Magic Square Construction Algorithms and Their Applications

Krishnappa H K*, N K Srinath** and Ramakanth Kumar P***

In recreational mathematics, a magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the sum of n numbers in all rows, all columns and both diagonals is the same constant called the magic number. The magic square in normal is represented using $n \times n$ matrix. A normal magic square contains the integers from 1 to n^2. Normal magic squares exist for all orders $n \geq 1$, except $n = 2$. The magic constant for normal magic squares of order n is given by $n(n^2 + 1)/2$. There are several methods for constructing the magic square of any given order. This paper proposes algorithms to obtain the magic square of any given order n. Some of the algorithms are straight forward and others are designed using the divide and conquer technique.

Keywords: Magic square, Magic constant, Divide and conquer, Increment-decrement, Swap

Introduction

The magic square is represented by n x n matrix. The normal magic square contains numbers from 1 to n^2 (Ellis *et al.*, 2002). There exists a magic square for all orders $n \geq 1$, except $n = 2$. Several methods exist for construction of magic square of any given order (www.wikipedia.org).

The magic squares have several applications in fields of discrete and combinatorial mathematics, and also in the area of graph theory. One such application of magic square is in graph labeling. It has been proved using the magic square of order n that, "There exists a vertex magic total labeling for all complete graph K_n" (Krishnappa *et al.*, 2009 and 2010).

Our process of constructing a magic square of order n is divided into the following categories and subcategories:

- Magic square of order n, where n is odd.
- Magic square of order N, where N is even.

* Assistant Professor, Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, India; and is the corresponding author. E-mail: hk_krit@yahoo.co.in

** Professor, Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, India. E-mail: srinath_nk@yahoo.com

*** Professor, Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, India. E-mail: pramakanth2000@gmail.com

– Magic square of order 4.
– Magic square of order N, for all $N \equiv 2 \bmod 4$.
– Magic square of order N, for all $N \equiv 0 \bmod 4$.

2. Algorithm for Magic Square of Order n, Where n is Odd

It is trivial for $n = 1$, it consists of a single cell containing the number 1.

H Coxeter had given a simple rule for generating a magic square, when n is odd (Ellis *et al.*, 2002).

- Start with 1 in the middle of the top row;
- Then go up and left, assigning numbers in increasing order to empty squares;
- If you fall off the square, imagine the same square as tiling the plane and continue;
- If a square is occupied, move down instead and continue.

Figure 1: Magic Square of Order 3

6	1	8
7	5	3
2	9	4

The magic square of order 3 is shown in Figure 1, which is formed using the Coxeter rule.

The following algorithm can be used to construct any magic square of order $n \geq 1$, where n is odd.

2.1 Algorithm for Magic Square of Odd Order (n)

//Description: This creates a magic square of order n, n being odd.

//Input: A positive integer n.

//Output: The n x n square matrix which is a magic square of order n.

Step 1:

//Check if n is odd.

if((n mod 2) = 0)

    then return 0;

Step 2:

//Initialization

//Initialize all the entries of matrix to 0

for i <– 0 to n – 1 do

    for j <– 0 to n – 1 do

        square[i, j] <– 0

//Initialize the position of the first entry:

square[0, $(n - 1)/2$] <– 1

//Initialize the next position of row{i} and column{j}

i <– 0; j <– $(n - 1)/2$

Step 3:

//Move up left and fill the empty square with next integer.

for key <–2 to n^2 do

    if($i \geq 1$) then

        k<– $i - 1$

    else

        k <– $n - 1$

    if($j \geq 1$) then

        l <– $j - 1$

    else

        l <– $n - 1$

if square [k, l] ≥ 1, then

    i <– $(i + 1)$ mod n

else

    i <– k

    j <– 1

square [i, j] <– key

return square.

As another example, let us construct a magic square of order 5 (Figure 2).

Figure 2: Magic Square of Order 5

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

The time to initialize and obtain the square is $\theta(n^2)$. The third 'for loop' in which the key ranges over 2 to n^2 is iterated $n^2 - 1$ times and each iteration takes $\theta(l)$ time. So, this 'for loop' takes $\theta(n^2)$ time, since there are n^2 positions at which the algorithm must place a number. We see that $\theta(n^2)$ is the best bound for an algorithm for the magic square problem considered by us.

3. Magic Square of Order *N*, Where *N* is Even

In this section, we provide algorithms to construct magic square of even order. We further divide the process into three parts.

- Magic square of order 4.
- Magic square of order *N*, for all $N \equiv 2 \bmod 4$.
- Magic square of order *N*, for all $N \equiv 0 \bmod 4$.

3.1 Magic Square of Order 4

The following algorithm can be used to construct a magic square of order 4:

3.1.1 Algorithm for Magic Square of Order_4(n)

```
//This algorithm generates the magic square of order 4.
//For convenience let us consider four variables
//TopIndex_X, TopIndex_Y, BottomIndex_X, BottomIndex_Y.
//Also consider two intermediate variables
//TopDownValue and BottomUpValue.

//Input: The value n which is bounded to 4.
//Output: A square matrix M, which is a 4 x 4 matrix, the magic square of order 4.

//Step 1: Initialization
//Indices
TopIndex_X <– 0;
TopIndex_Y <– 0;
BottomIndex_X <– n – 1;
BottomIndex_Y <– n – 1;

//Starting values:
TopDownValue <– 1;
BottomUpValue <– n2;
```

```
//Step 2: Repetitively fill the remaining entries:
    while (TopIndex_X < BottomIndex_X) do
        while (TopIndex_Y < n ) do
            M[TopIndex_X][TopIndex_Y] <– TopDownValue;
            M[BottomIndex_X][BottomIndex_Y] <– BottomUpValue;

            TopIndex_Y <– TopIndex_Y + 1;
            BottomIndex_Y <– BottomIndex_Y – 1;
            TopDownValue <– TopDownValue + 1;
            BottomUpValue <– BottomUpValue – 1;

            if( TopIndex_Y # n/2) then
                swap(TopDownValue, BottomUpValue);

    TopIndex_X <– TopIndex_X + 1;
    BottomIndex_X <– BottomIndex_X – 1;
    TopIndex_Y <– 0;
    BottomIndex_Y <– n – 1;
return M
```

Example: For $n = 4$, the working of the algorithm is depicted in Figure 3.

Figure 3: Working of the Algorithm for $n = 4$

(a)

1			
			16

(b)

1	15		
		2	16

(c)

1	15	14	
	3	2	16

(d)

1	15	14	4
13	3	2	16

(e)

1	15	14	4
12			
			5
13	3	2	16

(f)

1	15	14	4
12	6		
		11	5
13	3	2	16

(g)

1	15	14	4
12	6	7	
	10	11	5
13	3	2	16

(h)

1	15	14	4
12	6	7	9
8	10	11	5
13	3	2	16

The overall time complexity of this algorithm is $\theta(n^2)$, since there are n^2 positions at which the algorithm must place a number.

3.2 Magic Square of Order N, for all $N \equiv 2 \bmod 4$

In this section, we present an algorithm to construct a magic square of order N, where N is a singly even number. We adopt the divide and conquer technique and regular swapping to obtain the magic square of order N. For convenience, let $N = 2n$, where n is odd. The order of the final matrix is N x N, which in turn is divided into $4n \times n$ matrices. In Section 2, we presented an algorithm to construct a magic square of order n, n being odd. By using this algorithm, we construct four intermediate matrices M1, M2, M3 and M4, each of these matrices is of order n, where n is odd. Finally, we combine these four intermediate matrices to get a single matrix of order N.

The algorithm is as follows:

3.2.1 Algorithm for Magic Square of Singly Even Number (N)

//This algorithm constructs a magic square M of order N, where $N \equiv 2 \bmod 4$.

//Input: The positive even integer N.

//Output: The N x N matrix M, which is a magic square of order N.

//Step 1:

//Let M1, M2, M3, and M4 represent magic squares of order n.

n <– N/2;

if (N mod 4 # 2) then

    return (0)

else

    M1 = Magic Square of Odd Order (n)

    //Copy M1 to other three intermediate matrices M2, M3, and M4.

    for i <– 0 to n – 1 do

        for j <– 0 to n – 1 do

            M2[i, j] <– M1[i, j] + n^2

            M3[i, j] <– M1[i, j] + $2n^2$

            M4[i, j] <– M1[i, j] + $3n^2$

//Step 2:

//Swap the elements of M1 with the elements of M4 in the following order.

```
//Start from the first row of M1, swap n/2 elements of all the rows of M1 except for
//(n/2 + 1)th row with the corresponding position of M4 matrix.
for i <– 0 to n/2 do
    for j <– 0 to n/2 – 1 do
        swap (M1[i, j], M4[i, j])
for i <– n/2+1 to n do
    for j <– n/2+1 to n do
        swap (M1[i, j], M4[i, j])
//For (n/2 + 1)th row alone, increase column by 1 and start swapping n/2 elements
of that row of M1 with the corresponding elements of matrix M4.
i <– n/2
for j <– 1 to n/2 do
    swap (M1[i, j], M4[i, j])
//Starting from the last column, swap n/2 – 1 column elements of M3 with the
//corresponding elements of M2.
for i <– 0 to n – 1 do
    for j <– (n – n/2)+1 to n – 1 do
        swap (M2[i, j], M3[i, j])
//Step 3:
//Construct the final matrix M in the following order
//    M1    M3
//    M4    M2
//Copy M1 to M.
for i <– 0 to n – 1 do
    for j <– 0 to n – 1 do
        M[i, j] <– M1[i, j]
```

```
//Copy M3 to M.
for l <– 0 to n – 1 do
j <– n; k <– 0;
    for i <– 0 to n – 1 do
            M[i, j] <– M3[i, k]
            j <– j + 1
            k <– k + 1
//Copy M4 to M.
k <– 0
for i <– n to 2n –1 do
    for j <– 0 to n – 1 do
            M[i, j] <– M4[k, j]
    k <– k + 1
//Copy M2 to M.
k <– 0
for i <– n to 2n – 1 do
    l<–0
    for j <– n to 2n – 1 do
            M[i, j] <– M2[k, l]
            l<–l + 1
    k <– k + 1
//The matrix M is the desired magic square of order N.
return (M)
```

Example: Let us construct a magic square of order 6 using magic square of order 3. Let M1 be a magic square of order 3, obtained by using the algorithm for magic square of odd order (3). For $N = 6$, the working of the algorithm is depicted in Figure 4.

Figure 4: Working of the Algorithm for $N = 6$

M1		
6	1	8
7	5	3
2	9	4

M2		
15	10	17
16	14	12
11	18	13

M3		
24	19	26
25	23	21
20	27	22

M4		
33	28	35
34	32	30
29	36	31

	6	1	8	24	19	26	
	7	5	3	25	23	21	
M1	2	9	4	20	27	22	M3
	33	28	35	15	10	17	
	34	32	30	16	14	12	
M4	29	36	31	11	18	13	M2

	33	1	8	24	19	26	
	7	32	3	25	23	21	
M1	29	9	4	20	27	22	M3
	6	28	35	15	10	17	
	34	5	30	16	14	12	
M4	2	36	31	11	18	13	M2

The overall time complexity of this algorithm is $\theta(n^2)$, since there are n^2 positions at which the algorithm must place a number.

3.3 Magic Square of Order N, for all $N \equiv 0 \bmod 4$

In this section, we present an algorithm to construct a magic square of order N, where N is a doubly even number. We adopt the divide and conquer technique and regular swapping to obtain the magic square of order N. For convenience, let $N = 2n$, where n is even. The order of the final matrix is N x N which, in turn, is divided into 4 n x n matrices. In sections 3.1 and 3.2, we presented algorithms to construct a magic square of order 4 and magic square of order n, for all $n \equiv 2 \bmod 4$. By using these algorithms we construct four intermediate matrices M1, M2, M3 and M4, each of these matrices is of order n. Finally, we combine these four intermediate matrices to get a single matrix of order N.

The algorithm is as follows:

3.3.1 Algorithm for Magic Square of Doubly Even Number (N)

//This algorithm constructs a magic square M of order N, where $N \equiv 0 \bmod 4$.

//Input: The positive even integer N.

//Output: The N x N matrix M, which is a magic square of order N.

//Step 1:

n <– N/2;

//Let M1, M2, M3, and M4 represent Magic squares of order n.

if (n = 4) then

M1 = Magic Square of Order_4(n)

else

if (n mod 4 = 2) then

M1 = Magic Square of Singly Even Number (n)

else

M1 = Magic Square of Doubly Even Number (n)

//Copy M1 to other three intermediate matrices M2, M3, and M4.

for i <– 0 to n – 1 do

for j <– 0 to n – 1 do

M2[i, j] <– M1[i, j] + n^2

M3[i, j] <– M1[i, j] + $2n^2$

M4[i, j] <– M1[i, j] + $3n^2$

//Step 2:

//Construct the final matrix M by using the intermediate matrices M1, M2, M3 and M4

if (n mod 4 = 2) then

//Construct the final matrix M in the following order

// M1 M3

// M4 M2

//Copy M1 to M.

for i <– 0 to n – 1 do

for j <– 0 to n –1 do

M[i, j] <– M1[i, j]

//Copy M3 to M.

for l <– 0 to n – 1 do

j <– n; k <– 0;

for i <– 0 to n – 1 do

M[i, j] <– M3[i, k]

```
                    j <– j + 1
                    k <– k + 1
//Copy M4 to M.
    k <– 0
    for i <– n to 2n – 1 do
            for j <– 0 to n – 1 do
                    M[i, j] <– M4[k, j]
    k <– k + 1
//Copy M2 to M.
    k <– 0
    for i <– n to 2n – 1 do
    l <– 0
            for j <– n to 2n – 1 do
                    M[i, j] <– M2[k, l]
                    l <– l+1
    k <– k + 1
if (n mod 4 = 0) then
//Construct the final matrix M in the following order
//    M1      M4
//    M3      M2
//Copy M1 to M.
for i <– 0 to n – 1 do
    for j <– 0 to n – 1 do
            M[i, j] <– M1[i, j]
//Copy M4 to M.
for l <– 0 to n – 1 do
j <– n; k <– 0;
    for i <– 0 to n – 1 do
            M[i, j] <– M4[i, k]
```

```
                j <– j + 1
                k <– k+1
//Copy M3 to M.
k <– 0
for i <– n to 2n – 1 do
    for j <– 0 to n –1 do
            M[i, j] <– M3[k, j]
    k <– k + 1
//Copy M2 to M.
k <– 0
for i <– n to 2n – 1 do
    l <– 0
    for j <– n to 2n – 1 do
            M[i, j] <– M2[k, l]
            l <– l + 1
    k <– k+1
//Step 3:
// Perform the row exchange to get the final magic square.
if (n mod 4 = 2) then
//Exchange the n/2 +1 number of middle rows of M1 and M3
for i <– ((n – 2)/4)+1 to n – ((n – 2)/4) do
    j <– 1; k <– n + 1;
    for l <– 1 to n do
            swap (M[i, j], M[i, k]
            j <– j + 1
            k <– k + 1
//Similarly exchange the n/2 +1 number of middle rows of M4 and M2
for i <– [((n – 2)/4) + 1] + n to [n – ((n – 2)/4)] + n do
```

```
        j <– 1; k <– n + 1;
        for l <– 1 to n do
                swap (M [i, j], M[i, k])
                j <– j + 1
                k <– k + 1
    if (n mod 4 = 0) then
    //Exchange the n/2 number of middle rows of M1 and M4
    for i <– (n /4)+1 to 3n/4 do
        j <– 1; k <– n + 1;
        for l <– 1 to n do
                swap (M [i, j], M[i, k])
                j <– j + 1
                k <– k + 1
    //Similarly exchange the n/2 number of middle rows of M3 and M2
    for i <– [(n/4) + 1] + n to [3n/4] + n do
        j <– 1; k <– n + 1;
        for l <– 1 to n do
                swap (M [i, j], M[i, k])
                j <– j + 1
                k <– k + 1
    //Now the matrix M is the desired magic square of order N.
    return (M)
```

Example: Let us construct a magic square of order 8 and 12 using magic squares of order 4 and 6, respectively. For $N = 8$, the working of the algorithm is depicted in Figure 5.

$n = N/2 = 8/2 = 4$ and arranging the matrices in the order:

M1	M4
M3	M2

Figure 5: Working of the Algorithm for $N = 8$

M1			
1	15	14	4
12	6	7	9
8	10	11	5
13	3	2	16

M2			
17	31	30	20
28	22	23	25
24	26	27	21
29	19	18	32

M3			
33	47	46	36
44	38	39	41
40	42	43	37
45	35	34	48

M4			
49	63	62	52
60	54	55	57
56	58	59	53
61	51	50	64

	1	15	14	4	49	63	62	52	
	12	6	7	9	60	54	55	57	
	8	10	11	5	56	58	59	53	
M1	13	3	2	16	61	51	50	64	M4
	33	47	46	36	17	31	30	20	
	44	38	39	41	28	22	23	25	
	40	42	43	37	24	26	27	21	
M3	45	35	34	48	29	19	18	32	M2

- $n/2 = 4/2 = 2$
- Exchange the middle 2 rows of M1 with the middle 2 rows of M4.
- Similarly exchange the middle 2 rows of M3 with the middle 2 rows of M2.

	1	15	14	4	49	63	62	52	
	60	54	55	57	12	6	7	9	
	56	58	59	53	8	10	11	5	
M1	13	3	2	16	61	51	50	64	M4
	33	47	46	36	17	31	30	20	
	28	22	23	25	44	38	39	41	
	24	26	27	21	40	42	43	37	
M3	45	35	34	48	29	19	18	32	M2

Finally, we take the sum of all the entries in each row, column and the two diagonals and verify whether all the values are same or not. In this example it is 260, since magic constant for n is given by $n(n^2 + 1)/2$.

Therefore, $8(8^2 + 1)/2 = 4 \times 65 = 260$.

Example: Let us now construct the magic square of order 12 using this algorithm. The working of the algorithm is depicted in Figure 6.

$N = 12, n = N/2 = 12/2 = 6.$

Let M1, M2, M3, and M4 represent the magic square of order 6. Since 2 mod 4, we use the algorithm of Section 3.2 and arrange the matrices in the order:

M1	M3
M4	M2

Figure 6: Working of the Algorithm for $N = 12$

M1					
33	1	8	24	19	36
7	32	3	25	23	21
29	9	4	20	27	22
6	28	35	15	10	17
34	5	30	16	14	12
2	36	31	11	18	13

M2					
69	37	44	60	55	62
43	68	39	61	59	57
65	45	40	56	63	58
42	64	71	51	46	53
70	41	66	52	50	48
38	72	67	47	54	49

M3					
105	73	80	96	91	98
79	104	75	97	95	93
101	81	76	92	99	94
78	100	107	87	82	89
106	77	102	88	86	84
74	108	103	83	90	85

M4					
141	109	116	132	127	134
115	140	111	133	131	129
137	117	112	128	135	130
114	136	143	123	118	125
142	113	138	124	122	120
110	144	139	119	126	121

	33	1	8	24	19	36	105	73	80	96	91	98	
	7	32	3	25	23	21	79	104	75	97	95	93	
	29	9	4	20	27	22	101	81	76	92	99	94	
	6	28	35	15	10	17	78	100	107	87	82	89	
	34	5	30	16	14	12	106	77	102	88	86	84	
M1	2	36	31	11	18	13	74	108	103	83	90	85	M3

Figure 6 (Cont.)

	141	109	116	132	127	134	69	37	44	60	55	62	
	115	140	111	133	131	129	43	68	39	61	59	57	
	137	117	112	128	135	130	65	45	40	56	63	58	
	114	136	143	123	118	125	42	64	71	51	46	53	
	142	113	138	124	122	120	70	41	66	52	50	48	
M4	110	144	139	119	126	121	38	72	67	47	54	49	M2

- $n/2 + 1 = 6/2 + 1 = 4$.
- Exchange the middle 4 rows of M1 with middle 4 rows of M3.
- Similarly exchange the middle 4 rows of M4 with the middle 4 rows of M2.

	33	1	8	24	19	36	105	73	80	96	91	98	
	79	104	75	97	95	93	7	32	3	25	23	21	
	101	81	76	92	99	94	29	9	4	20	27	22	
	78	100	107	87	82	89	6	28	35	15	10	17	
	106	77	102	88	86	84	34	5	30	16	14	12	
M1	2	36	31	11	18	13	74	108	103	83	90	85	M3
	141	109	116	132	127	134	69	37	44	60	55	62	
	43	68	39	61	59	57	115	140	111	133	131	129	
	65	45	40	56	63	58	137	117	112	128	135	130	
	42	64	71	51	46	53	114	136	143	123	118	125	
	70	41	66	52	50	48	142	113	138	124	122	120	
M4	110	144	139	119	126	121	38	72	67	47	54	49	M2

Finally, we take the sum of all the entries in each row, column and the two diagonals and verify whether all the values are same or not. In this example it is 870, since the magic constant for n is given by $n(n^2 + 1)/2$.

Therefore, $12(12^2 + 1)/2 = 870$.

The overall time complexity of this algorithm is $\theta(n^2)$, since there are n^2 positions at which the algorithm must place a number.

Conclusion

The paper uses divide and conquer technique to prove that there exists a magic square of order n for all $n \geq 1$, except for $n = 2$. The technique attempts to divide and swap in a regular pattern to attain the same. It is worth exploring more such applications of the magic square as done by Krishnappa *et al.* (2009 and 2010). ✌

References

1. Ellis Horwitz, Sartaj Sahani and Sanguthevar Rajasekaran (2002), *Fundamentals of Computer Algorithms*, pp. 34-36, Galgotia Publications Pvt. Ltd.
2. Krishnappa H K, Kishore Kothapalli and Venkaiah V Ch. (2009), "Vertex Magic Total Labelings of Complete Graphs", *AKCE Journal of Graphs and Combinatorics*, Vol. 6, No. 1, pp. 143-154.
3. Krishnappa H K, Srinath N K and Ramakanth Kumar P (2010), "Vertex Magic Total Labelings of Complete Graphs", *International Journal of Computers, Mathematical Sciences and Applications*, Vol. 4, Nos. 1-2, pp. 157-169.
4. Mac Dougall J A, Mirka Miller, Slamin and Wallis W D (2002), "Vertex Magic Total Labelings of Graphs", *Util. Math.*, Vol. 61, pp. 3-21.
5. www.wikipedia.org
6. Yuqing Lin and Mirka Miller (2001), "Vertex Magic Total Labelings of Complete Graphs", *Bull. Inst. Combin. Appl.*, Vol. 33, pp. 68-76.

Reference # 61J-2010-09-04-01