert varieties.

There are lots of counting games to play...



height of a permutation $= \#\{$ pairs (i, j) out of order $\}$.

$\#\{$permutations at height $d\} =$ coefficient of x^d in polynomial

$$(1)(1+x)(1+x+x^2)\cdots(1+x+\cdots x^{n-1}).$$

$\#\{$ascending paths bottom to top$\} =$

$$\tbinom{n}{2}!/1^{n-1}3^{n-2}5^{n-2}\cdots(2n-5)^2(2n-3)$$

Stanley's formula

(Formula says 16 ascending paths bottom to top in this picture.)

More complicated groups

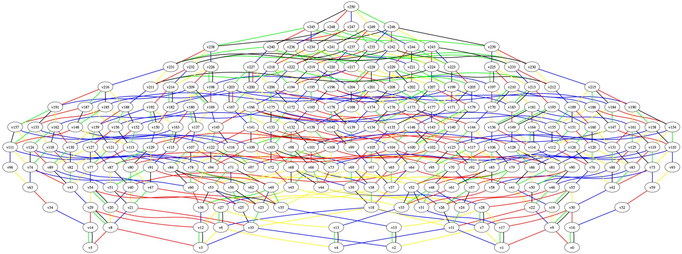
Picture just described (with $n!$ vertices) is for invertible $n \times n$ matrices. This is the basic example of a real reductive Lie group. Mathematicians and physicists look at lots of other reductive groups.

Each reductive group has a finite diagram describing how its big Schubert varieties are "inflated" from smaller ones. This one is for a 45-dimensional group called $SO(5,5)$.

For this group there are 251 Schubert varieties, but each arrow still means replace points by circles.

More complicated groups

Picture just described (with $n!$ vertices) is for invertible $n \times n$ matrices. This is the basic example of a real reductive Lie group. Mathematicians and physicists look at lots of other reductive groups.



Each reductive group has a finite diagram describing how its big Schubert varieties are “inflated” from smaller ones. This one is for a 45-dimensional group called $SO(5,5)$.

For this group there are 251 Schubert varieties, but each arrow still means replace points by circles.

More complicated groups

Picture just described (with $n!$ vertices) is for invertible $n \times n$ matrices. This is the basic example of a real reductive Lie group. Mathematicians and physicists look at lots of other reductive groups.

closeup view

Each reductive group has a finite diagram describing how its big Schubert varieties are “inflated” from smaller ones. This one is for a 45-dimensional group called $SO(5,5)$.

For this group there are 251 Schubert varieties, but each arrow still means replace points by circles.

More complicated groups

Picture just described (with $n!$ vertices) is for invertible $n \times n$ matrices. This is the basic example of a real reductive Lie group. Mathematicians and physicists look at lots of other reductive groups.

closeup view

Each reductive group has a finite diagram describing how its big Schubert varieties are “inflated” from smaller ones. This one is for a 45-dimensional group called $SO(5,5)$.

For this group there are 251 Schubert varieties, but each arrow still means replace points by circles.

More complicated groups

Picture just described (with $n!$ vertices) is for invertible $n \times n$ matrices. This is the basic example of a real reductive Lie group. Mathematicians and physicists look at lots of other reductive groups.

closeup view

Each reductive group has a finite diagram describing how its big Schubert varieties are “inflated” from smaller ones. This one is for a 45-dimensional group called $SO(5,5)$.

For this group there are 251 Schubert varieties, but each arrow still means replace points by circles.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n) \leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group $G \leftrightsquigarrow$ Schubert varieties for $G \leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n)$ $\leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group G $\leftrightsquigarrow$ Schubert varieties for G $\leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n)$ $\leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group G $\leftrightsquigarrow$ Schubert varieties for G $\leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n) \leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group $G \leftrightsquigarrow$ Schubert varieties for $G \leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n)$ $\leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group G $\leftrightsquigarrow$ Schubert varieties for G $\leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

What do you do with the pretty pictures?

Where we started:

systems of n linear eqns $\overset{\text{Gauss elim}}{\leftrightsquigarrow}$ group $GL(n)$ $\leftrightsquigarrow$ Schubert varieties $\leftrightsquigarrow$ graph with $n!$ vertices, arrows of $n-1$ colors.

Graph tells what cases can happen during Gaussian elimination; how Gaussian elimination changes with the system of equations; even which cases are most common.

Similarly:

math or physics problem $\overset{\text{repn theory}}{\leftrightsquigarrow}$ reductive group G $\leftrightsquigarrow$ Schubert varieties for G $\leftrightsquigarrow$ finite graph for inflating.

1979: David Kazhdan (Harvard) and George Lusztig (MIT) showed how to answer questions about representation theory by calculating in the finite graph.

Defined Kazhdan-Lusztig polynomial $P_{x,y}$ for x and y in the graph. Polynomial in q, non-neg integer coeffs.

Polynomial is non-zero only if y is above x in graph. Calculated by a recursion based on knowing all $P_{x',y'}$ for y' smaller than y.

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.
- Seek line up x (to x') same color as some line down y (to y').

  If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

  If not, (x, y) is primitive: no color down from y goes up from x.
- One hard calculation for each primitive pair (x, y).

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.
- Seek line up x (to x') same color as some line down y (to y').

  If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

  If not, (x, y) is primitive: no color down from y goes up from x.
- One hard calculation for each primitive pair (x, y).

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.

x'

- Seek line up x same color as some line down y.

y'

If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

If not, (x, y) is primitive: no color down from y goes up from x.

- One hard calculation for each primitive pair (x, y).

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.
- Seek line up $\overset{x'}{\underset{x}{|}}$ same color as some line down $\underset{y'}{\overset{y}{|}}$.

  If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

  If not, (x, y) is primitive: no color down from y goes up from x.
- One hard calculation for each primitive pair (x, y).

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.
- Seek line up x (to x') same color as some line down y (to y').

If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

If not, (x, y) is primitive: no color down from y goes up from x.

- One hard calculation for each primitive pair (x, y).

How the computation works

Now fixing a reductive group G and its graph of Schubert varieties.

- For each pair (x, y) of graph vertices, want to compute KL polynomial $P_{x,y}$.
- Induction: start with y's on bottom of graph, work up. For each y, start with $x = y$, work down.
- Seek line up $\overset{x'}{\overset{|}{x}}$ same color as some line down $\underset{y'}{\underset{|}{y}}$.

  If it's there, then $P_{x,y} = P_{x',y}$ (known by induction).

  If not, (x, y) is primitive: no color down from y goes up from x.
- One hard calculation for each primitive pair (x, y).

What to do for primitive pair (x, y)

- graph vertex $y \rightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \rightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.
- Pick line down y (line to y'); means $F_y \approx$ inflated from $F_{y'}$.
- Primitive means red line x (line to x') is also down from x.
- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \le z < y'} \mu(z, y') q^{(l(y')-l(z)-1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

$P_{x,y}$ describes how F_y looks near F_x.

- Pick line down $\begin{array}{c} y \\ | \\ y' \end{array}$; means $F_y \approx$ inflated from $F_{y'}$.
- Primitive means red line $\begin{array}{c} x \\ | \\ x' \end{array}$ is also down from x.
- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \leq z < y'} \mu(z, y') q^{(l(y') - l(z) - 1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.

- Pick line down $\begin{array}{c} y \\ | \\ y' \end{array}$; means $F_y \approx$ inflated from $F_{y'}$.
- Primitive means red line $\begin{array}{c} x \\ | \\ x' \end{array}$ is also down from x.
- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \leq z < y'} \mu(z, y') q^{(l(y')-l(z)-1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.

- Pick line down $\begin{array}{c} y \\ | \\ y' \end{array}$; means $F_y \approx$ inflated from $F_{y'}$.
- Primitive means red line $\begin{array}{c} x \\ | \\ x' \end{array}$ is also down from x.
- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \leq z < y'} \mu(z, y')q^{(l(y')-l(z)-1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.

- Pick line down y ; means $F_y \approx$ inflated from $F_{y'}$.

  y'

- Primitive means red line x is also down from x.

  x'

- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \leq z < y'} \mu(z, y') q^{(l(y')-l(z)-1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.
- Pick line down $\begin{array}{c} y \\ | \\ y' \end{array}$; means $F_y \approx$ inflated from $F_{y'}$.
- Primitive means red line $\begin{array}{c} x \\ | \\ x' \end{array}$ is also down from x.
- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \le z < y'} \mu(z, y') q^{(l(y')-l(z)-1)/2} P_{x',z}.$$

What to do for primitive pair (x, y)

- graph vertex $y \leftrightsquigarrow$ big Schubert variety F_y.
- lower vertex $x \leftrightsquigarrow$ little Schubert variety F_x.

  $P_{x,y}$ describes how F_y looks near F_x.

- Pick line down $\begin{array}{c} y \\ | \\ y' \end{array}$; means $F_y \approx$ inflated from $F_{y'}$.

- Primitive means red line $\begin{array}{c} x \\ | \\ x' \end{array}$ is also down from x.

- Geometry translates to algebra $P_{x,y} \approx P_{x',y'} + qP_{x,y'}$. Precisely:

$$P_{x,y} = P_{x',y'} + qP_{x,y'} - \sum_{x' \leq z < y'} \mu(z, y') q^{(l(y') - l(z) - 1)/2} P_{x',z}.$$

Forming the *Atlas* group

Between 1980 and 2000, increasingly sophisticated computer programs calculated special kinds of Kazhdan-Lusztig polynomials; none dealt with the complications attached to general real reductive groups.

In 2001, Jeff Adams at University of Maryland decided computers and mathematics had advanced far enough to begin interesting work in that direction.

Adams formed a research group *Atlas of Lie groups and representations*, aimed in part at producing software to make old mathematics widely accessible, and to find new mathematics.

A first goal was to calculate Kazhdan-Lusztig polynomials for real reductive groups.

Forming the *Atlas* group

Between 1980 and 2000, increasingly sophisticated computer programs calculated special kinds of Kazhdan-Lusztig polynomials; none dealt with the complications attached to general real reductive groups.

In 2001, Jeff Adams at University of Maryland decided computers and mathematics had advanced far enough to begin interesting work in that direction.

Adams formed a research group *Atlas of Lie groups and representations*, aimed in part at producing software to make old mathematics widely accessible, and to find new mathematics.

A first goal was to calculate Kazhdan-Lusztig polynomials for real reductive groups.

Forming the *Atlas* group

Between 1980 and 2000, increasingly sophisticated computer programs calculated special kinds of Kazhdan-Lusztig polynomials; none dealt with the complications attached to general real reductive groups.

In 2001, Jeff Adams at University of Maryland decided computers and mathematics had advanced far enough to begin interesting work in that direction.

Adams formed a research group *Atlas of Lie groups and representations*, aimed in part at producing software to make old mathematics widely accessible, and to find new mathematics.

A first goal was to calculate Kazhdan-Lusztig polynomials for real reductive groups.

Forming the *Atlas* group

Between 1980 and 2000, increasingly sophisticated computer programs calculated special kinds of Kazhdan-Lusztig polynomials; none dealt with the complications attached to general real reductive groups.

In 2001, Jeff Adams at University of Maryland decided computers and mathematics had advanced far enough to begin interesting work in that direction.

Adams formed a research group *Atlas of Lie groups and representations*, aimed in part at producing software to make old mathematics widely accessible, and to find new mathematics.

A first goal was to calculate Kazhdan-Lusztig polynomials for real reductive groups.

How do you make a computer do that?

- In June 2002, Jeff Adams asked Fokko du Cloux.
- In November 2005, Fokko finished the program.

  Wasn't that easy?

- In 2006, Jeff Adams proposed to calculate KL polynomials for the 248-dimensional *exceptional Lie group* E_8.

How do you make a computer do that?

- In June 2002, Jeff Adams asked Fokko du Cloux.
- In November 2005, Fokko finished the program.

  Wasn't that easy?

- In 2006, Jeff Adams proposed to calculate KL polynomials for the 248-dimensional *exceptional Lie group* E_8.

How do you make a computer do that?

- In June 2002, Jeff Adams asked Fokko du Cloux.
- In November 2005, Fokko finished the program.

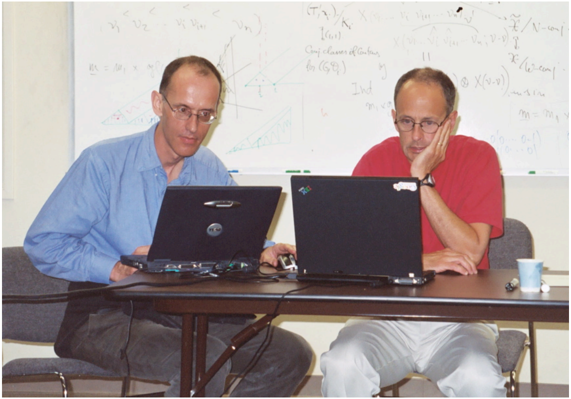
  Wasn't that easy?

- In 2006, Jeff Adams proposed to calculate KL polynomials for the 248-dimensional *exceptional Lie group* E_8.

How do you make a computer do that?

- In June 2002, Jeff Adams asked Fokko du Cloux.
- In November 2005, Fokko finished the program.

  Wasn't that easy?

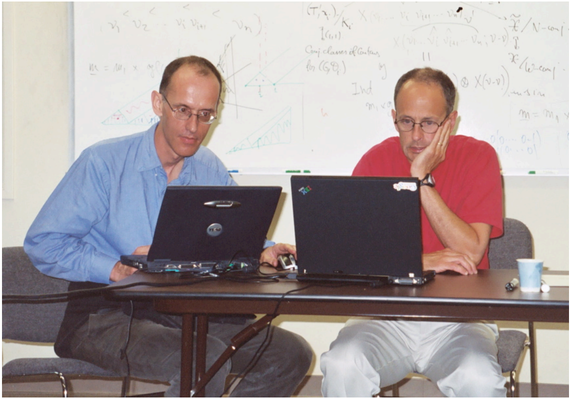


- In 2006, Jeff Adams proposed to calculate KL polynomials for the 248-dimensional *exceptional Lie group* E_8.

How do you make a computer do that?

- In June 2002, Jeff Adams asked Fokko du Cloux.
- In November 2005, Fokko finished the program.

  Wasn't that easy?



- In 2006, Jeff Adams proposed to calculate KL polynomials for the 248-dimensional *exceptional Lie group* E_8.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops ×6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys ⇝ 75G total RAM.
Feared 1 billion ⇝ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4\ \frac{\text{bytes}}{\text{coef}} \times 20\ \frac{\text{coefs}}{\text{poly}} \times\ ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys ⇝ 75G total RAM.
Feared 1 billion ⇝ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys ⇝ 75G total RAM.
Feared 1 billion ⇝ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys ⇝ 75G total RAM.
Feared 1 billion ⇝ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$\frac{4 \text{ bytes}}{\text{coef}} \times \frac{20 \text{ coefs}}{\text{poly}} \times$?? polys RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys $\rightsquigarrow$ 75G total RAM.
Feared 1 billion $\rightsquigarrow$ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??$ polys RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.
Hoped 400 million polys $\rightsquigarrow$ 75G total RAM.
Feared 1 billion $\rightsquigarrow$ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.

Hoped 400 million polys ⇝ 75G total RAM.

Feared 1 billion ⇝ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??$ polys RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.

Hoped 400 million polys $\rightsquigarrow$ 75G total RAM.

Feared 1 billion $\rightsquigarrow$ 150G total RAM.

What's the computer have to do?

TASK	COMPUTER RQMT
Make graph: 453,060 nodes, 8 edges at each	250M RAM, 10 minutes (latest software: thirty seconds)
List primitive pairs of vertices: 6,083,626,944	450M RAM, few seconds
Calculate polynomial for each primitive pair	Fetch few kB from memory, few thousand integer ops $\times$6 billion
Look for polynomial in store, add if it's new	$4 \frac{\text{bytes}}{\text{coef}} \times 20 \frac{\text{coefs}}{\text{poly}} \times ??\ \text{polys}$ RAM
Write number for poly in table	25G RAM

Big unknown: number of distinct polynomials.

Hoped 400 million polys $\rightsquigarrow$ 75G total RAM.

Feared 1 billion $\rightsquigarrow$ 150G total RAM.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  64435824 kB
```

Writing to disk took two days. Investigating why ⇝ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\left(\substack{\text{modular}\\ \text{arithmetic}}\right)} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why ⇝ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why $\rightsquigarrow$ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why ⇝ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why $\rightsquigarrow$ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one. . .

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why ⇝ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why $\rightsquigarrow$ output bug, so mod 251 answers no good.

Saga of the end times

11/06 Experiments by Birne Binegar on William Stein's computer `sage` showed we needed 150G.

11/28/06 Asked about pure math uses for 256G computer.

11/30/06 Noam Elkies told us we didn't need one...

$$\text{one 150G computation} \xrightarrow{\binom{\text{modular}}{\text{arithmetic}}} \text{four 50G computations}$$

12/03/06 Marc van Leeuwen made Fokko's code modular.

12/19/06 mod 251 computation on `sage`. Took 17 hours:

```
Total elapsed time = 62575s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   64435824 kB
```

Writing to disk took two days. Investigating why $\rightsquigarrow$ output bug, so mod 251 answers no good.

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on `sage`. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then `sage` crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  54995416 kB
```

12/23/06 Started mod 255 computation on `sage`, which crashed.

`sage` down til 12/26/06 (regional holiday in Seattle).

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on `sage`. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then `sage` crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  54995416 kB
```

12/23/06 Started mod 255 computation on `sage`, which crashed.

`sage` down til 12/26/06 (regional holiday in Seattle).

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on `sage`. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then `sage` crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  54995416 kB
```

12/23/06 Started mod 255 computation on `sage`, which crashed.

`sage` down til 12/26/06 (regional holiday in Seattle).

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on `sage`. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then `sage` crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:   54995416 kB
```

12/23/06 Started mod 255 computation on `sage`, which crashed.

`sage` down til 12/26/06 (regional holiday in Seattle).

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on sage. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then sage crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  54995416 kB
```

12/23/06 Started mod 255 computation on sage, which crashed.

sage down til 12/26/06 (regional holiday in Seattle).

The Tribulation (continued)

12/21/06 9 P.M. Started mod 256 computation on `sage`. Computed 452,174 out of 453,060 rows of KL polynomials in 14 hours, then `sage` crashed.

12/22/06 EVENING Restarted mod 256. Finished in just 11 hours

(hip, hip, HURRAH! hip, hip, HURRAH! `pthread_join(cheer[k], NULL);`):

```
Total elapsed time = 40229s.  Finished at l = 64, y = 453059
d_store.size() = 1181642979, prim_size = 3393819659

VmData:  54995416 kB
```

12/23/06 Started mod 255 computation on `sage`, which crashed.

`sage` down til 12/26/06 (regional holiday in Seattle).

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod 253·255·256 = 16,515,840.

One little computation for each of 13 billion coefficients.

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod 253·255·256 = 16,515,840.

One little computation for each of 13 billion coefficients.

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

So we've got mod 256...

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

So we've got mod 256...

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

So we've got mod 256. . .

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts ⇝ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older ⇝ thirty years wiser as team ⇝ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod 253·255·256 = 16,515,840.

One little computation for each of 13 billion coefficients.

So we've got mod 256...

12/26/06 sage rebooted. Wrote KL polynomials mod 255.

12/27/06 Started computation mod 253. Halfway, sage crashed.

consult experts $\rightsquigarrow$ probably not Sasquatch.

Did I mention sage is in Seattle?

Decided not to abuse sage further for a year.

1/3/07 Atlas members one year older $\rightsquigarrow$ thirty years wiser as team $\rightsquigarrow$ safe to go back to work.

Wrote KL polynomials mod 253 (12 hrs).

Now we had answers mod 253, 255, 256.

Chinese Remainder Theorem (CRT)

gives answer mod $253 \cdot 255 \cdot 256 = 16{,}515{,}840$.

One little computation for each of 13 billion coefficients.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-started CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

The Chinese Remainder

1/4/07 Marc van Leeuwen started his CRT software. On-screen counter displayed polynomial number: 0,1,2,3,...,1181642978. Turns out to be a bad idea.

1/5/07 MORNING Restarted CRT computation, with counter 0,4096,8192,12288,16536,...,1181642752,1181642978. Worked fine until `sage` crashed.

William Stein (our hero!) replaced hard drive with one with backups of our 100G of files mod 253, 255, 256.

1/5/07 AFTERNOON Re-restarted CRT computation.

1/6/07 7 A.M. Output file 7G too big: BUG in output routine.

1/7/07 2 A.M. Marc found output bug. Occurred only after polynomial 858,993,459; had tested to 100 million.

1/7/07 6 A.M. Re-re-restarted CRT computation.

In Which we Come to an Enchanted Place…

1/8/07 9 A.M. Finished writing to disk the KL polynomials for E_8.

In Which we Come to an Enchanted Place. . .

1/8/07 9 A.M. Finished writing to disk the KL polynomials for E_8.

So what was the point?

In the fall of 2004, Fokko du Cloux was at MIT, rooming with fellow Atlas member Dan Ciubotaru. Fokko was halfway through writing the software I've talked about: the point at which neither the end of the tunnel nor the beginning is visible any longer.

In Which we Come to an Enchanted Place...

1/8/07 9 A.M. Finished writing to disk the KL polynomials for E_8.

So what was the point?

In the fall of 2004, Fokko du Cloux was at MIT, rooming with fellow Atlas member Dan Ciubotaru. Fokko was halfway through writing the software I've talked about: the point at which neither the end of the tunnel nor the beginning is visible any longer.

Walking home after a weekend of mathematics, Dan said, "Fokko, look at us. We're spending Sunday alone at work."

In Which we Come to an Enchanted Place...

1/8/07 9 A.M. Finished writing to disk the KL polynomials for E_8.

So what was the point?

In the fall of 2004, Fokko du Cloux was at MIT, rooming with fellow Atlas member Dan Ciubotaru. Fokko was halfway through writing the software I've talked about: the point at which neither the end of the tunnel nor the beginning is visible any longer.

Walking home after a weekend of mathematics, Dan said,
"Fokko, look at us. We're spending Sunday alone at work."

Fokko was startled by this remark, but not at a loss for words.
"I don't know about you, but I'm having the time of my life!"

In Which we Come to an Enchanted Place...

1/8/07 9 A.M. Finished writing to disk the KL polynomials for E_8.

Fokko du Cloux

December 20, 1954–November 10, 2006