PSet4 Report

Ali Abolhassanzadeh Mahani

November 7, 2020

1 Standard Deviation

I proved this on a piece of paper and later, I scanned it using *CamScanner* and it's available here.

2 Random Walk

I made a function `rw_next()` that takes the current postion and moves to left / right with a given probability. Then I simulate the random walk and take the data. I took the data in 100 choice intervals, from 100 to 1000 and ran the simulation 10000 times in the `rw_simul()` function. The plots are available in Fig2)

3 Random Walk with Trap (Random)

I simulated this problem using a class called `Drunk`. This is the actualization of a drunk person dangling around in 1-D. I set the path to be from 0 to 19 and traps at -1 and 20. If the drunk gets to those points, it perishes and I can then get its lifetime. I ran the simulation for 10000 drunk men and took the mean. The resulting graph is plotted in Fig3

4 Random Walk with trap (Deterministic)

This time, I simulated and used the `profile` library in python to find the exact run time of my algorithm. The resulted graph is available in Fig4.

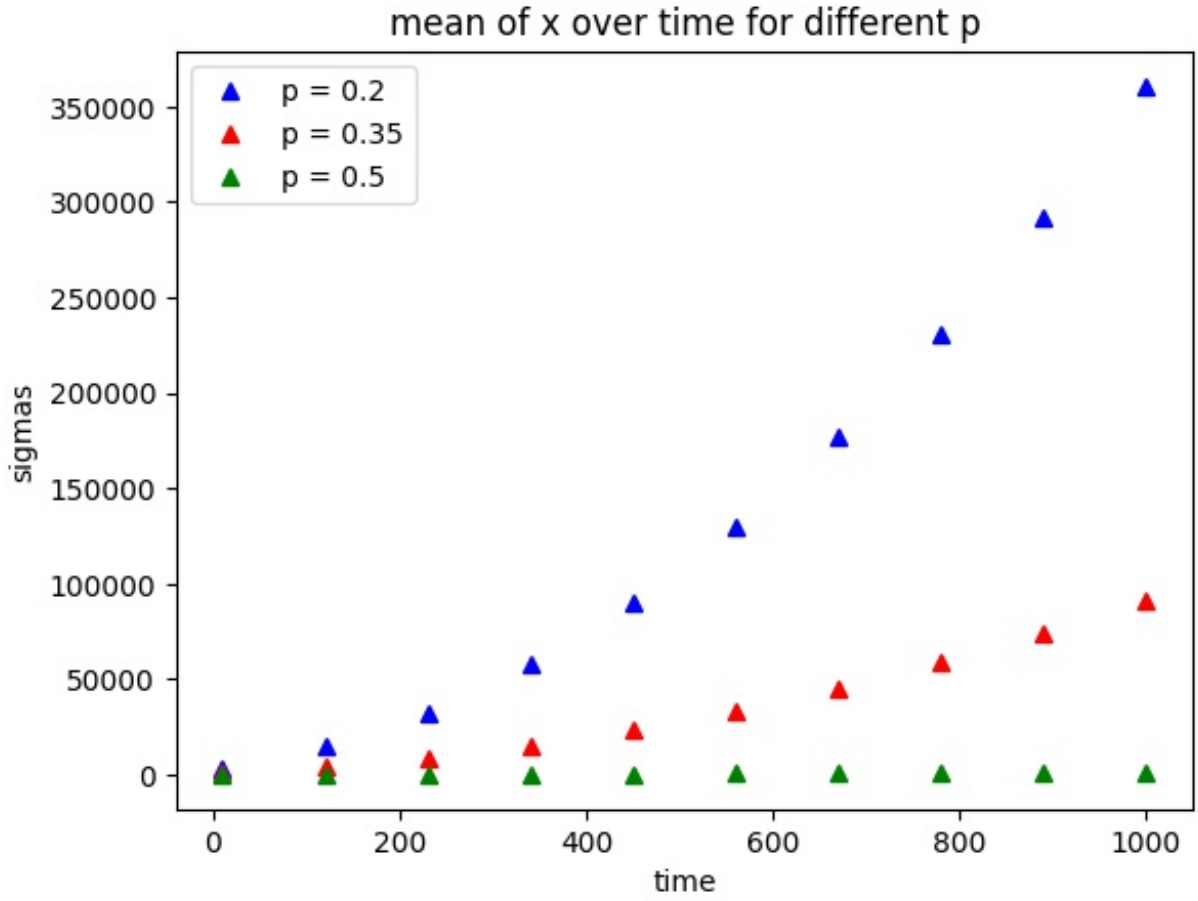
I did the same for the random generation algorithm here we shall see the difference. My system has an *Intel Core i5 8th gen* CPU and 8 GB of DDR4 RAM. the result are:

- **Deterministic:** $0.636s$
- **Random:** $139.667s$

This is an example where deterministic algorithms perform much faster!

5 2-D Random Walk

This time, I simulated the `Drunk` person to be a girl on a 2-D surface. Almost the same as the 1-D instance, but returns r^2. I ran the simulation with time intervals of 100 for 10 intervals. The plot is available in Fig5



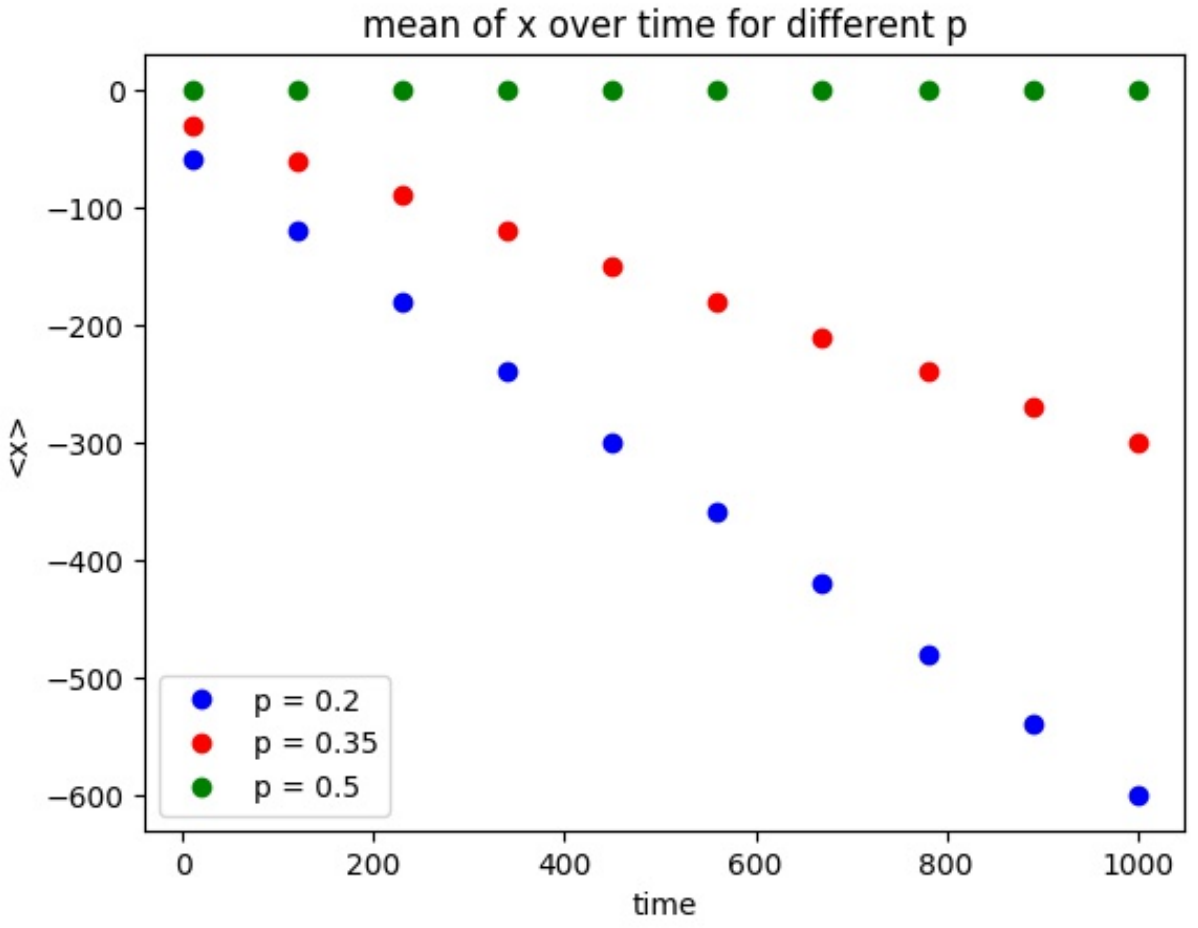


Figure 1: The plot of σ^2 (top) and $< x >$ (bottom) over time for different probabilities. Took 10000 runs and got the mean.

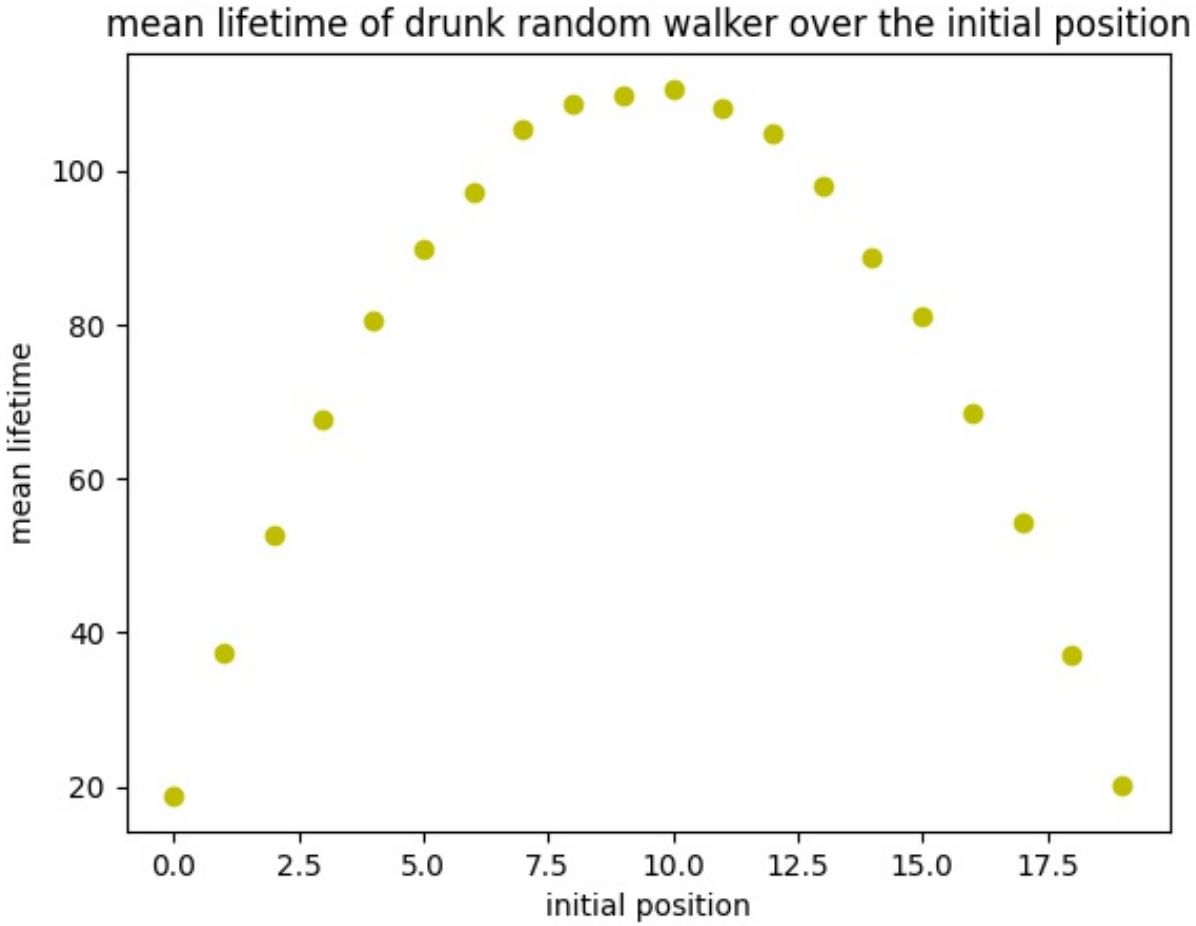


Figure 2: mean lifetime over different indices of the path. traps at -1 and 19. Ran 10000 instance simulations

6 Aggressive Layer Deposition

I made a `RandWalker` class that generates a particle at a random x and the highest y that I chose to be 199. The length of the seed is 201 for symmetry that proved to play no role in the simulation whatsoever :-)
I chose the buffer length to be 10. If the particle exceeds that buffer, I reset the position with a random x position again.

I made 2000 particles and gave them room to random walk and stick to the seed. Then I visualized the grid using `plt.pcolor()`.

I also tried changing colors but apparently something is wrong with it. Maybe the color code intervals.

The plot is in Fig6

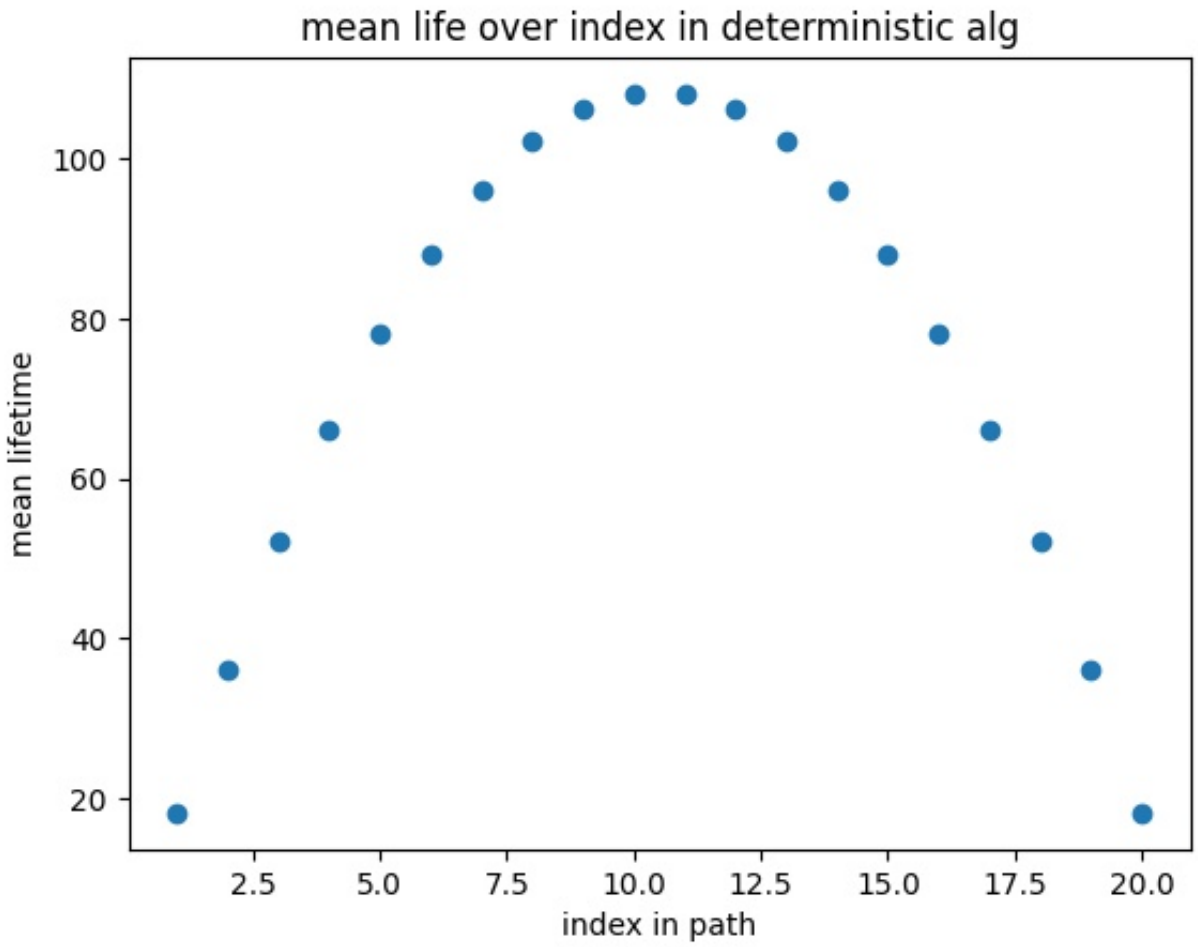


Figure 3: mean lifetime over index in path which the drunk person begins from. Deterministic algorithm.

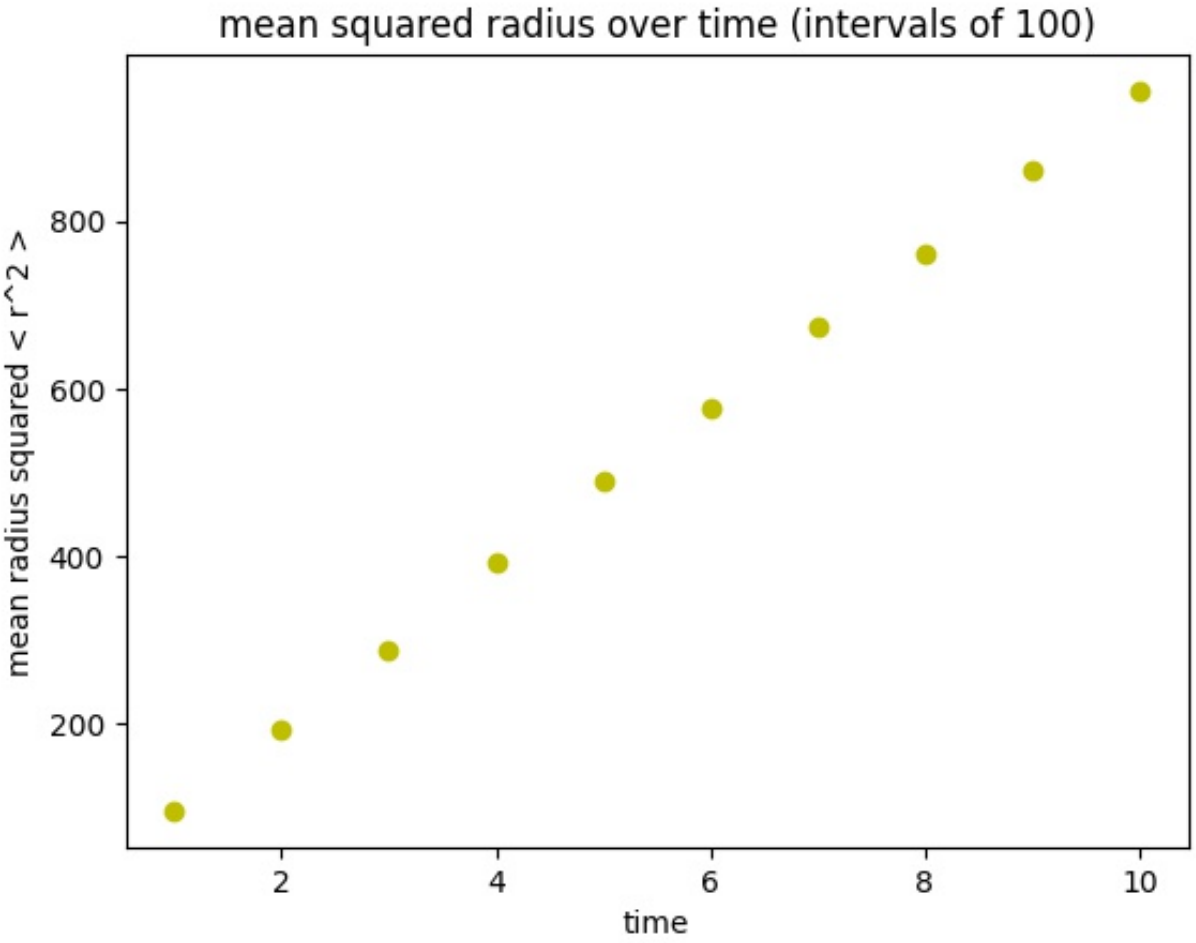


Figure 4: dependence of The distance squared r^2 over time intervals of 100 choices.

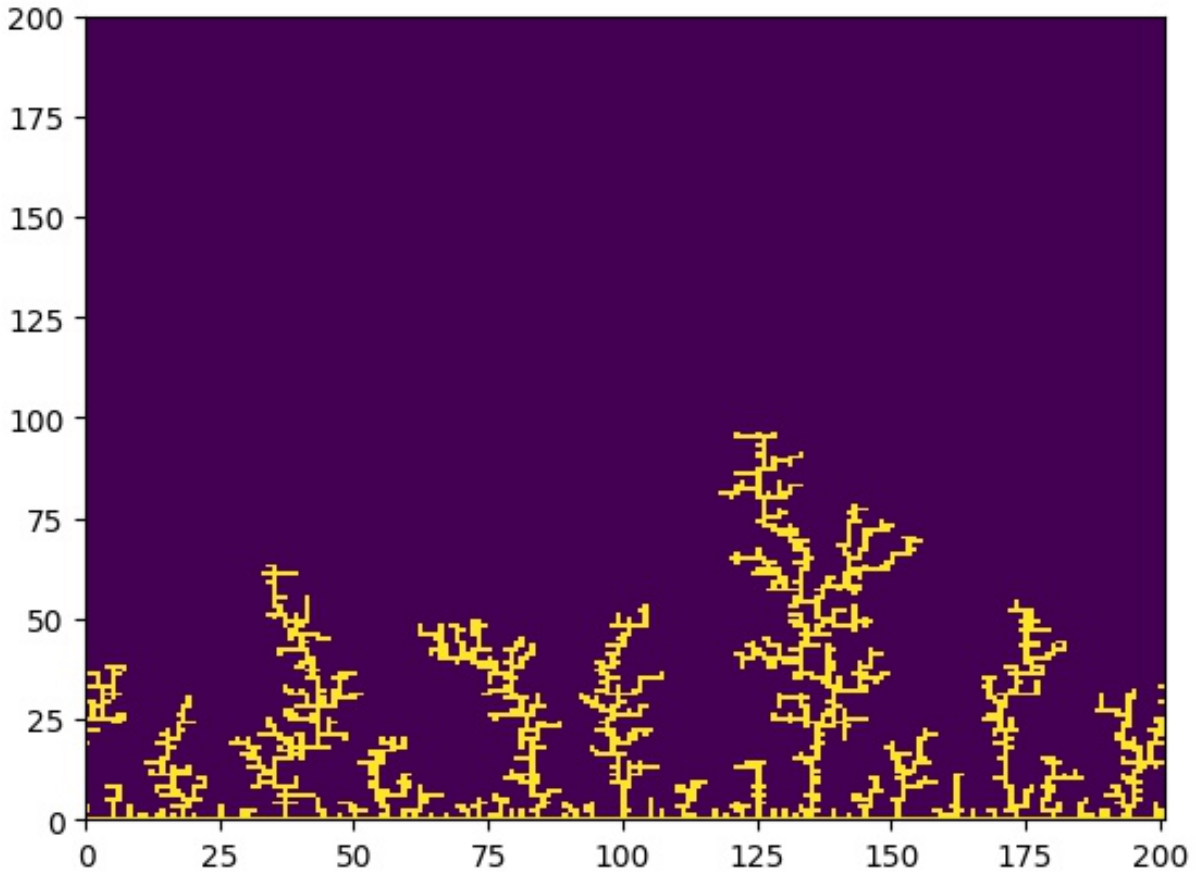


Figure 5: deposited 2000 randomly positioned particles using random walk simulation.