The math behind the Siamese method of generating magic squares

1. What is a magic square

A magic square consists of the distinct positive integers 1, 2, ... n^2, such that the sum of the n numbers in any horizontal, vertical, or main diagonal line is always the same magic constant.

If the rows and columns sum to the magic constant (so ignoring the main diagonals) it is called a semimagic square.

Several ways exist to automatically generate magic squares, among which the most famous one is the Siamese method to generate odd-sized magic squares. It is also called the de la Loubere's method. The method is purported to have been first reported in the West when de la Loubere returned to France after serving as ambassador to Siam.

2. Example of the classic constructions

This example is taken from the excellent description on Math Forum. I reproduce it almost literally here in case that link ever goes away.

Start with an empty **n** x **n** square, where **n is odd**. We'll begin with **n = 5** and denote its blank entries with dots.

```
.  .  .  .  .
.  .  .  .  .
.  .  .  .  .
.  .  .  .  .
.  .  .  .  .
```

The problem of making this into a magic square is one of counting from 1 to 25 (in general n^2) as one runs through the boxes of the square in some order. The basic rule for doing this is that one counts diagonally upwards to the right. For example, if one has written a 12 in the 2nd position of the 4th row

```
.   .   .   .   .
.   .   .   .   .
.   .   .   .   .
.  12   .   .   .
.   .   .   .   .
```

then the numbers 13,14,15 will be placed thus:

```
.   .   .   .  15
.   .   .  14   .
.   .  13   .   .
. 12   .   .   .
.   .   .   .   .
```

Let's see this in action.

First you have to know where to start: begin in the middle of the top row with a 1. (Here we for the first time use the assumption that n is odd, which guarantees that there **is** a middle square in the top row). If you don't start there, the row and column sums will be ok, but the diagonals won't add to the magic sum.

```
.  .  1  .  .
.  .  .  .  .
.  .  .  .  .
.  .  .  .  .
.  .  .  .  .
```

Now, the first thing one observes is that it is impossible to move diagonally upwards from this position, since we are already at the top of the square. This is an example of what can go wrong with the basic method of moving diagonally upwards to the right. There are actually three things that can go wrong:

1. You could be at the top edge of the square so you can't go up.
2. You could be at the right edge of the square so you can't move right.
3. The square you would like to put the next number in may already be occupied.

Here is what to do in those cases:
1. If you are at the top edge, pretend that **the top edge is pasted to the bottom edge** and come up through the bottom. Thus, after 1, we place 2 as follows:

```
.   .   1   .   .
.   .   .   .   .
.   .   .   .   .
.   .   .   .   .
.   .   .   2   .
```

In other words, we pretend that the bottom row is the row above the top row. Moving diagonally upwards to the right means moving up one row and over one column to the right. So the 2 goes in the bottom row one column to the right of the column containing the 1.

2. If you are at the right edge, pretend that **the right edge is pasted to the left edge** and that **the left edge is immediately to the right of the right edge**. We are in this situation after counting to 3, since after the above diagram, we have

```
    .    .    1    .    .

    .    .    .    .    .

    .    .    .    .    .

    .    .    .    .    3

    .    .    .    2    .
```

and apparently have no place to go. But moving diagonally upwards to the right is the same as moving right one column and up one row. Moving right one column from the rightmost column puts us in the leftmost column, so 4 must be in the leftmost column, and moving up one row we put the 4 in the row above the row containing the 3. So we get

```
    .    .    1    .    .

    .    .    .    .    .

    4    .    .    .    .

    .    .    .    .    3

    .    .    .    2    .
```

3. What if there is a number already occupying the square one would like to move into? We are in this situation after we place the number 5. Indeed, continuing from the previous diagram, we place the 5 diagonally upwards and to the right of the 4 and get

```
    .    .    1    .    .

    .    5    .    .    .

    4    .    .    .    .

    .    .    .    .    3

    .    .    .    2    .
```

and when we next try to place the 6, the square we would like to put the 6 in already has the 1 in it! When this happens, the rule is to abandon, just this once, the plan of moving diagonally upwards to the right and instead just drop down one square from the square one is in presently. So the 6 will be placed directly below the 5.

```
    .    .    1    .    .
```

```
.   5   .   .   .
4   6   .   .   .
.   .   .   .   3
.   .   .   2   .
```

After that, one continues counting normally:

```
.   .   1   8   .
.   5   7   .   .
4   6   .   .   .
.   .   .   .   3
.   .   .   2   .
```

Eventually, one gets the whole **5x5** square in this way:

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

The magic constant is 65 in this example, and indeed all rows, columns and the two main diagonals sum to 65.

The application of the basic rule of counting diagonally upwards, with the modifications above for handling the edges and the case where a number is in the way, is pretty straightforward. But there is one case that requires a little thought. Namely, when one gets to 15, where does one go next?

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

Since 15 is in the top row, 16 would normally go in the bottom row. Since 15 is on the right edge, 16 would normally go on the left edge. The position which is in the bottom row and the left edge is the lower left corner. That is where we want to put the 16. Unfortunately, it is already occupied by the number 11. So there is a number in the way, and the rule is then to drop down to the square below the 15. That is why the 16 is directly below the 15.

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

All other instances of the rules are a lot easier to figure out.

The same method was used to make a **3x3** magic square:

```
8 1 6
3 5 7
4 9 2
```

It is directly obvious that this construction is is really only one of a whole family. Instead of generating the diagonals going up/right, we could also have chosen to go down/left or up/left or down/right, appropiately changing the step we do when we come to an already filled in square.

But the family is more general in fact. Consider for example the follwing description of the pyramid method taken from Mathematics Enrichment.

The Pyramid method or extended diagonals consists of three steps:

1. Draw a pyramid of same size squares as the magic square's squares, on each side of the magic square. The pyramid should be two less, in number of squares on its base, than the number of squares on the side of the magic square.
2. Sequentially place the numbers 1 to n^2 of the nxn magic square in the diagonals as shown in figure 1.
3. Relocate any number not in the nxnsquare to the opposite hole inside the square (shaded).
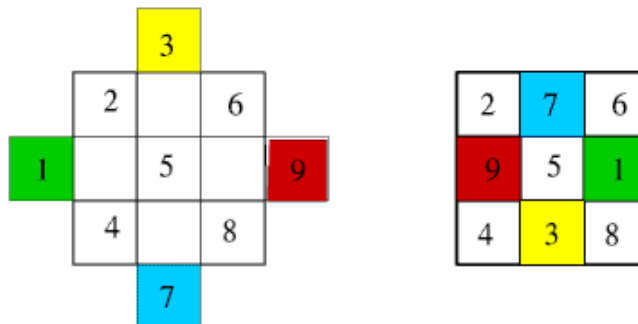
The same Pyramid method can be used for any odd order magic square as shown below for the 5x5 square.
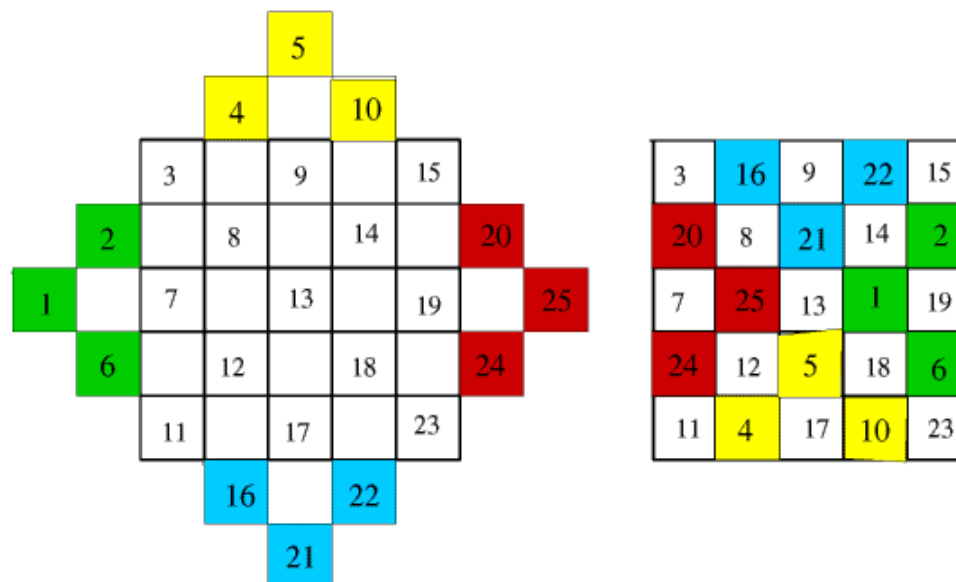
Now if you carefully observe the resulting magic squares, you see that it is in fact just the standard Siamese method, but with a different starting point (one to the right of the center) and another displacement in case a square is already occupied (go 2 to the right)

The standard dispacement also does not have to be a simple diagonal. Consider for example using a knights jump down right as standard move and moving up two as exceptional move:

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

However, not just everything works. Consider for example going one to the right as standard move and one down as exceptional move:

```
 1  2  3  4  5
 7  8  9 10  6
13 14 15 11 12
19 20 16 17 18
```

```
        25 21 22 23 24
```

Here even the rows and columns don't sum to the magic constant, and it's obvious that moving the starting point won't change this (you'll still end up with a row containing a permutation of 1,2,3,4,5 and another row containing a permutation of 21,22,23,24,25 which obviously don't have the same sum).

So the question arises: what exactly **are** the rules ?

3. Formalizing the construction

Let's denote the amount of times we applied the standard move by i (start counting at 0), and the amount of times we did the exceptional move by j (again start counting at 0). And let's apply a coordinate system to the square, where x runs from left to right and y runs from top to bottom, again both counting from 0. And let's call the x and y coordinates of the initial number a' and b' respectively. Let's also call the dispacements of the normal step in the x and y direction u' and v'.

So for the first few steps in the Siamese method we get something like:

```
x = i*u'+a'
y = i*v'+b'
```

In the first example we started in the middle of the top row, so with a=(n-1)/2 and b = 0, and a standard dispacement of (1, -1), so that gives:

```
x =  i+(n-1)/2
y = -i
```

Oops, we already get into trouble at i=1, where y goes negative (the point where in the construction we should wrap from top to bottom). And at i = (n+1)/2 we get x=n, the point where we should wrap from right to left. We can take care of the wrapping by taking x and y modulo n, the size of the square, so we get:

```
x = (i*u'+a')%n
y = (i*v'+b')%n
```

In modular arithmetic it's directly obvious that i=n is equivalent to i=0, so we will get to the point (a', b') again. This might be when we must do the exceptional move. However, it's also possible that already for some lower value of i we got back to the initial point. So let's call the lowest value for which this happens i0 (so we know 0 < i0 <= n). This implies that i0*u' = k * n. Now consider the multiplicative factors of n. Each of them must be in either i0 or u. If none are in u', they are all in i0, which must then be a multiple of n, and, since we know i0<=n, i0=n. So i0 != n can only happen when gcd(u', n) = f != 1. Let's again consider (with f either 1 or not):

```
i0*f*(u'/f)  = k*f*(n/f)
```

or

```
i0*(u'/f)  = k*(n/f)
```

where u'/f has no factors in common with n/f (since gcd(u', n) = f). So again all factors in n/f must be in i0, so i0=m*(n/f). The same reasoning can be applied to the y-coordinate, so i0=m'*(n/f'). The smallest non-zero number fulfilling both of these is lcm(n/f, n/f'), and for this number we have:

```
i*u = m *(n/f)*f*(u'/f)  = m *n*(u'/f)  = 0  (mod n)
i*u = m'*(n/f)*f*(v'/f)  = m'*n*(v'/f)  = 0  (mod n)
```

(a = b (mod n) is an alternative notation for a%n = b%n)
So we see that i0 = lcm(n/gcd(n, u'), n/gcd(n, v')) = n/gcd(n, u', v')

So each time i is a multiple of i0, we apply the exceptional case by adding and extra offset of p' and q' to x and y. So we get:

```
x = (i*u'+j*p'+a')%n
y = (i*v'+j*q'+b')%n
```

Notice that this is slightly different from the way we formulated the exceptional displacement up to this point, since that used to be relative to i0-1, so it correspondeds to (p'+u', q'+v'). This is (0, 1) in the original example, so (p', q') = (-1, 2), so the complete formula is:

```
x = ( i-j+(n-1)/2)%n
y = (-i+2*j)%n
```

Again it is obvious that in general when j is a multiple of n, this is equivalent to j=0, but it's possible that already before that point we end up in the same place. Let's call the first j for which that happens j0. Like before we conclude j0 = n/gcd(n, p', q'). So i has really i0 essentially different values, and j has j0 essentially different values. So we end up with at most i0*j0 different values for x and y. Since i0 <= n and j0 <= n we see that i0 = j0 =n, otherwise not all squares will get filled. This also means gcd(n, p', q') = gcd(n, u', v') = 1.

It's also interesting to see that i and j are symmetrical, so at least for filling up the matrix it doesn't matter if you exchange the normal displacement with the exceptional displacement (we don't say anything yet about if that gives a magic square or not).

So we have for both i and j basically n different values [0..n-1]. And each of these n^2 combinations gives a certain (x, y) coordinate where we fill in j*n+i+1. We also found a necessary condition for the (x, y) coordinates to be all different, but don't know yet if the condition is sufficient. A necessary and sufficient condition will in fact be if for every (x, y) we can find a (i, j) mapping to it. So let's try to invert the basic formula:

```
i*u'*q'+j*p'*q' = x*q'-a'*q'  (mod n)
i*v'*p'+j*q'*p' = y*p'-b'*p'  (mod n)
----------------------------------
i*(u'*q'-v'*p') = (x-a')*q'-(y-b')*p'  (mod n)
```

and simularly:

```
j*(u'*q'-v'*p') = (y-b')*u'-(x-a')*v'  (mod n)
```

This is obviously solvable if gcd(u'*q'-v'*p', n) = 1. (we will prove this by accident later when we are considering sequences). Let's denote the inverse element with 1/(u'*q'-v'*p'), so we get:

```
i = ((x-a')*q'-(y-b')*p')/(u'*q'-v'*p')  %n
j = ((y-b')*u'-(x-a')*v')/(u'*q'-v'*p')  %n
```

or, after introducing new names for constants:

```
u =  q'/(u'*q'-v'*p')              (mod n)
v = -v'/(u'*q'-v'*p')              (mod n)
p = -p'/(u'*q'-v'*p')              (mod n)
q =  u'/(u'*q'-v'*p')              (mod n)
```

```
a = (b'*p'-a'*q')/(u'*q'-v'*p')     (mod n)
b = (a'*v'-b'*u')/(u'*q'-v'*p')     (mod n)
```

we get:

```
i = (x*u+y*p+a)%n
j = (x*v+y*q+b)%n
```

This is in fact of exactly the same form as we started from, so viewing magic squares in the (i,j) space and the (x, y) space seems to be some kind of dual.

What if gcd(u'*q'-v'*p', n) = g != 1 ? Then

```
i*g*((u'*q'-v'*p')/g) = (x-a')*q'-(y-b')*p'  (mod n)
```

for every x and y. So if we take x=(a'+1)%n and y=b', we see that q' must be a multiple of g, and in the same way so must p'. So we are in the case gcd(n, p', q') != 1, and we know that the mapping won't be complete. So gcd(u'*q'-v'*p', n) = 1 is in fact necessary and sufficient for the inversion to work. And since the new formula is exactly like the original, we also see that gcd(u*q-v*p, n) is also necessary and sufficient for the inversion back to work. And if the inversion works in one direction, it also works in the other direction.

Let's quickly go back to the question if the conditions

```
gcd(n, u', v') = 1
gcd(n, p', q') = 1
```

are sufficient. Consider (u', v') = (2, 5), (p', q') = (7, 11) and n = 39. In that case u'*q'-v'*p' = -13 = 26 (mod 39), which is not relatively prime to 39. We know this is not invertible, so indeed the condition can't in fact be sufficient. And indeed if you consider (we ignore any offsets here, it's easy enough to do the same if they are there):

```
x = 2*i + 7*j  (mod 39)
y = 5*i +11*j  (mod 39)
```

Multiplying the second by 7 and subtracting the first times 11 gives:

```
7*y - 11 *x = 13*j  (mod 39)
```

So we see that the real reason is that the equations are not independent, and so not all (x, y) combinations are allowed. For example when x is 0, y must be 0, 13 or 26.

4. Example inversions

Let's go back to the original example:

```
x =  i-j+(n-1)/2   (mod n)
y = -i+2*j         (mod n)
```

So (u', v', p', q') = (1, -1, -1, 2). The determinant (the obvious name to give to u'*q'-v'*p') is in this case det = 2 - 1 = 1 (so gcd(det, n) = 1 for all n), whose inverse is obviously also 1. So we get (u, v, p, q) = (2, 1, 1, 1)

```
i = 2*x+y+1        (mod n)
j =   x+y-(n-1)/2 (mod n)
```

For another example we'll take the pyramid method, which we can write as:

```
x =  i+j+(n+1)/2   (mod n)
y = -i+j+(n-1)/2   (mod n)
```

So (u', v', p', q') = (1, -1, 1, 1). Therefore det = 1+1 = 2. And since we are only considering the cases where n is odd, we also have gcd(n, 2) = 1. More interesting is in this case the inverse, which is obviosuly (n+1)/2 since this is the smallest number such that 2*m = k*n+1 for k and m positive integers. So we get (u, v, p, q, a, b) = ((n+1)/2, (n+1)/2, -(n+1)/2, (n+1)/2, -(n+1)/2, 0) or:

```
i = (x-y-1)*(n+1)/2        (mod n)
j = (x+y  )*(n+1)/2        (mod n)
```

This case is particularly interesting since it shows an example of displacements that are not a constant, but multiples of (n+1)/2. And due to the duality this makes us suspect that displacements of this form can also be used in the Siamese method (this suspicion will turn out to be true. You can get magic squares like that). Also notice that -(n+1)/2 = (n-1)/2 (mod n), so you can consider them as displacements that are multiples of (n-1)/2 in the other direction.

5. Demanding the square is semi-magic

So now let's consider:

```
value(x, y) = i+j*n

where:

i = (x*u+y*p+a)%n
j = (x*v+y*q+b)%n

and:

gcd(u*q-p*v, n) = 1
```

Notice that I sneakily subtracted one from the value. This doesn't really matter, since subtracting one from every value in the square makes no difference whatsoever to the magicness or semi-magicness of a square (though it changes the value of the magic constant obviously).

We know that all magic squares generated with the Siamese method are of this form. What about the other direction ? Due to the invertability, we know that all values from 1 to n^2 will appear in the generated square. But obviously it's not always magic, for example:

```
i = x%n
j = y%n
```

Leads to this very obviously non-magic square:

```
 0  1  2  3  4
 5  6  7  8  9
10 11 12 13 14
15 16 17 18 19
20 21 22 23 24
```

Clearly we must demand that the sum of the rows and columns gives the magic constant.

Let's first determine this magic constant. The total sum of all elements in the square is obviously the mean value times the number of elements, so TotalSum = $n^2*(n^2-1)/2$. There are n rows, each of which sum to the magic constant S, and all of which sum to the TotalSum, so

```
S = TotalSum/n = n*(n²-1)/2 = n*(n-1)*(n+1)/2 = n * [n*(n-1)/2 + (n-1)/2]
```

Next let's get rid of that (a, b) offset. Since we know every (i,j) pair is realised, let's consider $(x, y) = (x_0, y_0)$ for which (i, j) = (0, 0), so

```
0 = x₀*u+y₀*p+a      (mod n)
0 = x₀*v+y₀*q+b      (mod n)
```

which we subtract from the basic formula to get:

```
i = (x-x₀)*u+(y-y₀)*p      (mod n)
j = (x-x₀)*v+(y-y₀)*q      (mod n)
```

This basically corresponds to taking off the top x_0 rows of the square and reattaching them to the bottom, and removing the left y_0 rows and attaching them to the right. This obviously does not change either the row or column sums (though it does change the diagonals). So since we are only going to demand semi-magicness in this section, it's enough to study:

```
value(x, y) = i+j*n

where:

i = x*u+y*p      (mod n)
j = x*v+y*q      (mod n)

and:

gcd(u*q-p*v, n) = 1
```

Let's fix y = 0, and let x run over [0..n-1]. then obviously the consecutive values of i and j are:

```
i  0  u%n  2*u%n  ....  (n-1)*u%n
j  0  v%n  2*v%n  ....  (n-1)*v%n
```

If we denote the sum over n multiples of u with sum(u, n), we get as condition for the total rowsum:

```
S = n * (n * (n-1)/2) + (n * (n-1)/2) = n * sum(v, n) + sum(u, n)
```

So we are led to study the sum of sequences of the form:

```
0  u%n  2*u%n  ....  (n-1)*u%n
```

The key question to ask here is if any of these values (after the first one) will be 0. Let's assume that the first multiplier for which this happens is m and that μ=gcd(u, n). So

```
gcd(u/μ, n/μ) = 1

m*μ*(u/μ) = k * μ * (n/μ)
m*(u/μ) = k * (n/μ)
```

All factors of n/μ are not in u/μ, so they must appear in m, so m is a multiple of n/μ. In the same way k must be a multiple of u/μ, so the smallest solution is

```
m = n/μ
k = u/μ
```

So the sequence to sum will look like:

```
gcd(u, n) = μ
U = u/μ
N = n/μ
gcd(U, N) = 1

μ times (0 U*μ%n 2*U*μ%n .. (N-1)*U*μ%n)
```

For example, if n=6, u=4 then μ = gcd(6, 4) = 2 and the sequence is

```
0 4 2 0 4 2
```

Which is indeed 2 times

```
0 2*2%6 2*2*2%6
0    4     2
```

So now let's consider one such subsequence. We can obviously divide out the common factor μ to get the sequence:

```
0 U%N 2*U%N .... (N-1)*U%N
```

Which in the example indeed corresponds to

```
0    2    1
```

where each element then has to be multiplied by μ=2. We also know that no other 0 except the first one will appear. And we know that each other value in the sequence will only appear once (otherwise walk back from the first one with a given value to 0. Walk equally many steps back from the second one. This must also be 0. But there is only one 0, so the positions must have been the same). So in fact the sequence is a permutation of 0..N-1 (as is again obvious to see in the example).

Since it will in fact be a permutation, there will be a 1 in the sequence, so there exists an m such that m*U%N = 1 if and only if gcd(U, N) = 1 (if you go back to the sequence for the case gcd(u, n) = μ you can see that the smallest non-zero value that will appear is μ). Which is the condition we still needed to prove for the inversion formula.

At this point we have all we need to work out the general sum. First the sequence with the factor μ divided out:

```
    sum =    0 1     2 .... N-1           (writing the sequence forward)
    sum = N-1 N-2 N-3 ....   0            (writing the sequence backward)
  ----------------------------
  2*sum = N-1+N-1+N-1 .... N-1 = N*(N-1)
    sum = N*(N-1)/2
```

So adding back the μ factor and that we have this sequence μ times gives:

```
    sum(u, n)  = μ² * N * (N-1)/2 = μ² * (n/μ) * ((n/μ)-1)/2 = n * (n-μ)/2
```

Notice that this is always less or equal to n*(n-1)/2 with equality if and only if μ=1. Combining this with

```
    S = n * (n * (n-1)/2) + (n * (n-1)/2) = n * sum(v, n) + sum(u, n)
```

shows sum(v, n) = n * (n-1)/2 and sum(u, n) = n * (n-1)/2 which will be the case if and only if gcd(u, n) = gcd(v, n) = 1.

We've also proven that if gcd(u,n) = 1 then x*u%n will assume all values in [0..n-1]. So adding a constant value to that before the modulo operation will just be a permutation of these same values. So if we consider a different row with j != 0, the row will still have the same sum. So all rows will sum to the magic constant.

And applying the same reasoning to j gives that all column sum to the magic constant if and only if gcd(p, n) = gcd(q, n) = 1.

Notice that at this point we haven't really used yet that n is odd. So can this method lead to even squares ? Since u, v, p and q must be relatively prime to n, all of them must be odd. but then det = u*q-p*v will be even, and cannot be relatively prime to n. Which is why the Siamese method won't work for even sized magic squares

6. Demanding the square is magic

So now let's consider:

```
    value(x', y') = i+j*n
```

```
    where:
```

```
    i = (x'*u+y'*p+a')%n
    j = (x'*v+y'*q+b')%n
```

```
    and:
```

```
    gcd(u*q-p*v, n) = 1
```

```
        gcd(u, n) = gcd(v, n) = gcd(p, n) = gcd(q, n) = 1
```

We know that every magic square generated with the Siamese method is of this form. What about the other direction ? Due to the invertability we know that every value will appear in the resulting magic square. Due to the conditions on u, v, p and q we know that all sums and rows sum to the magic constant. What remains is to study the diagonals.

The two diagonals meet in the center. To make the situation more symmetric, we will move the (x, y) coordinates so that (0, 0) is the center. So we introduce

```
    x = x'-(n-1)/2       (mod n)
    y = y'-(n-1)/2       (mod n)
```

so:

```
    i = (x+(n-1)/2)*u+(y+(n-1)/2)*p+a' = x*u+y*p+(a'+(u+p)*(n-1)/2) (mod n)
    j = (x+(n-1)/2)*v+(y+(n-1)/2)*q+b' = x*v+y*q+(b'+(v+q)*(n-1)/2) (mod n)
```

So by defining

```
    a = a'+(u+p)*(n-1)/2        (mod n)
    b = b'+(v+q)*(n-1)/2        (mod n)
```

We get the familiar:

```
    value(x, y) = i+j*n

    where:

    i = (x*u+y*p+a)%n
    j = (x*v+y*q+b)%n

    and:

    gcd(u*q-p*v, n) = 1
    gcd(u, n) = gcd(v, n) = gcd(p, n) = gcd(q, n) = 1
```

So let's run x and y from -(n-1)/2 to (n-1)/2. We get the following sequences

```
    i: [-(n-1)/2*(u+p)+a]%n, [(1-(n-1)/2)*(u+p)+a]%n .... a%n .... [((n-1)/2-1)*(u+p)+a]%n [(n-1)/2*(u+p)+a]%n
    j: [-(n-1)/2*(v+q)+b]%n, [(1-(n-1)/2)*(v+q)+b]%n .... b%n .... [((n-1)/2-1)*(v+q)+b]%n [(n-1)/2*(v+q)+b]%n
```

Let's denote the sum of the i sequence as sum(u+p, a, n), then we get for this diagonal:

```
    S = n * (n * (n-1)/2) + (n * (n-1)/2) = n * sum(v+q, b, n) + sum(u+p, a, n)
```

For the other diagonal we let x run from -(n-1)/2 to (n-1)/2 while y runs from (n-1)/2 to -(n-1)/2. This gives:

```
    i: [-(n-1)/2*(u-p)+a]%n, [(1-(n-1)/2)*(u-p)+a]%n .... a%n .... [((n-1)/2-1)*(u-p)+a]%n [(n-1)/2*(u-p)+a]%n
    j: [-(n-1)/2*(v-q)+b]%n, [(1-(n-1)/2)*(v-q)+b]%n .... b%n .... [((n-1)/2-1)*(v-q)+b]%n [(n-1)/2*(v-q)+b]%n
```

so

```
S = n * (n * (n-1)/2) + (n * (n-1)/2) = n * sum(v-q, b, n) + sum(u-p, a, n)
```

So this time we study a sequence of the form

```
(-(n-1)/2*u+a)%n, ((1-(n-1)/2)*u+a)%n, ... (-u+a)%n, a%n, (u+a)%n, ... (((n-1)/2-1)*u+a)%n, ((n-1)/2*u+a)%n
```

It will be interesting to see if the value a%n is repeated in this sequence. so let's again call gcd(u, n) = μ. We will get a%n again if (m*u+a) = a (mod n), so if m*u = 0 (mod n) which happens for the first time for m = n/μ = N as we showed higher. This time we will use that n is odd, so all factors are odd, so m is odd and we can write it as m = 2*k+1. So we can write the repeated sequence as

```
(-k*u+a)%n, ((1-k)*u+a)%n, .... (-u+a)%n, a%n, (u+a)%n, ... ((k-1)*u+a)%n, (k*u+a)%n
```

which will appear μ times. Since we want to try to get rid of a factor μ again, let's write a as a=å*μ+A with 0 <= A < μ , so the sequence is

```
(-k*U*μ+å*μ+A)%n, ((1-k)*U*μ+å*μ+A)%n, .... (-U*μ+å*μ+A)%n, å*μ+A%n, (U*μ+å*μ+A)%n, ... ((k-1)*U*μ+å*μ+A)%n, (k*U*μ+å*μ+A)%n
```

This sequence consists of multiples of μ (which divides n) plus A where 0 <= A < μ. So in fact A will never cause this multiple of μ to go over a multiple of n, so we can move it outside the modulo operation. So we get μ times the sequence:

```
(-k*U*μ+å*μ)%n+A, ((1-k)*U*μ+å*μ)%n+A, .... (-U*μ+å*μ)%n+A, å*μ%n+A, (U*μ+å*μ)%n+A, ... ((k-1)*U*μ+å*μ)%n+A, (k*U*μ+å*μ)%n+A
```

So if we remove n times A from the elements of the sequence, we get

```
(-k*U*μ+å*μ)%n, ((1-k)*U*μ+å*μ)%n, .... (-U*μ+å*μ)%n, å*μ%n, (U*μ+å*μ)%n, ... ((k-1)*U*μ+å*μ)%n, (k*U*μ+å*μ)%n
```

Which is of the form we already handled, and equals sum(u, n) (the å additions again only cause an extra permutation, but do not change the sum). So

```
sum(u, a, n) = sum(u, n) + n * A = n*(n-μ)/2+n*A = n*(n+2*A-μ)/2
```

So the demand that:

```
S = n * (n * (n-1)/2) + (n * (n-1)/2) = n * sum(v+q, b, n) + sum(u+p, a, n)
```

leads us to consider:

```
n * (n * (n-1)/2) + (n * (n-1)/2) = n * (n*(n+2*B-ð)/2) + n*(n+2*A-μ)/2
n * (2*B-ð+1) + (2*A-μ+1) = 0
```

Let's consider 2*A-μ+1. The possible values of A are in [0..μ-1], so the possible values of this run in [-(μ-1)..μ-1] and μ <= n. So this value can never compensate for a multiple of n. Therefore:

```
      2*A-μ+1 = 0
      2*B-ð+1 = 0
```

or, if we consider it our job to determine a and b once u, v, p and q are known:

```
      A = (μ-1)/2
      B = (ð-1)/2
```

Each diagonal will lead to a condition of this kind. They must of course be compatible.

Let's recap what we worked out in the previous section:

```
      μ = gcd(u+p, n)
      A = a%μ
      ð = gcd(v+q, n)
      B = b%ð

      2*A = μ-1
      2*B = ð-1

      μ' = gcd(u-p, n)
      A' = a%μ'
      ð' = gcd(v-q, n)
      B' = b%ð'

      2*A' = μ'-1
      2*B' = ð'-1
```

So the compatibility conditions become:

```
      a = s * μ + A = s' * μ' + A'
      b = t * ð + B = t' * ð' + B'
```

with s, s', t and 't integers

```
      s * μ + A = s' * μ' + A'
      s * μ + (μ-1)/2 = s' * μ' + (μ'-1)/2
      (2*s+1)*μ = (2*s'+1)*μ'
```

So we must find out more about gcd(μ, μ') = g. Since g | μ and g | μ' and since μ | n, we see that g | n (| is a notation for "divides"). Also g | u+p and g | u-p, so summing these we see g | 2*u. Since we know n is odd, we get μ is odd and therefore g is odd, so g | u. but that means that g | gcd(n, u). But gcd(u, n) = 1, so g = 1.

So each factor of μ' must appear in 2*s+1, and each factor of μ must appear in 2*s'+1, so:

```
      2*s'+1 = (2*k+1) * μ
      2*s+1  = (2*k+1) * μ'
```

With k an integer (we know the factor is odd because 2*s'+1 is odd). So:

```
a = s * μ + (μ-1)/2 = ((2*s+1)*μ-1)/2 = ((2*k+1)*μ*μ'-1)/2 = k*μ*μ'+(μ*μ'-1)/2
```

Formulating a in terms of μ' leads to the same formula, so we found the necessary and sufficient condition for compatibility of a. And in the same way we find:

```
b = ((2*l+1)*ð*ð'-1)/2 = l*ð*ð'+(ð*ð'-1)/2
```

And translating back to (x, y) we see:

```
a' = ((2*k+1)*μ*μ'-1)/2-(u+p)*(n-1)/2 = k'*μ*μ'+(1-u-p)*(μ*μ'-1)/2
b' = ((2*l+1)*ð*ð'-1)/2-(v+q)*(n-1)/2 = l'*ð*ð'+(1-v-q)*(ð*ð'-1)/2
```

In the last step I used that μ | n, μ' | n, and gcd(μ, μ') = 1, so μ*μ' | n. And since all of these must be odd, we can write n = (2*M+1)*μ*μ', and in a simular way that n = (2*E+1)*ð*ð'. Substituting this and some rearrangement and renaming of the arbitrary constant then gives the result.

7. Main formula

So the final conclusion is that we get a magic square generated with the Siamese method if and only if:

```
value(x, y) = i+j*n

where:

i = (x*u+y*p+k*μ*μ'+(1-u-p)*(μ*μ'-1)/2)%n
j = (x*v+y*q+l*ð*ð'+(1-v-q)*(ð*ð'-1)/2)%n

and:

gcd(u*q-p*v, n) = gcd(u, n) = gcd(v, n) = gcd(p, n) = gcd(q, n) = 1
μ  = gcd(u+p, n)
μ' = gcd(u-p, n)
ð  = gcd(v+q, n)
ð' = gcd(v-q, n)

and k and l are arbitrary integers.
```

We can also write i and j as:

```
i = ((2*x-n+1)*u+(2*y-n+1)*p+(2*k'+1)*μ*μ'-1)/2%n
j = ((2*x-n+1)*v+(2*y-n+1)*q+(2*l'+1)*ð*ð'-1)/2%n
```

8. Example formulas

Let's first consider the original example again.

```
(u, v, p, q) = (2, 1, 1, 1)
μ  = gcd(3, n)
μ' = gcd(1, n) = 1
```

```
ð  = gcd(2, n) = 1
ð' = gcd(0, n) = n
```

So we must distinguish two cases:
   1. gcd(n, 3) = 1

```
i = (2*x+y+k')%n
j = (x+y-(n-1)/2)%n
```

   2. gcd(n, 3) = 3 or n = 3*(2*m+1)

```
i = (2*x+y+3*k'+1)%n
j = (x+y-(n-1)/2)%n
```

And if we want a formula that handles both cases, we get:

```
i = (2*x+y+3*k'+1)%n
j = (x+y-(n-1)/2)%n
```

In the previous case it was very interesting that the i formula had an almost arbitrary offset. It was in a way caused by μ and μ' being small constants. Let's try to get that effect on both i and j:

```
(u, v, p, q) = (2, 2, 1, -1)
det = u*q-v*p = -2-2 = -4, which is prime relative to all odd numbers, so the square will be magic.
µ  = gcd(3, n)
µ' = gcd(1, n) = 1
ð  = gcd(1, n) = 1
ð' = gcd(3, n)
```

Again we must distinguish two cases:
   1. gcd(n, 3) = 1

```
i = (2*x+y+k')%n
j = (2*x-y+l')%n
```

   2. gcd(n, 3) = 3 or n = (2m+1)*3

```
i = (2*x+y+3*k'+1)%n
j = (2*x-y+3*l')%n
```

And if we want a formula that handles both cases, we get:

```
i = (2*x+y+3*k'+1)%n
j = (2*x-y+3*l')%n
```

If we for example take k'=1 and l'=0, and add 1 to all values again, we get for n=5 the following magic square:

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

It's interesting to note that the average value, 13, does not appear in the center.

9. Demanding the square is associative

So what if we decided to move the 13 to the center ? We can take two columns from the right side and reattach them at the left, and take one row from the top and attach it to the bottom. This sort of operation would obviously not impact the semi-magicness of the square, but it might disturb the diagonals, so the result might not be a magic square.

In the example under consideration 2 would move to the top left, and this fact of course completely determines this "translation". In general, we can decide to move the number at (x_0, y_0) to $(0, 0)$ so we go from the original

```
i = x*u+y*p+a      (mod n)
j = x*v+y*q+b      (mod n)

x' = x-x₀          (mod n)
y' = y-y₀          (mod n)
```

to

```
i = (x'+x₀)*u+(y'+y₀)*p+a = x'*u+y'*p+(a+x₀*u+y₀*p) = x'*u+y'*p+a'  (mod n)
j = (x'+x₀)*v+(y'-y₀)*q+b = x'*v+y'*q+(b+x₀*v+y₀*q) = x'*v+y'*q+b'  (mod n)

a' = a+x₀*u+y₀*p   (mod n)
b' = b+x₀*v+y₀*q   (mod n)
```

(I used the (mod n) notation here instead of %n to stress that this works whether we are considering the range [0..n-1] or [-(n-1)/2..(n-1)/2] or any other such range). So every translation leaves the multipliers for x and y unchanged, but just changes the offsets. And since there are n^2 different values we can move to $(0, 0)$ (each of which must give different offsets since the value on $(0, 0)$ is different), and n*n essentially different values for the offsets, each change in offsets also corresponds to a translation (notice how this reasoning fails if the multipliers are not linear independent since not all $(0, 0)$ results would be distinguishable).

So if we choose the average value in the center and use a coordinate system with $(0, 0)$ in the middle, it's obvious that for $(0, 0)$ we must have $i = j = (n-1)/2$, so:

```
i = (x*u+y*p+k*n+(n-1)/2)%n
j = (x*v+y*q+l*n+(n-1)/2)%n
```

(I left in the k and l to stress the complete set of solutions. In practise they of course have no influence and you just drop them).

If you now consider two points in positions symmetric with respect to the center, their x and y coordinates have opposite sign, so equally much gets added or subtracted from $(n-1)/2$. And since 0 an $n-1$ are equally far from $(n-1)/2$, if the modulo wraps -1 or lower up, the other position is at n or higher and gets wrapped down equally much. So the sum of the values at symmetric positions is constant, and in fact twice the average (and now central) value, $(n-1)/2*n+(n-1)/2 = (n-1)/2*(n+1) = (n^2-1)/2$

In fact, a square is called "associative" if all positions symmetric to the center sum to n^2+1 (this definition is for squares that start counting at 1). So that is n^2-1 for 0-based squares, so moving the average value to the center makes a semi-magic siamese square associative.

And since the whole square is on average filled with the average value and all pairs (except the center) sum to twice the average value, every associative semi-magic square must have the average value in the center, so the two conditions are equivalent for siamese squares.

Using that $n = (2*M+1)*\mu*\mu'$ and $n = (2*E+1)*\eth*\eth'$ gives for the offsets:

```
a = ((2*k+1)(2*M+1)*µ*µ'-1)/2
b = ((2*l+1)(2*E+1)*ð*ð'-1)/2
```

which is compatible with the condition for the diagonals to sum to the magic constant, so every associative semi-magic Siamese square is magic. So moving the average value to the center in the example in fact didn't destroy the magicness. On the contrary, it's a way you can make any semi-magic Siamese square magic.

Translating back to zero-based coordinates for (x, y) gives:

```
i = (x*u+y*p+(1-u-p)*(n-1)/2)%n
j = (x*v+y*q+(1-v-q)*(n-1)/2)%n
```

Going back to our standard example (u, v, p, q) = (2, 1, 1, 1) gives:

```
i = (2*x+y+1)%n
j = (  x+y-(n-1)/2)%n
```

which is the exact formula we found for the example inversions. So the classic construction gives associative magic squares.

The other example we recently looked at was (u, v, p, q) = (2, 2, 1, -1). This gives:

```
i = (2*x+y+1)%n
j = (2*x-y)%n
```

Notice how choosing u+p and v+q to be odd tends to get rid of the ugly offsets.

10. Demanding the square is panmagic

When **all** the diagonals, including those obtained by "wrapping around" the edges, of a magic square sum to the same magic constant, the square is said to be a panmagic square.

In this example from MathWorld you see at the left the normal magic square condition and to the right the conditions for all wrapped diagonals (in both directions):
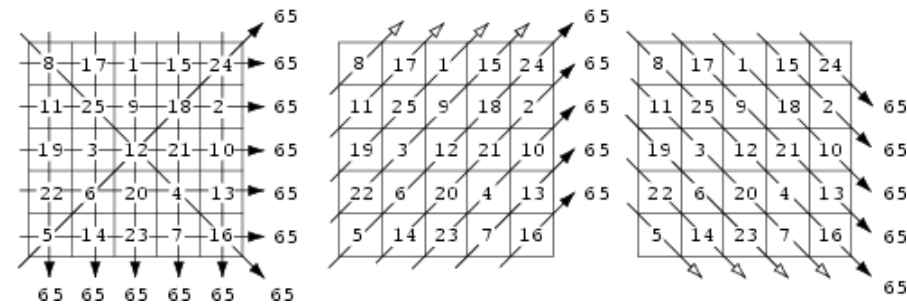
If you go back to the sequences we studied when demanding a square is magic, it's easy to see that e.g. moving z times one down from the main diagonal corresponds to adding z*p to the i entries and z*q to the j entries. Each of these must give a component of the magic constant (We only demanded that all diagonals sum to the same value, but since in this way we cover all positions, this value must be the magic constant. And the diagonal compatibility condition will also be fullfilled). So:

```
n * (n-1)/2 = sum(u+p, a+z*p, n) = sum(u+p, n) + (a+z*p)%µ * n
n * (n-1)/2 = sum(v+q, a+z*q, n) = sum(v+q, n) + (a+z*q)%ð * n
```

(moving the diagonals to the right instead of down would correspond to adding z*u and z*v and will of course lead to the same conclusions, since in the end they represent the same diagonals). Since each of these must give the same result irrespective of z, it follows that (a+z*p)%µ is constant. Considering z=0 and z=1 we see that µ | p, and assuming this is also enough to to make (a+z*p)%µ constant. But we also have µ | u+p, so µ | u. And since µ | n, we also see that µ | gcd(u, p), so µ=1. Doing the same for the others, we see that the necessary and sufficient condition for a semi-magic square to be panmagic is:

```
µ = µ' = ð = ð' = 1
```

So a Siamese square is panmagic if and only if:

```
value(x, y) = i+j*n

where:

i = (x*u+y*p+k)%n
j = (x*v+y*q+l)%n

and:

gcd(u*q-p*v, n) = 1
gcd(u,   n) = gcd(v,   n) = gcd(p,   n) = gcd(q,   n) = 1
gcd(u+p, n) = gcd(u-p, n) = gcd(v+q, n) = gcd(v-q, n) = 1

and k and l are arbitrary integers.
```

We can easily make the square associative too by choosing k and l so that they are of the form we determined higher.

Looking again at the examples, we see that (u, v, p, q) = (2, 1, 1, 1) implies ð' = n, so the classic construction never gives a panmagic square (except n=1).

Considering (u, v, p, q) = (2, 2, 1, -1)we see that we need gcd(3, n) = 1 as necessary and sufficient condition. So the squares generated by

```
i = (2*x+y+k)%n
j = (2*x-y+l)%n
```

are panmagic if n is not a multiple of 3, so of the form n = 6*m±1

What happens in the previous example is pretty typical. Siamese squares that are a multiple of 3 never seem to be panmagic. Why ? u+p, u-p, u and p may obviously not be multiples of 3, otherwise their gcd with n is at least 3, so u = ±1 (mod 3) and p = ±1 (mod 3) which means that either the sum or the difference must be 0 (mod 3), so either u+p = 0 (mod 3) or u-p = 0 (mod 3), which is not allowed. So the Siamese method never generates panmagic squares whose size is a multiple of 3. And in fact no panmagic squares of size 3 exist at all, Siamese or not.

11. Decomposing a magic square into magic carpets

In a size n square, we can write each value as a+n*b with 0 <= a, b < n. So when we write the a and n*b squares separate, we get for example from the magic square generated by:

```
i = (2*x+y+4)%n
j = (2*x-y)%n
```

the following decomposition:

```
 4 11 23  5 17        4  1  3  0  2         0 10 20   5 15
20  7 19  1 13        0  2  4  1  3        20  5 15   0 10
16  3 10 22  9   =    1  3  0  2  4    +   15  0 10 20   5
12 24  6 18  0        2  4  1  3  0        10 20  5 15   0
 8 15  2 14 21        3  0  2  4  1         5 15  0 10 20
```

or, if we divide out the factor 5 in the last square we get these two patterns:

```
4  1  3  0  2        0  2  4  1  3
0  2  4  1  3        4  1  3  0  2
1  3  0  2  4        3  0  2  4  1
2  4  1  3  0        2  4  1  3  0
3  0  2  4  1        1  3  0  2  4
```

These two squares of course correspond to the i and j squares since the values in a Siamese square are i+n*j. One possible name for this kind of regular patterns with repeated values is "magic carpets".

In for example the i square you can clearly see how moving to the right each time adds u (2), while moving down each time adds p (1). Moving along the diagonal from top-left to bottom-right adds u+p (3), while moving from bottom-left to top-right adds u-p (1). In the j square moving to the right adds v (2), while moving down adds q(-1) and the diagonals add v+q and v-q, all of this modulo 5 of course.

But we in fact studied this kind of sequence already when calculating sums over it. In particular, we studied when the patterns repeats, which didn't happen if the step was relatively prime to n. And since gcd(u, n)=1, we know that the values in the i row won't repeat for a semi-magic square. The same can be concluded for the columns and all of this for the j square too. Such a square with multiple values none of which repeat on any row or column are often called "latin squares", especially when they are written with letters instead of numbers. E.g. we could write the two example squares with letters as:

```
E   B   D   A   C           a   c   e   b   d
A   C   E   B   D           e   b   d   a   c
B   D   A   C   E           d   a   c   e   b
C   E   B   D   A           c   e   b   d   a
D   A   C   E   B           b   d   a   c   e
```

And writing the two together gives a so-called "Graeco-Latin square" (normally written with greek and latin letters, but since most greek letters aren't in latin-1, i use upper- and lowercase letters:

```
Ea  Bc  De  Ab  Cd
Ae  Cb  Ed  Ba  Dc
Bd  Da  Ac  Ce  Eb
Cc  Ee  Bb  Dd  Aa
Db  Ad  Ca  Ec  Be
```

which gives us another way to represent a semi-magic square.

Demanding that the diagonals don't repeat any value in the i square corresponds to $gcd(u+p, n) = gcd(u-p, n) = 1$ which if you combine it with the corresponding demand for the j square is obviously the condition we found for panmagic squares, so such carpets don't just give magic squares as you might have expected, but are directly panmagic. Demanding uniqueness on one diagonal obviously also does the same for the parallel broken diagonals, so this is explains why the panmagic condition enforced the relative primeness in the diagonal directions.

Another condition that was needed for a magic square was that each (i, j) combination was possible (so that each value actually appears in the generated square). So if you consider any value in the i square, and look at the corresponding positions in the j-square, you must get each j value exactly once (and the other way round of course).

As an application consider the following (non Siamese of course) 4x4 panmagic square:

```
 0   7  12  11
13  10   1   6
 3   4  15   8
14   9   2   5
```

which has the interesting extra property that each sub 2x2 square (even when wrapping over the edges) adds to the magic constant, for example 0+7+13+10 = 30. Can we make the same kind of thing true for our Siamese squares ?

So consider a rectangular window of size w x h which would give the same sum of values wherever you place it (including wrapping). Choose a certain value in the j square, and consider all the position of this value as places to put the top-left corner of the rectangle. Fixing this value fixes all the j-value in all other positions in the rectangle since moving right is adding v and moving down is adding q (modulo n). so the complete j contribution is fixed.

In the i square, we now have the rectangle positioned with each possible value once in the upper-left corner, and this value again fixes all other i entries by the constant offsets u and p. Now consider the two i-rectangles with for example 0 and 1 in the upper-left corner. All values in the second rectangle are one higher than in the first rectangle, except for any cases of n-1, which would become n and wrap to 0. So if the first rectangle has z times n-1, we find that we add 1 w*h times and subtract z times n when going from the first rectangle to the second. And the sum must remain constant, so w*h = z*n (so in fact w*h fixes z). The second square will now contain z times the value 0. Now consider the rectangle with 1 in the upper left and the one with 2 in the upper left. Again we see we must have z times n-1, and the third rectangle will again have z times 0, but all zeroes got one added, so we also have z times 1. Repeating this process, we see in fact that in every rectangle every value will appear z times. And all of this was irresprective of the value of j we choose, so this is true for every rectangle in the i-square. In fact, up to this point we only used the surface of the area, so it's in fact true for every shape.

Now consider two rectangles shifted by 1 down, for example:

```
x   x   x   .   .                         .   .   .   .   .

x   x   x   .   .                         X   X   X   .   .

x   x   x   .   .                         X   X   X   .   .

.   .   .   .   .                         X   X   X   .   .

.   .   .   .   .                         .   .   .   .   .
```

Since both rectangles contain the exact same set of values (w*h/n times every value), the row that gets removed from the top must be a permutation of the row that gets added at the bottom. So subrows of width w at a distance h from each other are each others permutations. Now consider two such subrows shifted by 1 right:

```
x1  x2  x3  .   .                         .   X2  X3  X4  .

.   .   .   .   .                         .   .   .   .   .

.   .   .   .   .                         .   .   .   .   .

x5  x6  x7  .   .                         .   X6  X7  X8  .

.   .   .   .   .                         .   .   .   .   .
```

The top subrow of x's is a permutation of the bottom subrow. The same for the two subrows of X's. Going from the left pattern to the right, at the top we remove x1 and add X4. And at the bottom we remove x5 and add X8. So there are two possibilities:
  1. x1 is the same as X4
     Since every value appears only once on every row, this means that the subrow must in fact be the whole row (possibly multiple times). So w=k*n is a trivial solution (of course the bottom is also the whole row, and x5 = X8).
  2. x1 is different from X4
     In that case the same thing must have gotten removed and added at the bottom, so x1 = x5 and X4 = X8. Which, due to the uniqueness of values in a column, means that the height is a full column (or a multiple), so again we have a trivial solution h = l*n.

So the conclusion is that in a Siamese semi-magic square there never are non-trivial rectangles with a constant sum.

A very simple exercise is to prove that the following shape has a constant sum in every 5x5 panmagic Siamese quare:

```
.    .    X    .    .
.    X    X    X    .
.    .    X    .    .
.    .    .    .    .
.    .    .    .    .
```

[Ton Hospel](#)