Partially Completed Magic Squares Solver

Zijian Zhang (zz2795), Cynthia Zhang (cz2696)

Introduction

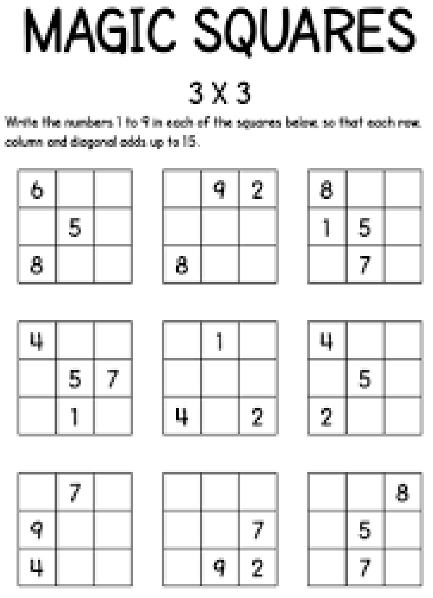
A magic square is composed of a square array of numbers, usually positive integers, in which the sums of the numbers in each row, each column, and both main diagonals are the same. The order of the magic square n is the size of the sides, and the magic constant M of a magic square is the constant sum. If the square only contains the positive integer $1, 2, ..., n^2$, the magic square is called normal, and its magic constant is $M = n * \frac{n^2+1}{2}$. Generating magic squares with $n > 5$ is still an open challenge.

The goal of this magic square solver is to solve normal partially filled magic squares with a size smaller or equal to 5. The techniques are similar to those used in Sudoku. We have not decided exactly which algorithm to implement, but it will likely include generating permutations and deducing values of unfilled squares. It is also possible to expand the solver to non-normal magic squares if time permits.

Single Thread Algorithm

Here is a brute force sample algorithm, we will likely add additional optimizations, such as checking the sum of completed rows and columns and terminating early if they don't sum to the magic constant. We could also explore if there's any mathematical relationship between different permutations and attempt to eliminate more possibilities at each iteration.

1. Choose a blank position as the start point, try to fill in a number that has not been filled.
2. Repeat step 1, unitil the square is totally filled.
3. Check if the filled square satisfies the requirement of magic square. If it satisfies, then output, otherwise backtrace to step1 and choose another number.



MAGIC SQUARES

3 X 3

Write the numbers 1 to 9 in each of the squares below, so that each row, column and diagonal adds up to 15.

6		
	5	
8		

	9	2
8		

8		
1	5	
	7	

4		
	5	7
	1	

	1	
4		2

4		
	5	
2		

	7	
9		
4		

		7
	9	2

		8
	5	
	7	

Parallel Implementation

At the first step of the single thread algorithm, the process of trying different numbers can be separated as individual tasks, and thus can be dispatched to threads. Each thread can calculate one or more possible permutations. Due to additional optimization, each permutation would have unequal workloads, and we need to balance the workload between each thread. Additionally, we could try to involve parallelism within each permutation, such as calculating two rows at the same time.