Self-Avoiding Walks On a Rectangular Grid

Harin Lim, Evan Zhang, Alice Zhao

March 2020

1 Problem Statement

A self-avoiding walk on a rectangular lattice is a path from the bottom left lattice points to the top right lattice points in which no vertex and no edge is revisited. Counting these walks involves recursive functions as well as traditional counting methods. Different categories of self-avoiding walks can be explored. Examples include walks that move in exactly three of the four cardinal directions, minimum length walks, walks of a certain length, walks on a certain related grid sizes (such as 2 by N for various N).

2 Definitions

1) Minimum-length path: a path where only two directions, right and up, are allowed

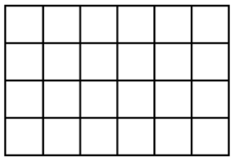
2) Let $(0, 0)$ be the bottom left corner of a grid. The positive x-direction is rightwards, and the positive y-direction is upwards.

3 Grids with Two Directions

3.1 For a m x n rectangular grid, there are $\binom{m+n}{m}$ minimum-length paths

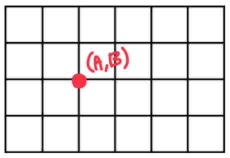
Proof: To travel from the bottom-left corner to the top-right corner of a m x n grid, a player must go right m times and go up n times. A minimum-length path is then a unique permutation of m right moves and n up moves for a total of $m + n$ moves.

For example, in a $6x4$ rectangular grid, a player must move right 4 times and up 6 times to reach $(4, 6)$ from $(0, 0)$.



We can visualize this player's moves as RRRRUUUUUU. Each minimum-length path must be a unique permutation of these moves, and using simple counting principles, we know that there are $\frac{10!}{6!4!}$ moves here, or $\binom{10}{4}$ (which is equal to $\binom{10}{6}$). The same logic extends to the general m x n grid: there must be m right moves, R, and n upwards moves, U, and the total number of permutations is equal to $\frac{(m+n)!}{m!n!} = \binom{m+n}{m}$, proving the conjecture.

3.2 For a m x n rectangular grid, there are $\binom{a+b}{a}\binom{m+n-a-b}{m-a}$ minimum-length paths that must go through a point (a, b)



Proof: Split the rectangular grid into two separate rectangular grids, one with corner points $(0, 0), (a, 0), (a, b), (b, 0)$ and the other with corner points $(a, b), (a, n), (m, n), (m, b)$.

Now you can split the question into finding two minimum-length paths, one from $(0, 0)$ to (a, b), and one from (a, b) to (m, n). This is because any walk that deviates from such a path would not be able to hit either (a, b) or (m, n). Thus, using our findings from Conjecture 5.1, there are $\binom{a+b}{a}\binom{m+n-a-b}{m-a}$ minimum-length paths in a m x n rectangular grid that go through the point (a, b).

3.3 For a m x n rectangular grid, there are $\binom{m+n}{m} - \binom{a+b}{a}\binom{m+n-a-b}{m-a}$ minimum-length paths that avoid a point (a, b)

Proof: Using our findings from Conjecture 5.2, there are $\binom{a+b}{a}\binom{m+n-a-b}{m-a}$ minimum-length paths that must go through a point (a, b). Thus, using our findings from Conjecture 5.1 and the Principle of Inclusion-Exclusion, there are $\binom{m+n}{m} - \binom{a+b}{a}\binom{m+n-a-b}{m-a}$ minimum length paths that avoid a point (a,b).

3.4 There are 2^n paths in three directions from $(0, 0)$ to $(1, n)$ on an 1 x n grid

Proof: Note that there must be n upwards moves in such a path. For the first move, the two cases are to either move upwards, or move right and then move upwards. After each upwards move, there is a choice to either continue upwards or to make a left/right move and then be forced to continue upwards. With n upwards moves needed and 2 choices after each, there are a total of 2^n unique paths from $(0, 0)$ to $(1, n)$.

4 Grids with Three Directions

Consider a 3 x 4 graph in three directions. Let these directions be up, right, and left. Instead of starting paths from the bottom left corner to the top right corner, it becomes more manageable to work backwards from the top right corner.

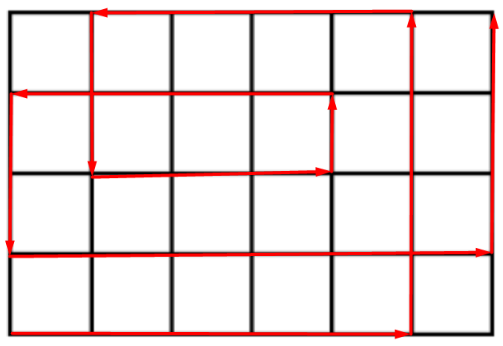
To reach the top right corner, the path must pass through at least one of the two points directly adjacent to the last point. Examining the point below the end destination, we can observe that there are exactly five paths from that point to the last point using three directions of motion. Examining the four other points on the same row, we can observe that there are also exactly five paths from each point to the last point using three directions of motion.

Starting from any point on the row below, or the third row of points from the top, there are exactly five paths in which the last point is the first point that the path reaches on the second row. There are five paths from each point on the second row to the top right corner, thus, there are 25 paths in total from any point on the third row to the top right corner. Similarly, examining the fourth row, there are exactly five unique paths from any point on the fourth row to the first point that the path reaches on third row. As there are 25 paths from any point on the third row to the top right corner, there are 125 paths from any point on the fourth row to the top right corner. Thus, for a 3 x 4, which has 4 rows and 5 columns of points, there are 125 total paths from the bottom left corner to the top right corner.

To generalize this situation, consider an m x n graph. There are exactly $n+1$ paths from any point on the second row from the top to the top right corner. There are exactly $n+1$ paths from any point on the third row to the first point the path reaches on the second row. Thus, there are $(n+1)^2$ paths from any point on the third row to the top right corner. Continuing for m rows, we find that with each additional row, the number of total paths is multiplied by $n+1$. Thus, for an m x n graph in three directions, where the directions are up, left, and right, there are a total of $(n+1)^m$ paths from the bottom left corner to the top right corner.

5 Grids with Four Directions

When movement is restricted to two directions, the number of paths is feasible, as it is just a simple combination. When movement is restricted to three directions, the number of paths becomes less manageable, though still possible as the number of paths increase in a slow exponential. However, consider movement not being restricted whatsoever. The number of paths increases at an alarming rate.

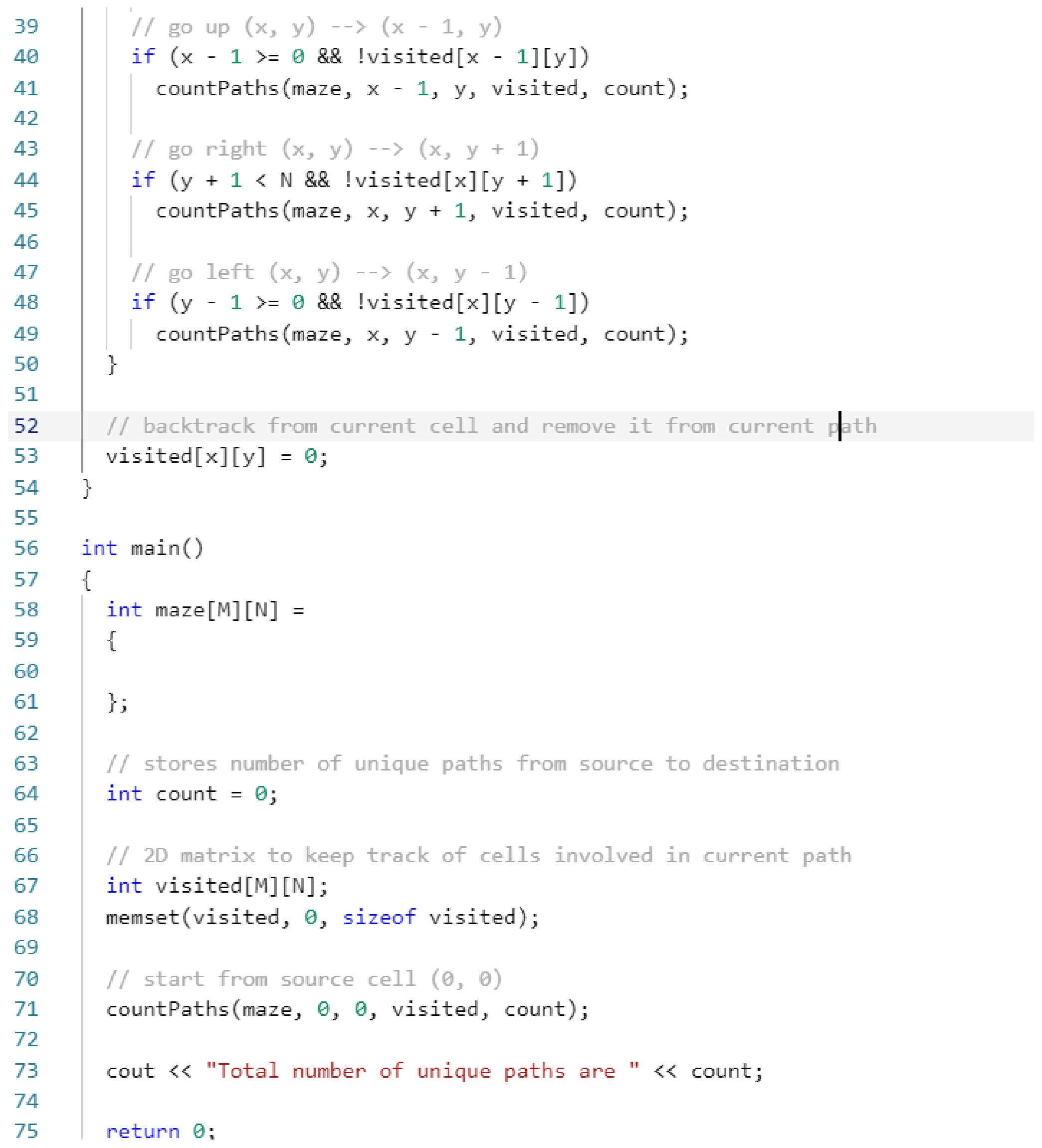


Thus, our group found and modified C++ code where the computer would find the number of paths when movement is not restricted. The code is the following:

5.1 C++ code

Link: https://repl.it/@clcironic/LightgrayObviousWeb

```cpp
#include <iostream>
#include <cstring>
using namespace std;

// M <= N, where M = x+1 and N = y+1 in a x by y grid

#define M 4
#define N 4

// Check if cell (x, y) is valid or not
bool isValidCell(int x, int y)
{
  if (x < 0 || y < 0 || x >= M || y >= N)
    return false;

  return true;
}

void countPaths(int maze[M][N], int x, int y, int visited[M][N], int& count)
{
  // if destination (bottom-rightmost cell) is found,
  // increment the path count
  if (x == M - 1 && y == N - 1)
  {
    count++;
    return;
  }

  // mark current cell as visited
  visited[x][y] = 1;

  // if current cell is a valid and open cell,
  if (isValidCell(x, y) && maze[x][y] == 0)
  {
    // go down (x, y) --> (x + 1, y)
    if (x + 1 < N && !visited[x + 1][y])
      countPaths(maze, x + 1, y, visited, count);
```



```cpp
    // go up (x, y) --> (x - 1, y)
    if (x - 1 >= 0 && !visited[x - 1][y])
      countPaths(maze, x - 1, y, visited, count);

    // go right (x, y) --> (x, y + 1)
    if (y + 1 < N && !visited[x][y + 1])
      countPaths(maze, x, y + 1, visited, count);

    // go left (x, y) --> (x, y - 1)
    if (y - 1 >= 0 && !visited[x][y - 1])
      countPaths(maze, x, y - 1, visited, count);
  }

  // backtrack from current cell and remove it from current path
  visited[x][y] = 0;
}

int main()
{
  int maze[M][N] =
  {

  };

  // stores number of unique paths from source to destination
  int count = 0;

  // 2D matrix to keep track of cells involved in current path
  int visited[M][N];
  memset(visited, 0, sizeof visited);

  // start from source cell (0, 0)
  countPaths(maze, 0, 0, visited, count);

  cout << "Total number of unique paths are " << count;

  return 0;
```

5.2 Analysis

x and y	1	2	3	4	5	6
1	2	-	-	-	-	-
2	4	12	-	-	-	-
3	8	38	184	-	-	-
4	16	125	976	8512	-	-
5	32	414	5382	79384	1262816	-
6	64	1369	29739	752061	20562673	575780564
7	128	4522	163496	7110272	336067810	-
8	256	14934	896476	67005561	-	-
9	512	49322	4913258	630588698	-	-
10	1024	162899	26932712	-	-	-

To analyze and model the data, we set the y value constant and varied the x values. We put the numbers into LoggerPro and then modeled the number of paths, P, using the curve function e^{Cx} for a constant C. For practical reasons, we did not include data where the value of $P > 1x10^9$.

We got the following models:

y $= 1 \Rightarrow P = e^{0.6931x}$

y $= 2 \Rightarrow e^{1.2x}$

y $= 3 \Rightarrow e^{1.711x}$

y $= 4 \Rightarrow e^{2.251x}$

y $= 5 \Rightarrow e^{2.805x}$

y $= 6 \Rightarrow e^{3.362x}$

We ended where $y = 6$ because there is not enough data to sufficiently continue to model higher constant y values.

Next, we plotted the C value for each of the models versus the constant y value, and then put a linear fit $(y = mx + b)$ through it.

We get the model $y = 0.5343x + 0.1337$.

Thus, for variable x, y values, we get the model $P = e^{(0.5343y+0.1337)x}$

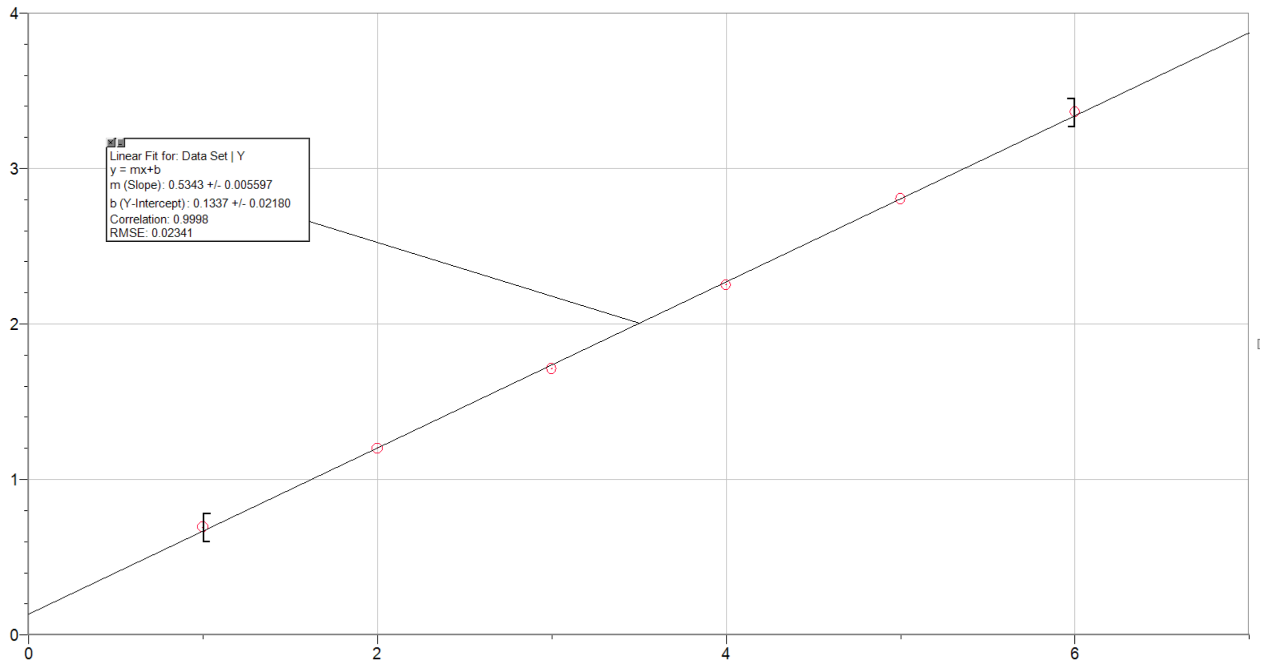


Figure 1: Logger Pro Graph

6 Comparisons between 2,3,4 directions

For 2 directions, $P = \binom{x}{x+y}$

For 3 directions, $P = (x+1)^y$ when the third direction is moving horizontally.

For 4 directions, $P = e^{(0.5343y+0.1337)x}$

7 Further Investigation

Scenarios to investigate next include:

1) Generalizations for 2 x n grids

2) The number of walks of a certain length from point A to point B