Matrix Transformation Notes

David Prager Branner

14 December, 2013

1 Rotation

Make a model of how the various kinds of points move when a matrix is rotated:

a (0,0)	f (1,0)	b (2,0)
…	e (1,1)	…
d (0,2)	…	c (2,2)

Initial state

$\longrightarrow$

d (0,0)	…	a (2,0)
…	e (1,1)	f (2,1)
c (0,2)	…	b (2,2)

After 90° rotation clockwise

It may be useful to tabulate these single-point movements, to make your function easier to check:

type	label	location	location
corner	a	(0,0)	(2,0)
corner	b	(2,0)	(2,2)
corner	c	(2,2)	(0,2)
corner	d	(0,2)	(0,0)
center	e	(1,1)	(1,1)
non-corner, non-center	f	(1,0)	(2,1)

This rotation can be effected by applying the following code to every point in the matrix:

```
def rotate_point(x, y, side_length):
    return ((side_length - 1 - y), x)
```

In the example above, `side_length = 3`. Check the results of the function against the various edge cases.

Similarly, the following will bring about counterclockwise rotation:

```
def rotate_point(x, y, side_length):
    return (y, (side_length - 1 - x))
```

2 Flipping

Flipping is a different kind of transformation from rotation, because it turns the "face" of the matrix away from us. We can flip along a row ("vertically"), a column, or a diagonal. Let's use the diagonal running from lower left to upper right. As before, make a model:

a (0,0)	f (1,0)	b (2,0)
…	e (1,1)	…
d (0,2)	…	c (2,2)

Initial state

$\longrightarrow$

c (0,0)	…	b (2,0)
…	e (1,1)	f (2,1)
d (0,2)	…	a (2,2)

After 90° rotation clockwise

Tabulate for reference:

type	label	location	location
corner	a	(0,0)	(2,2)
corner	b	(2,0)	(2,0)
corner	c	(2,2)	(0,0)
corner	d	(0,2)	(0,2)
center	e	(1,1)	(1,1)
non-corner, non-center	f	(1,0)	(2,1)

Note that the points forming the diagonal on which we are flipping do not change position. All the other points move to a place symmetrically opposite their place of origin across the diagonal.

```
def flip_on_diag_lowerleft_to_upperright(x, y, side_length):
    # points change place only if not on this diagonal
    if x + y !== side_length:
        return side_length - 1 - y, side_length - 1 - x
    return x, y
```