Κούτρας Δημήτριος

Αριθμώς Μητρώου: 355

Για λόγους απλότητας έχω χωρίσεις το πρόβλημα σε δύο υποπρογράμματα. Το πρώτο ασχολείται με κυψελιδωτά αυτόματα τα οποία έχουν δυαδικές καταστάσεις, ο κανόνας δίνεται σε δεκαδικό σύστημα και η αρχική κατάσταση έχει όλες τις τιμές μηδενικές εκτός απο την κεντρική. Τρέχω ενα σύνολο απο πειράματα για διαφορετικούς συνδιασμούς της ακτίνας, η οποία παίρνεις τιμες [1, 2, 3] και για τους κανόνες [30, 60, 90, 110, 182, 250]. Τα παραπάνω βήματα φαίνονται και στην Εικόνα 1.

```
54  rows = 100
55  columns = rows
56  matrix = np.zeros([rows,columns], dtype=int)
57
58  # setting initial conditions
59  _, index = matrix.shape
60  matrix[0][int(index/2)] = 1
61
62  radiusList = [1, 2, 3]
63
64  ruleList = [30, 90, 60, 110, 182, 250]
```

Εικόνα 1

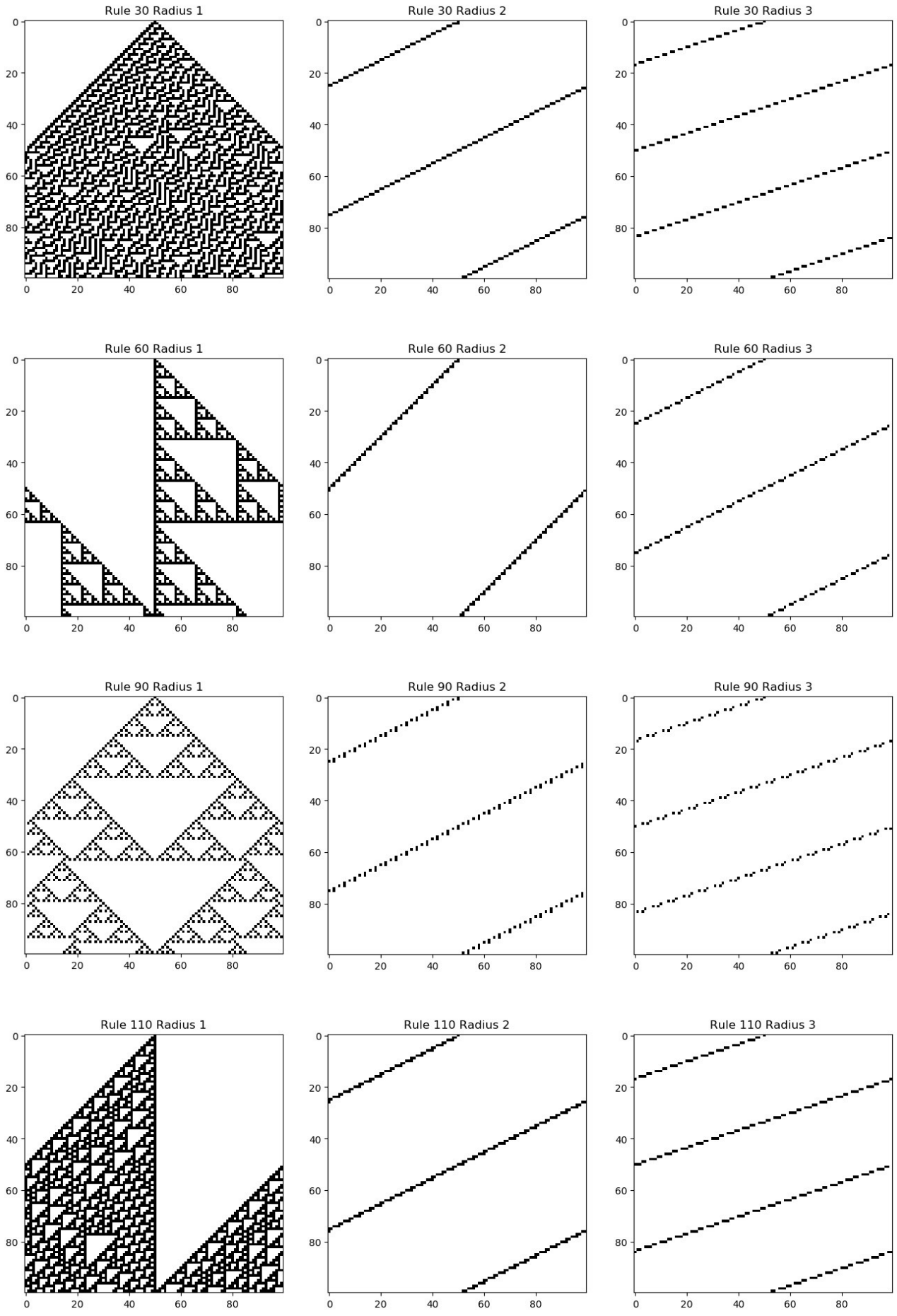
Αφού ορίσουμε τις παραμέτρους του προβλήματος το πρόγραμμα θα δημιουργήσεις ενα lookup table (με την μορφή dictionary), το οποίο θα αντιστοιχεί τις τιμές απο μια γειτονία απο κυψελίδες σε μια μοναδική τιμή. Για παράδειγμα για τον κανόνα 30 και ακτίνα ίση με 1 το λεξικό (dictionary) θα έχει την μορφή που φαίνεται παρακάτω.

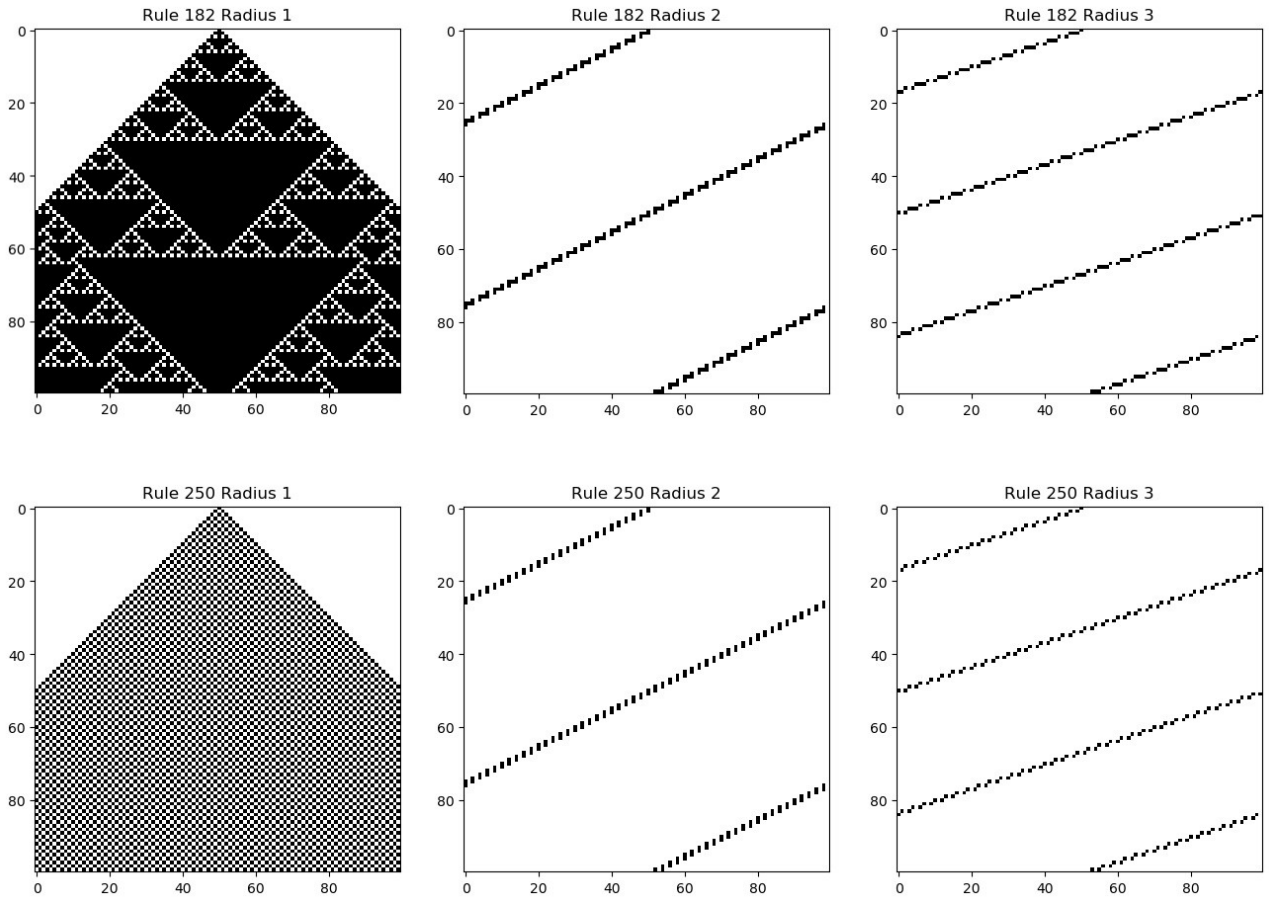
```
Rule : 30  Radius : 1
Lookup table : {'111': '0', '110': '0', '101': '0', '100': '1', '011': '1', '010': '1', '001': '1', '000': '0'}
```

Τα lookup table που προκύπτους μπορείτε να τα δείτε στον αρχείο “output1.txt” η συνάρτηση που δημιουργεί το pattern_dic είναι η rule_dictionary(rule).

Στη συνέχει η συνάρτηση simulation θα αρχίσει να “γεμίζει” τον πίνακα matrix (που αποτελεί το περιβάλλον μας, δηλαδή εκφράζει την μεταβολή του Κ.Α ως προς τον χρόνο). Αυτό το κάνει μέσω μιας επαναληπτικής διαδικασίας κατα την οποία χρησιμοποιεί την συνάρτηση step. Ουσιαστικά πάντα υπολογίζουμε το μεσαίο στοιχείο του Κ.Α, για την επόμενη κατάσταση, με βάση τις τιμές απο την μεσαία κυψελίδα και τους γειτονες της, της προηγούμενης κατάστασης. Μέσω της συνάρτηση numpy.roll() εξασφαλίζουμε οτι έχουμε περιοδικές συνθήκες.

Τα αποτελέσματα απο τα παραπάνω πειράματα φαίνονται παρακάτω


Rule 30 Radius 1
Rule 30 Radius 2
Rule 30 Radius 3
Rule 60 Radius 1
Rule 60 Radius 2
Rule 60 Radius 3
Rule 90 Radius 1
Rule 90 Radius 2
Rule 90 Radius 3
Rule 110 Radius 1
Rule 110 Radius 2
Rule 110 Radius 3
0
20
40
60
80



Αυτό που παρατηρούμε είναι οτι για ακτίνα μεγαλύτερη του ένα το αποτέλεσμα μας είναι πολύ αρραιό. Αυτό συμβαίνει διότι όταν έχουμε ακτίνα 2 ή 3 ο μέγιστος κανόνας που μπορούμε να έχουμε είναι 4294967496 και 3.4e+38, άρα όσο πιο μικρός είναι ο κανόνας μας (στο δεκαδικό σύστημα) τόσο περισσότερες τιμές (δηλαδή συνδιασμοί κελιών) θα έχουν αποτέλεσμα μηδεν.

Το δεύτερο πρόλβημα μας αφορά Κ.Α στα οποία υπάρχουν περισσότερες απο μία καταστάσεις. Σε αυτή την περίπτωση ο κανόνας δεν γίνεται να είναι δεκαδικός γιατι το αυθαίρετο συστημα μας δεν μπορεί να συνδεθεί μεσω μιας απλής συνάρτησης με το δεκαδικο και η δημιουργία ενος πίνακα που θα περιέχει όλες τις τιμές του αυθαίρετου συστήματος μας απαιτεί τεράστια μνήμη (ακμή και στην πίο απλή περίπτωση, με τρείς καταστάσεις και ακτίνα ένα θα υπάρχουν 3^3^3 στοιχεία, φανταστείτε τι θα γίνει για περισότερς κλάσσεις-καταστάσεις!).

```
# number of distinct values-states (binary = 2)
distinct = 3

examples = 10
distinctList = [3, 4, 5, 6]
radList = [1, 2, 3]

for distinct in distinctList:
    for rad in radList:
        radius = rad
```

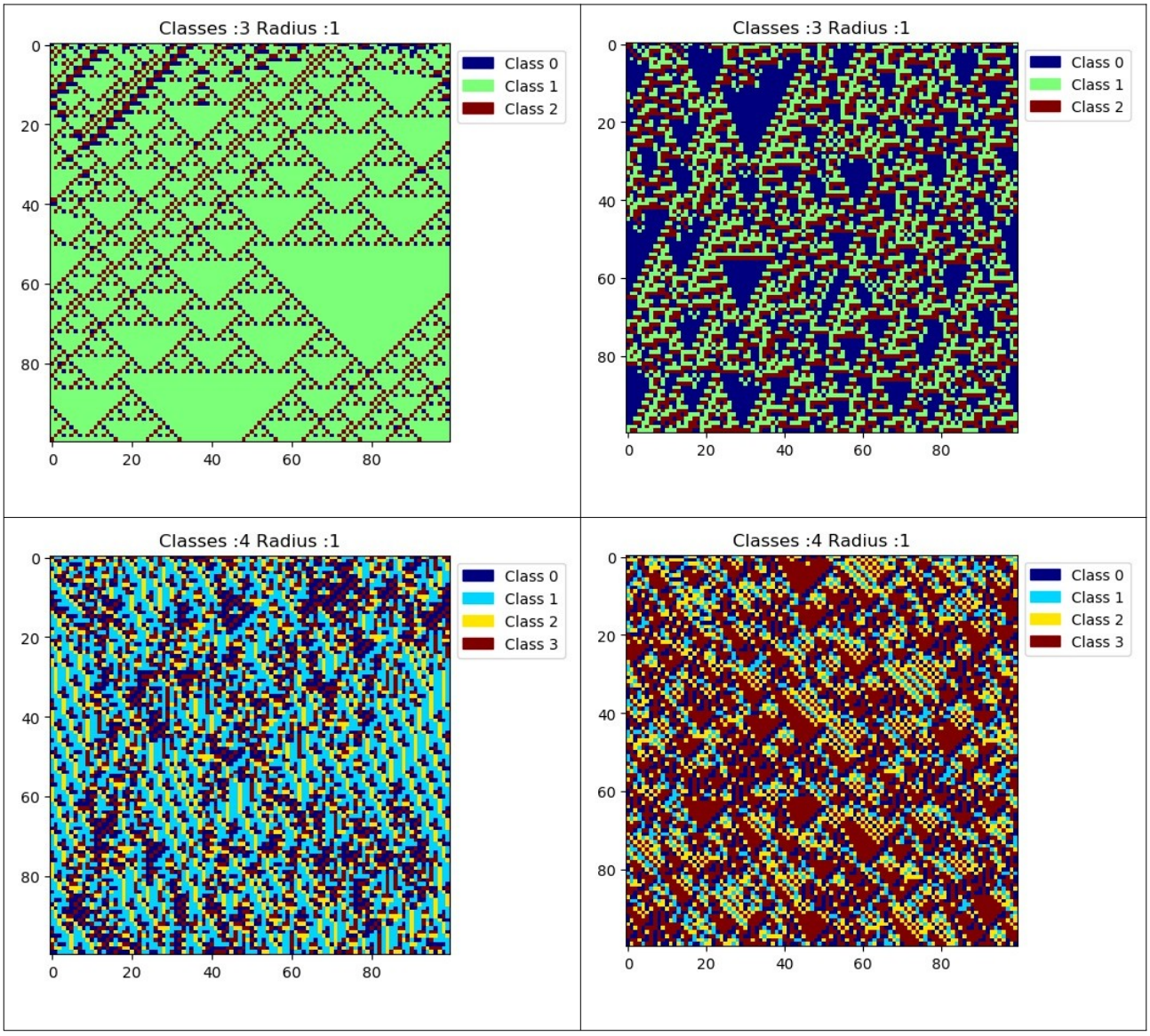
Όπως φαίνεται δοκίμασα διαφορετικούς συνδιασμούς καταστάσεων (απο 3 έως 6) και ακτίνας (απο 1 έως 3).

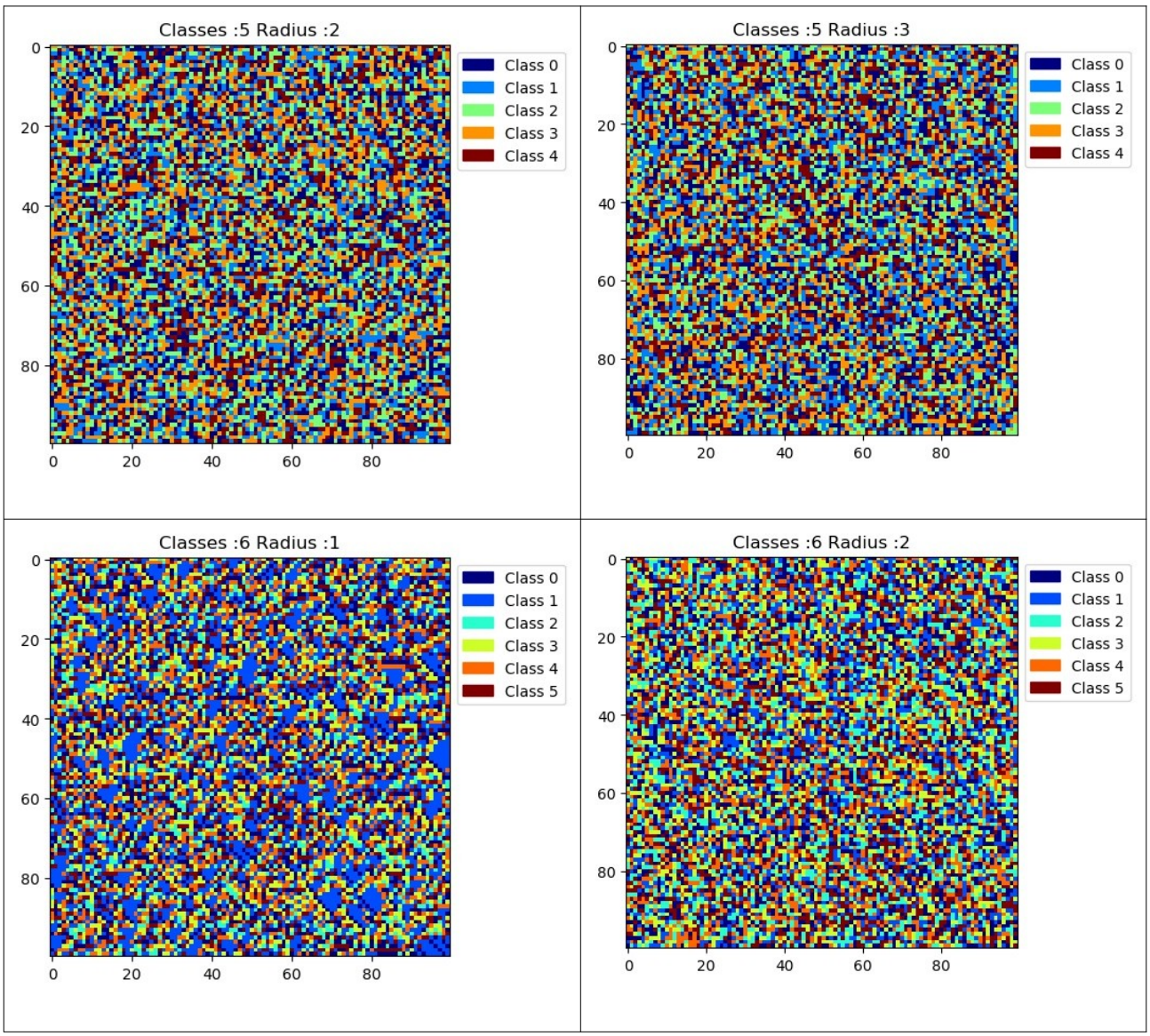
Το lookup table σε αυτό το πρόβλημα προκύπτει με την χρήση του Cartesian Product, το οποίο αποτελεί μια μαθηματική πράξη που επιστρέφει όλους τους δυνατούς συνδιασμούς για έναν αριθμό

συνόλων. Ένα απλό παράδειγμα για 3 διακριτές καταστάσεις και ακτίνα ίσης με 1 φαίναιεται παρακάτω

Lookup table : {'000': '1', '001': '1', '002': '2', '010': '0', '011': '1', '012': '1', '020': '0', '021': '1', '022': '1', '100': '0', '101': '0', '102': '1', '110': '2', '111': '2', '112': '1', '120': '1', '121': '1', '122': '2', '200': '2', '201': '0', '202': '2', '210': '1', '211': '1', '212': '2', '220': '2', '221': '2', '222': '1'}

Μερικά αποτελέσμτα ακολουθούν παρακάτω





Αυτό που παρατηρώ είναι ότι όσο πιο μικρός είναι ο αριθμός των διακριτών καταστάσεων και η ακτίνα του Κ.Α τόσο πιο πιθανό είναι να εντοπίσουμε μοτίβα στην λειτουργία του συστήματος μας. Περισσότερα αποτελέσματα μπορείτε να βρείτε στο repository που έχω δημιουργήσει https://github.com/dimikout3/cellularAutomata.git