Basically, the algorithm will be broken into 3 parts:

1. The base raw 2D board data will be generated varied by the specified character value spacing only(to be explained later)
2. The generated data will have its portion groups and portions in groups swapped by a specified combination only(to be explained later)
3. The character value spacing and swapping combination will be generated by random

**Character Value Spacing**

Suppose characters and are 2 valid digits in Sudoku, then the character value spacing between them is . For instance:

1. The character value spacing between and is
2. The character value spacing between and is

**Portion Group**

For a sudoku board with size , a portion group is a group of the 1st to th/th to th/th to th/…/th to thportions. So a row group is that of such rows and a column group is that of such columns.

The importance of this concept is that, no matter how these portion groups are swapped within each other, every grid will still remain unchanged, meaning that the board won’t become invalid.

**Portions In Group**

Portions in group are those In the same portion group. So rows in group are those in the same row group and columns in group are those in the same column group.

The importance of this concept is that, no matter how these portions are swapped within the same portion group, every grid will still remain unchanged, meaning that the board won’t become invalid.

**Swapping Combination**

Given n elements to be swapped within each other, a swapping combination is a list of consecutive indices , where , arranged in a specific order. To simplify its usage, the smallest is and the largest one is .

For instance, for 6 elements, [0, 1, 2, 3, 4, 5], [2, 5, 4, 3, 1, 0] and [5, 4, 3, 2, 1, 0] are swapping combinations, meaning that the new index of the element with old index i should be l[i], where l is the list being the swapping combination.

**Base Raw 2D Board Data**

Basically, for a board with size and character value spacing , the character value of a square with row index and column index is , where is the row group index, is the base character value offset per row index increase, is the character value spacing per column index increase, and is the board length.

A row, column or grid is said to be valid if every square inside has different modulo of its length.

If and are [coprime](https://en.wikipedia.org/wiki/Coprime_integers), then the above character value formula will ensure that all rows, columns and grids are valid.

Lemma 1 -

Proof of Lemma 1:

,

– (1)

-(2)

As (1) + (2) leads to a contradiction, Lemma 1 holds.

Lemma 2–

Proof of Lemma 2:

Therefore Lemma 2 holds.

**Row Validity**

By Lemma 2,

As and are [coprime](https://en.wikipedia.org/wiki/Coprime_integers), must be divisible by , which contradicts the fact that .

So the character value of squares within the same row must be different from each other, meaning that every row must be valid.

**Column Validity**

Similarly, due to Lemma 1 and 2, the character value of squares within the same column must be different from each other, meaning that every column must be valid.

**Grid Validity**

For any 2 distinct squares in the same grid, let their coordinates be and respectively.

By Lemma 2,

is the row group index,

As and are coprime and must not be divisible by , meaning that there’s a contradiction.

Therefore the character value of squares within the same grid must be different from each other, meaning that every grid must be valid.