Counting Latin Squares

Jeffrey Beyerl

August 24, 2009

Motivation

On the Summer 2009 computational prelim there were the following questions:

Motivation

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.

Motivation

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, 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.

Examples

You are familiar with some Latin Squares already.

Examples

You are familiar with some Latin Squares already.

- Sudoku Puzzles are 9×9 Latin Squares with some additional constraints.

Examples

You are familiar with some Latin Squares already.

- Sudoku Puzzles are 9×9 Latin Squares with some additional constraints.
- The multiplication table for a finite group is a Latin Square.

Examples

You are familiar with some Latin Squares already.

- Sudoku Puzzles are 9×9 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

9	4	6	8	3	2	7	1	5
1	5	2	6	9	7	8	3	4
7	3	8	4	5	1	2	9	6
8	1	9	7	2	6	5	4	3
4	7	5	3	1	9	6	8	2
2	6	3	5	4	8	1	7	9
3	2	7	9	8	5	4	6	1
5	8	4	1	6	3	9	2	7
6	9	1	2	7	4	3	5	8

Example: Klein's four group

·	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Example: A Quasigroup

$\cdot$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	a	e
c	c	b	e	a

```
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.
```

Enumerating Latin Squares

```
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;
```

Enumerating Latin Squares

1. What insight does this program give to counting Latin Squares?

Enumerating Latin Squares

1. What insight does this program give to counting Latin Squares?
2. Backtracking algorithms are very difficult to analyze

1. What insight does this program give to counting Latin Squares?
2. Backtracking algorithms are very difficult to analyze
3. ...at least for me

Enumerating Latin Squares

1. What insight does this program give to counting Latin Squares?
2. Backtracking algorithms are very difficult to analyze
3. ...at least for me
4. The order to look at the tiles has an impact on the runtime.

Enumerating Latin Squares

1. What insight does this program give to counting Latin Squares?
2. Backtracking algorithms are very difficult to analyze
3. ...at least for me
4. The order to look at the tiles has an impact on the runtime.
5. 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.

Enumerating Latin Squares

- So how many Latin Squares are there?

- So how many Latin Squares are there?

1	1
2	2
3	12
4	576
5	161280

- So how many Latin Squares are there?

1	1
2	2
3	12
4	576
5	161280
6	812851200
7	61479419904000
8	108776032459082956800
9	5524751496156892842531225600
10	9982437658213039871725064756920320000
11	776966836171770144107444346734230682311065600000

Lower Bounds

- $n!$ - reordering columns.

Lower Bounds

- $n!$ - reordering columns.
- $n!(n-1)!$ - reordering columns and rows.

Lower Bounds

- $n!$ - reordering columns.
- $n!(n-1)!$ - reordering columns and rows.
- $n!(n-2)!\left\lfloor \frac{(n-1)!}{e} + \frac{1}{2} \right\rfloor$ - reordering the columns, considering derangements for the second row, then reordering the other rows.

Lower Bounds

- $n!$ - reordering columns.
- $n!(n-1)!$ - reordering columns and rows.
- $n!(n-2)!\left\lfloor \frac{(n-1)!}{e} + \frac{1}{2} \right\rfloor$ - 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.

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

Row/Column Permutations

- 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 $(1, 2, 3, 4, ..., n)$ 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)

Enumerating Latin Squares - the reduced problem

n	Latin Squares	Equivalence classes	isotrophy classes	paratopy classes
1	1	1	1	1
2	2	1	1	1
3	12	1	1	1
4	576	4	2	2
5	161280	56	2	2
6	$\approx 8 \times 10^{8}$	9408	22	12
7	$\approx 6 \times 10^{13}$	$\approx 1 \times 10^{7}$	564	147
8	$\approx 1 \times 10^{20}$	$\approx 5 \times 10^{11}$	1676267	283657
9	$\approx 5 \times 10^{27}$	$\approx 3 \times 10^{17}$	115618721533	19270853541
10	$\approx 9 \times 10^{36}$	$\approx 7 \times 10^{24}$	$\approx 2 \times 10^{17}$	$\approx 3 \times 10^{16}$
11	$\approx 7 \times 10^{47}$	$\approx 5 \times 10^{33}$	?	Unknown

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

Future Work

- None
- (...On this problem)