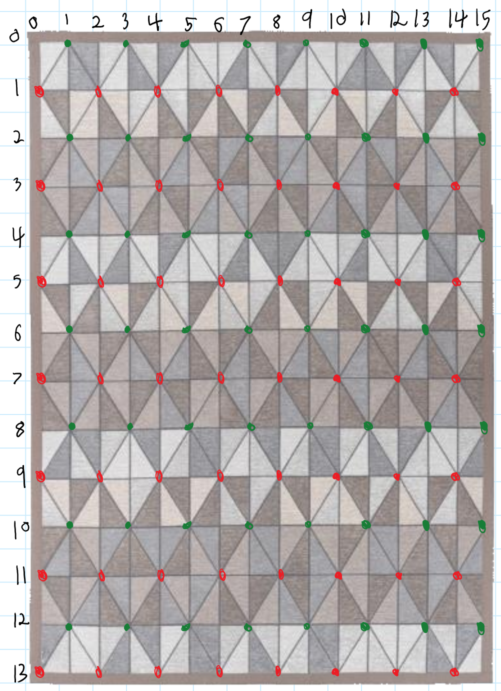
# Matt Parker's Carpet Problem

Wednesday, April 4, 2018 1:04 PM



From <http://www.narma.ee/collection/treski-t65>

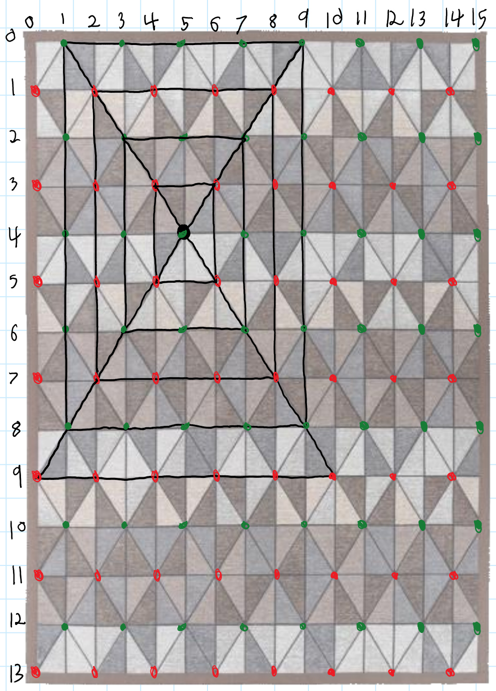
We'll split the carpet into a coordinate grid, 15 units wide (x) and 13 units tall (y).
Since every triangle has a diagonal line (can't be all right angles), every triangle on the carpet must have a corner on a di agonal line, so we can ignore the points not along diagonal lines.
These diagonal lines intersect at coordinates where x is odd and y is even (green points), and where x is even and y is odd ( red points).

If we have a function triangles_around(x,y,x_max,y_max) that takes in a coordinate (and the rug size) and returns the total n umber of unique triangles around the point (that won't be counted by other points), we can sum these with the following equation:

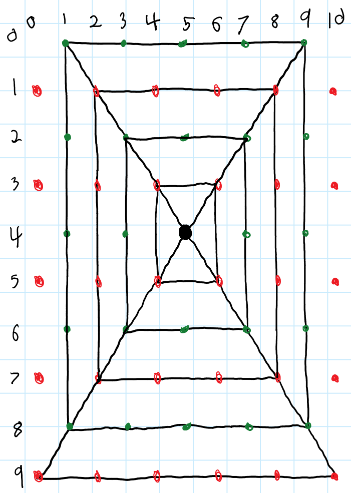
$$\sum_{x=0}^{x_{max}}\left(\sum_{y=0}^{y_{max}}\left(\begin{cases} triangles\_around(x,y,x_{max},y_{max}), & x\%2=0\ XOR\ y\%2=0 \\ 0, & ELSE \end{cases}\right)\right)$$

Translated into python code:

```
def total_triangles(x_max, y_max):
    total = 0
    for x in xrange(0, x_max+1):
        for y in xrange(0, y_max+1):
            val = 0
            if x%2 ^ y%2:
                val = triangles_around(x, y, x_max, y_max)
            total = total + val
    return total
```



From <http://www.narma.ee/collection/treski-t65>

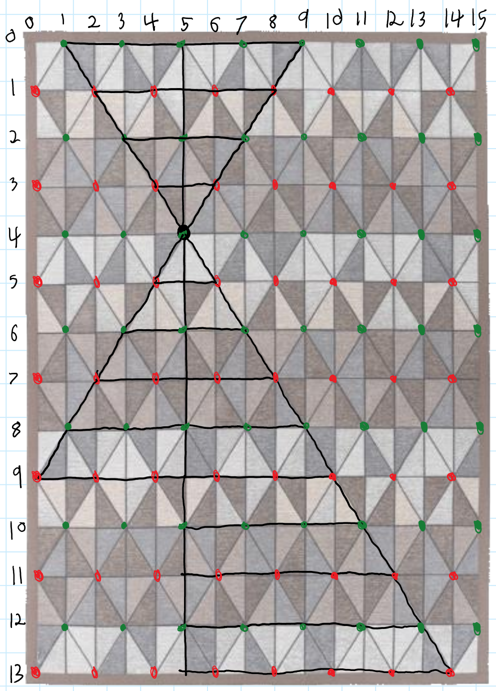


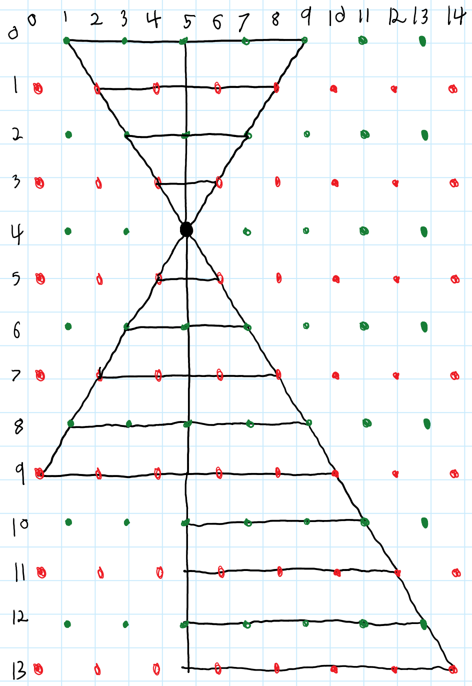
We will focus on the green point at x=5, y=4, or (5,4) for an arbitrary example.
First, we'll focus on the isosceles triangles shown here. The line extending up and to the left will extend by x or y units, whichever is smaller. Also, the line extending down and to the right can extend by (15-x) or (13-y) units, whichever is smaller. The number of triangles above the point will be either the length of the top left line or the top right line, whichever is smaller. This would be the minimum of x, y, and (15-x), which I will show as min([x,y,(15-x)]). This makes the total number of isosceles triangles around the point:

```
isosc_around(x,y,x_max,y_max) = min([x,y,(x_max-x)])
                              + min([x,y,(y_max-y)])
                              + min([x,(y_max-y),(x_max-x)])
                              + min([y,(y_max-y),(x_max-x)])
```

In python:

```
def isosc_around(x,y,x_max,y_max):
    top = min([x,y,(x_max-x)])
    left = min([x,y,(y_max-y)])
    bottom = min([x,(y_max-y),(x_max-x)])
    right = min([y,(y_max-y),(x_max-x)])
    return sum([top, left, bottom, right])
```





Shown here are the right triangles around the point. There are also some triangles not shown to the left and right of the point, but these will all be counted by the opposite (non-right-angled) point on the triangle. For example, the triangle between (1,0),(1,4), and (5,4) is not counted by (5,4), but it will be included by (1,0). These triangles each correspond to a single diagonal reaching out as far as possible in each direction, so the total number of right triangles is:

right_around(x,y,x_max,y_max) = min([x,y])
+ min([x,(y_max-y)])
+ min([y,(y_max-y)])
+ min([(y_max-y),(x_max-x)])

In python:

```
def right_around(x,y,x_max,y_max):
    top_left = min([x,y])
    bottom_left = min([x,(y_max-y)])
    top_right = min([(x_max-x),y])
    bottom_right = min([(y_max-y),(x_max-x)])
    return sum([top_left, bottom_left, top_right, bottom_right])
```

We can now define triangles_around as:
triangles_around(x,y,x_max,y_max) = right_around(x,y,x_max,y_max)
+ isosc_around(x,y,x_max,y_max)

In python:

```
def triangles_around(x,y,x_max,y_max):
    return right_around(x,y,x_max,y_max) + isosc_around(x,y,x_max,y_max)
```

From <http://www.narma.ee/collection/treski-t65>

Since the rug in the picture on Twitter has a diagonal down and to the right in the top left corner, instead of up and to the  right (as shown here), I will have to make a slight modification to the equations, as the points are at coordinates where x a  nd y are either both even or both odd, instead of one or the other.  This can be accounted for with a simple negation of the condition  in the equation:

$$\sum_{x=0}^{x_{max}}\left(\sum_{y=0}^{y_{max}}\left(\begin{cases} triangles\_around(x,y,x_{max},y_{max}), & x\%2=0\ XNOR\ y\%2=0 \\ 0, & ELSE\end{cases}\right)\right)$$

Translated into python code:

```
def total_triangles(x_max, y_max, down_right):
    total = 0
    for x in xrange(0, x_max+1):
        for y in xrange(0, y_max+1):
            val = 0
            if x%2 ^ y%2 ^ down_right:
                val = triangles_around(x, y, x_max, y_max)
            total = total + val
    return total
```

All python code together:

```
def isosc_around(x,y,x_max,y_max):
    top = min([x,y,(x_max-x)])
    left = min([x,y,(y_max-y)])
    bottom = min([x,(y_max-y),(x_max-x)])
    right = min([y,(y_max-y),(x_max-x)])
    return sum([top, left, bottom, right])

def right_around(x,y,x_max,y_max):
    top_left = min([x,y])
    bottom_left = min([x,(y_max-y)])
    top_right = min([(x_max-x),y])
    bottom_right = min([(y_max-y),(x_max-x)])
    return sum([top_left, bottom_left, top_right, bottom_right])

def triangles_around(x,y,x_max,y_max):
    return right_around(x,y,x_max,y_max) + isosc_around(x,y,x_max,y_max)


def total_triangles(x_max, y_max, down_right):
    total = 0
    for x in xrange(0, x_max+1):
        for y in xrange(0, y_max+1):
            val = 0
            if x%2 ^ y%2 ^ down_right:
                val = triangles_around(x, y, x_max, y_max)
            total = total + val
    return total
```