Counting Latin Squares

Jeffrey Beyerl

August 24, 2009
On the Summer 2009 computational prelim there were the following questions:
On the Summer 2009 computational prelim there were the following questions:

- Write a program, which given $n$ will enumerate all Latin Squares of order $n$. 
On the Summer 2009 computational prelim there were the following questions:

- Write a program, which given $n$ will enumerate all Latin Squares of order $n$.
- Does the structure of your program suggest a formula for the number of Latin Squares of size $n$? If it does, use the formula to calculate the number of Latin Squares for $n = 6, 7, 8, \text{ and } 9$. 
Definition

A Latin Square is an $n \times n$ table with entries from the set \{1, 2, 3, ..., n\} such that no column nor row has a repeated value.
You are familiar with some Latin Squares already.
You are familiar with some Latin Squares already.

- Sudoku Puzzles are $9 \times 9$ Latin Squares with some additional constraints.
You are familiar with some Latin Squares already.

- Sudoku Puzzles are $9 \times 9$ Latin Squares with some additional constraints.
- The multiplication table for a finite group is a Latin Square.
You are familiar with some Latin Squares already.

- Sudoku Puzzles are $9 \times 9$ Latin Squares with some additional constraints.
- The multiplication table for a finite group is a Latin Square.
- The multiplication table for a quasigroup is a Latin Square.
Example: Sudoku

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>9</strong></td>
<td><strong>4</strong></td>
<td><strong>6</strong></td>
<td><strong>8</strong></td>
<td><strong>3</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
<td><strong>1</strong></td>
<td><strong>5</strong></td>
</tr>
<tr>
<td><strong>1</strong></td>
<td><strong>5</strong></td>
<td><strong>2</strong></td>
<td><strong>6</strong></td>
<td><strong>9</strong></td>
<td><strong>7</strong></td>
<td><strong>8</strong></td>
<td><strong>3</strong></td>
<td><strong>4</strong></td>
</tr>
<tr>
<td><strong>7</strong></td>
<td><strong>3</strong></td>
<td><strong>8</strong></td>
<td><strong>4</strong></td>
<td><strong>5</strong></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>9</strong></td>
<td><strong>6</strong></td>
</tr>
<tr>
<td><strong>8</strong></td>
<td><strong>1</strong></td>
<td><strong>9</strong></td>
<td><strong>7</strong></td>
<td><strong>2</strong></td>
<td><strong>6</strong></td>
<td><strong>5</strong></td>
<td><strong>4</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td><strong>7</strong></td>
<td><strong>5</strong></td>
<td><strong>3</strong></td>
<td><strong>1</strong></td>
<td><strong>9</strong></td>
<td><strong>6</strong></td>
<td><strong>8</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>6</strong></td>
<td><strong>3</strong></td>
<td><strong>5</strong></td>
<td><strong>4</strong></td>
<td><strong>8</strong></td>
<td><strong>1</strong></td>
<td><strong>7</strong></td>
<td><strong>9</strong></td>
</tr>
<tr>
<td><strong>3</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
<td><strong>9</strong></td>
<td><strong>8</strong></td>
<td><strong>5</strong></td>
<td><strong>4</strong></td>
<td><strong>6</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>8</strong></td>
<td><strong>4</strong></td>
<td><strong>1</strong></td>
<td><strong>6</strong></td>
<td><strong>3</strong></td>
<td><strong>9</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>9</strong></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
<td><strong>4</strong></td>
<td><strong>3</strong></td>
<td><strong>5</strong></td>
<td><strong>8</strong></td>
</tr>
</tbody>
</table>
Example: Klein’s four group

<table>
<thead>
<tr>
<th></th>
<th>e</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>e</td>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>c</td>
<td>e</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>e</td>
</tr>
</tbody>
</table>
### Example: A Quasigroup

$$
\begin{array}{|c|c|c|c|c|}
\hline
. & e & a & b & c \\
\hline
e & e & a & b & c \\
\hline
a & a & e & c & b \\
\hline
b & b & c & a & e \\
\hline
c & c & b & e & a \\
\hline
\end{array}
$$
function enumerate(int xPosition, int yPosition)
    if row at xPosition is not valid: reset and return
    if column at yPosition is not valid: reset and return
    if last position: record Latin Square, reset and return.
    for $i = 1$ to $n$
        set next position to $i$.
        enumerate(next position).
    reset and return.
public void Enumerate(Coordinate coord) throws IOException
    if(!isValidRow(coord.x)) {
        entries[coord.x][coord.y] = 0;
        return;
    }
    if(!isValidCol(coord.y)) {
        entries[coord.x][coord.y] = 0;
        return;
    }
    if(coord.y == n-1 && coord.x == n-1) {
        AddValidSquare();
        entries[coord.x][coord.y] = 0;
        return;
    }
    for(int i=1; i <= n; i++) {
        Coordinate nextPlace = next(coord);
        entries[nextPlace.x][nextPlace.y] = i;
        Enumerate(nextPlace);
        entries[nextPlace.x][nextPlace.y] = 0;
    }
    return;
What insight does this program give to counting Latin Squares?
1. What insight does this program give to counting Latin Squares?
2. Backtracking algorithms are very difficult to analyze.

Jeffrey Beyerl  Counting Latin Squares
What insight does this program give to counting Latin Squares?

Backtracking algorithms are very difficult to analyze...

...at least for me
What insight does this program give to counting Latin Squares?

Backtracking algorithms are very difficult to analyze

...at least for me

The order to look at the tiles has an impact on the runtime.
What insight does this program give to counting Latin Squares?

Backtracking algorithms are very difficult to analyze...

...at least for me

The order to look at the tiles has an impact on the runtime.

Less than the $n^{n^2}$ possibilities from brute force.
Why backtracking?

1. Very easy to conceptualize
Why backtracking?

1. Very easy to conceptualize
2. Fairly easy to code
Why backtracking?

1. Very easy to conceptualize
2. Fairly easy to code
3. ...If you don’t try to make it too complicated at first and have to rewrite the entire thing
Why backtracking?

1. Very easy to conceptualize
2. Fairly easy to code
3. ...If you don’t try to make it too complicated at first and have to rewrite the entire thing
4. ......Like I did
Why backtracking?

1. Very easy to conceptualize
2. Fairly easy to code
3. ...If you don't try to make it too complicated at first and have to rewrite the entire thing
4. ......Like I did
5. ........On the Prelim
Why backtracking?

1. Very easy to conceptualize
2. Fairly easy to code
3. ...If you don’t try to make it too complicated at first and have to rewrite the entire thing
4. ......Like I did
5. ..........On the Prelim
6. Generalizes to other types of puzzles (In particular KenKen easily.)
So how many Latin Squares are there?

<table>
<thead>
<tr>
<th>n</th>
<th>Number of Latin Squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>576</td>
</tr>
<tr>
<td>5</td>
<td>161,280</td>
</tr>
<tr>
<td>6</td>
<td>8,128,512</td>
</tr>
<tr>
<td>7</td>
<td>614,794,199,040</td>
</tr>
<tr>
<td>8</td>
<td>10,877,603,245,908,295,680</td>
</tr>
<tr>
<td>9</td>
<td>5,524,751,496,156,892,842,531,256,000</td>
</tr>
<tr>
<td>10</td>
<td>9,982,437,658,213,039,871,725,064,756,920,320,000</td>
</tr>
<tr>
<td>11</td>
<td>7,769,668,361,717,701,441,074,443,467,342,306,823,110,656,000,000</td>
</tr>
</tbody>
</table>
So how many Latin Squares are there?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>576</td>
</tr>
<tr>
<td>5</td>
<td>161280</td>
</tr>
</tbody>
</table>
So how many Latin Squares are there?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>576</td>
</tr>
<tr>
<td>5</td>
<td>161280</td>
</tr>
<tr>
<td>6</td>
<td>812851200</td>
</tr>
<tr>
<td>7</td>
<td>61479419904000</td>
</tr>
<tr>
<td>8</td>
<td>108776032459082956800</td>
</tr>
<tr>
<td>9</td>
<td>5524751496156892842531225600</td>
</tr>
<tr>
<td>10</td>
<td>9982437658213039871725064756920320000</td>
</tr>
<tr>
<td>11</td>
<td>7769668361717701441074443467342306823110656000000</td>
</tr>
</tbody>
</table>
Lower Bounds

- $n!$ - reordering columns.

- $(n!)^2$ - A combinatorics textbook. (found on Wikipedia)
\begin{itemize}
  \item $n!$ - reordering columns.
  \item $n!(n - 1)!$ - reordering columns and rows.
\end{itemize}
Lower Bounds

- $n!$ - reordering columns.
- $n!(n - 1)!$ - reordering columns and rows.
- $n!(n - 2)! \left[ \frac{(n-1)!}{e} + \frac{1}{2} \right]$ - reordering the columns, considering derangements for the second row, then reordering the other rows.
- $n!$ - reordering columns.
- $n!(n - 1)!$ - reordering columns and rows.
- $n!(n - 2)! \left( \frac{(n-1)!}{e} + \frac{1}{2} \right)$ - reordering the columns, considering derangements for the second row, then reordering the other rows.
- $\frac{(n!)^{2n}}{n^{n^2}}$ - A a combinatorics textbook. (found on Wikipedia) (Better than the above for $n \geq 6$).
Why Lower Bounds?

No exact formula is known. Sloane's On-Line Encyclopedia of Integer Sequences lists the problem as "hard." Exact values are only known through $n = 11$ (possibly $n = 12$).
Why Lower Bounds?

- No exact formula is known.
Why Lower Bounds?

- No exact formula is known.
- Sloane’s On-Line Encyclopedia of Integer Sequences lists the problem as “hard”
Why Lower Bounds?

- No exact formula is known.
- Sloane’s On-Line Encyclopedia of Integer Sequences lists the problem as “hard”
- Exact values are only known through $n = 11$ (possibly $n = 12$).
Simplifying the problem

There are two equivalence relations that can be put on Latin Squares
Simplifying the problem

- There are two equivalence relations that can be put on Latin Squares
- ...This is useful so that one need only count the number of equivalence classes
If two Latin Squares are the same up to row/column permutations, they are equivalent (in this relation)
If two Latin Squares are the same up to row/column permutations, they are equivalent (in this relation).

...A canonical representative is a reduced Latin Square.
Row/Column Permutations

- If two Latin Squares are the same up to row/column permutations, they are equivalent (in this relation).
- A canonical representative is a reduced Latin Square.
- Which has the permutation \((1, 2, 3, 4, \ldots, n)\) across the first row and down the first column.
Row/Column Permutations and renaming elements

- If two Latin Squares are the same up to row/column permutations and renaming the elements, they are equivalent (in this relation - called isotopy)
<table>
<thead>
<tr>
<th>$n$</th>
<th>Latin Squares</th>
<th>Equivalence classes</th>
<th>isotrophy classes</th>
<th>paratopy classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>576</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>161280</td>
<td>56</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>$\approx 8 \times 10^8$</td>
<td>9408</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>$\approx 6 \times 10^{13}$</td>
<td>$\approx 1 \times 10^7$</td>
<td>564</td>
<td>147</td>
</tr>
<tr>
<td>8</td>
<td>$\approx 1 \times 10^{20}$</td>
<td>$\approx 5 \times 10^{11}$</td>
<td>1676267</td>
<td>283657</td>
</tr>
<tr>
<td>9</td>
<td>$\approx 5 \times 10^{27}$</td>
<td>$\approx 3 \times 10^{17}$</td>
<td>115618721533</td>
<td>19270853541</td>
</tr>
<tr>
<td>10</td>
<td>$\approx 9 \times 10^{36}$</td>
<td>$\approx 7 \times 10^{24}$</td>
<td>$\approx 2 \times 10^{17}$</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$\approx 7 \times 10^{47}$</td>
<td>$\approx 5 \times 10^{33}$</td>
<td>?</td>
<td>Unknown</td>
</tr>
</tbody>
</table>
Other Applications

- Orthogonal Arrays (They are one)
Other Applications

- Orthogonal Arrays (They are one)
- Error Correcting Codes
Other Applications

- Orthogonal Arrays (They are one)
- Error Correcting Codes
- P=NP?
Future Work

- None
None

(...On this problem)