Jane Street Expelled Puzzle

https://www.janestreet.com/puzzles/current-puzzle/

Perhaps using the fewest words (one) of any puzzle on record the May 2020 Janet Street Puzzle is explained below:

1	2	3	4	5	6	7	8	9	10	11
1	75	2	5	3	9	4	17	20	7	???

When faced with something akin to a spatial IQ question, finding and establishing patterns in the data is always the first and last step. Although, it was not immediately obvious to me (**spoiler alert, stop here and try the puzzle now if you wish**) a row in the infinite grid generates a subsequent row by oscillating in ever greater swings about the diagonal entry.

This observation is probably best understood with an example:

The next to last row in the grid gives the final row by writing down the closest value left of the diagonal entry. In this case this would be the **11** highlighted in blue. The next value to write down is the closest value right of the diagonal. In this case this would be the **2** highlighted in green. This left-right oscillation about the diagonal continues until all values to the left of the diagonal have been used and the diagonal value **17** has been expelled forever from the grid. The diagonal for this new row is the 6 in red.

14 15 6 9 13 16 11 17 2 18 8 19 20 21 22 23 24 25 …

11 2 16 18 13 8 9 19 6 20 15 21 14 22 23 24 25 26 …

The remaining question then is what is the meaning of the cryptic table at the puzzle's end?

1	2	3	4	5	6	7	8	9	10	11
1	75	2	5	3	9	4	17	20	7	???

Again, not immediately obvious (to me) the entries give the row/column values where the given index first appears/expelled in the diagonal of the infinite grid. For example: one appears in the first row and first column. Three appears in the second row and second column. Six as can be seen above appears in the ninth row and ninth column. Interestingly, two does not appear until the 75th row and 75th column. So, to finish the puzzle, one needs to find where 11 occurs. Here some ad-hoc Python code (see below) shows that 11 occurs at the **416th row/column!** Thus solving the puzzle.

Using the Python code gives the following row/column values in the grid for the first 25 numbers (1 to 25):

1	**2**	3	4	5	6	7	8	9	10	**11**	12	13	14	15	**16**	17	18	19	20	21	22	**23**	24	25
1	**75**	2	5	3	9	4	17	20	7	**416**	6	132	27	11	**575**	8	16	12	26	108	10	**2194**	13	242

Interestingly, there does not appear to be any correlation between the size of a number and whether it appears earlier or later in the grid. The highlighted values 2, 11, 16, and 23 being clear examples of this.

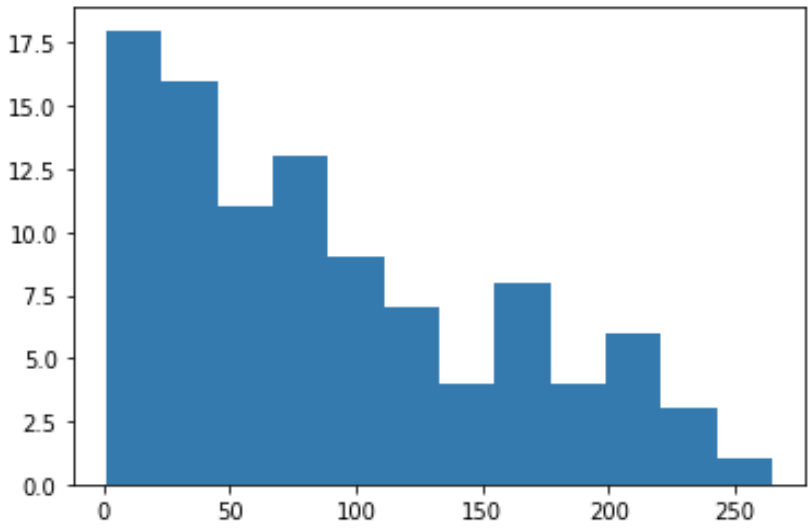
As expected, when entering this sequence into http://oeis.org/ there is no match. Once the puzzle's solution is posted perhaps someone will upload the sequence to the site.

Below are three histograms displaying the diagonal values found for grid size n = 100, 1000, and 10000. Due to size constraints only the data for n = 100 is reproduced here. A natural scaling among the plots is clearly present.

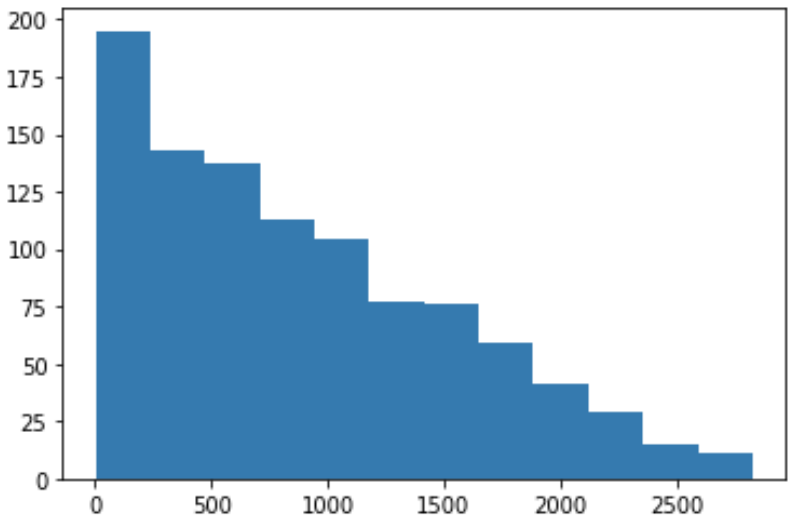
n = 100

NB. 2 rendered in red at index 75

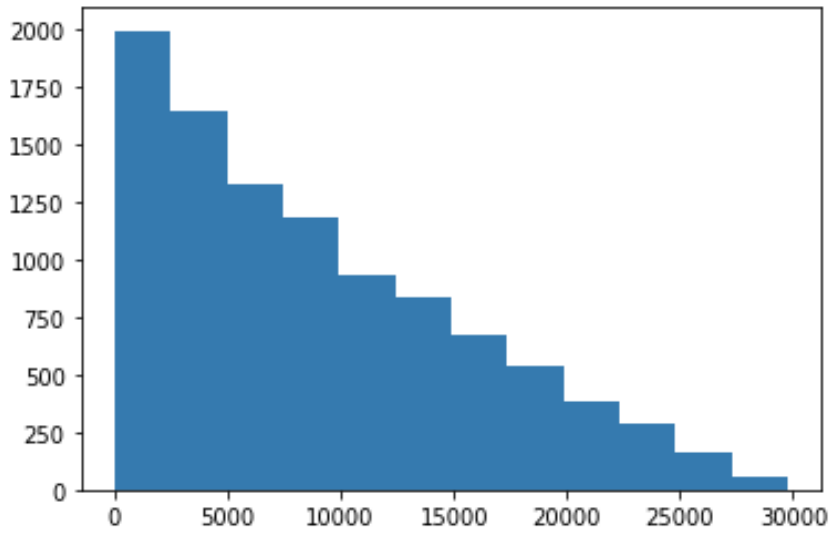
[1, 3, 5, 7, 4, 12, 10, 17, 6, 22, 15, 19, 24, 33, 31, 18, 8, 44, 35, 9, 39, 55, 26, 42, 29, 20, 14, 32, 58, 78, 76, 52, 38, 68, 74, 59, 67, 101, 27, 47, 88, 75, 61, 109, 50, 124, 54, 113, 41, 102, 119, 84, 34, 40, 136, 105, 71, 92, 131, 108, 28, 171, 169, 112, 89, 166, 46, 158, 64, 82, 116, 77, 126, 138, **2**, 140, 208, 218, 182, 159, 201, 206, 220, 174, 81, 80, 225, 198, 99, 237, 62, 165, 107, 265, 170, 202, 194, 195, 151, 228]



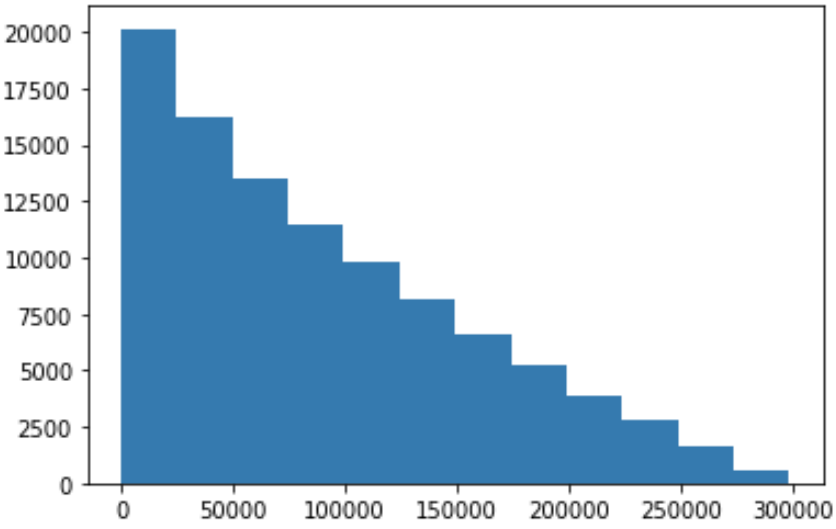
n = 1000



n = 10000



n= 100000



Questions

1. Is it clear or is a proof required that every positive integer will be eventually expelled?

For example, a computer check of the first 100000 diagonal entries failed to find 48!

2. Is there a computationally efficient means to determine the row/column index of a given positive value in the infinite grid?

The last histogram for n = 100000, took several hours to compute on my Mac Book 1.8 GHz Intel Core i5

3. Lastly, are there any fix points, i.e where the value of the integer on the diagonal is the same as its row/column index other than the the value 1?

A computer check up to 10000 diagonal values gives evidence that this is **not** the case.

Below is the Python code used to solve the puzzle and to make some initial investigations into the questions discussed above. I would expect it would be more beneficial for anyone interested in this puzzle to **write their own code.** The code is included to show no more what went into the witches' brew.

```
import matplotlib.pyplot as plt
def getNext(l):
    next = []
    lowIndex = l[0]
    upIndex = l[0]

    while (lowIndex >0):
        lowIndex-=1
        next.append(l[1][lowIndex])
        upIndex+=1
        next.append(l[1][upIndex])
    last = next[len(next)-1]
    pad = [last+1, last+2, last+3, last+4]
    next = next + pad
    return next

lst = [1, [2, 3, 4, 5, 6, 7, 8, 9, 10]]
```

```
num = []
for i in range(1,50000):
    num.append(lst[1][i])
    temp = []
    lst[0]= i
    for item in getNext(lst):
        temp.append(item)
    lst[1] = temp
num = [1] + num

plt.hist(num, bins=12)
plt.show()
```