ISSN 2229-5518

Magic Square Generic Algorithm

Author: Samuel A.M.

Abstract:

Today in the world of competition , the creativity do hold their roots. And, because of these vital things only, the technology of today is at great peaks and expanding its influence all over the world. It's being the rule of nature that better things i.e. new technology, better algorithms, better analogies do overtake the old ones and to survive in this world of competition creativity is must. With the same thoughts , I am giving you a new angle of vision to magic square , which i had figured out through my research in late 2002.

Introduction

The reader must be curious to know about what magic square is ? What is magic in that square? What are its use? etc. In fact this n*n (square) do resembles a 2-Dimensional matrices in appearance, but the pattern of its number, this 2 D matrices becomes special. The specialty being that, the numbers are arranged in such a crucial pattern that its sum of rows, sum of columns, sum of diagonals are equal. This property of square matrices has wide applications and do unbelievable tasks. i.e. Magic.

Research and Merit of my algorithm:

In Quizzes, competition, and in general life , magic square is helpful.

The problem of a farmer who posses 49 cows, with the specialty to give milk according to their number i.e. if cow gives milk of about 45 liters per day, then it is numbered 45 and so on. This farmer is about to die and the problem is to distribute these cows among his 7 sons, such that each get equal milk.

The reader may think, if farmer. posses more cows, the distribution becomes tougher. The case may be different, but the property is same. when considering the case of a nation milk supply and we want to distribute it to the distributors equally then. This accurate distribution will create transparency and build trust too.

There are some algorithms present on this topic like arrow rule, Semi matrices, even magic square, Duler's magic Square, Multiplication magic square, diamond magic square, addition-multiplication magic square, doubly even magic square, alpha magic square etc.

My algorithm is based on simple probability theory and based on logics of permutation and combination.

Explanation of my algorithm's principle

The best way to understand probability is to understand with an example. So, let us consider a 3*3 matrices.

The middle element of square must be middle element of series. i.e. 5

1	2	3
4	5	6
7	8	9

Now , for n*n matrice,

then, series be 1,,,,,,n^2.

Then , mid-value= (1+n^2)/2

Note: here, n must be odd.

For 3*3 matrice

Pairs of 2 number i.e. (1,9), (2,8), (3,7), (4,6) which sums to 10. Now arrange these number pairs with 5 to generate sum 15 and fulfill magic square condition.

But, as the order increases the probability to arrange these numbers too increases.

i.e. for 5*5, the mid value=13

then,

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Now the pairs (1,2,24,25), (7,8,18,19), (3,4,22,23),(9,10,16,17),(5,6,20,21),(11,12,14,15)

Thus for n*n, Mid value= (n^2 +1)/2

Pairs of (n-1) combinations, in which number is selected (n-1)/2 from beginning and end and pair is formed. Total number of pairs is n+1.

The Algorithm along with explanation (Using C++)

1) Check n whether odd or even.

2) If even return to step 1.

3) Set M[n][n] =0 and a[n^2]= {1,2,3,..., n^2}

4) Set a1=0, a2=n^2 /* Lower and upper bound limits */

5) For Pair storing p [n+1][n-1] & p1=0 by using malloc().

6) Since, pairs stored dynamically, use the pattern,

```
while (l<(n+1)*(n-1))
{
*(p+l)=a[a1];++l;++a;++t;
        if(t==(n-1)/2)
                {
                t=0;
                l+=(n-1)/2;
                }
}
l=0;
while(l<(n+1)*(n-1))
{       if(*(p+l)==0)
        {
        *(p+l)=a[a2]; --a2;
        }
}
```

Now , to form pairs with 5 [in 3*3] or mid value, to form square, magic square.

ISSN 2229-5518

7)

.	.	.
.		

Now, in 3*3,

4 positions at which numbers to be filled,

Case : keeping one pair fixed and changing the other three pairs until pairs reach their initial stage. Thus fixing a pair at particular position and instantaneously changing the other pairs and repeat this process for all pairs at that particular position, until magic square is found.

For n*n

Check at (n+1) position

cases will be (n+1)

sequence number is n

if not magic square i.e. Sr=Sc=Sd then switch to next case.