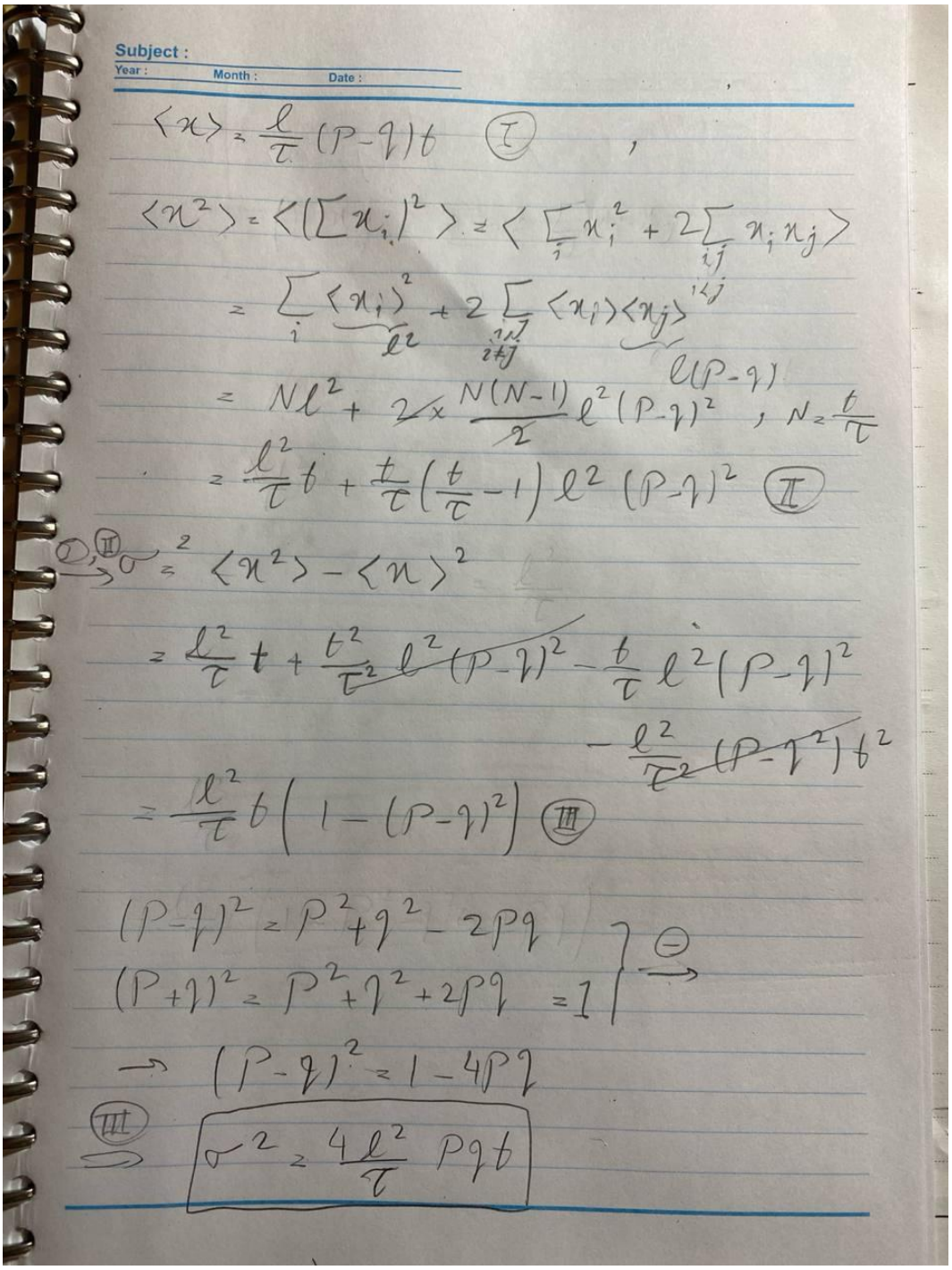
Random Walk

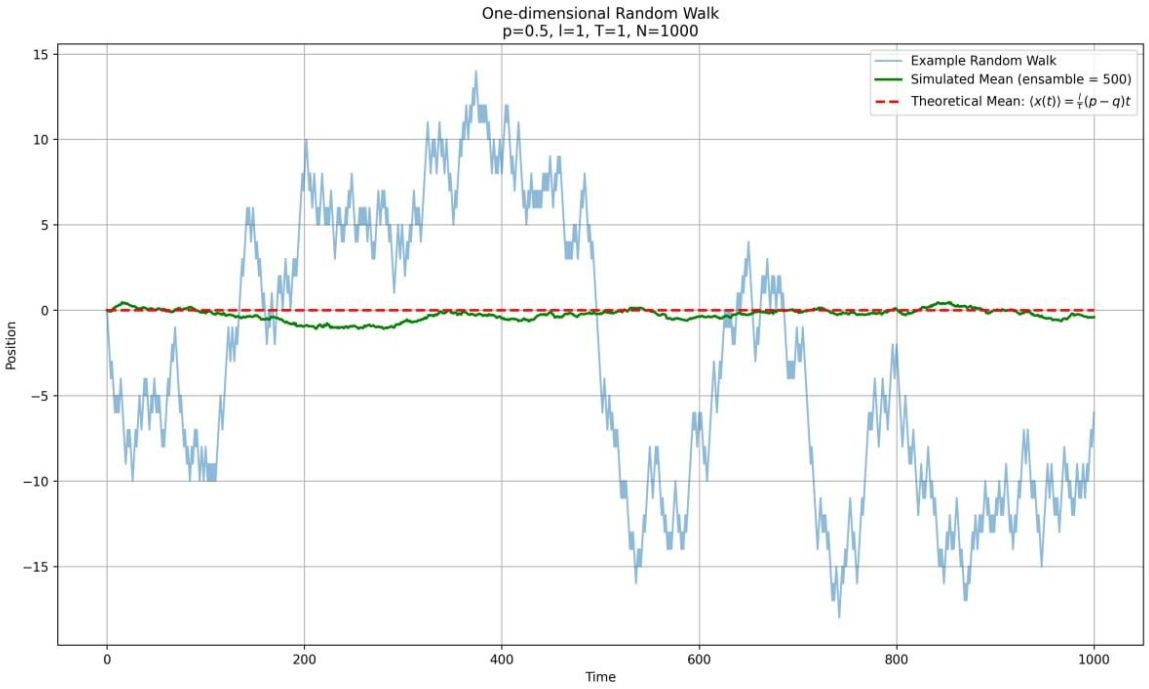
1. اثبات رابطه 7



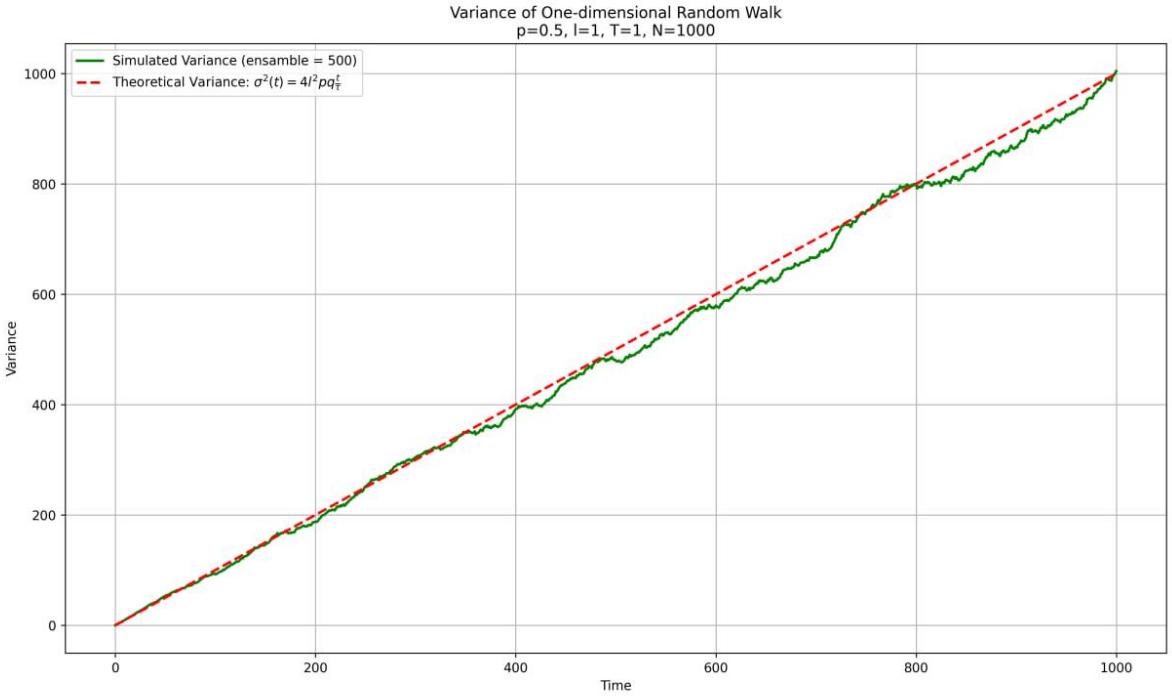
2. random_walk_1D

برای این برنامه تابع single_random_walk را برای یک ولگشت تعریف می کنیم و برای بررسی روابط 6 و 7 از روش آنسامبل گیری استفاده می کنیم.
در تابع single_random_walk خیلی ساده به صورت رندوم یکی از قدم های چپ یا راست را انتخاب می کنیم.
در تابع ensemble_avarage میانگین مکان و واریانس را محاسبه می کنیم.
در نهایت مقادیر بدست آمده از شبیه سازی را با مقدار تئوری در یک نمودار رسم می کنیم.

:P = 0.5

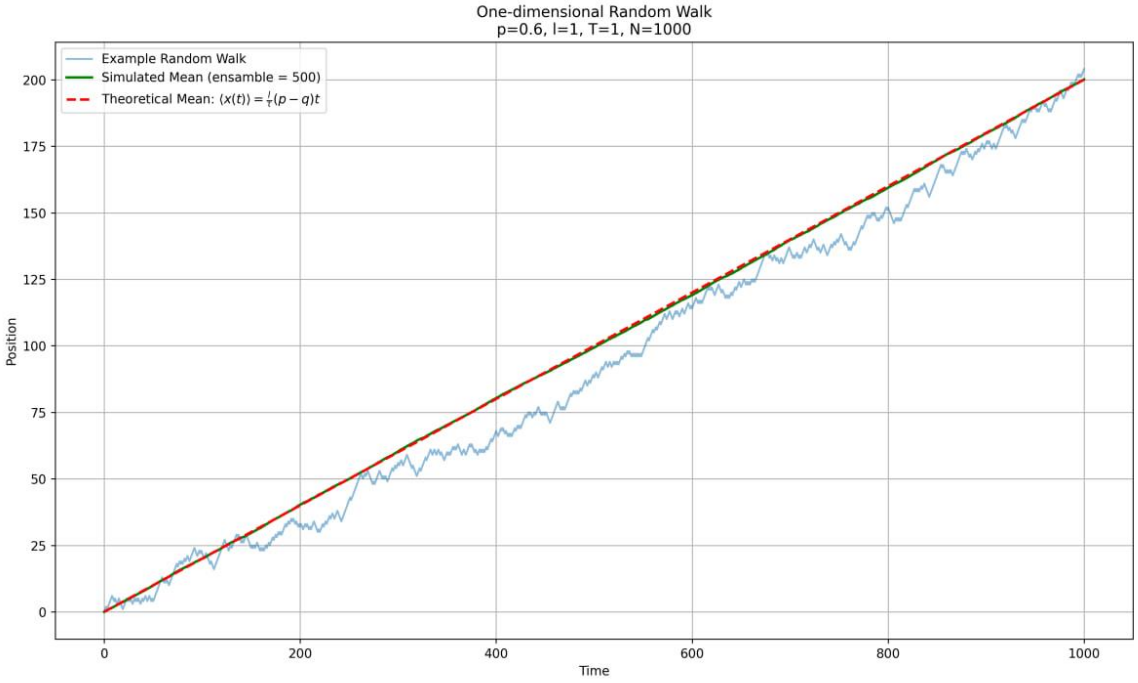


می بینیم که با تقریب خوبی مقدار شبیه سازی و تئوری برای $\langle x \rangle$ منطبق و مساوی صفر هستند.
در این نمودار مکان یک ولگشت نیز برای مثال رسم شده است.

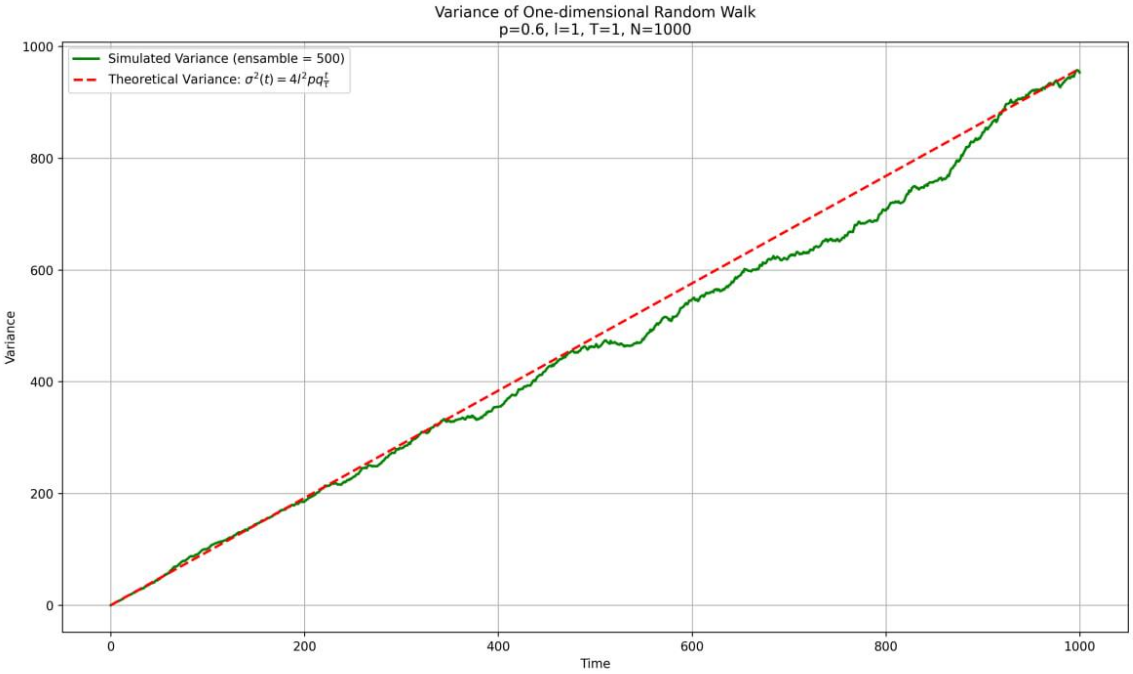


واریانس نیز با زمان زیاد می شود که خاصیت رندومنس بودن شبیه سازی است.

:P = 0.6

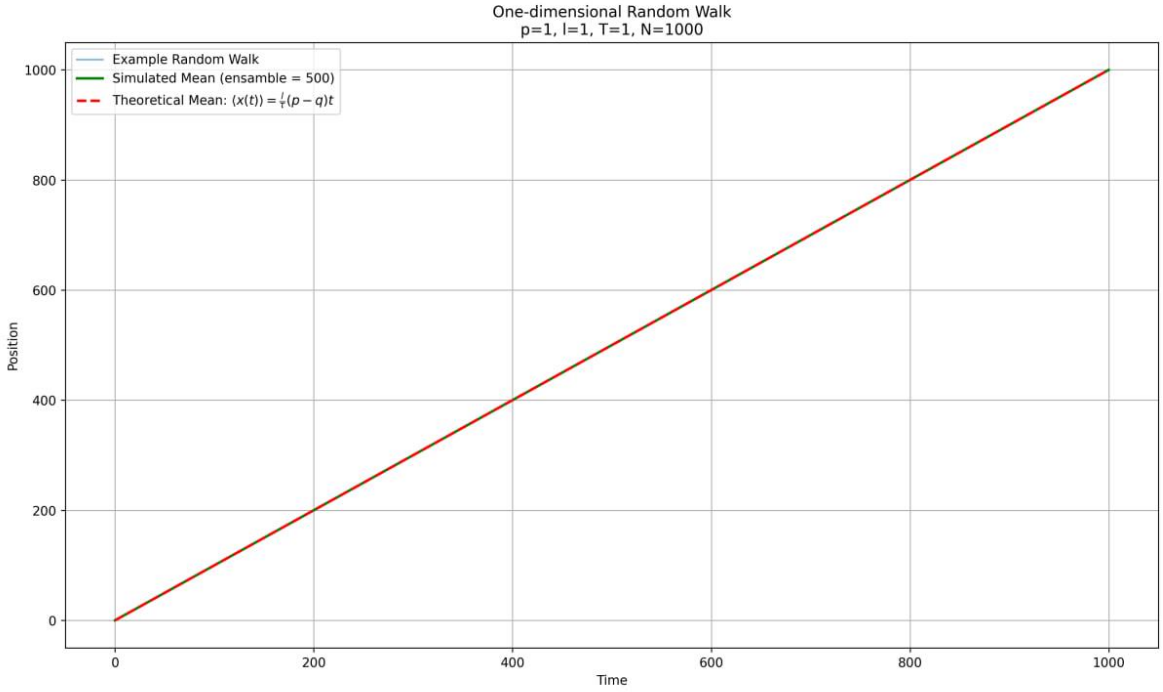


برای P = 0.6 میانگین x به سرعت به سمت راست تغییر جهت می دهد و با زمان به صورت خطی بیشتر می شود.

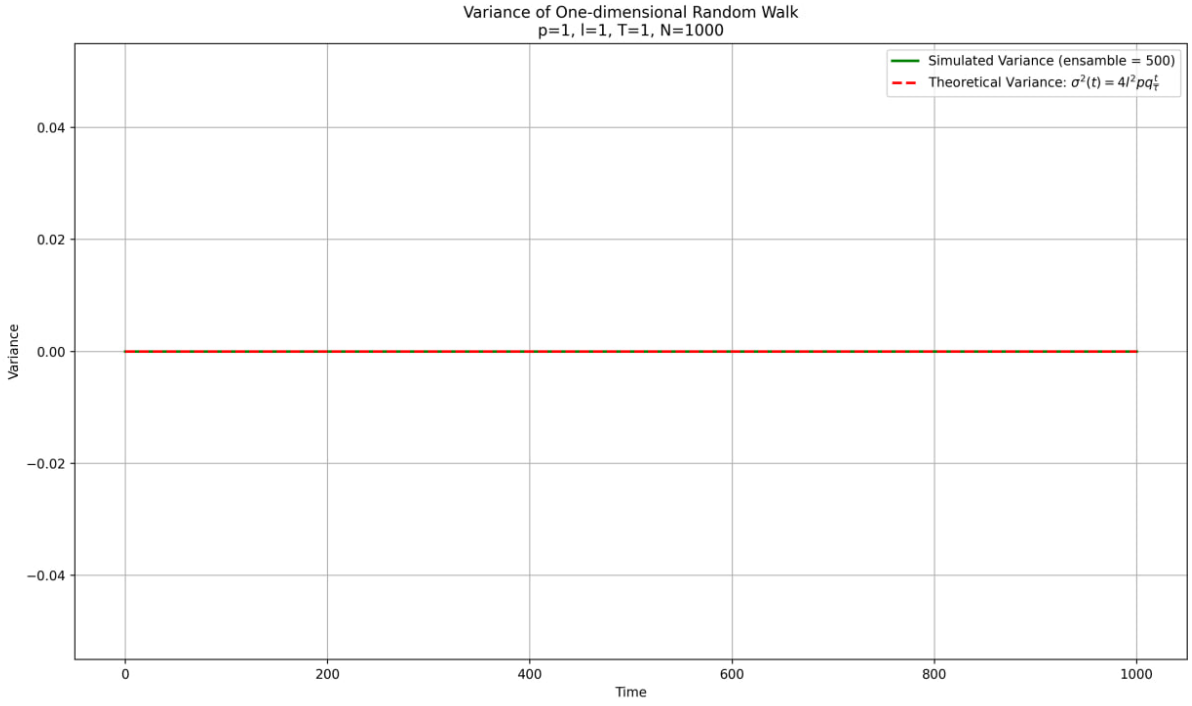


ولی واریانس تغییری نمی کند.

:P = 1



برای P = 1 به طور دقیق شبیه سازی و تئوری و ولگشت مثال با هم منطبق می شوند، چون ولگشت فقط یک انتخاب دارد و نتایج بدیهی اند.

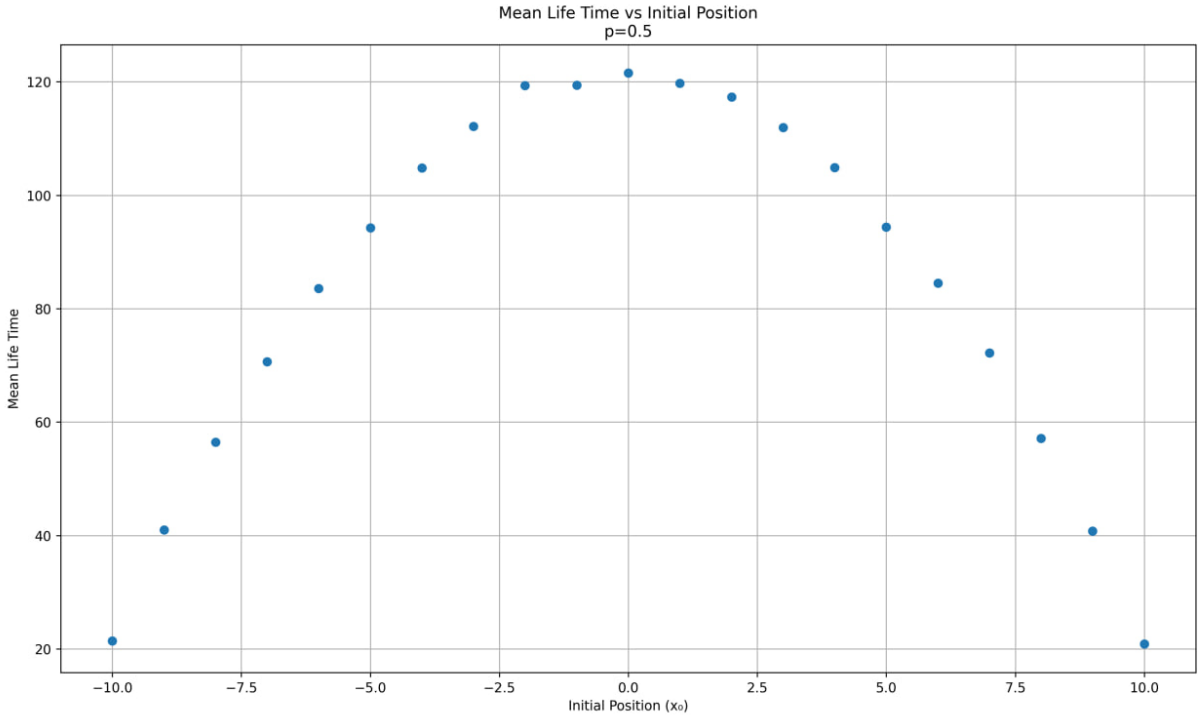


به دلیل بدیهی بودن نتایج و نبود رندومنس، واریانس برای این احتمال صفر است.

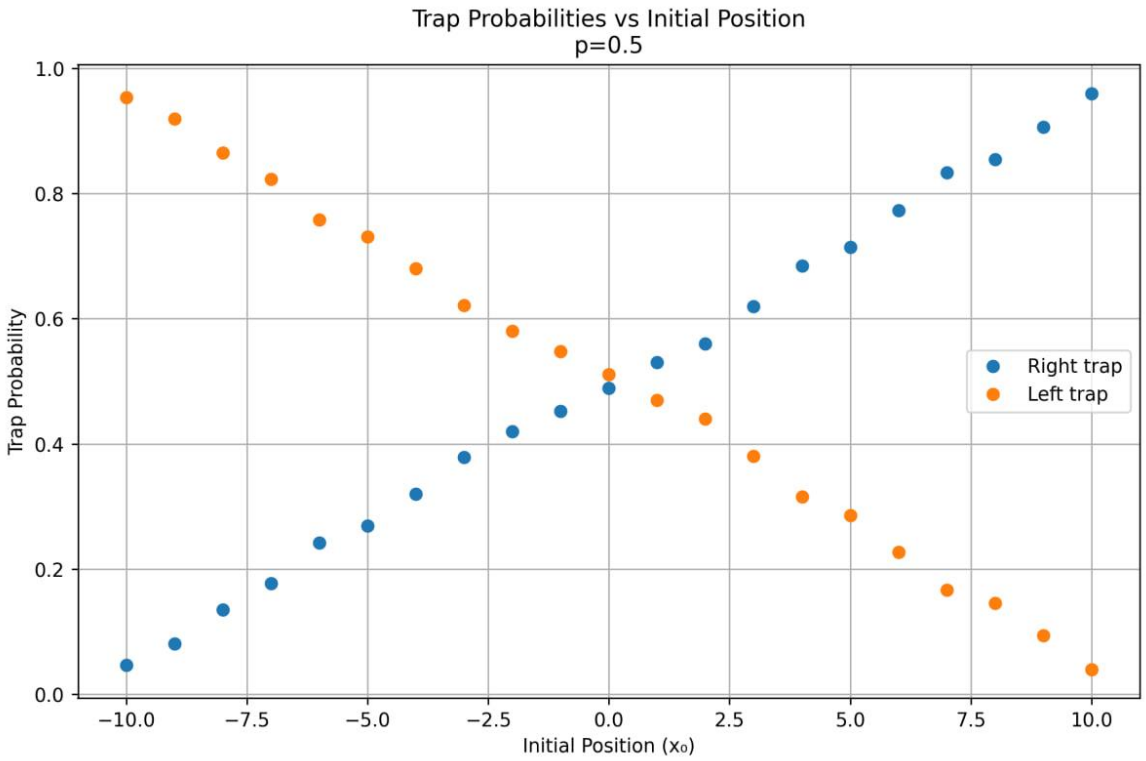
3. random_walk_trap

برای برنامه سوال قبل شرطی قرار می دهیم که اگر مکان ولگشت از 10 و -10 گذشت، شبیه سازی متوقف شود. برای طول عمر تعداد قدم ها را در τ ضرب می کنیم. و برای احتمال هر کدام از تله ها برای ولگشت در تله افتاده trap_id را ذخیره می کنیم. که برای تله راست 1 و برای تله چپ -1 است.
در نهایت روی بازه $-10 \leq x_0 \leq 10$ از طول عمر و احتمال تله ها آنسامبل گیری می کنیم.

:P = 0.5

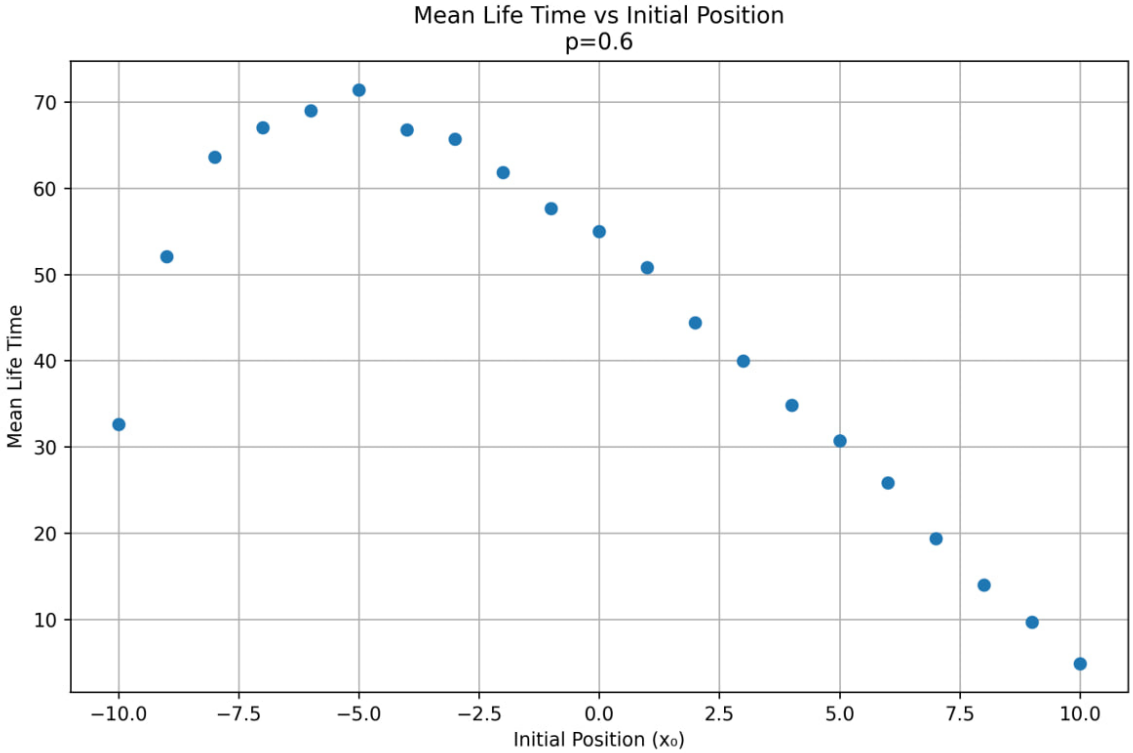


به نظر می سد میانگین طول عمر تابعیت سهمی دارد و برای $x_0 = 0$ بیشینه است.

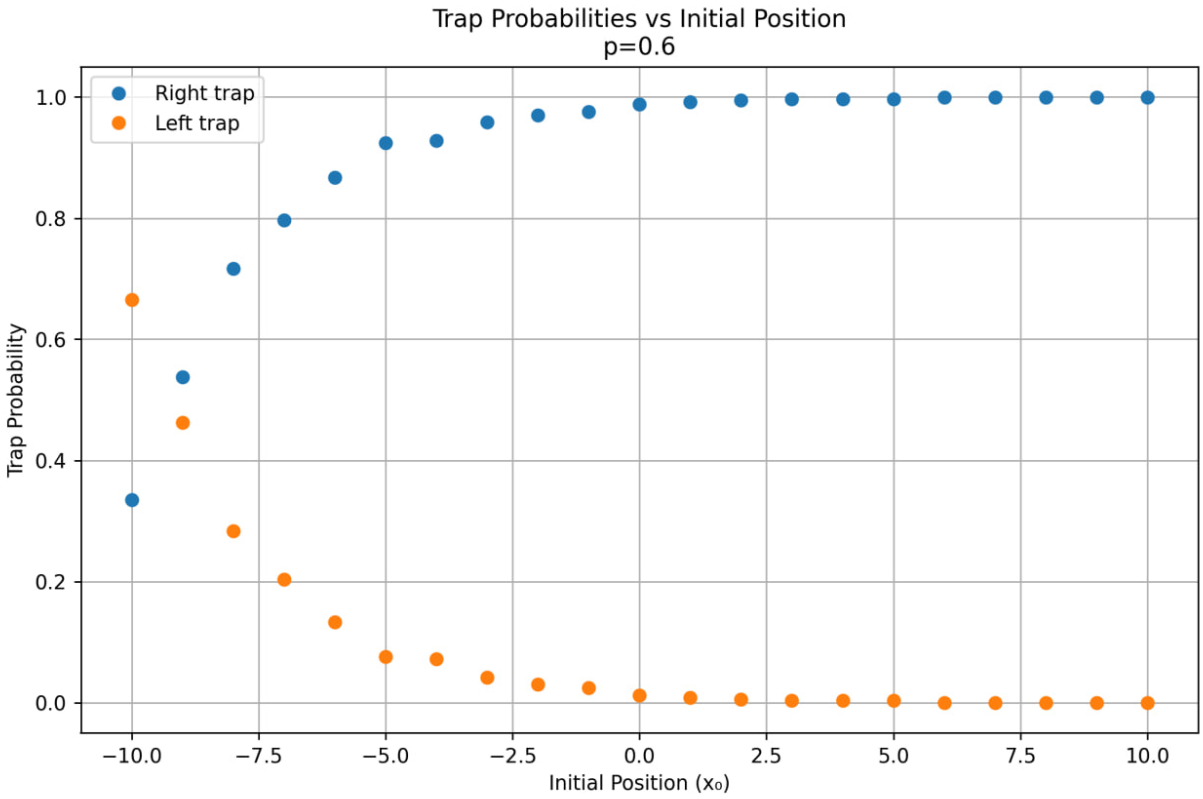


احتمال تله های راست و چپ به صورت خطی به ترتیب زیاد و کم می شوند.

:P = 0.6

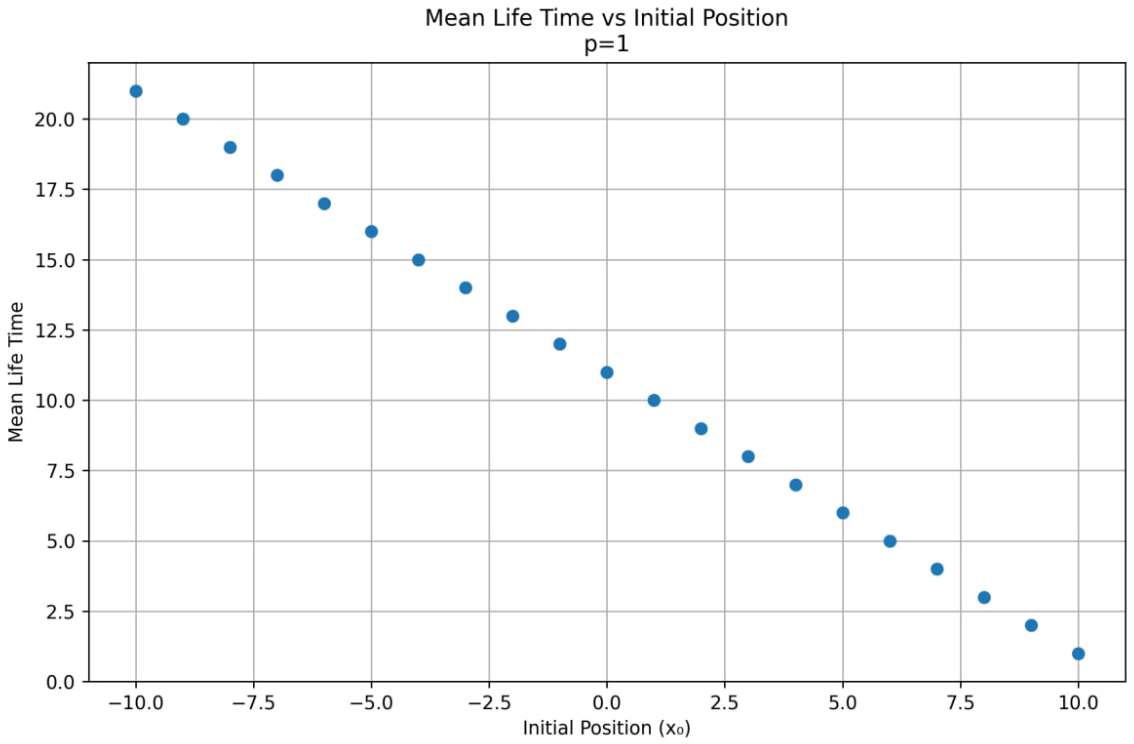


قله نمودار جابجا شده و با نزدیک شدن به تله راست، میانگین طول عمر تابعیت خطی پیدا می کند.

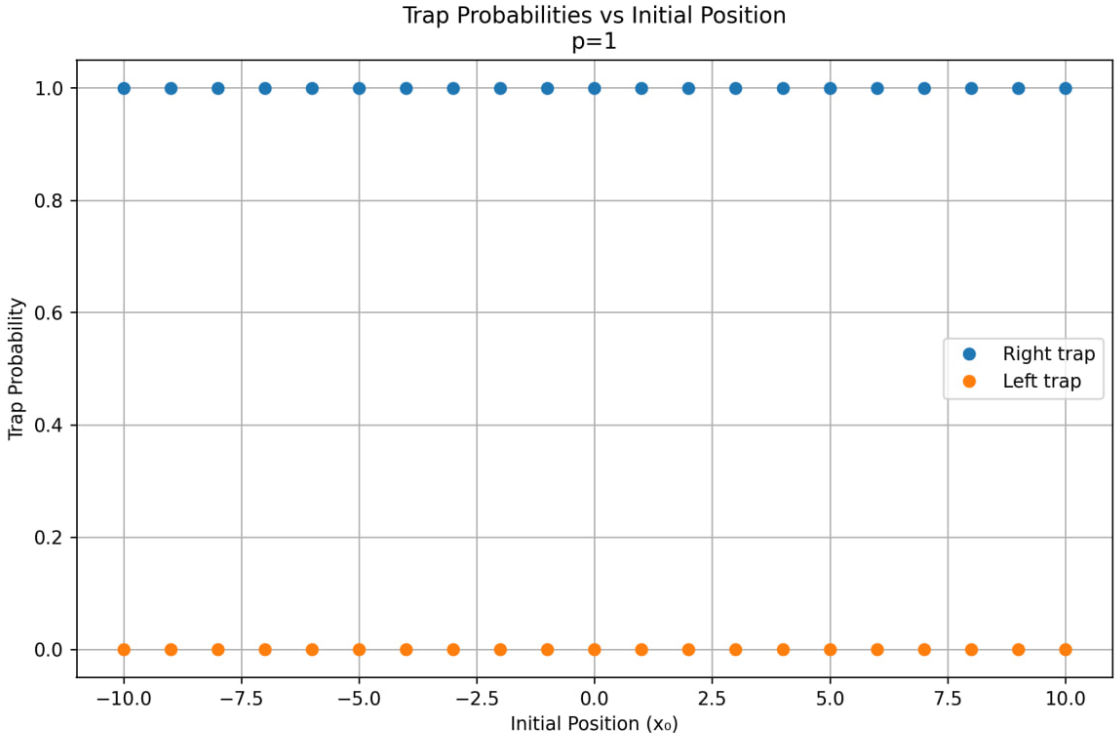


احتمال تله ها در ابتدا صفر و 1 نیستند و به سرعت تغییر کرده و ثابت می شوند.

:P = 1



طول عمر متوسط کامل تابعیت خطی پیدا می کند.



احتمال ها هم بدیهی است.

4. trap_enumeration

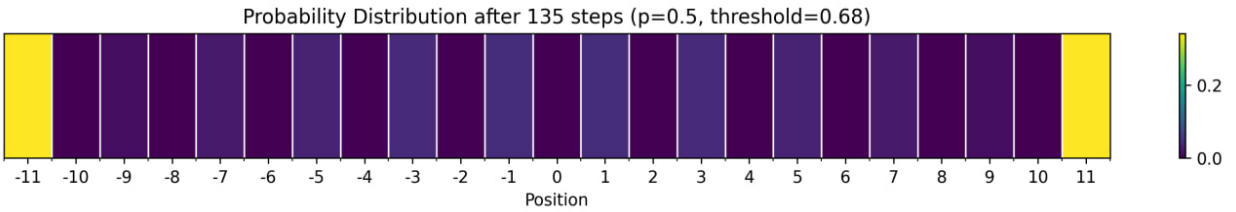
در الگوریتم روش سرشماری روی 21 خانه ایتریت می کنیم و احتمال هر خانه را از روی احتمال های خانه قبلی و بعدی بدست می آوریم. همچنین شرطی برای مرزها می گذاریم که احتمال از تله ها به خانه های کناری نریزد و داخل تله فیکس بماند.

شرط پایان محاسبه احتمال ها رسیدن جمع احتمالات دو تله به death_prob_threshold است.

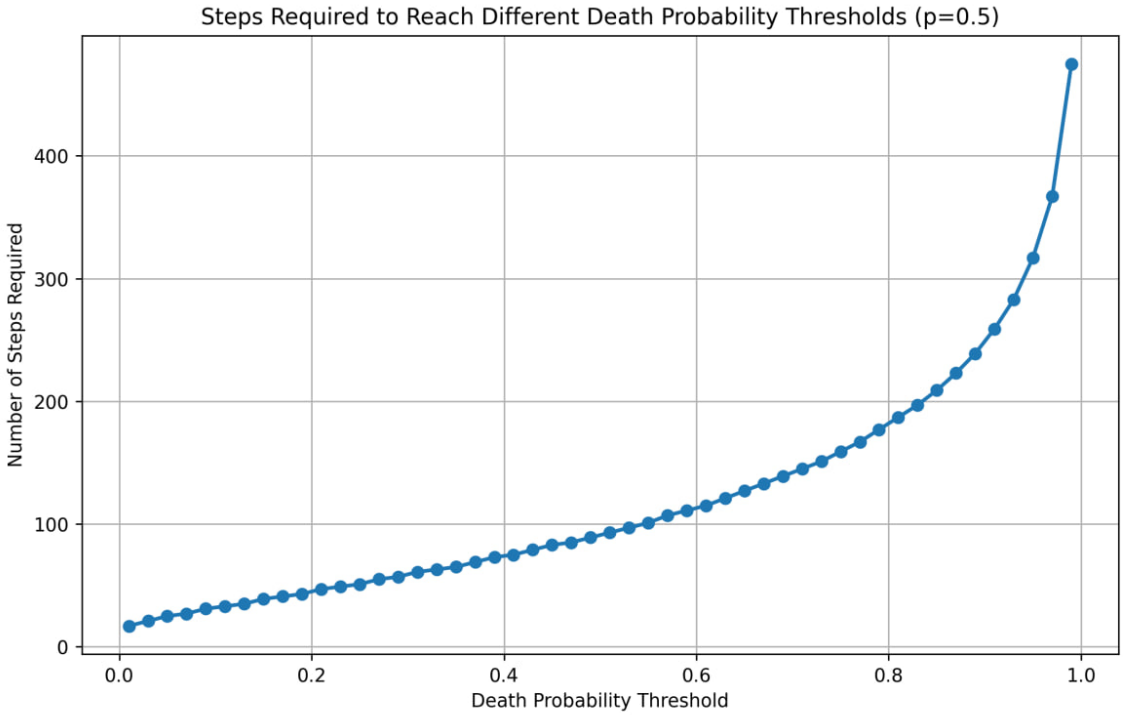
احتمال ها را به صورت heat map نمایش می دهیم.

این هیت مپ ها برای تعداد قدم مشخص هستند. الگوی یکی در میان برای احتمالات خانه ها مشاهده می شود.

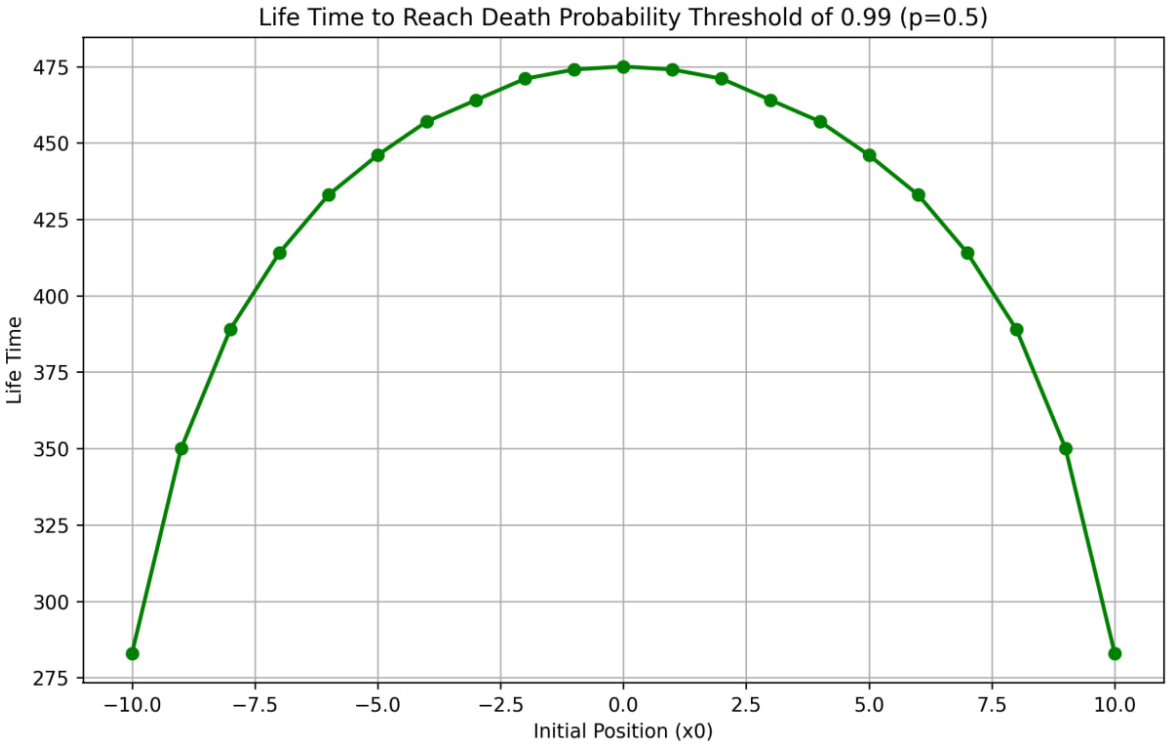
حال برای threshold مشخص هیت مپ را نمایش می دهیم.

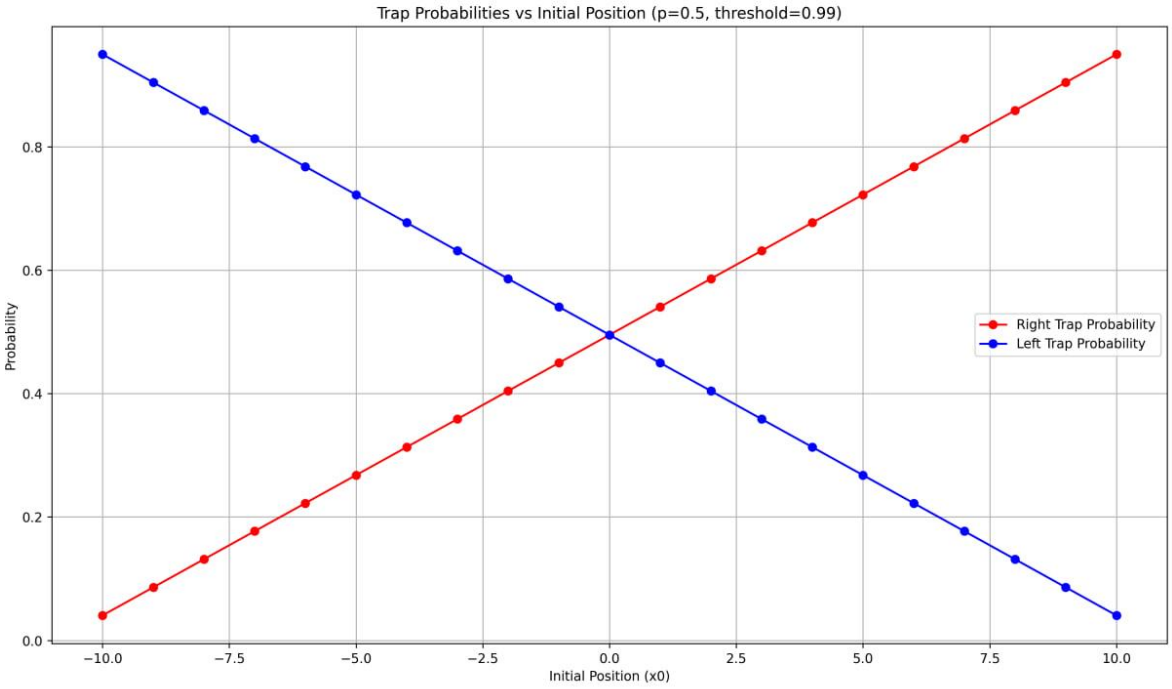


به ذهن می رسد که تابع توزیع ولشگت ها برای تعداد بالا گوسی است، پس از قانون 68% برای تابع گوسی شاید threshold = 0.68 مقدار خوبی برای بررسی مقادیر خواسته شده باشد. برای اطمینان از حدس خود نمودار N vs threshold را رسم می کنیم.



می بینیم که 0.68 نقطه خاصی نیست و بهتر است با threshold = 0.99 پیش برویم.
حال نتایج سوال قبل را تولید می کنیم.

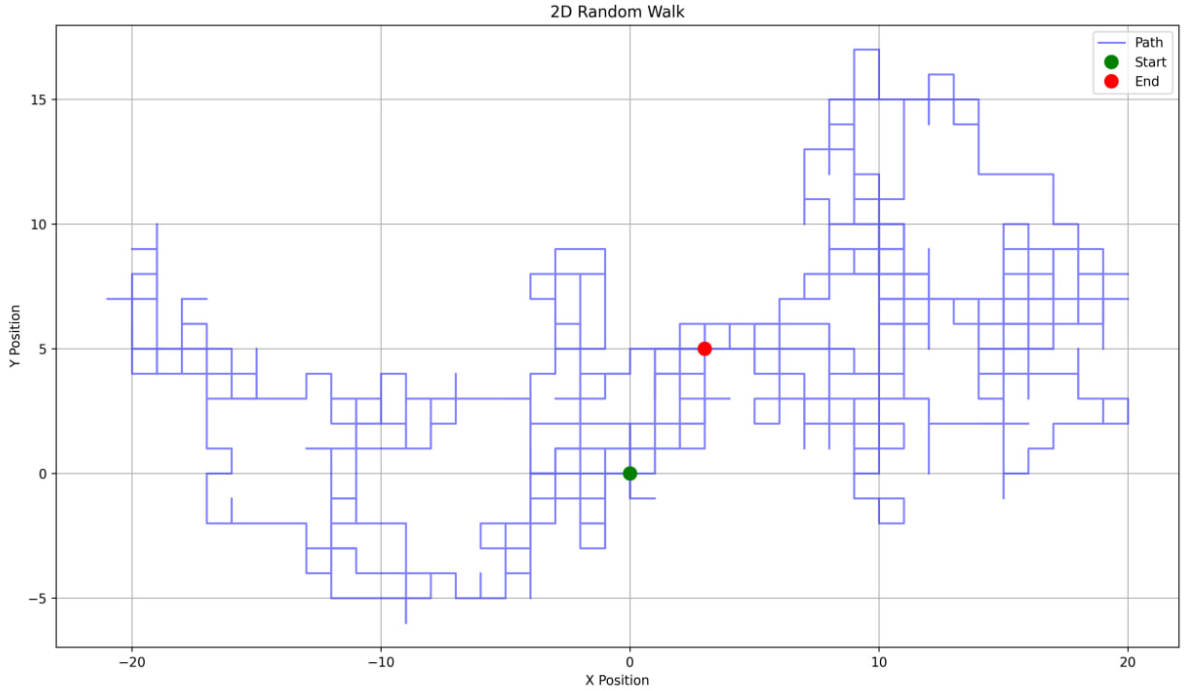




می بینیم که نتایج با سوال قبل تطابق دارند و دقیق هستند.

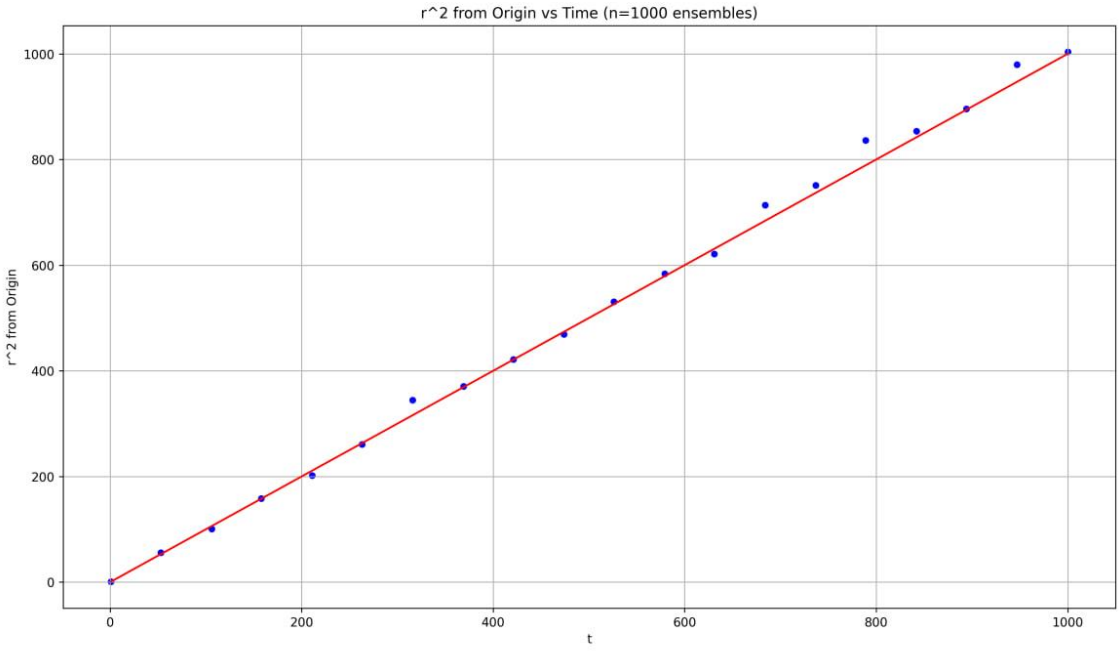
5. random_walk_2D

برای این الگوریتم خیلی ساده از بین بردار چهار قدم، به صورت رندوم یکی را انتخاب می کنیم و با مکان کنونی ولگشت جمع می کنیم.



نقاط شروع و پایان نیز مشخص شده اند.

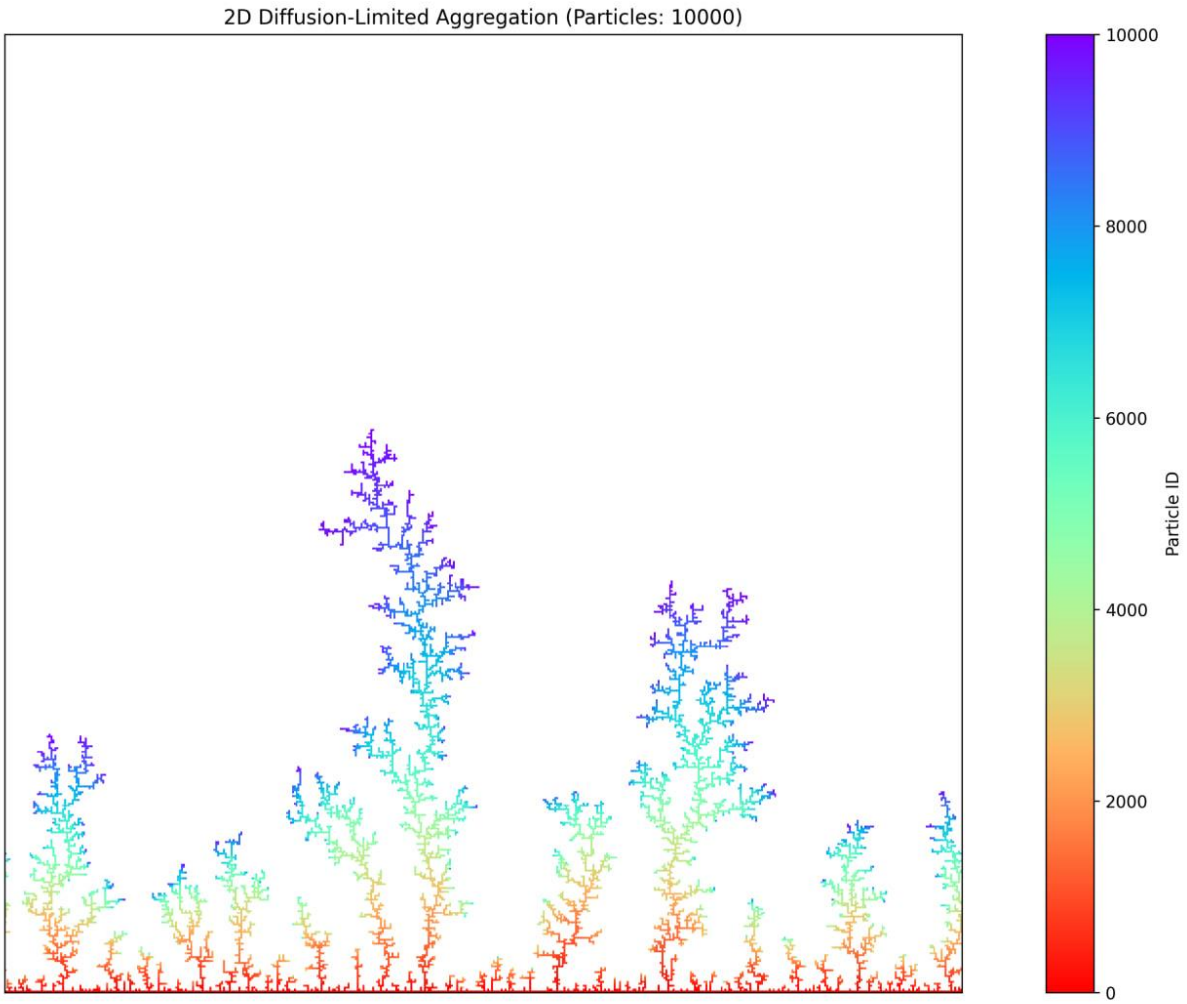
حال فاصله پایانی ولگشت از نقطه شروع را محاسبه می کنیم و برای زمان های مختلف روی آن آنسامبل می گیریم که نمودار رابطه 14 بدست آید.



خط قرمز مقدار تئوری است که منطبق بر نقاط است.

6. DLA

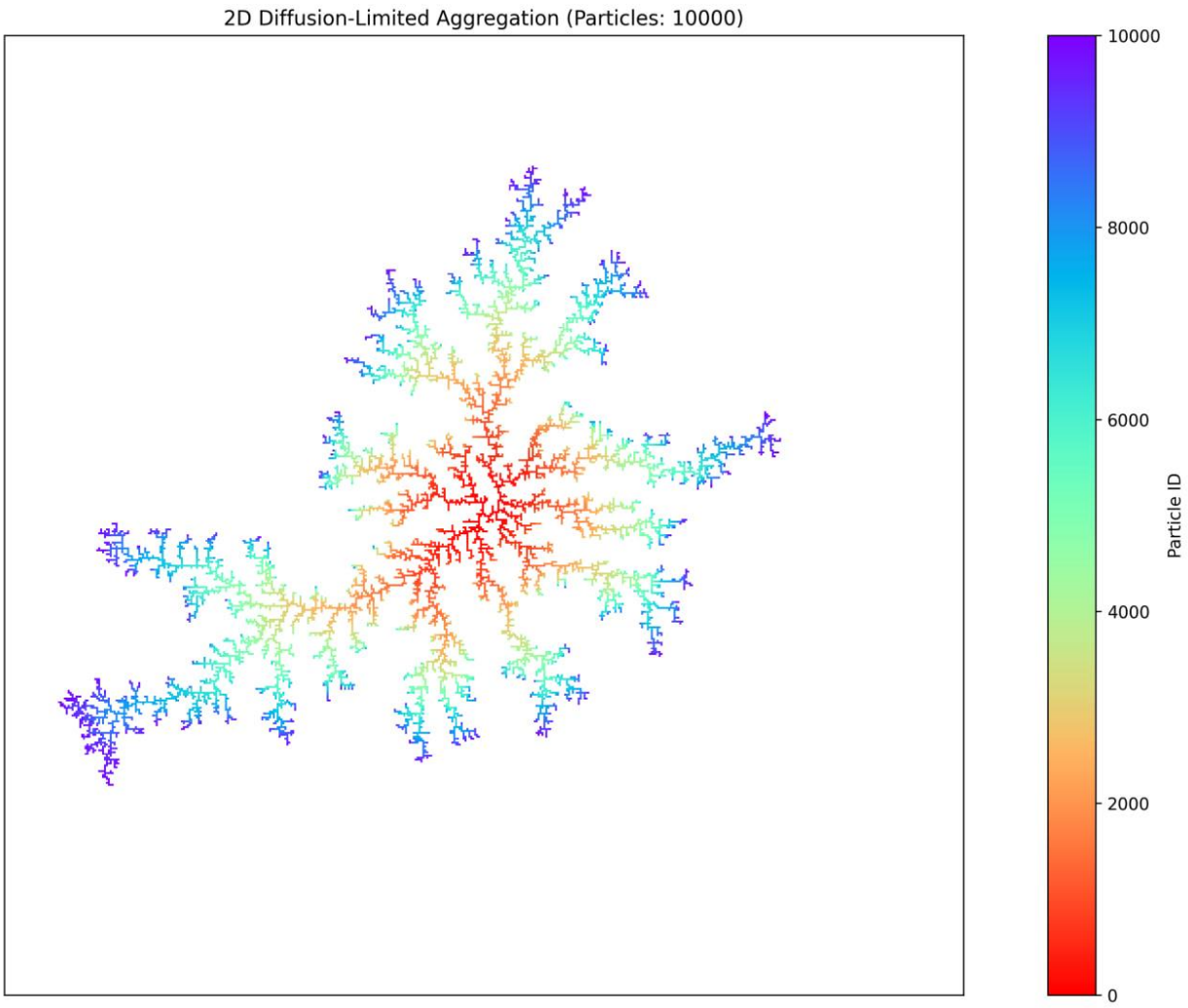
الگوریتم: از الگوریتم های لایه نشانی تابع get_neighbors را استفاده می کنیم. در تابع اصلی یک ذره را در محدوده بین launch_height و kill_height یا همان spawn_zone رها می کنیم تا ولگشت انجام دهد و در هر قدم دو شرط را بررسی می کنیم، یک اینکه ارتفاع ذره از kill_height بالاتر رفته است یا خیر، دو اینکه آیا ذره به ذره دیگری برخورد کرده است یا خیر. اگر برخورد کرده بود همان جا متوقف می شود و spawn_zone متناسب با بیشینه ارتفاع کل ذره ها بالاتر می رود.
در ماتریس grid رنگ ذرات color_value ذخیره می شود.



در این شبیه سازی رشد رقابتی مشاهده می شود.

7. circular_DLA, quadratic_DLA

دایروی: الگوریتم مانند سوال قبل است با این تفاوت که spaw_zone دایروی است، پس باید شعاع آن نسبت به خوشه زیاد شود. برای رها کردن ذره یک شعاع بین launch_height و kill_height و یک زاویه بین 0 تا 2π به صورت رندوم انتخاب می شود.

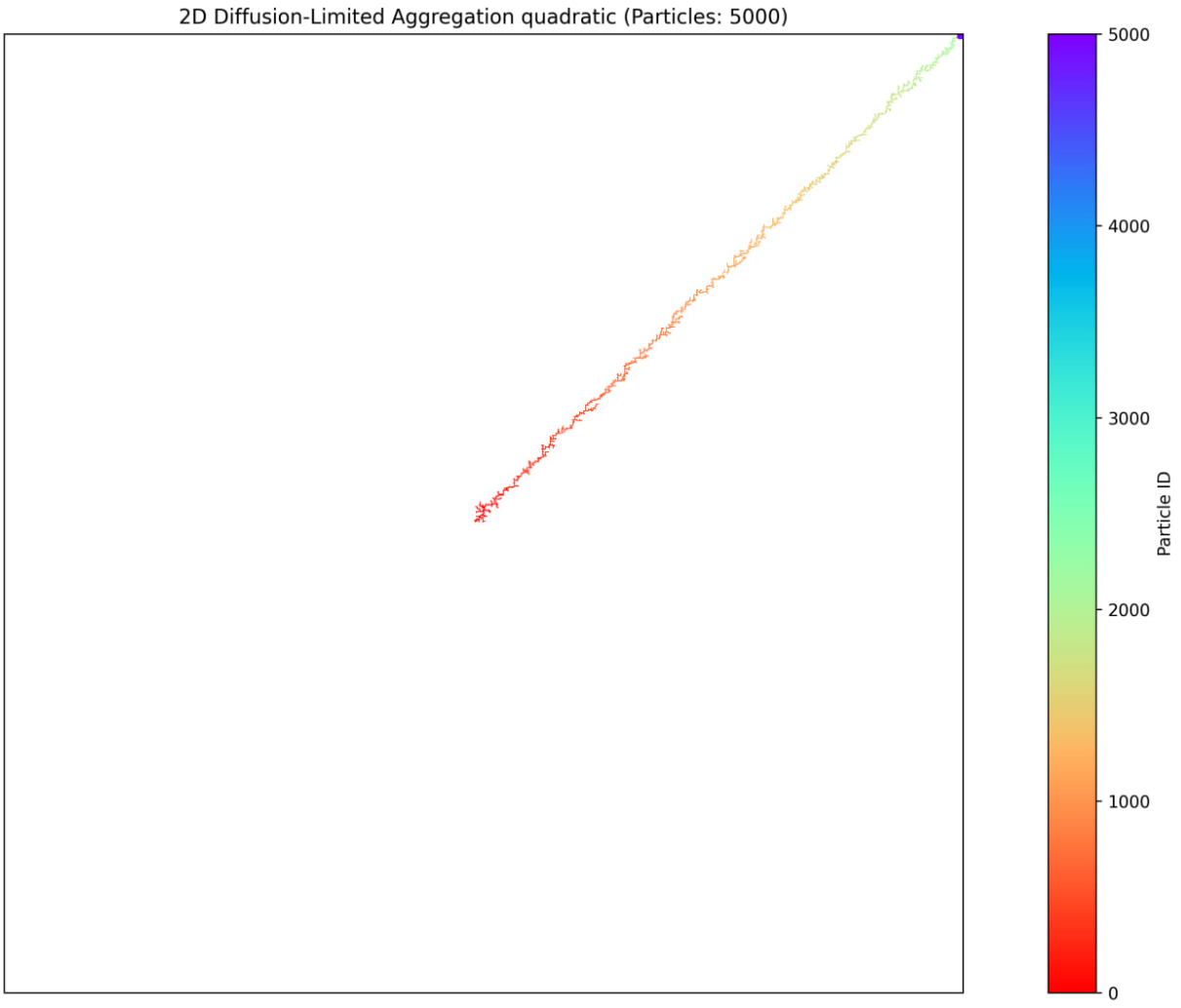


خوشه های بسیار زیبایی تولید می شوند. در طول زمان مشاهده می کنیم بعضی شاخه ها *spawn zone* را دور تر می کنند و برتری بیشتری در جذب ذرات پیدا می کنند.

مربعی: برای *spawn_zone* مربعی طول مربع باید نسبت به اندازه خوشه زیاد شود. که به نوبه خود چالش بر انگیز بود. برای اینکار dist_x, dist_y *را* تعریف می کنیم که فاصله ذره از مرکز شروع خوشه است، سپس چک می کنیم هر وقت یکی از این مقادیر پس از چسبیدن ذره ای بیشتر شد، طول spaw_zone بیشتر شود. باید توجه کنیم که *spawn_zone* از صفحه خارج نشود.

```
82                if max(dist_x, dist_y) > max(max_x, max_y):
83                    max_x = max(max_x, dist_x)
84                    max_y = max(max_y, dist_y)
85                    kill_l = min(height // 2, L // 2, kill_l + 1)
86                    launch_l = min(
87                        height // 2 - spawn_zone_l,
88                        L // 2 - spawn_zone_l,
89                        launch_l + 1,
90                    )
```

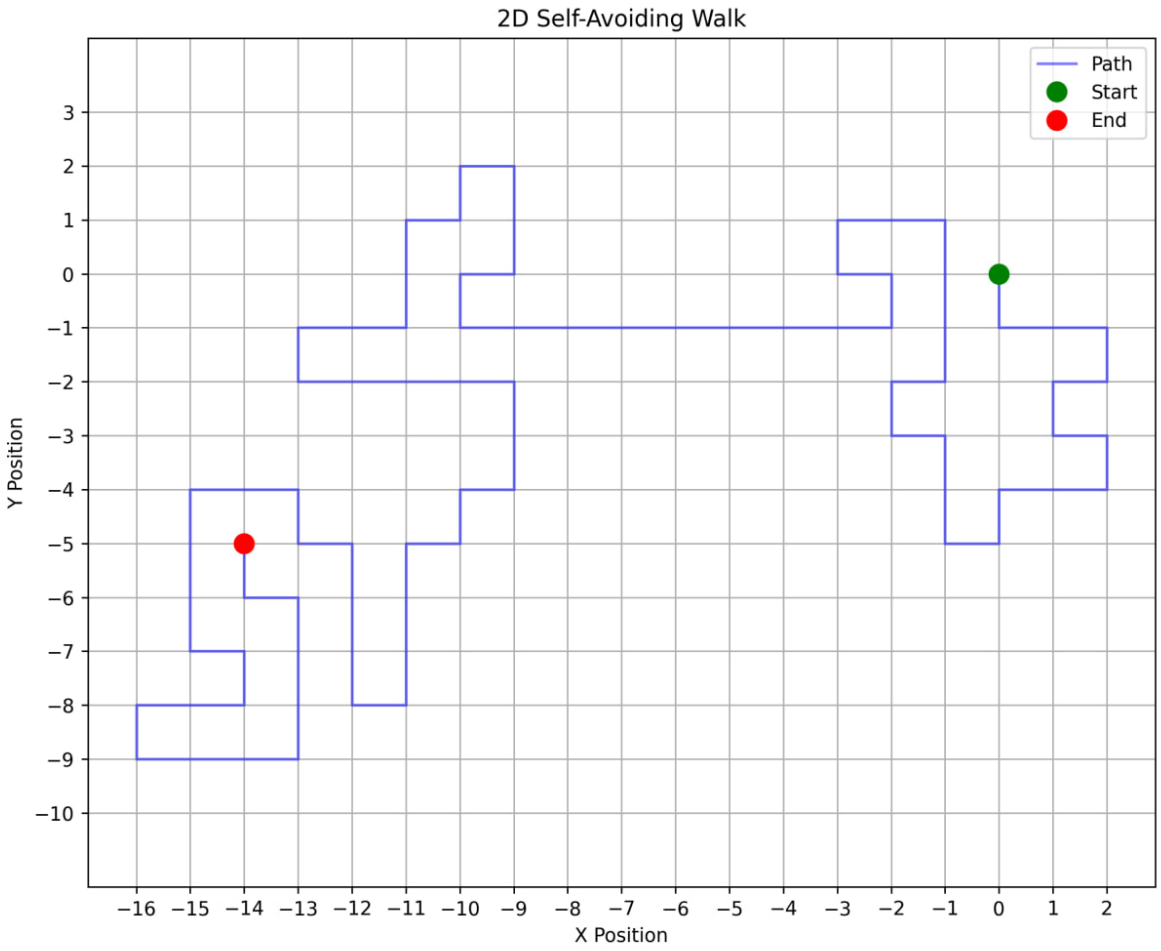
نتیجه شبیه سازی برای 5000 ذره به صورت زیر است.



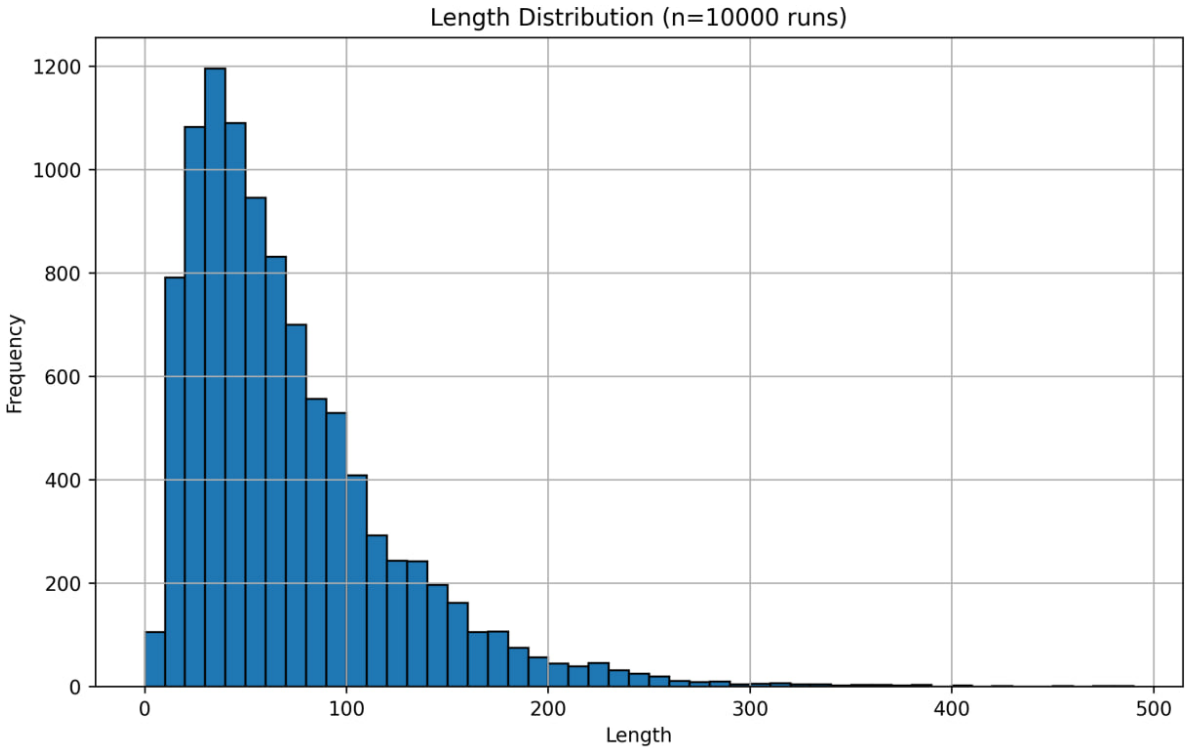
اختلاف کاملا مشاهده می شود. فقط یک شاخه با زاویه 45 درجه طول مربع را زیاد می کند و شانس رشد زیادی پیدا می کند.

8. SAW

برای کد ولگشت دو بعدی حافظه را در لیست visited ذخیر می کنیم و چهار قدم step_vectors را با استفاده از این حافظه فیلتر می کنیم و به لیتس available_steps تبدیل می کنیم و از بین اعضای این لیست رندوم انتخاب می کنیم. اگر available_steps خالی بود یعنی ولگشت به بن بست رسیده است.



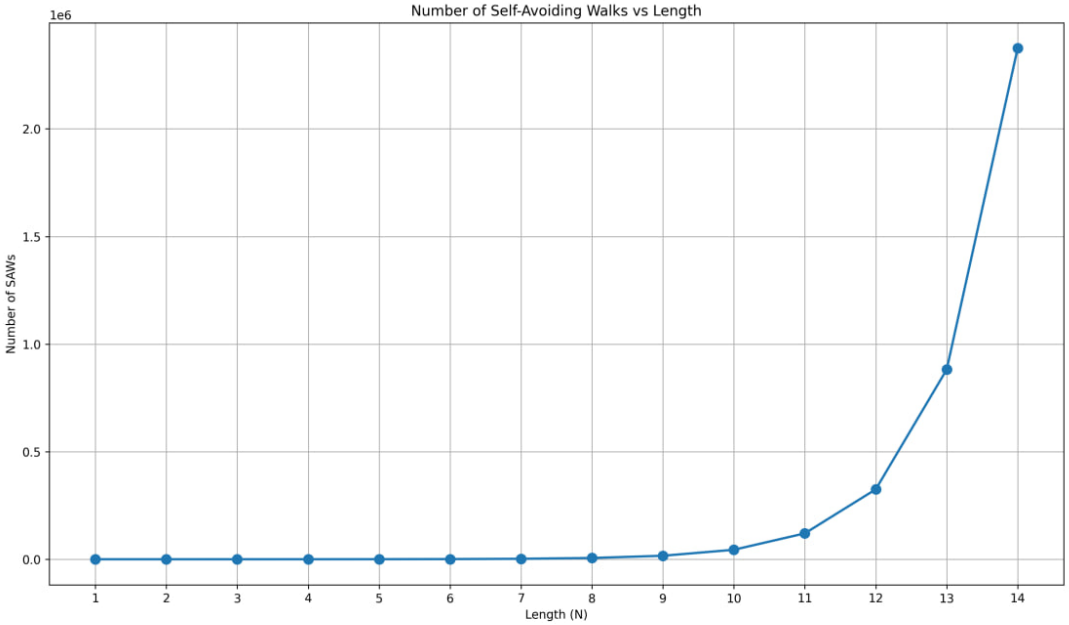
تابع توزیع طول های ولگشت را رسم می کنیم.



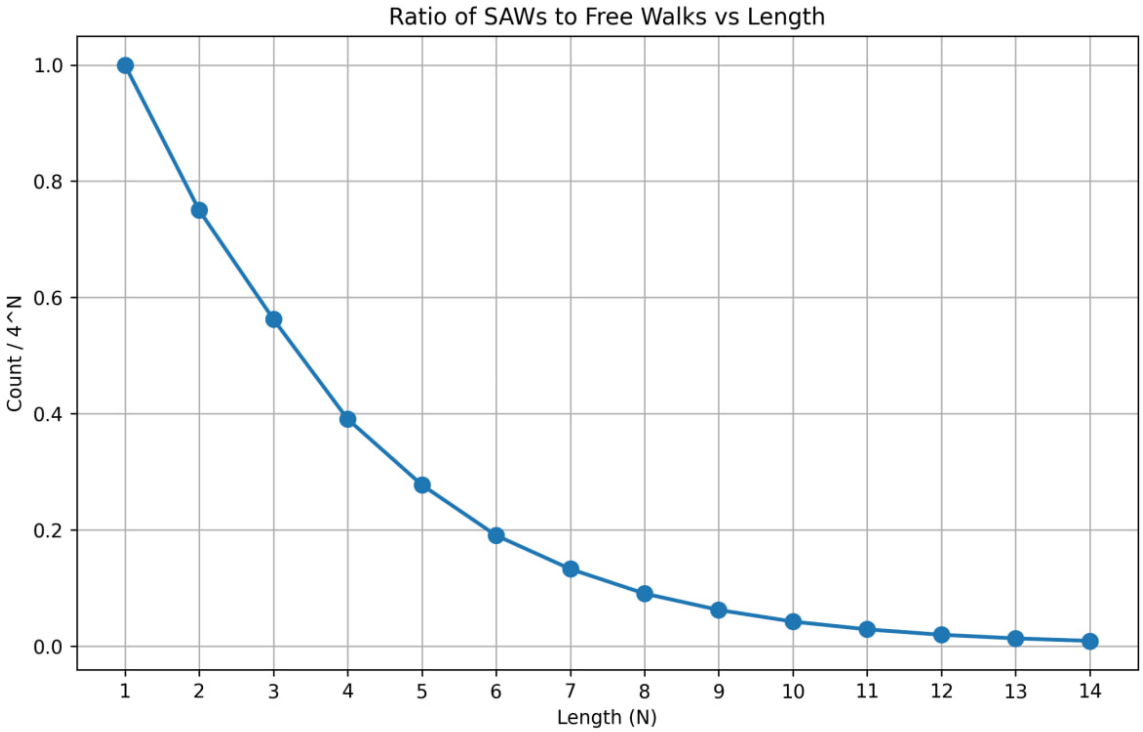
تابع توزیع رشد سریعی دارد و میانگین آن تقریبا روی 40 است.

9. possible_SAWs

الگوریتم: تابع بازگشتی explore را تعریف می کنیم که تمام حالات ممکن را برای قدم حال حاضر امتحان می کند و این تابع را در خودش کال می کنیم تا وقتی که به طول N برسیم.
نمودارگشت های ممکن بر حسب طول را رسم می کنیم.



تعداد گشت های ممکن به صورت نمایی افزایش می یابد و برای طول های بیشتر از 15 زمان محاسبه بسیار زیادی نیاز دارد.
حال نسبت تعدادها را به تعدادها گشت های آزاد رسم می کنیم.



در ابتدا این دو تعداد با هم برابرند و با افزایش طول این نسبت کمتر و کمتر می شود.