Random Walks: An Analysis of Non-self-intersecting Paths

Marcus Sammer

Follow this and additional works at: https://knowledge.e.southern.edu/senior_research

Part of the Physical Sciences and Mathematics Commons

Recommended Citation
https://knowledge.e.southern.edu/senior_research/90

This Article is brought to you for free and open access by the Southern Scholars at KnowledgeExchange@Southern. It has been accepted for inclusion in Senior Research Projects by an authorized administrator of KnowledgeExchange@Southern. For more information, please contact jspears@southern.edu.
Random Walks:
An Analysis of Non-self-intersecting Paths

by

Marcus Sammer

Southern Scholars Project
4-9-99
INTRODUCTION

Given a two dimensional square grid it is simple to count the number of possible paths for any given length from a fixed origin. For to determine these values, one can use a simple formula. This formula is given by $P(n) = 4^n$ where $P$ is the number of possible paths, and $n$ is the given length. This formula is determined by noting that at each step in the path there is the option of going one of four directions: North, South, East, or West, if these directions are the arbitrarily defined possibilities on the grid.

The next natural step is to make the problem more interesting by adding restrictions. One possible restriction is to allow only non-self-intersecting paths. This means that no path can cross any point on the grid more than once. For very short path lengths it is convenient to draw all of the possible paths and count the non-self-intersecting ones. But for paths longer than about four steps it becomes cumbersome to draw them all by hand. For longer paths, it is not difficult to write a computer program that traces through each possible path and counts the non-self-intersecting ones. But because the total number of paths is an exponential function of $n$, even a computer is limited in the length of paths it can to trace through and count. So because it is not even possible for a computer to trace through and determine all of the non-self-intersecting paths, another formula is needed to calculate these values. Consequently, I devoted my research to that end. I became familiar with some of the properties of non-self-intersecting random paths and worked toward the goal of creating and proving a simple formula for the number of non-self-intersecting paths of any given length.

METHODS

Computer programs written in C++ are used to trace through the possible paths starting at
a fixed origin on a two dimensional square grid. Program 1 (appendix J) uses a recursive algorithm to trace through the paths using a two dimensional array (a computer storage type) to emulate the two dimensional square grid. It uses another two dimensional array to tabulate the number of non-self-intersecting paths that end at each position on the grid. Programs 2 through 5 (appendices K through N) each use the same basic algorithm with slight modifications. These programs treat each path as a string of the letters N, S, E, and W, standing for North, South, East, and West respectively. These programs use a recursive algorithm to generate all of the possible strings of N, S, E, and W of a given length, and then they test the strings for loops. If a loop is found in a path, then the path intersects itself and thus is not included with the non-self-intersecting paths. This testing relies on the fact that for a string representing a path to contain a loop, the string must contain a substring in which the number of North’s equals the number of South’s and the number of West’s equals the number of East’s.

The data provided by these programs were then analyzed for patterns either by hand using pencil and paper or imported into a spreadsheet for easier manipulation.

RESULTS

The bulk of my research consisted of generating and analyzing data on non-self-intersecting paths. For in order to come up with a formula I first needed to find some sort of pattern in the data and figure out a way of determining the factors involved in making a path self-intersect so that I could distinguish the non-self-intersecting paths from the self-intersecting ones without tracing through each path. Here I will explain the data that I have calculated, studied, and included in this report.

Appendix A lists the number of non-self-intersecting paths for lengths up to 20 steps;
these were generated by Program 1 (Appendix J). Also in appendix A is the total number of paths, calculated using the formula $4^n$. The number of self-intersecting paths included in this index was calculated by subtracting the number of non-self-intersecting paths from the total number of paths.

Appendix Q contains the patterns generated by Program 1 (Appendix J) in tabulating the number of paths that end at each point on the grid. The $n=4$ pattern, for example, is the pattern generated by paths of length four. Appendix H contains the notation that will be used in describing these patterns. For example, $C1L1$ denotes the number at the four tips of each pattern found in appendix Q. Furthermore, $L1$ by itself denotes the diagonals found on the outside of each pattern, and $C1$ represents the center row or middle column of each pattern.

Generated by Program 2 (Appendix K), appendix B lists the number of loops of each size contained in the paths of lengths up to 10 steps. Generated by Program 3 (Appendix L), appendix C lists the same information as appendix B, except that loops that are completely contained within other loops are not included. That is, if a substring $S$ of a string representing a path is determined to form a loop, proper substrings of $S$ that also form loops are not included in the totals.

Generated by Program 4 (Appendix M), appendix D lists the number of loops of every size found in paths that were non-self-intersecting at $n-1$ steps, but intersect themselves on the $n$th step. And because they intersect on the last step, they are called end loops. It may also be noted that these data contain the number of simple loops of length $n$; this is the number of paths of length $n$ that are non-self-intersecting expect that they end at the same point as they began. These may be found by looking at the number of end loops of length $n$ with a total path of length $n$.

Generated by Program 5 (Appendix N), appendices E through G list the number of paths that end at three different points along the row $C1$. They are categorized by the number of
corners or turns in each path. A corner is the place where a path changes direction. They can be found by looking for the places where adjacent letters in a string, which represents a path, are different from each other.

**DISCUSSION**

The results from Programs 2 through 4 have not produced anything that would suggest that they will be useful in determining a formula for the number non-self-intersecting paths of length n. The most productive line of inquiry has come from looking at the patterns generated by Program 1. It is immediately apparent that the data form diamond shapes with alternating diagonals of positive integers and zero's. The diamond is the familiar shape created by the set of points that are equidistant from the origin in “taxicab” geometry, which deals with limiting travel on a grid to the four directions North, South, East, and West. The alternating rows of zeros and positive integers come from the fact that paths of even length cannot end on the same diagonals as paths of odd lengths.

Besides these features, the data are also interesting because the outside diagonals look like the rows of Pascal’s Triangle (appendix O). In fact, by stacking the \( L_n \) diagonals, where \( n=1, 2, 3, \ldots \), one does obtain Pascal’s Triangle. This is because for every non-self-intersecting path of length n that ends at a point adjacent to some point P but never touches P there is a non-self-intersecting path of length \( n+1 \) that ends at point P. And conversely, for every non-self-intersecting path of length \( n+1 \) that ends at point P there is a non-self-intersecting path of length n that ends at a point adjacent to point P but never touches P. Thus to determine the number of paths of length n that will end at a given point at which no paths of length \( n-1 \) end, one may take the sum of the numbers of paths of length \( n-1 \) that end at points adjacent to the given point. And
thus by looking at the nth pattern one may determine the number of paths that will end on the L1 diagonals of the (n+1)th pattern. This comes from summing the two numbers from the L1 diagonals of the nth pattern that are adjacent to the desired points. Since there are well known formulas for the rows of Pascal’s Triangle, counting the number of paths that end on an L1 diagonal is consequently made easy.

Inside the L1 diagonals, the patterns become more difficult. One way to approach the problem is to look at the numbers for an individual position in each pattern, starting with the n=1 pattern and continuing for increasing values of n. For example, the C1L1 position contains a one in every pattern. The C1L3 position is not constant, but taking the difference between subsequent numbers, one finds that each number is two greater than the previous number. The C1L5 position is not constant, and neither are the differences found by subtracting subsequent numbers. But if one takes the difference between the subsequent numbers generated by taking the initial differences is constant. In general, when analyzing the C1Ln position for odd n (where n is a number like 1, 3, 5, 7, ...), I have found that the difference between consecutive numbers becomes constant after taking the difference n-1 times (appendix P). Furthermore, I have found that the constant difference arrived at for the C1Ln position is the middle entry of the nth row (i.e. the row corresponding to the value used for n) of Pascal’s Triangle.

If the same process is applied to the C2 and higher rows, one obtains similar results. But interestingly, the constants obtained from row Cn correspond to vertical slices of Pascal’s Triangle offset from the center by n-1 slices.

The tables of differences presented in appendix P each take the form shown in appendix I. In each case, the \( a_{0n} \) \( (n=0, 1, 2, \ldots) \) column is constant, and for some integer i, the \( a_{in} \) column contains the numbers from the appendix Q patterns for the position being analyzed. For example,
the C1L3 numbers are found in column $a_{2n}$. The differences between subsequent numbers in this column are located in column $a_{1n}$ and the constant differences between the subsequent numbers of the $a_{1n}$ column are found in the $a_{0n}$ column. In this case, everything to the left of column $a_{2n}$ is irrelevant. Because of the method used to create these pattern, it is clear that the entry $a_{ij}$, where $i$ and $j$ are positive integers, can be found by adding together the entries $a_{(i-1)j}$ and $a_{(i-1)(j-1)}$.

These patterns, which are shown in appendix P and obtained from data in appendix Q, look like they may lead to a formula for the number of non-self-intersecting paths that end at any point. By taking the sum of these values for every possible endpoint, one may derive a formula for the total number of non-self-intersecting paths. But two things must be done in order to meet this goal. First, assuming that the data all follow the described pattern, one must use this information to create and prove a formula that actually gives the desired values. Second, one must show that the data will actually continue to follow the pattern for all possible positions and all possible path lengths.

The first task is partly done. Assuming a pattern of the form in appendix I, with $a_{0n} = a_{00}$ for all integers $n$, I have created a formula for the total number of paths that end at any point:

$$a_{kn} = a_{k0} + a_{(k-1)0}n + a_{(k-2)0}n(n-1) + \ldots + a_{00}n(n-1)(n-2)\ldots(n-(k-1))$$

Here, $a_{kn}$ is the entry in a pattern described by appendix I, which corresponds with the number of non-self-intersecting paths of one particular length that end at some particular position. I am currently working on the proof of this formula. But unfortunately, this formula includes $k+1$ arbitrary variables ($a_{k0}$, $a_{(k-1)0}$, \ldots, $a_{00}$). Currently the values of these variables can only be determined by tracing through paths and counting non-self-intersecting paths. Another method of calculating the numbers $a_{20}$, $a_{(3-1)0}$, \ldots, $a_{10}$, $a_{00}$ must be determined in order to take full advantage
of this formula.

In order to try to determine a method for calculating the arbitrary variables without tracing through paths, I am currently using the data from program 5 (appendix N), which categorizes non-self-intersecting paths by the number of corners in them. I am hoping that this method of inquiry will also provide enough insight into the origin of this pattern that I will be able to prove that the pattern continues indefinitely.

CONCLUSION

Though I have made much progress in understanding non-self-intersecting paths through my study, this project is not near completion. Finding a simple formula is nowhere in sight, although the formula I have created may, with more information, reduce the problem to one that a computer will be able to handle for even large path lengths. And although much of the data collected have not contributed toward finding the desired formula, they have illuminated several new ways of viewing the problem which may be useful in the future. And some of them, like the number of simple loops of length n, would make interesting projects to study individually.

Although further research should continue along these lines, there are several other ways of looking at the problem that may prove useful. One may plot the number of self-intersecting paths of length n that end at each point on the grid. One could also count the endpoints or even the points of intersection of those paths that are non-self-intersecting at n-1 steps by intersect on the nth step. There are probably other ways of representing the paths that lend themselves to study in different manners. And if a simple formula were to be found, one could easily add complexity to the problem by adding one or more dimensions. The problem still has plenty of room for study.
Although a solution to this problem has not been found, this paper examines the techniques I used for collecting and analyzing data on non-self-intersecting paths. This information should provide a starting point for further investigation and illuminate the productive areas of research.
Appendix A

The number of non-self-intersecting (good) paths of length up to \( n = 20 \), generated by program 1 (see appendix J), along with the number of self-intersecting (bad) paths and the total number of paths.

<table>
<thead>
<tr>
<th>( n )</th>
<th>good</th>
<th>bad</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>28</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>156</td>
<td>256</td>
</tr>
<tr>
<td>5</td>
<td>284</td>
<td>740</td>
<td>1024</td>
</tr>
<tr>
<td>6</td>
<td>780</td>
<td>3316</td>
<td>4096</td>
</tr>
<tr>
<td>7</td>
<td>2172</td>
<td>14212</td>
<td>16384</td>
</tr>
<tr>
<td>8</td>
<td>5916</td>
<td>59620</td>
<td>65536</td>
</tr>
<tr>
<td>9</td>
<td>16268</td>
<td>245876</td>
<td>262144</td>
</tr>
<tr>
<td>10</td>
<td>44100</td>
<td>1004476</td>
<td>1048576</td>
</tr>
<tr>
<td>11</td>
<td>120292</td>
<td>4074012</td>
<td>4194304</td>
</tr>
<tr>
<td>12</td>
<td>324932</td>
<td>16452284</td>
<td>16777216</td>
</tr>
<tr>
<td>13</td>
<td>881500</td>
<td>66227364</td>
<td>67108864</td>
</tr>
<tr>
<td>14</td>
<td>2374444</td>
<td>266061012</td>
<td>268435456</td>
</tr>
<tr>
<td>15</td>
<td>6416596</td>
<td>1067325228</td>
<td>173741824</td>
</tr>
<tr>
<td>16</td>
<td>17245332</td>
<td>4277721964</td>
<td>4294967296</td>
</tr>
<tr>
<td>17</td>
<td>46466676</td>
<td>17133402508</td>
<td>17179869184</td>
</tr>
<tr>
<td>18</td>
<td>124658732</td>
<td>68594818004</td>
<td>68719476736</td>
</tr>
<tr>
<td>19</td>
<td>335116620</td>
<td>274542790324</td>
<td>274877906944</td>
</tr>
<tr>
<td>20</td>
<td>897697164</td>
<td>1098613930612</td>
<td>1099511627776</td>
</tr>
</tbody>
</table>
Appendix B

The number of loops contained in paths of lengths up to 10 steps. This information was adapted from output from program 2 (see appendix K).

1 steps

2 steps
total loops of length 2 is 4

3 steps
total loops of length 2 is 32

4 steps
total loops of length 2 is 192
total loops of length 4 is 36

5 steps
total loops of length 2 is 1024
total loops of length 4 is 288

6 steps
total loops of length 2 is 5120
total loops of length 4 is 1728
total loops of length 6 is 400

7 steps
total loops of length 2 is 24576
total loops of length 4 is 9216
total loops of length 6 is 3200

8 steps
total loops of length 2 is 114688
total loops of length 4 is 46080
total loops of length 6 is 19200
total loops of length 8 is 4900

9 steps
total loops of length 2 is 524288
total loops of length 4 is 221184
total loops of length 6 is 102400
total loops of length 8 is 39200

10 steps
total loops of length 2 is 2359296
Appendix B

total loops of length 4 is 1032192
total loops of length 6 is 512000
total loops of length 8 is 235200
total loops of length 10 is 63504
Appendix C

The number of loops contained in paths of lengths up to 10 steps, excluding loops that are contained within another loop. This information was adapted from output from program 3 (see appendix L)

<table>
<thead>
<tr>
<th>Steps</th>
<th>Total loops of length 2</th>
<th>Total loops of length 4</th>
<th>Total loops of length 6</th>
<th>Total loops of length 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>192</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
<td>384</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>24576</td>
<td>2048</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>114688</td>
<td>10240</td>
<td>1152</td>
<td>112</td>
</tr>
<tr>
<td>7</td>
<td>524288</td>
<td>49152</td>
<td>6144</td>
<td>896</td>
</tr>
<tr>
<td>8</td>
<td>10240</td>
<td>1152</td>
<td>6144</td>
<td>896</td>
</tr>
<tr>
<td>9</td>
<td>1152</td>
<td>6144</td>
<td>896</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C

10 steps
total loops of length 2 is 2359296
total loops of length 4 is 229376
total loops of length 6 is 30720
total loops of length 8 is 5376
total loops of length 10 is 560
Appendix D

The number of good paths, and the number of loops of each length in paths that were non-self-intersecting at n-1 steps, but self-intersect on the nth step. This information was adapted from output from program 4 (see appendix M)

1 steps
4 good paths

2 steps
12 good paths
total number of end loops of length 2 is 4

3 steps
36 good paths
total number of end loops of length 2 is 12

4 steps
100 good paths
total number of end loops of length 2 is 36
total number of end loops of length 4 is 8

5 steps
284 good paths
total number of end loops of length 2 is 100
total number of end loops of length 4 is 16
total number of end loops of length 6 is 24

6 steps
780 good paths
total number of end loops of length 2 is 284
total number of end loops of length 4 is 48
total number of end loops of length 6 is 120

7 steps
2172 good paths
total number of end loops of length 2 is 780
total number of end loops of length 4 is 128
total number of end loops of length 6 is 40

8 steps
5916 good paths
total number of end loops of length 2 is 2172
total number of end loops of length 4 is 368
total number of end loops of length 6 is 120
Appendix D

total number of end loops of length 8 is 112

9 steps
16268 good paths
total number of end loops of length 2 is 5916
total number of end loops of length 4 is 992
total number of end loops of length 6 is 312
total number of end loops of length 8 is 176

10 steps
44100 good paths
total number of end loops of length 2 is 16268
total number of end loops of length 4 is 2768
total number of end loops of length 6 is 888
total number of end loops of length 8 is 488
total number of end loops of length 10 is 560

11 steps
120292 good paths
total number of end loops of length 2 is 44100
total number of end loops of length 4 is 7472
total number of end loops of length 6 is 2392
total number of end loops of length 8 is 1272
total number of end loops of length 10 is 872

12 steps
324932 good paths
total number of end loops of length 2 is 120292
total number of end loops of length 4 is 20528
total number of end loops of length 6 is 6616
total number of end loops of length 8 is 3576
total number of end loops of length 10 is 2248

13 steps
881500 good paths
total number of end loops of length 2 is 324932
total number of end loops of length 4 is 55392
total number of end loops of length 6 is 17848
total number of end loops of length 8 is 9624
total number of end loops of length 10 is 5776
total number of end loops of length 12 is 4656
Appendix D

14 steps
2374444 good paths
total number of end loops of length 2 is 881500
total number of end loops of length 4 is 150784
total number of end loops of length 6 is 48760
total number of end loops of length 8 is 26488
total number of end loops of length 10 is 16144
total number of end loops of length 12 is 11416
total number of end loops of length 14 is 16464

15 steps
6416596 good paths
total number of end loops of length 2 is 2374444
total number of end loops of length 4 is 406256
total number of end loops of length 6 is 131448
total number of end loops of length 8 is 71272
total number of end loops of length 10 is 43280
total number of end loops of length 12 is 28568
total number of end loops of length 14 is 25912

16 steps
17245332 good paths
total number of end loops of length 2 is 6416596
total number of end loops of length 4 is 1099808
total number of end loops of length 6 is 356600
total number of end loops of length 8 is 194280
total number of end loops of length 10 is 118752
total number of end loops of length 12 is 79280
total number of end loops of length 14 is 61720
total number of end loops of length 16 is 94016
Appendix E.

The number of non-self-intersecting paths that end at position CIL3 (see appendix H), grouped by the number of corners in each path. This information was adapted from output from program 5 (see appendix N)

1 steps
0 total good paths

2 steps
0 total good paths

3 steps
Total paths with 2 turns is 2
2 total good paths

4 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 4
6 total good paths

5 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 8
Total paths with 4 turns is 2
12 total good paths

6 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 12
Total paths with 4 turns is 6
20 total good paths

7 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 16
Total paths with 4 turns is 12
30 total good paths

8 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 20
Total paths with 4 turns is 20
42 total good paths
Appendix E

9 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 24
Total paths with 4 turns is 30
56 total good paths

10 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 28
Total paths with 4 turns is 42
72 total good paths
Appendix F.

The number of non-self-intersecting paths that end at position C1L5 (see appendix H), grouped by the number of corners in each path. This information was adapted from output from program 5 (see appendix N)

1 steps
0 total good paths

2 steps
0 total good paths

3 steps
0 total good paths

4 steps
0 total good paths

5 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 4
6 total good paths

6 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 8
Total paths with 4 turns is 10
20 total good paths

7 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 12
Total paths with 4 turns is 22
Total paths with 5 turns is 12
Total paths with 6 turns is 6
54 total good paths

8 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 16
Total paths with 4 turns is 36
Total paths with 5 turns is 36
Total paths with 6 turns is 24
Total paths with 7 turns is 12
Appendix F

126 total good paths

9 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 20
Total paths with 4 turns is 52
Total paths with 5 turns is 72
Total paths with 6 turns is 60
Total paths with 7 turns is 48
Total paths with 8 turns is 6
260 total good paths

10 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 24
Total paths with 4 turns is 70
Total paths with 5 turns is 120
Total paths with 6 turns is 120
Total paths with 7 turns is 120
Total paths with 8 turns is 30
486 total good paths

11 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 28
Total paths with 4 turns is 90
Total paths with 5 turns is 180
Total paths with 6 turns is 210
Total paths with 7 turns is 240
Total paths with 8 turns is 90
840 total good paths

12 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 32
Total paths with 4 turns is 112
Total paths with 5 turns is 252
Total paths with 6 turns is 336
Total paths with 7 turns is 420
Total paths with 8 turns is 210
1364 total good paths
Appendix G.

The number of non-self-intersecting paths that end at position C1L7 (see appendix H), grouped by the number of corners in each path. This information was adapted from output from program 5 (see appendix N)

1 steps
0 total good paths

2 steps
0 total good paths

3 steps
0 total good paths

4 steps
0 total good paths

5 steps
0 total good paths

6 steps
0 total good paths

7 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 8
Total paths with 4 turns is 10
Total paths with 5 turns is 8
28 total good paths

8 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 12
Total paths with 4 turns is 30
Total paths with 5 turns is 32
Total paths with 6 turns is 12
Total paths with 7 turns is 4
92 total good paths

9 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 16
Total paths with 4 turns is 52
Total paths with 5 turns is 80
Appendix G

Total paths with 6 turns is 74
Total paths with 7 turns is 28
Total paths with 8 turns is 4
256 total good paths

10 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 20
Total paths with 4 turns is 76
Total paths with 5 turns is 152
Total paths with 6 turns is 196
Total paths with 7 turns is 140
Total paths with 8 turns is 68
654 total good paths

11 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 24
Total paths with 4 turns is 102
Total paths with 5 turns is 248
Total paths with 6 turns is 390
Total paths with 7 turns is 404
Total paths with 8 turns is 282
Total paths with 9 turns is 80
Total paths with 10 turns is 20
1552 total good paths

12 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 28
Total paths with 4 turns is 130
Total paths with 5 turns is 368
Total paths with 6 turns is 668
Total paths with 7 turns is 884
Total paths with 8 turns is 768
Total paths with 9 turns is 400
Total paths with 10 turns is 140
Total paths with 11 turns is 40
3428 total good paths

13 steps
Total paths with 2 turns is 2
Total paths with 3 turns is 32
Appendix G

Total paths with 4 turns is 160
Total paths with 5 turns is 512
Total paths with 6 turns is 1042
Total paths with 7 turns is 1644
Total paths with 8 turns is 1680
Total paths with 9 turns is 1200
Total paths with 10 turns is 540
Total paths with 11 turns is 240
Total paths with 12 turns is 20
7072 total good paths
Appendix H

Notation used for describing patterns generated by program 1.

C1L1
  C2L1 C1L2 C2L1
  C3L1 C2L2 C1L3 C2L2 C3L1
  ... C3L2 C2L3 C1L4 C2L3 C3L2 ... 
  C3L1 C3L2 ... C2L4 C1L5 C2L4 ... C3L2 C3L1
  C2L1 C2L2 C2L3 C2L4 ... C1L6 ... C2L4 C2L3 C2L2 C2L1
  C1L1 C1L2 C1L3 C1L4 C1L5 C1L6 ... C2L6 C1L5 C1L4 C1L3 C1L2 C1L1
  C2L1 C2L2 C2L3 C2L4 ... C1L6 ... C2L4 C2L3 C2L2 C2L1
  C3L1 C3L2 ... C2L4 C1L5 C2L4 ... C3L2 C3L1
  ... C3L2 C2L3 C1L4 C2L3 C3L2 ... 
  C3L1 C2L2 C1L3 C2L2 C3L1
  C2L1 C1L2 C2L1
  C1L1
Appendix I

Notation used for describing the difference patterns (appendix P)

\[
\begin{array}{cccccccccccc}
\ldots & a_{90} & a_{80} & a_{70} & a_{60} & a_{50} & a_{40} & a_{30} & a_{20} & a_{10} & a_{00} \\
\ldots & a_{91} & a_{81} & a_{71} & a_{61} & a_{51} & a_{41} & a_{31} & a_{21} & a_{11} & a_{01} \\
\ldots & a_{92} & a_{82} & a_{72} & a_{62} & a_{52} & a_{42} & a_{32} & a_{22} & a_{12} & a_{02} \\
\ldots & a_{93} & a_{83} & a_{73} & a_{63} & a_{53} & a_{43} & a_{33} & a_{23} & a_{13} & a_{03} \\
\ldots & a_{94} & a_{84} & a_{74} & a_{64} & a_{54} & a_{44} & a_{34} & a_{24} & a_{14} & a_{04} \\
\ldots & a_{95} & a_{85} & a_{75} & a_{65} & a_{55} & a_{45} & a_{35} & \ldots & \ldots & \ldots & \ldots \\
\end{array}
\]
Appendix J

Program 1. This C++ program, contained in three files, recursively traces paths using a two dimensional array to represent a two dimensional square grid. The number of paths that end at each point is recorded, generating a pattern for each path length.

```cpp
/*
 * File: path.h
 * Purpose: Header file for Path class.
 */

class Path {
    public:
        void display(int steps);
    protected:
    private:
        int goQ;
        void goNorthQ;
        void goSouth;
        void goEast;
        void goWest();
        void initialize(int steps);

        int other_than_north;
        int hits;
        unsigned int good;
        unsigned int bad;
        int x;
        int y;
        int N;
        int n;

        unsigned int map[101][101];
        unsigned int result[101][101];
};
```
Appendix J

/*
File: path.cc
Purpose: Implementation for the Path class
*/

#include "path.h"
#include <iostream.h>

void Path::goNorth() {
//cout << "goNorth" << endl;
y++;
n++;
if (go) {
    map[x][y]++;
goWest();
goNorth();
    if (other_than_north)
        goEast();
}

    map[x][y]--;
}
n--;
y--;
}

void Path::goSouth() {
//cout << "goSouth" << endl;
y--;
n++;
if (go) {
    map[x][y]++;

goEast();
goSouth();
goWest();
    map[x][y]--;
}
n--;
y++;
}
Appendix J

void Path::goEast() {
//cout << "goEast" << endl;
    x++;  
    n++;  
    if ( go() ) {  
        map[x][y]++;  
        goNorth();  
        goEast();  
        goSouth();  
        map[x][y]--;  
    }  
    n--;  
    x--;  
}

void Path::goWest() {
//cout << "goWest" << endl;
    other_than_north++;  
    x--;  
    n++;  
    if ( go() ) {  
        map[x][y]++;  
        goSouth();  
        goWest();  
        goNorth();  
        map[x][y]--;  
    }  
    n--;  
    x++;  
    other_than_north--;  
}

int Path::go() {  
//cout << "go" << endl;  
//cout << "n " << n << " N " << N << endl;  
    if( map[x][y] > 0 ) {  
        return 0;  
    }  
    if( n == N ) {
Appendix J

good++;

result[x][y]++;
result[2*N-x][y]++;
result[x][2*N-y]++;
result[2*N-x][2*N-y]++;

result[y][x]++;
result[2*N-y][x]++;
result[y][2*N-x]++;
result[2*N-y][2*N-x]++;

return 0;
}
return 1;
}

void Path::display( int steps) {
//cout << "display" << endl;
if ( steps< 40) {
 initialize(steps);
 other_than_north=0;
 N = steps;
 x = steps;
 y = steps;

 map[x][y]=1;
 goNorth();

cout << N << "\t" << good*8-4 << endl;
for (int i=0; i<steps*2+1; i++) {
 for (int j=0; j<steps*2+1; j++) {
 if ( j==0 || i==steps || j==2*steps && i==steps ||
 j==steps && i==0 || j==steps && i==2*steps ) {

cout << result[i][j]-1;
 }
 else {
 cout << result[i][j];
 }

cout << "\t";
 }
}
Appendix J

```cpp
void Path::initialize(int maxSteps) {
    //cout << "initialize" << endl;
    n = 0;
    hits = 0;
    good = 0;
    bad = 0;

    for (int c = 0; c < 101; c++) {
        for (int b = 0; b < 101; b++) {
            map[c][b] = 0;
            result[c][b] = 0;
        }
    }
}
```
Appendix J

/*
  File: test.cc
  Purpose: This file actually uses the Path class
*/

#include "path.h"

main() {
    Path x;
    for( int i=1; i<30; i++ ) {
        x.display(i);
    }
}
Program 2. This program counts all of the loops categorized by length found in all of the paths of any given length.

#include <iostream.h>

char stack[50];
int top = -1;

char analyzer[50][4];
int data[25];

analyzeStack() {
    for (int x=0; x<=top; x++) {
        for (int y=0; y<4; y++) {
            analyzer[x][y]=0;
        }
    }
    for (int x=0; x<=top; x++) {
        switch( stack[x] ) {
            case 'N':
                for (int y=0; y<=x; y++) {
                    analyzer[y][0]++;
                }
                break;
            case 'E':
                for (int y=0; y<=x; y++) {
                    analyzer[y][1]++;
                }
                break;
            case 'S':
                for (int y=0; y<=x; y++) {
                    analyzer[y][2]++;
                }
                break;
            case 'W':
                for (int y=0; y<=x; y++) {
                    analyzer[y][3]++;
                }
                break;
        }
        for (int y=0; y<=x; y++) {
            if ( analyzer[y][0] == analyzer[y][2] && analyzer[y][1] == analyzer[y][3] ) {
                // cout << "loop of size " << x-y+1 << endl;
            }
        }
    }
}
Appendix K

```cpp
    data[(x-y+1)/(2-1)]++;
}
}
}
}
}

displayStack() {
    if (top<0) cout << "warningDisplay" << endl;
    for (int x=0; x<=top; x++) {
        // cout << \n << stack[x];
    }
    // cout << endl;
    analyzeStack();
}

push( char x ) {
    top++;
    stack[top] = x;
}

char pop() {
    if (top<0) {
        // cout << "warningPop" << endl;
        top=-1;
        return 'e';
    }
    top--;
    return stack[top+1];
}

void go( int steps);

void goNorth( int steps ) {
    push('N');
    go(steps-1);
}

void goEast( int steps ) {
    push('E');
    go(steps-1);
}

void goSouth( int steps ) {
```
Appendix K

```c
void goWest( int steps ){
    push('W');
    go(steps-1);
}

void go( int steps ){
    if ( steps > 0 ) {
        goNorth( steps);
        goEast( steps);
        goSouth( steps);
        goWest( steps);
    } else {
        displayStack();
    }
    pop();
}

main() {
    for (int x=1; x<=10; x++) {
        for (int y=0; y<x/2; y++) {
            data[y]=0;
        }
        go(x);
        cout << x << " steps" << endl;
    }
    cout << "total loops of length " << 2*(y+1) << " is " << data[y] << endl;
    }
}
```
Program 3. This program counts all of the loops categorized by length except for loops contained in other loops.

```c
#include <iostream.h>

char stack[50];
int top = -1;

char analyzer[50][4];
int data[25];

analyzeStack() {
    for (int x=0; x<=top; x++) {
        for (int y=0; y<4; y++) {
            analyzer[x][y]=0;
        }
    }
    int earliestGood=0;
    for (int x=0; x<=top; x++) {
        switch( stack[x] ) {
            case 'N':
                for (int y=0; y<=x; y++) {
                    analyzer[y][0]++;
                }
                break;
            case 'E':
                for(int y=0; y<=x; y++){
                    analyzer[y][1]++;
                }
                break;
            case 'S':
                for (int y=0; y<=x; y++) {
                    analyzer[y][2]++;
                }
                break;
            case 'W':
                for (int y=0; y<=x; y++) {
                    analyzer[y][3]++;
                }
                break;
        }
    }
    for (int y=0; y<=x; y++) {
        if ( analyzer[y][0] == analyzer[y][2] && analyzer[y][1] == analyzer[y][3] ) {
```
Appendix L

```cpp
if (y+1 > earliestGood) {
    cout << "loop of size " << x-y+1 << endl;
    data[ (x-y+1) / 2 - 1 ]++;
    if (y+1 > earliestGood) {
        earliestGood = y+1;
    }
}
}
}
}

displayStack() {
    if (top<0) cout << "warningDisplay" << endl;
    for( int x=0; x<=top; x++ ) {
        cout << stack[x];
    }
    cout << endl;
    analyzeStack();
}

push( char x ) {
    top++;
    stack[top] = x;
}

char pop() {
    if (top<0) {
        // cout << "warningPop" << endl;
        top--;
        return 'e';
    }
    top--;
    return stack[top+1];
}

void go( int steps );

void goNorth( int steps ) {
    push('N');
    go(steps-1);
}
```
Appendix L

void goEast( int steps ) {
    push('E');
    go(steps-1);
}

void goSouth( int steps ) {
    push('S');
    go(steps-1);
}

void goWest( int steps ) {
    push('W');
    go(steps-1);
}

void go( int steps ) {
    if ( steps > 0 ) {
        goNorth( steps );
        goEast( steps );
        goSouth( steps );
        goWest( steps );
    } else {
        displayStack();
    }
    pop();
}

main () {
    for (int x=1; x<=6; x++) {
        for (int y=0; y<x/2; y++) {
            data[y]=0;
        }
        go(x);
        cout << x << " steps" << endl;
        for (int y=0; y<x/2; y++) {
            cout << "total loops of length " << 2*(y+1) << " is " << data[y] << endl;
        }
    }
}
Appendix M

Program 4. This program counts the number of loops, categorized by length, in paths that are good at length n-1 but self intersect on the nth step

```c
#include <iostream.h>

char stack[50];
int good = 0;
int top = -1;

char analyzer[50][4];
int data[25];

analyzeStack() {
    int loop = 0;
    for (int x = 0; x <= top; x++) {
        for (int y = 0; y < 4; y++) {
            analyzer[x][y] = 0;
        }
    }
    for (int x = 0; x <= top; x++) {
        switch (stack[x]) {
            case 'N':
                for (int y = 0; y <= x; y++) {
                    analyzer[y][0]++;
                }
                break;
            case 'E':
                for (int y = 0; y <= x; y++) {
                    analyzer[y][1]++;
                }
                break;
            case 'S':
                for (int y = 0; y <= x; y++) {
                    analyzer[y][2]++;
                }
                break;
            case 'W':
                for (int y = 0; y <= x; y++) {
                    analyzer[y][3]++;
                }
                break;
        }
    }
    for (int y = 0; y <= x; y++) {
```
Appendix M

if ( analyzer[y][0] == analyzer[y][2] && analyzer[y][1] == analyzer[y][3] ) {
    // cout << "loop of size " << x-y+1 << endl;
    loop++;
    if(x==top && loop==1) {
        data[ (x-y+1) / 2 - 1 ]++;
    }
}
}
if (loop==0) {
    good++;
}
}

displayStack() {
    if (top<0) cout << "warningDisplay" << endl;
    for( int x=0; x<=top; x++ ) {
        // cout << ',' << stack[x];
    }
    // cout << endl;
    analyzeStack();
}

push( char x ) {
    top++;
    stack[top] = x;
}

char pop() {
    if (top<0) {
        // cout << "warningPop" << endl;
        top=-1;
        return 'e';
    }
    top--;
    return stack[top+1];
}

void go( int steps );
void goNorth( int steps ) {
    push('N');
    go(steps-1);
Appendix M

```c
void goEast( int steps ) {
    push('E');
    go(steps-1);
}

void goSouth( int steps ) {
    push('S');
    go(steps-1);
}

void goWest( int steps ) {
    push('W');
    go(steps-1);
}

void go( int steps ) {
    if ( steps > 0 ) {
        goNorth( steps );
        goEast( steps );
        goSouth( steps );
        goWest( steps );
    } else {
        displayStackQ();
        popQ();
    }
}

main() {
    for (int x=1; x<=20; x++) {
        good=0;
        for (int y=0; y<x/2; y++) {
            data[y]=0;
        }
        go(x);
        cout << x << " steps" << endl;
        cout << good << " good paths" << endl;
        for (int y=0; y<x/2; y++) {
            cout << "total number of end loops of length " << 2*(y+1) << " is " << data[y] << endl;
        }
    }
}
```
Appendix N

Program 5. This program counts the number of paths, categorized by the number of corners, from the origin to a specified point, with a specified length.

#include <iostream.h>

int left;
int up;
int max;

char stack[50];
int top = -1;

char analyzer[50][4];
int good=0;

int num_turns[50];

int analyzeStack() {
    int loop=0;
    for (int x=0; x<=top; x++) {
        for (int y=0; y<4; y++) {
            analyzer[x][y]=0;
        }
    }
    for (int x=0; x<=top; x++) {
        switch( stack[x] ) {
            case 'N':
                for (int y=0; y<=x; y++) {
                    analyzer[y][0]++;
                }
                break;
            case 'E':
                for (int y=0; y<=x; y++) {
                    analyzer[y][1]++;
                }
                break;
            case 'S':
                for (int y=0; y<=x; y++) {
                    analyzer[y][2]++;
                }
                break;
            case 'W':
                for (int y=0; y<=x; y++) {
                    analyzer[y][3]++;
                }
        }
    }
    return loop;
}
Appendix N

break;
}
for (int y=0; y<=x; y++) {
if ( analyzer[y][0] == analyzer[y][2] && analyzer[y][1] == analyzer[y][3] ) {
    loop++;
}
}
if( loop==0 ) {
    if( left <= top+1 ) {
        if((analyzer[0][3]-analyzer[0][1]==top+1-left)&&(analyzer[0][0]-analyzer[0][2]==up)) {
            good++;  
            int turns=0; 
            for( int y=0; y<top; y++ ) {  
                if( stack[y] != stack[y+1] ) {  
                    turns++;  
                }
            }
            cout << "good path with " << turns << " turns ";
            num_turns[turns]++;
            return 1;
        }
    }
}
return 0;
}

displayStack() {
    if( analyzeStack() ) {
        for( int x=0; x<=top; x++ ) {
            cout << stack[x];
        }
        cout << endl;
    }
}
push( char x ) {
    top++;  
    stack[top] = x;
}
char pop() {
    if( top<0 ) {
        // cout << "warningPop" << endl;
        top=-1;
        return 'e';
    }
Appendix N

}  
  top--;  
  return stack[top+1];
}

void go(int steps);

void goNorth(int steps) {
  push('N');
  go(steps-1);
}

void goEast(int steps) {
  push('E');
  go(steps-1);
}

void goSouth(int steps) {
  push('S');
  go(steps-1);
}

void goWest(int steps) {
  push('W');
  go(steps-1);
}

void go(int steps) {
  if (seps > 0) {
    goNorth(steps);
    goEast(steps);
    goSouth(steps);
    goWest(steps);
  } else {
    displayStackQ;
  }
  pop();
}

main() {
  cout << "What point do you want to examine?" << endl  
    << "How many positions in from the left: ";
  cin >> left;
  cout << "How many positions up from the middle: ";
  }
Appendix N

    cin >> up;
    cout << "What maximum path length do you want: ";
    cin >> max;
    cout << "Position (" << left <<", " << up <<")"<< endl;

    for (int x=1; x<=max; x++) {
        for (int y=0; y<50; y++) {
            num_turns[y]=0;
        }
        good=0;
        cout << x << " steps" << endl;
        go(x);
        for (int y=0; y<50; y++) {
            if (num_turns[y] != 0) {
                cout << "Total paths with " << y << " turns is " << num_turns[y] << endl;
            }
        }
        cout << good << " total good paths" << endl;
    }
}
Appendix O

Pascal’s Triangle

<table>
<thead>
<tr>
<th>vertical slice #</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 2</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 3</td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 4</td>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 5</td>
<td></td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 6</td>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 7</td>
<td></td>
<td>1</td>
<td>6</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>row 8</td>
<td></td>
<td>1</td>
<td>7</td>
<td>21</td>
<td>35</td>
<td>35</td>
<td>21</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>row 9</td>
<td></td>
<td>1</td>
<td>8</td>
<td>28</td>
<td>56</td>
<td>70</td>
<td>56</td>
<td>28</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>row 10</td>
<td></td>
<td>1</td>
<td>9</td>
<td>36</td>
<td>84</td>
<td>126</td>
<td>126</td>
<td>84</td>
<td>36</td>
<td>8</td>
</tr>
</tbody>
</table>
Difference patterns for sequences of numbers from selected positions

<table>
<thead>
<tr>
<th>C1L1</th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1L3</th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
</tr>
<tr>
<td>42</td>
<td>14</td>
</tr>
<tr>
<td>56</td>
<td>16</td>
</tr>
<tr>
<td>72</td>
<td>18</td>
</tr>
<tr>
<td>90</td>
<td>20</td>
</tr>
<tr>
<td>110</td>
<td>22</td>
</tr>
<tr>
<td>132</td>
<td>24</td>
</tr>
<tr>
<td>156</td>
<td>26</td>
</tr>
<tr>
<td>182</td>
<td>28</td>
</tr>
<tr>
<td>210</td>
<td>30</td>
</tr>
<tr>
<td>240</td>
<td>32</td>
</tr>
<tr>
<td>272</td>
<td>34</td>
</tr>
<tr>
<td>306</td>
<td>36</td>
</tr>
<tr>
<td>342</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix P

**C1L5**

<table>
<thead>
<tr>
<th></th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>34</td>
</tr>
<tr>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>126</td>
<td>134</td>
</tr>
<tr>
<td>260</td>
<td>226</td>
</tr>
<tr>
<td>486</td>
<td>354</td>
</tr>
<tr>
<td>840</td>
<td>524</td>
</tr>
<tr>
<td>1364</td>
<td>742</td>
</tr>
<tr>
<td>2106</td>
<td>1014</td>
</tr>
<tr>
<td>3120</td>
<td>1346</td>
</tr>
<tr>
<td>4466</td>
<td>1744</td>
</tr>
<tr>
<td>6210</td>
<td>2214</td>
</tr>
<tr>
<td>8424</td>
<td>2762</td>
</tr>
<tr>
<td>11186</td>
<td>3394</td>
</tr>
<tr>
<td>14580</td>
<td>4116</td>
</tr>
<tr>
<td>18696</td>
<td></td>
</tr>
</tbody>
</table>

**C1L7**

<table>
<thead>
<tr>
<th></th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>64</td>
</tr>
<tr>
<td>92</td>
<td>164</td>
</tr>
<tr>
<td>256</td>
<td>398</td>
</tr>
<tr>
<td>654</td>
<td>898</td>
</tr>
<tr>
<td>1552</td>
<td>1876</td>
</tr>
<tr>
<td>3428</td>
<td>3644</td>
</tr>
<tr>
<td>7072</td>
<td>6634</td>
</tr>
<tr>
<td>13706</td>
<td>11418</td>
</tr>
<tr>
<td>25124</td>
<td>18728</td>
</tr>
<tr>
<td>43852</td>
<td>29476</td>
</tr>
<tr>
<td>73328</td>
<td>44774</td>
</tr>
<tr>
<td>118102</td>
<td>65954</td>
</tr>
<tr>
<td>184056</td>
<td>94588</td>
</tr>
<tr>
<td>278644</td>
<td></td>
</tr>
</tbody>
</table>

**C1L9**

<table>
<thead>
<tr>
<th></th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>328</td>
</tr>
<tr>
<td>468</td>
<td>872</td>
</tr>
<tr>
<td>1340</td>
<td>2218</td>
</tr>
<tr>
<td>3558</td>
<td>5366</td>
</tr>
<tr>
<td>8924</td>
<td>12336</td>
</tr>
<tr>
<td>21260</td>
<td>26928</td>
</tr>
<tr>
<td>48188</td>
<td>55832</td>
</tr>
<tr>
<td>104020</td>
<td>110158</td>
</tr>
<tr>
<td>214178</td>
<td>207456</td>
</tr>
<tr>
<td>421634</td>
<td>374296</td>
</tr>
<tr>
<td>795930</td>
<td>649478</td>
</tr>
<tr>
<td>1445408</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix P

#### C2L1 differences

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### C2L3 differences

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td>26</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>52</td>
<td>40</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>92</td>
<td>57</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>149</td>
<td>77</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>226</td>
<td>100</td>
<td>26</td>
<td>3</td>
</tr>
<tr>
<td>326</td>
<td>126</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>452</td>
<td>155</td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td>607</td>
<td>187</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>794</td>
<td>222</td>
<td>38</td>
<td>3</td>
</tr>
<tr>
<td>1016</td>
<td>260</td>
<td>41</td>
<td>3</td>
</tr>
<tr>
<td>1276</td>
<td>301</td>
<td>44</td>
<td>3</td>
</tr>
<tr>
<td>1577</td>
<td>345</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>1922</td>
<td>392</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>2314</td>
<td>442</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2756</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Appendix P

<table>
<thead>
<tr>
<th>C2L5</th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>29</td>
</tr>
<tr>
<td>45</td>
<td>73</td>
</tr>
<tr>
<td>118</td>
<td>168</td>
</tr>
<tr>
<td>286</td>
<td>350</td>
</tr>
<tr>
<td>636</td>
<td>665</td>
</tr>
<tr>
<td>1301</td>
<td>1169</td>
</tr>
<tr>
<td>2470</td>
<td>1928</td>
</tr>
<tr>
<td>4398</td>
<td>3018</td>
</tr>
<tr>
<td>7416</td>
<td>4525</td>
</tr>
<tr>
<td>11941</td>
<td>6545</td>
</tr>
<tr>
<td>18486</td>
<td>9184</td>
</tr>
<tr>
<td>27670</td>
<td>12558</td>
</tr>
<tr>
<td>40228</td>
<td>16793</td>
</tr>
<tr>
<td>57021</td>
<td>22025</td>
</tr>
<tr>
<td>79046</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C2L7</th>
<th>differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>144</td>
</tr>
<tr>
<td>220</td>
<td>378</td>
</tr>
<tr>
<td>598</td>
<td>934</td>
</tr>
<tr>
<td>1532</td>
<td>2180</td>
</tr>
<tr>
<td>3712</td>
<td>4794</td>
</tr>
<tr>
<td>8506</td>
<td>9924</td>
</tr>
<tr>
<td>18430</td>
<td>19383</td>
</tr>
<tr>
<td>37813</td>
<td>35879</td>
</tr>
<tr>
<td>73692</td>
<td>63280</td>
</tr>
<tr>
<td>136972</td>
<td>106914</td>
</tr>
<tr>
<td>243886</td>
<td>173904</td>
</tr>
<tr>
<td>417790</td>
<td>273538</td>
</tr>
<tr>
<td>691328</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix P

#### C3L1 Differences

<table>
<thead>
<tr>
<th>Value</th>
<th>Difference 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>17</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>18</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>190</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### C3L3 Differences

<table>
<thead>
<tr>
<th>Value</th>
<th>Difference 1</th>
<th>Difference 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>31</td>
<td>32</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>55</td>
<td>63</td>
<td>50</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>118</td>
<td>113</td>
<td>72</td>
<td>26</td>
<td>4</td>
</tr>
<tr>
<td>231</td>
<td>185</td>
<td>98</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>416</td>
<td>283</td>
<td>128</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>699</td>
<td>411</td>
<td>162</td>
<td>38</td>
<td>4</td>
</tr>
<tr>
<td>1110</td>
<td>573</td>
<td>200</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>1683</td>
<td>773</td>
<td>242</td>
<td>46</td>
<td>4</td>
</tr>
<tr>
<td>2456</td>
<td>1015</td>
<td>288</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>3471</td>
<td>1303</td>
<td>338</td>
<td>54</td>
<td>4</td>
</tr>
<tr>
<td>4774</td>
<td>1641</td>
<td>392</td>
<td>58</td>
<td>4</td>
</tr>
<tr>
<td>6415</td>
<td>2033</td>
<td>450</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>8448</td>
<td>2483</td>
<td>512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10931</td>
<td>2995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13926</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Appendix P

#### C4L1 Differences

<table>
<thead>
<tr>
<th>Value</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>15 6 1</td>
</tr>
<tr>
<td>35</td>
<td>21 7 1</td>
</tr>
<tr>
<td>56</td>
<td>28 8 1</td>
</tr>
<tr>
<td>84</td>
<td>36 9 1</td>
</tr>
<tr>
<td>120</td>
<td>45 10 1</td>
</tr>
<tr>
<td>165</td>
<td>55 11 1</td>
</tr>
<tr>
<td>220</td>
<td>66 12 1</td>
</tr>
<tr>
<td>286</td>
<td>78 13 1</td>
</tr>
<tr>
<td>364</td>
<td>91 14 1</td>
</tr>
<tr>
<td>455</td>
<td>105 15 1</td>
</tr>
<tr>
<td>560</td>
<td>120 16 1</td>
</tr>
<tr>
<td>680</td>
<td>136 17 1</td>
</tr>
<tr>
<td>816</td>
<td>153 18 1</td>
</tr>
<tr>
<td>969</td>
<td>171</td>
</tr>
<tr>
<td>1140</td>
<td></td>
</tr>
</tbody>
</table>

#### C4L3 Differences

<table>
<thead>
<tr>
<th>Value</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>139 134 87 32 5</td>
</tr>
<tr>
<td>259</td>
<td>273 221 119 37 5</td>
</tr>
<tr>
<td>532</td>
<td>494 340 156 42 5</td>
</tr>
<tr>
<td>1026</td>
<td>834 496 198 47 5</td>
</tr>
<tr>
<td>1860</td>
<td>1330 694 245 52 5</td>
</tr>
<tr>
<td>3190</td>
<td>2024 939 297 57 5</td>
</tr>
<tr>
<td>5214</td>
<td>2963 1236 354 62 5</td>
</tr>
<tr>
<td>8177</td>
<td>4199 1590 416 67 5</td>
</tr>
<tr>
<td>12376</td>
<td>5789 2006 483 72</td>
</tr>
<tr>
<td>18165</td>
<td>7795 2489 555</td>
</tr>
<tr>
<td>25960</td>
<td>10284 3044</td>
</tr>
<tr>
<td>36244</td>
<td>13328</td>
</tr>
<tr>
<td>49572</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix Q

#### n=1 pattern

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### n=2 pattern

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### n=3 pattern

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### n=4 pattern

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### n=5 pattern

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
n=6 pattern

0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 6 0 6 0 0 0 0 0 0
0 0 0 0 0 15 0 20 0 15 0 0 0 0 0
0 0 0 20 0 26 0 26 0 20 0 0 0 0 0
0 0 15 0 24 0 20 0 24 0 15 0 0 0 0
0 6 0 26 0 16 0 16 0 26 0 6 0 0 0
1 0 20 0 20 0 0 0 0 20 0 20 0 1
0 6 0 26 0 16 0 16 0 26 0 6 0 0 0
0 0 15 0 24 0 20 0 24 0 15 0 0 0 0
0 0 0 20 0 26 0 26 0 20 0 0 0 0 0
0 0 0 0 15 0 20 0 15 0 0 0 0 0 0
0 0 0 0 0 6 0 6 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
<table>
<thead>
<tr>
<th>n=7 pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 7 0 7 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 21 0 30 0 21 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 35 0 52 0 52 0 35 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 35 0 55 0 54 0 55 0 35 0 0 0 0</td>
</tr>
<tr>
<td>0 0 21 0 55 0 45 0 45 0 55 0 21 0 0 0</td>
</tr>
<tr>
<td>0 7 0 52 0 45 0 28 0 45 0 52 0 7 0 0</td>
</tr>
<tr>
<td>1 0 30 0 54 0 28 0 28 0 54 0 30 0 1 0</td>
</tr>
<tr>
<td>0 7 0 52 0 45 0 28 0 45 0 52 0 7 0 0</td>
</tr>
<tr>
<td>0 0 21 0 55 0 45 0 45 0 55 0 21 0 0 0</td>
</tr>
<tr>
<td>0 0 0 35 0 55 0 54 0 55 0 35 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 35 0 52 0 52 0 35 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 21 0 30 0 21 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 7 0 7 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Appendix Q
n=8 pattern

<p>| | | | | | | | | | | | | | | | | | | | | |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 28 | 0 | 42 | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 0 | 92 | 0 | 92 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 70 | 0 | 118 | 0 | 126 | 0 | 118 | 0 | 70 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 56 | 0 | 120 | 0 | 118 | 0 | 118 | 0 | 120 | 0 | 56 | 0 | 0 | 0 | 0 |
| 0 | 0 | 28 | 0 | 118 | 0 | 112 | 0 | 92 | 0 | 112 | 0 | 118 | 0 | 28 | 0 | 0 | 0 | 0 | 0 |
| 0 | 8 | 0 | 92 | 0 | 118 | 0 | 76 | 0 | 76 | 0 | 118 | 0 | 92 | 0 | 8 | 0 | 0 | 0 | 0 |
| 1 | 0 | 42 | 0 | 126 | 0 | 92 | 0 | 0 | 0 | 92 | 0 | 126 | 0 | 42 | 0 | 1 | 0 | 0 | 0 |
| 0 | 8 | 0 | 92 | 0 | 118 | 0 | 76 | 0 | 76 | 0 | 118 | 0 | 92 | 0 | 8 | 0 | 0 | 0 | 0 |
| 0 | 0 | 28 | 0 | 118 | 0 | 112 | 0 | 92 | 0 | 112 | 0 | 118 | 0 | 28 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 120 | 0 | 118 | 0 | 118 | 0 | 120 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 0 | 92 | 0 | 92 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 28 | 0 | 42 | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| n=9 pattern |     |     |     | 1   |     |     |     |     |     |     |     |     |     |     | 0   | 0   | 0   | 0   | 0   | 0   |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0           | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 9   | 0   | 9   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 0   | 36  | 0   | 56  | 0   | 36  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 0   | 84  | 0   | 149 | 0   | 149 | 0   | 84  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 126 | 0   | 231 | 0   | 260 | 0   | 231 | 0   | 126 | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 126 | 0   | 259 | 0   | 286 | 0   | 286 | 0   | 259 | 0   | 126 | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 84  | 0   | 259 | 0   | 277 | 0   | 256 | 0   | 277 | 0   | 259 | 0   | 84  | 0   | 0   | 0   | 0   | 0   |
| 0           | 9   | 0   | 149 | 0   | 286 | 0   | 220 | 0   | 140 | 0   | 220 | 0   | 286 | 0   | 149 | 0   | 9   | 0   | 0   | 0   | 0   |
| 1           | 0   | 56  | 0   | 260 | 0   | 256 | 0   | 140 | 0   | 140 | 0   | 256 | 0   | 260 | 0   | 56  | 0   | 1   |
| 0           | 9   | 0   | 149 | 0   | 286 | 0   | 220 | 0   | 140 | 0   | 220 | 0   | 286 | 0   | 149 | 0   | 9   | 0   | 0   | 0   |
| 0           | 0   | 36  | 0   | 231 | 0   | 277 | 0   | 220 | 0   | 220 | 0   | 277 | 0   | 231 | 0   | 36  | 0   | 9   |
| 0           | 0   | 0   | 84  | 0   | 259 | 0   | 277 | 0   | 256 | 0   | 277 | 0   | 259 | 0   | 84  | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 126 | 0   | 259 | 0   | 286 | 0   | 286 | 0   | 259 | 0   | 126 | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 126 | 0   | 231 | 0   | 260 | 0   | 231 | 0   | 126 | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 84  | 0   | 149 | 0   | 149 | 0   | 84  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 36  | 0   | 56  | 0   | 36  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 9   | 0   | 9   | 0   | 9   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0           | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |</p>
<table>
<thead>
<tr>
<th>n=10</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n=11</td>
<td>pattern</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 55 0 90 0 55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 165 0 326 0 326 0 165 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 330 0 699 0 840 0 699 0 330 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 462 0 1026 0 1301 0 1301 0 1026 0 462 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1176 0 1500 0 1552 0 1552 0 1176 0 462 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1546 0 1467 0 1532 0 1532 0 1546 0 1176 0 330 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1467 0 1157 0 1157 0 1467 0 1500 0 699 0 55 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1340 0 744 0 744 0 1340 0 1552 0 1301 0 326 0 11 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 744 0 1157 0 1157 0 744 0 1340 0 1532 0 1301 0 326 0 11 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1157 0 1340 0 1340 0 1157 0 1532 0 1301 0 326 0 11 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1467 0 1500 0 1552 0 1552 0 1467 0 1500 0 699 0 55 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1546 0 1546 0 1546 0 1546 0 1546 0 1500 0 699 0 55 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1552 0 1552 0 1552 0 1552 0 1552 0 1552 0 1552 0 1552 0 1552</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1026 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301 0 1301</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 840 0 699 0 699 0 330 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 326 0 326 0 326 0 165 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 55 0 90 0 55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 11 0 11 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Appendix Q
n=12 pattern

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Appendix Q