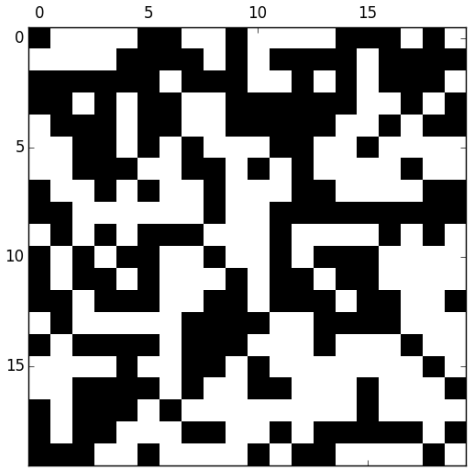
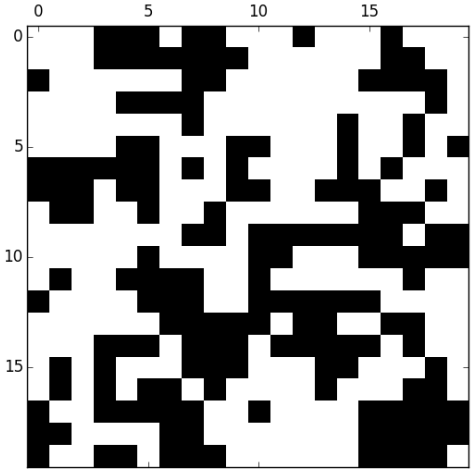
A) 40,000 iterations at each temperature performed.

T = 10



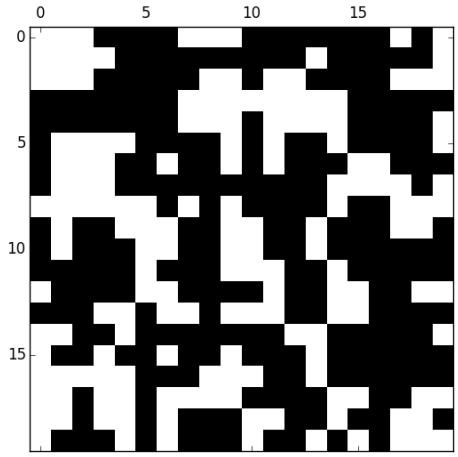
The largest cluster is about 5x5. There aren't really any well-defined clusters though.

T = 5



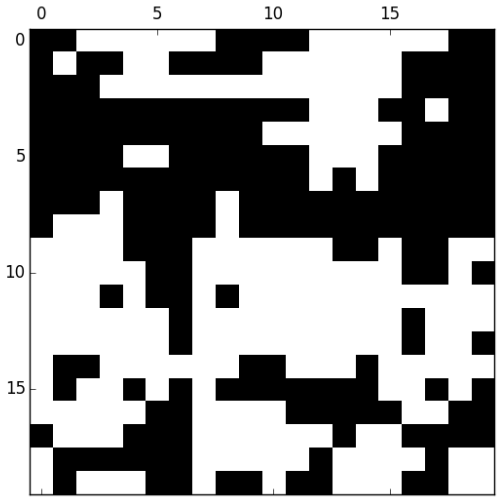
The largest cluster is about 10x10.

T = 4



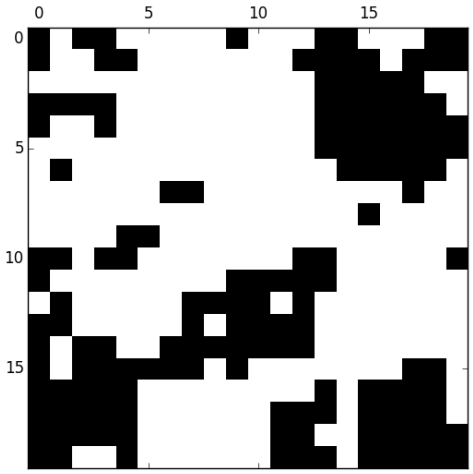
There aren't really any well-defined clusters. The largest "pseudo-cluster" is about 10x10.

T = 3



The largest cluster is about 10x20.

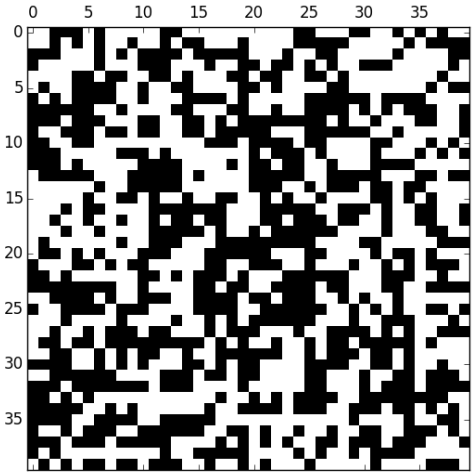
T = 2.5



The largest cluster is about 10x10.

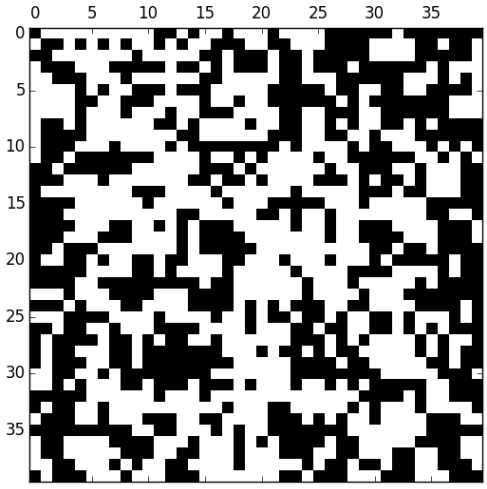
B) 160,000 iterations at each temperature performed.

T = 10



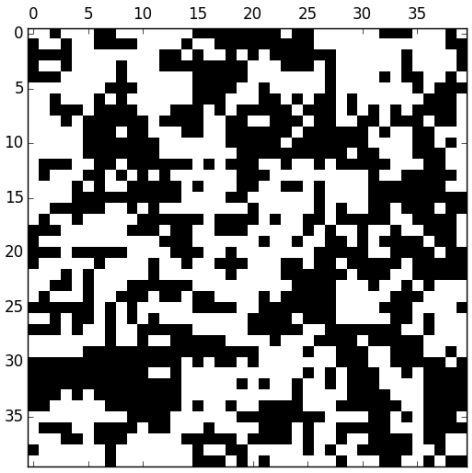
The largest cluster is about 5x5. There are not really any well-defined clusters though.

T = 5



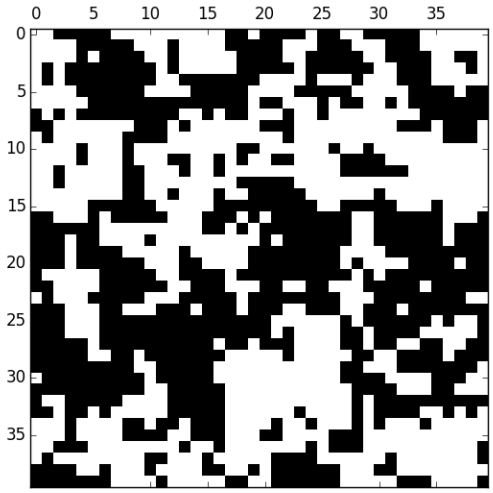
The largest cluster is about 10x10.

T = 4



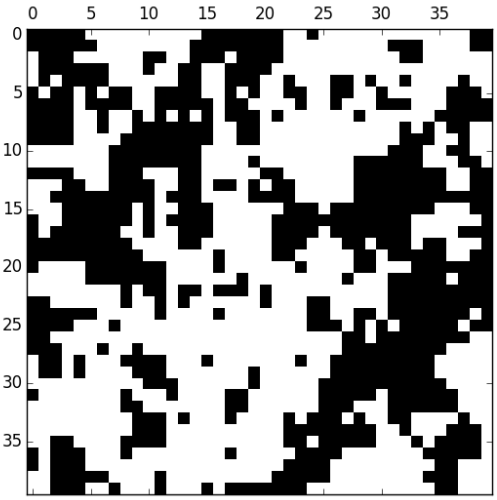
The largest cluster is about 10x20.

T = 3



The largest cluster is about 20x40.

T = 2.5

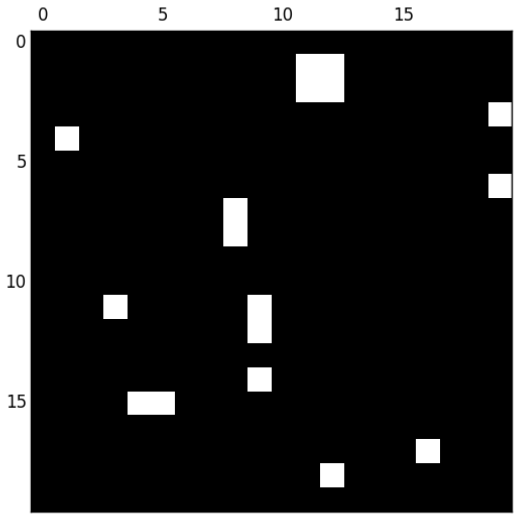


The largest cluster is about 20x40.

Generally, the clusters in the 40x40 lattice are larger than the clusters in the 20x20. However, the larger lattice also allows for larger clusters to form meaning this increase in cluster size could be due primarily to the increase in lattice size.

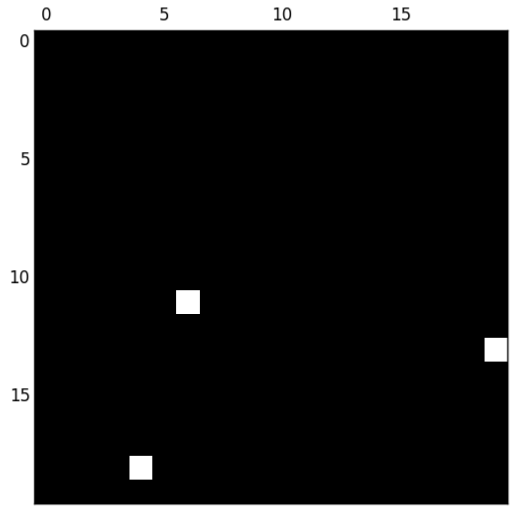
C)

T = 2



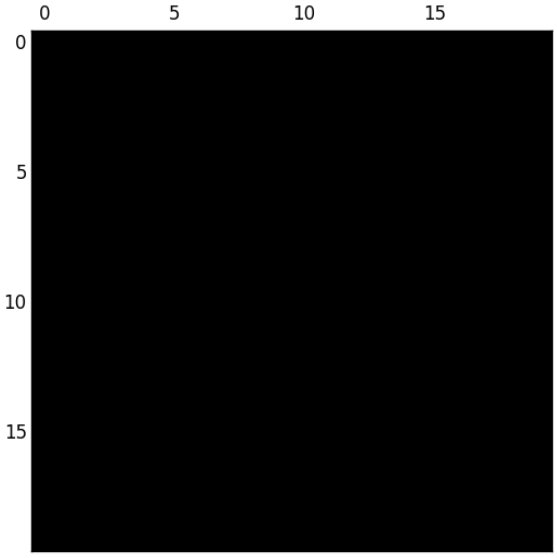
The average magnetization is about 95% up and 5% down.

T = 1.5



The average magnetization is about 99% up and 1% down.

T = 1

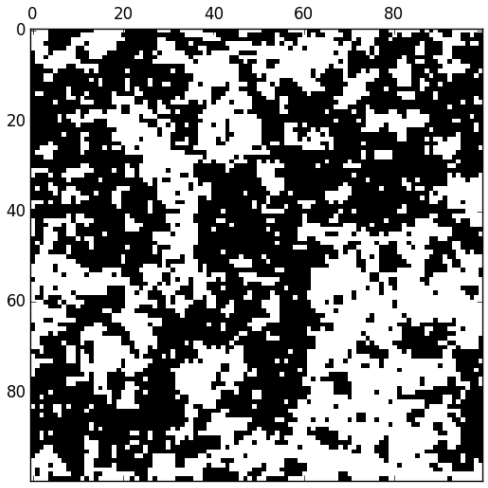


The average magnetization is 100% up.

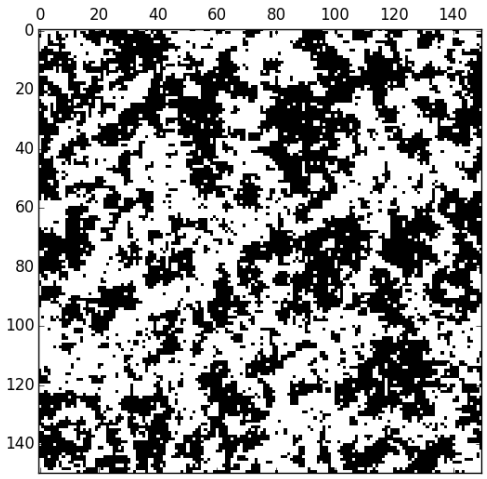
D) When running the program for a 20x20 lattice at T = 2.5, the magnetization kept flipping. It would go from being almost entirely black to being almost entirely white again and again. This comes from the Boltzmann factor that provides a random chance of flipping when flipping is energetically unfavorable. If this was not present in the program, the lattice would go to one state and stay there; however, this factor allows a dipole to flip, then allows more to flip around it.

E)

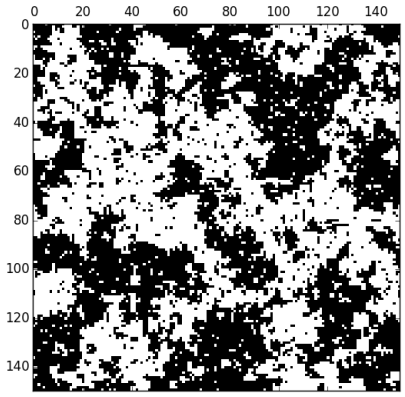
T = 2.5 S = 100



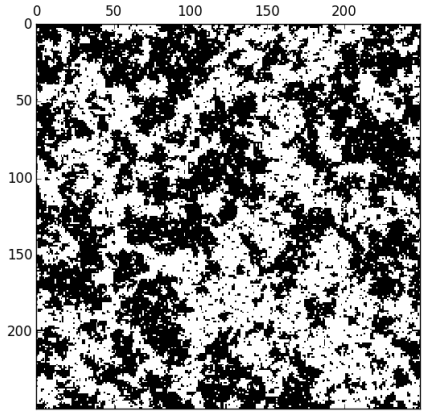
T = 2.45 S = 150



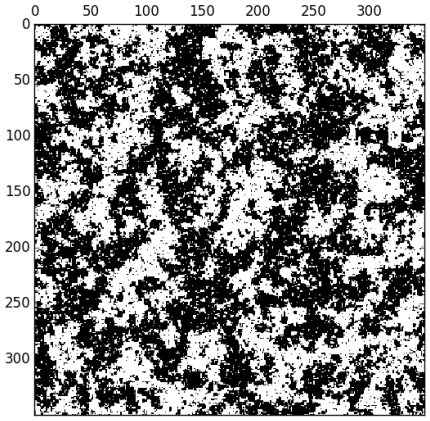
T = 2.35 S = 150



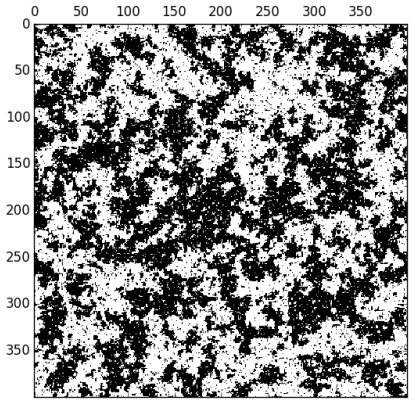
T = 2.35 S = 250



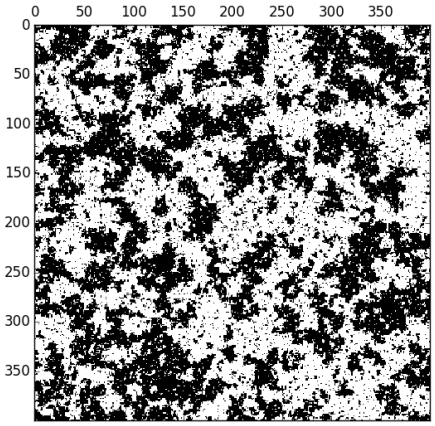
T = 2.35 S = 400



T = 2.3 S = 400



T = 2.27 S = 400



As the temperature goes from 2.5 to 2.27, the average cluster size appears to remain relatively constant. This means that it isn't really plausible that the cluster size tends to infinity as the temperature goes towards the critical temperature.

Written in Python 3.4

```
# Matt Meyers PHYS 310-010 Computer Assignment 3

import random
import math
import matplotlib.pyplot as plt
import copy
import time

SIZE = 400
T = 2.3

def initialize(s):
    for i in range(0, SIZE):
        for j in range(0, SIZE):
            if random.random() < .5:
                s[i][j] = 1
            else:
                s[i][j] = -1
    return s

def deltaU(i, j, s):
    if i == 0:
        top = s[SIZE-1][j]
    else:
        top = s[i-1][j]
    if i == SIZE-1:
        bottom = s[0][j]
    else:
        bottom = s[i+1][j]
    if j == 0:
        left = s[i][SIZE-1]
    else:
        left = s[i][j-1]
    if j == SIZE-1:
        right = s[i][0]
    else:
        right = s[i][j+1]
    return 2 * s[i][j] * (top + bottom + left + right)

def main():
    s = [[0 for x in range(SIZE)] for x in range(SIZE)]
    s = initialize(s)

    p = copy.deepcopy(s)
    for i in range(0, SIZE):
        for j in range(0, SIZE):
            if p[i][j] == 1:
                p[i][j] = 0
            else:
                p[i][j] = -p[i][j]
    plt.matshow(p, cmap=plt.cm.gray)
    plt.show(block=False)
    plt.savefig("Initial_S"+str(SIZE)+"_T"+str(T)+".png")
    plt.hold(False)
    time.sleep(1)
```

```
    for iteration in range(100*(SIZE**2)):
        i = random.randint(0, SIZE-1)
        j = random.randint(0, SIZE-1)
        Ediff = deltaU(i, j, s)
        if Ediff <= 0:
            s[i][j] = -s[i][j]
        else:
            if random.random() < math.exp(-Ediff/T):
                s[i][j] = -s[i][j]

        # This code animates the graph. Remove #s to see animation. Otherwise leave
commented to see initial and final.
        #r = copy.deepcopy(s)
        #for i in range(0, SIZE):
        #    for j in range(0, SIZE):
        #        if r[i][j] == 1:
        #            r[i][j] = 0
        #        else:
        #            r[i][j] = -r[i][j]
        #plt.matshow(r, fignum=False, cmap=plt.cm.gray)
        #plt.draw()
        #plt.hold(False)

    r = copy.deepcopy(s)
    for i in range(0, SIZE):
        for j in range(0, SIZE):
            if r[i][j] == 1:
                r[i][j] = 0
            else:
                r[i][j] = -r[i][j]
    plt.matshow(r, fignum=False, cmap=plt.cm.gray)
    plt.savefig("Final_S"+str(SIZE)+"_T"+str(T)+".png")
    plt.show(block=True)

main()
```