شبیه‌سازی رایانه‌ای در فیزیک
تمرین اول: Cellular Automata

سینا معمر ۹۵۱۰۲۳۱۶

۱۸ شهریور ۱۴۰۰

۱ قانون کلاه

۱.۱ اتوماتای سلولی یک بعدی

از آنجایی که سوالات ۱ تا ۳ همگی در دسته‌ی cellular automata های یک بعدی قرار می‌گیرند و تنها در شرایط اولیه، زمان و قانون تفاوت دارند، کلاس CA1D در فایل automata.py پیاده‌سازی و در این ۳ سوال از این کلاس استفاده شده است. در زیر به تشریح ساختار و عملکرد این کلاس می‌پردازیم.
برای ساخت یک object از این کلاس به دو ورودی "تعداد سلول‌ها" و "شرایط اولیه نیاز" است. برای ایجاد شرایط اولیه تصادفی از "rand" استفاده و در غیر این صورت باید لیستی از شماره‌ی خانه‌های روشن فرستاده شود. برای ذخیره وضعیت سلول‌ها از کلاس bitarray استفاده می‌کنیم تا تنها یک بیت از حافظه برای هر سلول اشغال شود.
برای انجام شبیه‌سازی تابع render را باید صدا بزنیم. برای این کار به "شماره قانون" و "زمان" نیاز داریم. شیوه کار این تابع به این صورت است که ابتدا با استفاده از تابع __LUT__ یک دیکشنری از ۸ حالت ممکن برای همسایگی‌ها و وضعیت بعدی سلول متناظر با آن‌ها می‌سازیم. به این صورت که شماره قانون را به مبنای ۲ برده و لیست حاصل شده را پیمایش می‌کنیم و نتیج را به صورت {key, value} به LUT اضافه می‌کنیم. در ادامه‌ی تابع render، روی لیست سلول‌ها پیمایش می‌کنیم و وضعیت همسایه‌های آن را به صورت یک عدد ۳ رقمی در مبنای ۲ به یک string تبدیل کرده و به این طریق key معادل با وضعیت بعدی سلول در LUT مان را به دست می‌آوریم و لیست سلول‌های جدید را می‌سازیم و به لیست grids اضافه می‌کنیم.
در پایان برای نمایش عکس حاصل، تابع show را فراخوانی می‌کنیم. این تابع برای نمایش عکس از لیست سلول‌های ما، ابتدا وضعیت هر سلول را به Not آن تبدیل می‌کند تا سلول‌های روشن به رنگ سیاه نمایش داده بشوند. سپس تابع imshow از کتاب‌خانه matplotlib صدا زده می‌شود.

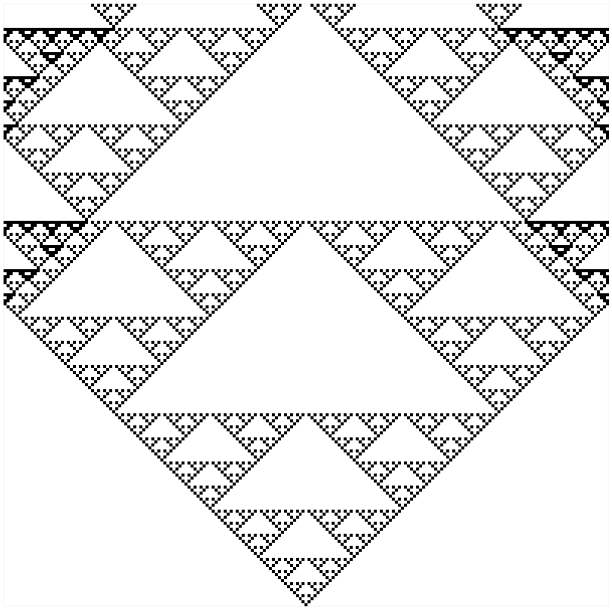
۲.۱ مسئله گسترش رفتار

مسئله‌ی گسترش رفتار یک نمونه از CA یک بعدی است. طبق گفته مسئله، هر سلول تنها در صورتی در مرحله بعدی روشن خواهد بود که تنها یکی از همسایه‌های چپ و راستش روشن باشند، یعنی یکی از حالت‌های روبرو: {100, 110, 001, 011}. و در باقی حالات، خاموش خواهد بود.
در نتیجه همان طور که در جدول ؟؟ مشاهده می‌شود، قانون ما $(01011010)_2$ یا همان ۹۰ خواهد بود. همچنین ۲۰۱ سلول داریم که در زمان ۰ تنها سلول ۱۰۱ روشن است. برای نشان دادن پخش شدن این مد در جامعه، ۲۰۰

111	110	101	100	011	010	001	000
0	1	0	1	1	0	1	0

جدول ۱: قانون مسئله گسترش رفتار

واحد زمانی آن را بررسی می‌کنیم. کد این بخش از سوال در فایل q1.py قابل مشاهده است.



شکل ۱: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۹۰ و ۲۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن

همان طور که در شکل ؟؟ مشاهده می‌شود، چگونگی گسترش مد در این جامعه، مشابه مثلث سرپینسکی است. نکته قابل توجه این شکل آن است که چگونگی پخش شدن مد در هر اسکیلی، مشابه با چگونگی پخش شدن مد در اسکیل‌های کوچک‌تر همان جامعه است که کاملا متناسب با توقع ما از مسئله است.

۲ قوانین ۲۵۶گانه

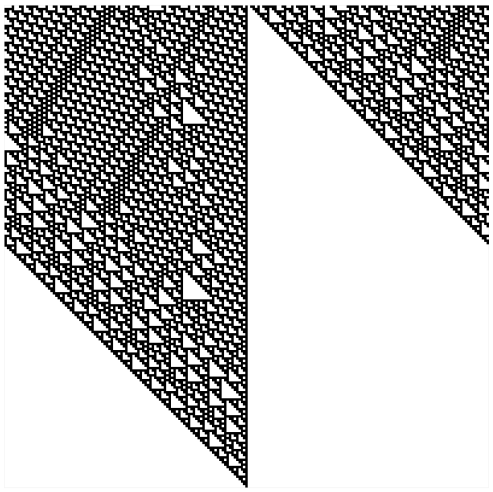
برای این سوال نیز همانند بخش قبل عمل می‌کنیم و از کلاس CA1D استفاده می‌کنیم. شرایط اولیه هر دو یکسان و به طول ۲۰۱ با تنها یک سلول روشن در وسط است. نمودار هر دوی آن‌ها را برای ۲۰۰ واحد زمانی به دست می‌آوریم. نمودار قانون ۱۱۰ در شکل ؟؟ و نمودار قانون ۷۵ در شکل ؟؟ قابل مشاهده است.

۱.۲ بهینه‌سازی حافظه

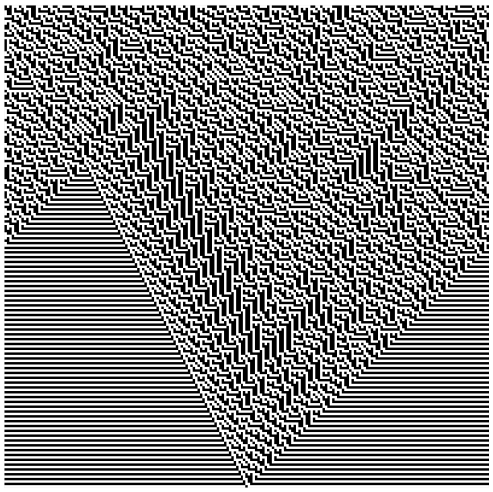
برای بهینه‌سازی حافظه، می‌توانیم از متغیرهای boolean و یا bit برای ذخیره‌سازی وضعیت سلول‌ها استفاده کنیم تا تنها یک بیت از حافظه را به ازای هر سلول اشغال کنیم. همچنین می‌توان به جای ذخیره کردن وضعیت تمام سلول‌ها در هر مرحله، تنها شماره سلول‌هایی که وضعیت‌شان تغییر کرده است را ذخیره کنیم. که این کار تحت شرایطی که تعداد سلول‌هایی که تغییر وضعیت داده‌اند کمتر از حد $\frac{N}{log_2 N}$ باشد، به کاهش حافظه کمک می کند ولی از طرف دیگر، پیچیدگی محاسبات و زمان اجرا را افزایش می‌دهد. علاوه بر این از آنجایی که در نهایت برای نمایش گرافیکی نیاز به لیستی از وضعیت تک تک سلول‌ها در هر مرحله داریم، پس عملا این کار بی‌تاثیر خواهد بود.

۳ Wolfram Classification

در این بخش نیز همانند بخش‌های پیشین، از کلاس CA1D استفاده می‌کنیم. برای به دست آوردن شکل‌های ذکر شده در سوال، از شرایط اولیه‌ای به طول ۲۰۱ که تنها سلول مرکزی روشن باشد، استفاده می‌کنیم و ۳۰۰ واحد زمانی آن را روشن می‌گذاریم.



شکل ۲: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۱۰ و ۲۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



شکل ۳: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۷۵ و ۲۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن

همچنین برای مشخص کردن دسته‌بندی آن‌ها، باید از شرایط اولیه تصادفی شروع کنیم و بر اساس ظاهر و ویژگی‌های شکل به دست آمده، این کار را انجام دهیم.

۱.۳ قانون ۴۶

این قانون در مبنای ۲ به صورت $(00101110)_2$ است که با قانون $110 = (01101110)_2$ تنها در رقم ۷ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح‌های به نسبت منظمی ولی به صورت تقریبا تصادفی شکل گرفته است. پس در کلاس ۴ قرار می‌گیرد.

۲.۳ قانون ۷۸

این قانون در مبنای ۲ به صورت $(01001110)_2$ است که با قانون $110 = (01101110)_2$ تنها در رقم ۶ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که دیده می‌شود، به یک طرح ثابت و پایدار می‌رسیم. پس این قانون در کلاس ۲ قرار می‌گیرد.

۳.۳ قانون ۱۰۲

این قانون در مبنای ۲ به صورت $(01100110)_2$ است که با قانون $110 = (01101110)_2$ تنها در رقم ۴ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح به دست آمده پیچیده است ولی در عین حال به طور نسبی دارای نویز کمی است و به طور موضعی تکرار می‌شوند. پس در کلاس ۴ قرار می‌گیرد.

۴.۳ قانون ۱۰۶

این قانون در مبنای ۲ به صورت $(01101010)_2$ است که با قانون $(01101110)_2 = 110$ تنها در رقم ۳ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح به دست آمده پیچیده و تصادفی و همراه با نویز بسیار است. پس در کلاس ۳ قرار می‌گیرد.

۵.۳ قانون ۱۰۸

این قانون در مبنای ۲ به صورت $(01101100)_2$ است که با قانون $(01101110)_2 = 110$ تنها در رقم ۲ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح به دست آمده به دو الگو ثابت که به طور تناوبی تکرار می‌شوند، می‌رسد. پس در کلاس ۲ قرار می‌گیرد.

۶.۳ قانون ۱۱۱

این قانون در مبنای ۲ به صورت $(01101111)_2$ است که با قانون $(01101110)_2 = 110$ تنها در رقم ۱ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح‌های به نسبت منظمی ولی به صورت تقریبا تصادفی شکل گرفته است. پس در کلاس ۴ قرار می‌گیرد.

۷.۳ قانون ۱۲۶

این قانون در مبنای ۲ به صورت $(01111110)_2$ است که با قانون $(01101110)_2 = 110$ تنها در رقم ۵ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح‌های به نسبت منظمی ولی به طور تصادفی و همراه با نویز کم شکل گرفته است. پس در کلاس ۴ قرار می‌گیرد.

۸.۳ قانون ۲۳۸

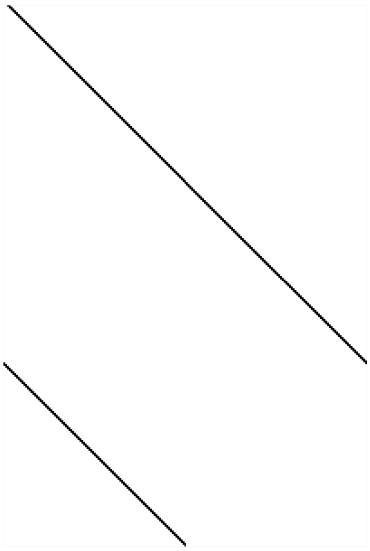
این قانون در مبنای ۲ به صورت $(11101110)_2$ است که با قانون $(01101110)_2 = 110$ تنها در رقم ۸ ام متفاوت است. نمودار حاصل از این قانون را در شکل ؟؟ مشاهده می‌کنیم.
همچنین برای دسته‌بندی کردن آن باید نمودار این قانون با شرایط اولیه تصادفی را هم داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که دیده می‌شود، همه‌ی سلول‌ها نهایتا به یک وضعیت یکسان می‌رسند و دارای طرحی همگن می‌شوند. پس در کلاس ۱ قرار می‌گیرد.

۹.۳ قانون ۱۱۰

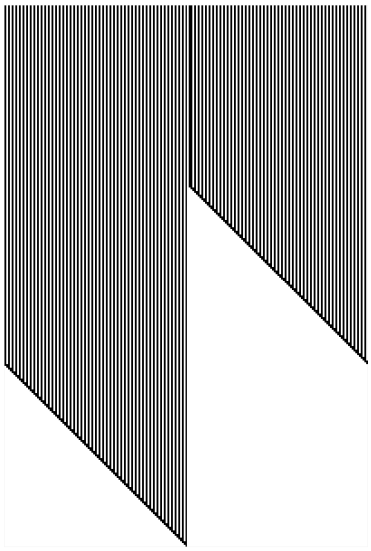
برای دسته‌بندی کردن این قانون باید نمودار آن با شرایط اولیه تصادفی را داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح‌های به نسبت منظمی ولی به طور تصادفی و همراه با نویز کم شکل گرفته است. پس در کلاس ۴ قرار می‌گیرد.

۱۰.۳ قانون ۷۵

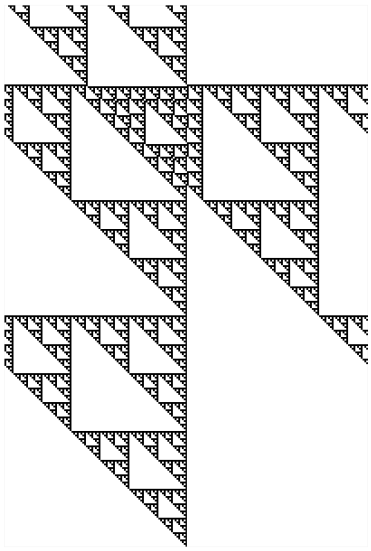
برای دسته‌بندی کردن این قانون باید نمودار آن با شرایط اولیه تصادفی را داشته باشیم، که این نمودار را در شکل ؟؟ می‌توان مشاهده نمود. همان طور که در شکل دیده می‌شود، طرح به دست آمده پیچیده و تصادفی و همراه با نویز بسیار است. پس در کلاس ۳ قرار می‌گیرد.



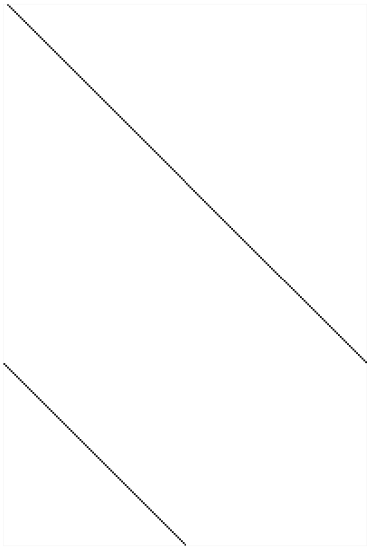
شکل ۴: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۴۶ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



شکل ۵: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۷۸ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن

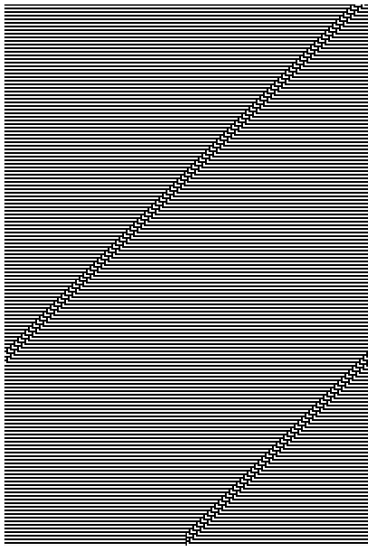


شکل ۶: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۲ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن

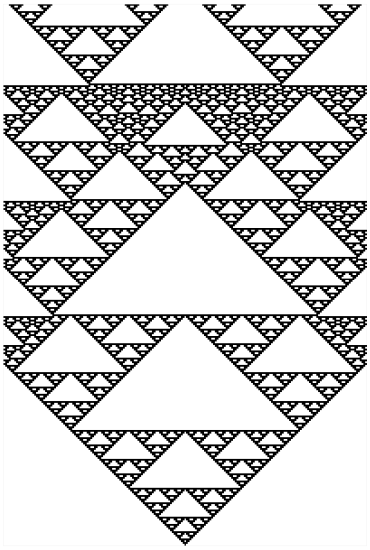


شکل ۷: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۶ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن

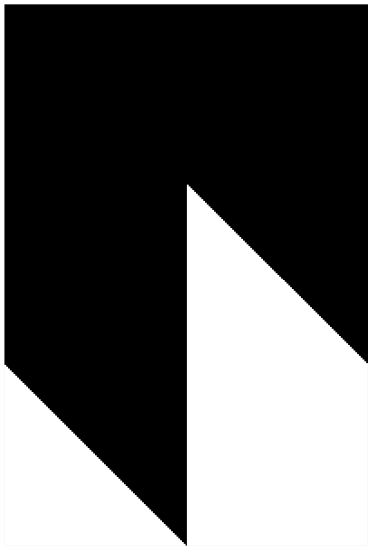
شکل ۸: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۸ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



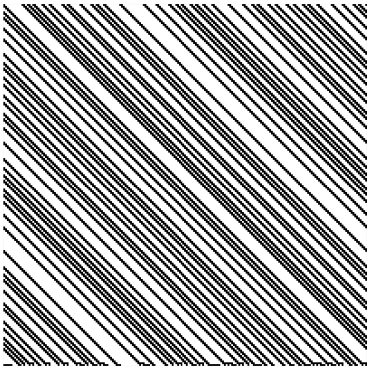
شکل ۹: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۱۱ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



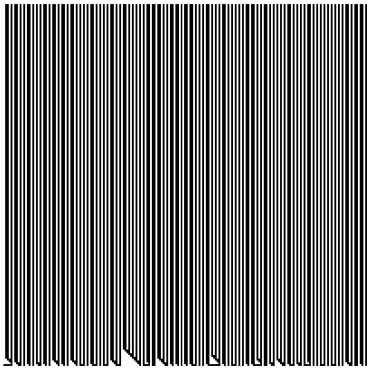
شکل ۱۰: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۲۶ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



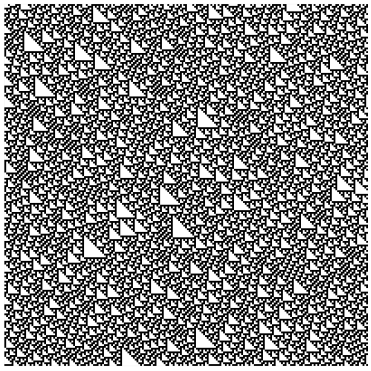
شکل ۱۱: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۲۳۸ و ۳۰۰ دوره زمانی، شرایط اولیه: تنها خانه‌ی ۱۰۱ روشن



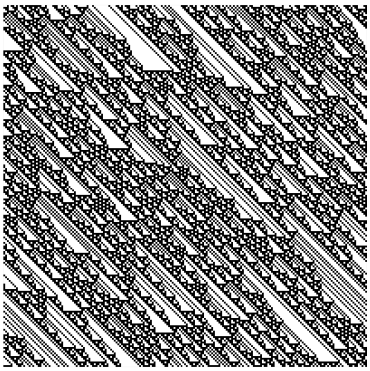
شکل ۱۲: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۴۶ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



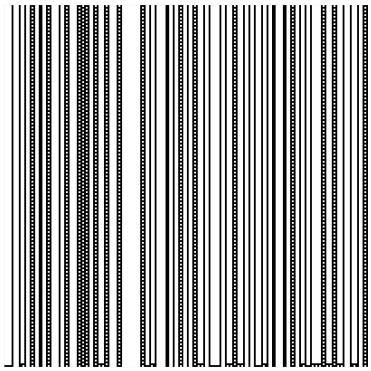
شکل ۱۳: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۷۸ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



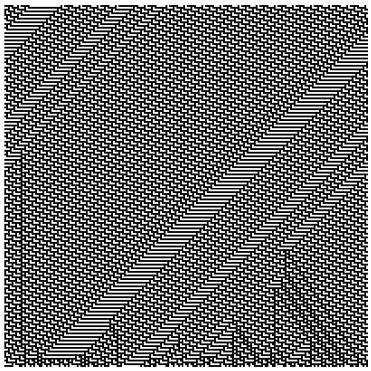
شکل ۱۴: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۲ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



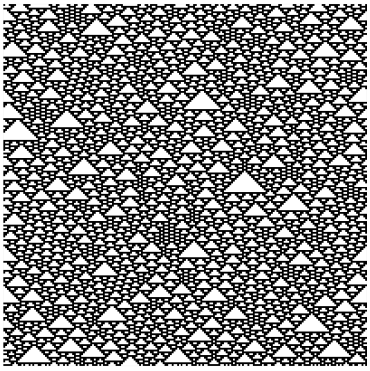
شکل ۱۵: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۶ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



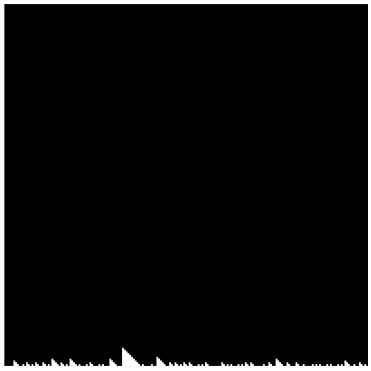
شکل ۱۶: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۰۸ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



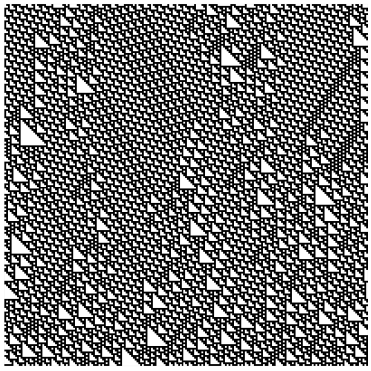
شکل ۱۷: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۱۱ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



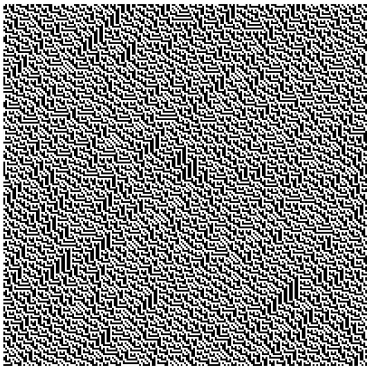
شکل ۱۸: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۲۶ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



شکل ۱۹: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۲۳۸ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



شکل ۲۰: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۱۱۰ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی



شکل ۲۱: اتوماتای سلولی یک بعدی به طول ۲۰۱ با قانون ۷۵ و ۲۰۰ دوره زمانی، شرایط اولیه‌ی تصادفی

Game of life ۴

در این قسمت نیز مشابه روندی که برای cellular automata یک بعدی انجام دادیم، پیش می‌رویم. به این منظور کلاس GameOfLife را در فایل automata.py پیاده‌سازی می‌کنیم.
برای ساختن یک object از این کلاس، به ۳ متغیر "طول"، "ارتفاع" و "شرایط اولیه" نیاز است. شرایط اولیه می‌تواند مقدار "rand" داشته باشد که به معنای شرایط اولیه تصادفی است، یا می‌تواند یک رشته به شکل یک ماتریس باشد که خانه‌های روشن در آن با کارکتر "*" مشخص شده‌اند و هم‌چنین می تواند لیستی از اندیس‌های خانه‌های روشن باشد. برای ذخیره‌سازی وضعیت سلول‌ها، از آرایه‌ای دو بعدی از متغیرهای boolean استفاده می‌کنیم که خانه‌های روشن معادل با مقدار False هستند.
برای انجام شبیه‌سازی باید تابع render را با مقدار "زمانی" که می خواهیم آن را روشن بگذاریم، فراخوانی کنیم. روش کار این تابع به این صورت است که روی تک تک خانه‌های این لیست دو بعدی پیمایش می‌کند و اندیس‌های آن خانه‌ها را به تابع __live_neighbors__ می‌فرستد تا تعداد همسایه‌های زنده‌ی آن سلول را به دست آوریم. این کار را هم به این صورت انجام می‌دهیم که روی لیست ۳ در ۳ همسایگی آن سلول پیمایش کرده و تعداد خانه‌های False را می‌شماریم. بعد از آن که تعداد همسایه‌های زنده را به دست آوردیم، نوبت تعیین کردن وضعیت سلول در مرحله‌ی بعد می‌رسد. وضعیت جدید سلول بر اساس وضعیت پیشین سلول و تعداد همسایه‌های زنده‌ی آن تعیین می‌شود پس این دو مقدار را به تابع __new_state__ می‌فرستیم و بر اساس قوانین تعیین شده در سوال، وضعیت جدید سلول را به دست می‌آوریم و به لیست جدیدمان اضافه می‌کنیم. در نهایت لیست کامل شده را، به لیست grids اضافه می‌کنیم.
در نهایت برای نمایش دادن انیمیشن حاصل از تغییرات سلول‌ها، تابع animate را باید فراخوانی کنیم. این تابع دو مقدار "interval" و "name" را می‌گیرد که اولی فاصله‌ی زمانی بین نمایش هر وضعیت را تعیین می‌کند و دومی اسم فایل ویدئوی ذخیره شده را. روش کار هم به این صورت است که ابتدا لیست دو بعدی هر مرحله را با استفاده از تابع imshow از کتاب‌خانه‌ی matplotlib به عکس تبدیل می‌کنیم و سپس عکس‌های به دست آمده را در یک لیست جدید ذخیره می‌کنیم. در نهایت این لیست را با تابع animation.ArtistAnimation از کتاب‌خانه‌ی matplotlib، به انیمیشن تبدیل و سپس آن را ذخیره می‌کنیم.

با استفاده از تابع تشریح شده در بالا، شرایط اولیه را مطابق شکل‌های داده شده اعمال می‌کنیم و می‌گذاریم که به مدت ۱۰ واحد زمانی روشن بمانند. ویدئوهای به دست آمده را در فولدر game_of_life_animations می‌توان مشاهده نمود.