Extra Credit Answer:

1. The algorithmic complexity of the current code is O(MN)
   Where,
   M = number of paths
   N = number of patterns
   This has a quadratic complexity and as the number of paths or patterns scale up, the complexity will grow and obtaining a solution becomes infeasible
2. Tested on 10K paths, 10K patterns, the sequential code took 27.82 seconds on an i7 processor with 64GB ram. The simulation will increase drastically as the size grows to order of Million's. File name: my_code_sequential.py
3. A possible solution to reduce the complexity is to parallelize the algorithm. This can be done using multiprocessing package of python. Given P number of processors, the algorithm complexity reduces to O(ceil(MN/P))
   Where,
   P = number of cores
4. Tested on 10K paths, 10K patterns, the parallel code took 8.23 seconds on an i7 processor with 64GB ram and 4 cores. File name: my_code_parallel.py
5. Both sequential and parallel algorithms generates same output