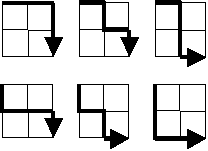
Problem 15: Lattice paths.

Suppose a 2 × 2 grid. Starting in the top left corner of the grid, and only being able to move to the right and down, there are exactly 6 routes to the bottom right corner.



How many such routes are there through a 20 × 20?

Solution.

Suppose a 0 × 0 grid exists as a point or a dot.

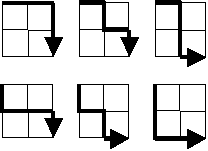


As unintuitive as it sounds, let’s assume that there is one path to the dot to itself.

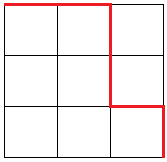
Now let’s look at a 1 × 1 grid, in other word a square. There can be two routes from the top left corner to the bottom right corner.

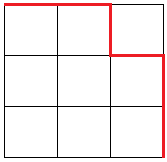
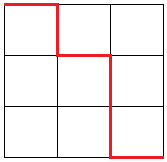


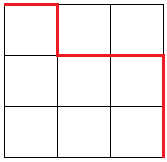
Now, as shown in the example above, a 2 × 2 grid will have 6 possible routes between the two corners.

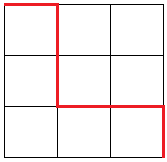
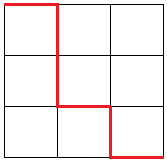
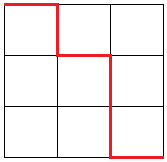
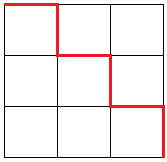
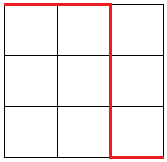


Let’s get into a 3 × 3 grid. First, here are 9 of the 20 possible routes.

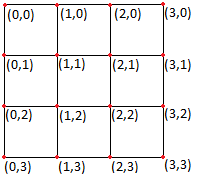




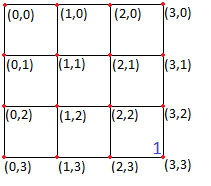




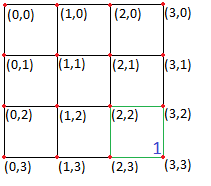
For the first demonstration, each intersection will correspond to a point (x, y) where x are the intersections on columns and y are the intersections on rows, going from 0 to n (see figure below).

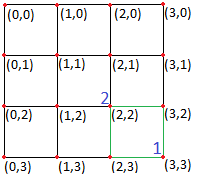


As mentioned for a 0 × 0 grid, a dot or a point will have one possible path. Therefore, we can mark the bottom right corner point (3, 3) as having one path to itself and we will mark this number of path on the grid to keep track (see figure below).

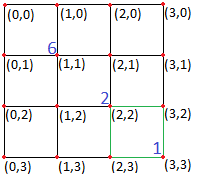


Now, let’s identify a 1 × 1 sub-grid highlighted green (see figure below on the left). By setting (2, 2) as the starting corner for the 1 × 1 highlighted sub-grid and (3, 3) as our target destination, two different possible routes can be identified, as shown previously for a 1 × 1 grid. We will mark (2, 2) with a **2** (see figure below on the right).





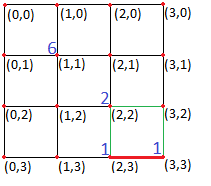
Similarly, we can mark (1, 1) as the starting point of a 2 × 2 sub-grid. As shown in the example presented in the problem, the 2 × 2 sub-grid contains 6 different paths (see figure below).



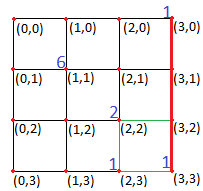
Since we have yet to solve a 3 × 3 grid, the value at point (0, 0) is still unknown. However, there are trivial cases in any n × n grid. Those are the points along the sides of the last row (y = 3) and of the last column (x = 3). In mathematical notation,

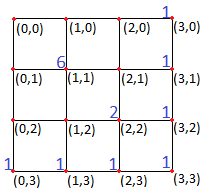
.

For example, let’s place our position at (2, 3). How many paths are there from (2, 3) to (3, 3)? Remember that we can only go right or down. So, from (2, 3), we arrive at the target (3, 3). Necessarily, there is only one path (see red line in figure below) and we mark the value 1 at (2, 3).



The same goes for every edge point defined in the set above. Let’s do another example and apply implication, this time with the point (3, 0) (top right corner). Illustrated below on the left, a red line highlights the unique path from (3, 0) to target (3, 3), meaning (3, 0) is marked with a value of 1. Now, if we move to (3, 1), it is the same path but 1 row below. The same reasoning is applied to (3, 2), (0, 3) and (1, 3) (see figure on the right below).



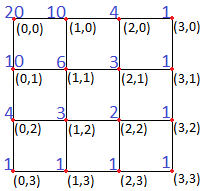


From the grid on the right above, simple arithmetic is applied to deduce the remaining values. For example, from point (1, 2), going right will land on (2, 2) which has 2 possible paths. If going down to (1, 3), there is only one possible path.

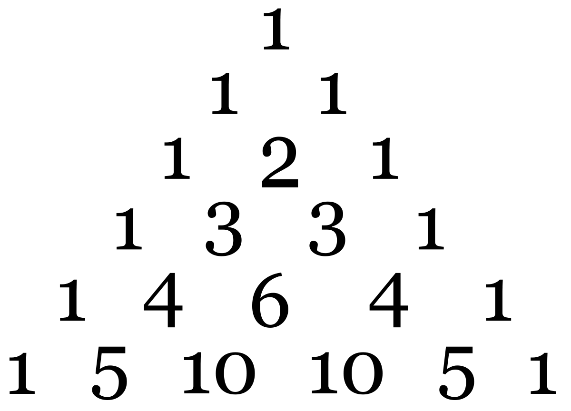
By adding the number of possible paths marked on the points from going right or down, a value is obtained and that value represents the number of possible paths from the initial position.

Therefore, for (1, 2), going right to (2, 2) gives 2 routes and going down to (1, 3) gives one route. By adding the values, there is (2 + 1) possible routes from (1, 2). By applying this arithmetic algorithm, we can complete any n × n grid.

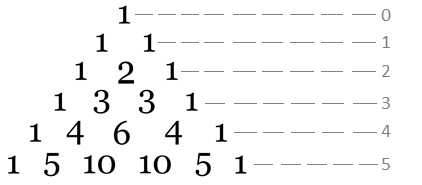
Here is the fully marked 3 × 3 grid below. To answer the problem above adapted to a 3 × 3 grid, there is 20 possible paths going from the top left corner to the bottom right corner by only right or down.



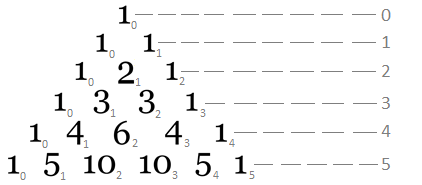
If you haven’t noticed yet, by turning an n × n grid by 180 degree or flip it upside down then flip it left right, it resembles the Pascal’s triangle. Below are the first 6 rows of the triangle. The arithmetic algorithm for each number is exactly the same as the algorithm used previously. In fact, starting on row 1 with 1, then on row two with [0 + 1] and [1 + 0], then on row three with [0 + 1], [1 + 1] and [0 + 1], then on row four with [0 + 1], [1 + 2], [2 + 1] and [1 + 0], then so on until n (where n ).



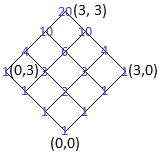
Each row is assigned a “row rank” as show below.



And each number on a row has a “term rank” as shown in subscript below.



By adding 10 and 10 together, we get 20 which is the answer for a 3 × 3 grid. In fact, the arithmetic algorithm provided to solve the grid below also helps draw out the Pascal’s triangle.



Notice that the solution to any sub-grid n × n with n ≤ 2 is marked in a 3 × 3 grid.

For

* n = 0, solution is 1;
* n = 1, solution is 2;
* n = 2, solution is 6.

Finally, for n = 3, solution is 20. Also notice that the solutions 1, 2, 6 and 20 are middle ranked terms.

Let t be the variable for term and k the term rank of t. For

* t = 1, it is middle ranked k = 0 on row 0;
* t = 2, it is middle ranked k = 1 on row 2;
* t = 6, it is middle ranked k = 2 on row 4;

For t = 20, it can be checked that it is middle ranked k = 3 on row 6. Notice yet another pattern with the rows 0, 2, 4, 6 and so on, in other words the pair rows, and the term ranks k = 0, 1, 2, 3, ..., where k = row-rank/2.

Another method of calculating the Pascal’s triangle is with the combinatorial equation

In general, for an n × n grid, the solution should be the middle ranked term k = n on the 2n-nth row of Pascal’s triangle (or k = n/2 on the n-nth row), defined as

We can develop the factorials to simplify the equation.

By rearranging the denominators, we can clean up this notation.

Conclusion.

For an n × n grid, the number of possible routes from top left corner to bottom right corner is equal to the solution the following combinatorial equation

For problem 15, a 20 × 20 grid will have