Name: ⟨ Julius Putra Tanu Setiaji ⟩ Matric: ⟨ A0149787E ⟩
Name: ⟨ Tan Zong Xian ⟩ Matric: ⟨ A0167343A ⟩

Algorithm: Phase 1: Initialisation

Step 1: Traverse the input puzzle to generate lists of sets containing non-zero numbers in each row, column and subsquare.

Step 2: Generate possible values for each cell using set difference of the domain and the existing numbers in the cell's row/column/subsquare as derived in step 1.

During this step, if any zero-value cell has a possibility set of size 1, push the cell's coordinates and its sole possible value onto a queue for processing in Phase 2. Phase 2: Set value and propagate constraint

While queue has elements:

Step 1: Pop from queue to get cell coordinates and value v.

Step 2: Set the cell in the answer matrix to v.

Step 3: For each cell in the same row, column or subsquare:

Step 3a: Remove v from its set of possibilities.

Step 3b: If the cell's possibility set size is 1 and its coordinates are not in the queue and its value in the answer matrix is zero:

Push the cell's coordinates and its sole possible value onto the queue.

Phase 3: Backtracking

Step 1: Find the cell with possibility set with the smallest possibility set size, and also size > 1.

Step 2: Spawn a copy of the current solver object, push a possibile value onto its queue and return to step 2. Return result if valid, backtrack to next possible value if invalid.

Complexity Analysis: Let n be the dimension of the puzzle i.e. 9. Time: Each step of the initialisation is $O(n^2)$ as they each traverse the puzzle once. In Phase 2 constraint propagation, when a cell's value is set, updating for same row, column and subsquare take $O(n)$ each. This is a constant $O(3n)$ which reduces to $O(n)$. The number of times this has to be done is one for each cell in the puzzle, giving a total of $O(n^3)$ in the ideal case without backtracking. Space: Each instance of the solver object has stores the answer matrix and the queue. The queue will not have duplicate elements so the queue's size is $O(n^2)$. The lists of sets used in Phase 1 Step 1 are also $O(n^2)$ each. Hence each instance of the solver object takse $O(n^2)$ space. Backtracking is depth-first search. For the provided puzzle, our recursion depth was 16. Hence we store at most $16O(n^2)$.