M20 Homework 5

Colin Skinner - 505795313

2022/07/29

1 The Shared Birthday Problem

1.1 Introduction

A variation on the traditional birthday problem, this problem answers the question: "how many people are required in a group before it is more likely than not that any two of their birthdays occur in the same week?" In other words, the script calculates the minimum number of people in a room to have two people with the same birthday within a week of each other. The script utilizes the `randi()` function to calculate a random birthday during the year and stops the code when a new person's birthday falls within a week of another of the previous birthdays. The code then repeats and displays the median number of people in that room when this condition is reached, plotting a histogram of these values as well.

1.2 Model and Theory

Simply put, if the difference between two birthdays sits at 7 or below (with days counted from 1 to 365), then those two days are within a week of each other. In the case that a birthday sits within week of the end of the year, the code must account for another year starting after it, and vice versa as displayed below:

...	362	363	364	365	1	2	3	4	...

In this example, the days 364 and 2 are within the same week, along with 364 and 4, but 362 and 4 are not. This is references later as wraparound. To account for this, we will test for 3 conditions when determining if a birthday-week collision exists. First will be the traditional method of being 6 or fewer days apart. Next will see if the day minus 365 will be within a week of another. This essentially tests to see if a day at the end of the year will be within a week of one at the beginning of the year (day 365 = day 0; day 360 = day -5), as subtracting 365 simulates wraparound from the year before. The third method adds 365 for days at the beginning of the year, simulating days from the year after (366 and beyond).

1.3 Methods and Pseudo-code

The main `MATLAB` script should carry out the following pseudo-code:

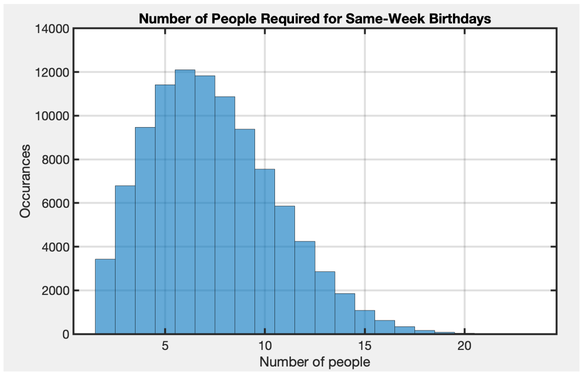
1. Create a *trials* array with size 10^4 (number of trials), preallocating space to record the number of people in the room for each trial
2. For loop that iterates variable i over all values in the $trialsarray$
   (a) The first index in an array called *birthdays* is set to a random integer from 1 to 365 (inclusive)
   (b) Create *common* array with days 6 more and 6 less than the birthday added (a birthday 7 days away would not be within a week)
   (c) Variable $found$ is set to logical 0 (false)
   (d) Initialize *count* equal to 1
   (e) While found is equal to 0 (false)
      i. Increment *count*
      ii. Add random birthday to *birthdays* array
      iii. If the added birthday is within the *common* array, OR the added birthday + or -365 is in it (to test for end of year wraparound), set $found$ to 1 (true)

```
        iv. Else add days within a week of the birthday to the common array
    (f) ith index of trials is set to current value for count
3. Display median of trials array
4. Plot histogram of trials data
```

1.4 Calculations and Results

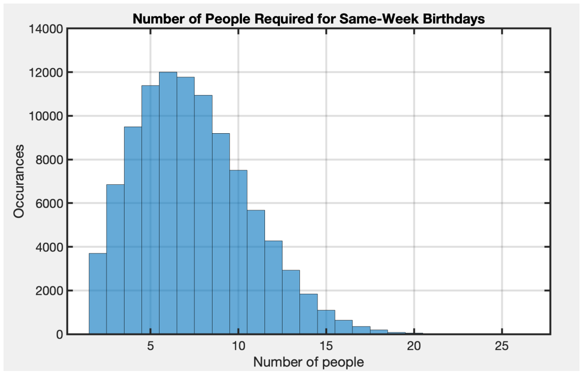
Script output with all days equally likely and the default seed of 0:

```
Which problem to test?
1
Median Number of People = 7
```



Script output with all days equally likely and the random seed:

```
Which problem to test?
1
Median Number of People = 7
```



1.5 Discussion and Conclusion

Over 10^4 trials, the script found that the median number of people in a room that produces birthday-week overlap is 7, and the plot of all trials' results creates a slightly skewed right curve with a peak at 6. Because there are outliers on the right, the median is slightly to the right of the peak at 7. Comparing the default vs. random graphs, there appears to be a greater maximum in the random graph, proving that the two seeds are in fact unequal.

In reality, not all birthdays are equally likely, which means that birthdays might be concentrated in certain parts of the year. Therefore, when compared to equally-likely birthdays, in reality, it is more likely that a given birthday will fall within a week of another previously chosen birthday.

2 Random Walk Collisions

2.1 Introduction

The goal of this problem is to determine whether a collision is more likely with one person moving or both people moving in a scenario with two people walk in random directions. Both people cannot move through the boundary of the square grid (in this problem 11x11), have a certain probability of staying still, and have an equal probability of traveling in any of the available directions to move. The script conducts 5000 trials for both scenarios, stopping the code when both people are on the same coordinate and recording the number of moves taken until that point. Then, it displays the median for both sets of trials (2 moving, 1 moving), and finally, it plots 2 histograms of the number of moves for both data sets.

2.2 Model and Theory

To calculate movement of the people, a cartesian representation is used, where their coordinates are placed on the grid and changed after each move, having the freedom to move in all 4 directions equally. But the boundary is finite. In order to prevent the people from moving outside of the grid, values for the boundary conditions need to be set. Because the default board size is 11, the maximum and minimum values are 5 and -5 for x and y.

To randomize the directions in the RandomWalk2D function, there needs to be 4 equally probably moves. The code accomplishes this by nesting two if-else statements, both with the condition of being greater than 0.5, which is the same as splitting in 4 with 0.25, 0.50, and 0.75.

2.3 Methods and Pseudo-code

The `MATLAB` script should carry out the following pseudo-code:

```
1. Create a trials array with size $10^4$ (number of trials), preallocating space to record the number of
   people in the room for each trial
2. Array BC for boundary conditions set to [-5, 5]
3. For loop that iterates variable i over all values in the trials
   (a) Initialize values for a and b
   (b) Set constant p for remaining idle (probability of remaining still is 1-4p)
   (c) Set a_path and b_path arrays to a and b respectively in order to track the two people's paths
   (d) Variable collide is set to logical 0 (false)
   (e) Initialize moves equal to 0
   (f) While $moves < 1000$ and NOT collide
       i. If $rand > (1 - 4p)$, call RandomWalk2D() function for a
       ii. Repeat for b
       iii. Add new a and b values to a_path and b_path
       iv. If $a == b$ set collide to 1 (true)
       v. Increment moves
   (g) Add moves to trials at index i
4. Display median of trials
5. Display histogram of trials
6. Repeat steps 1-4 without calling RandomWalk2D() for person B
```

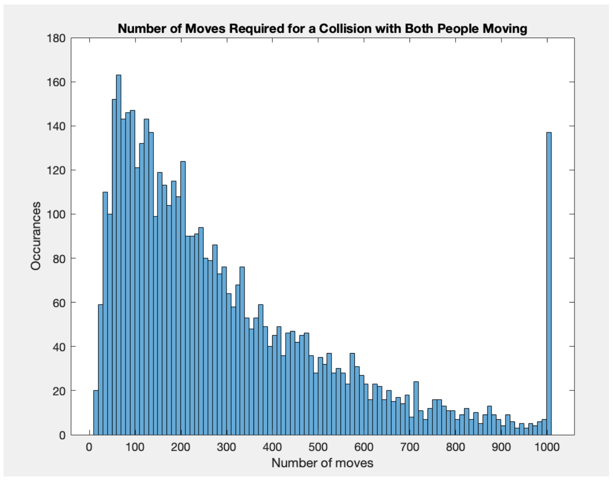
The function RandomWalk2D should carry out the following pseudo-code:

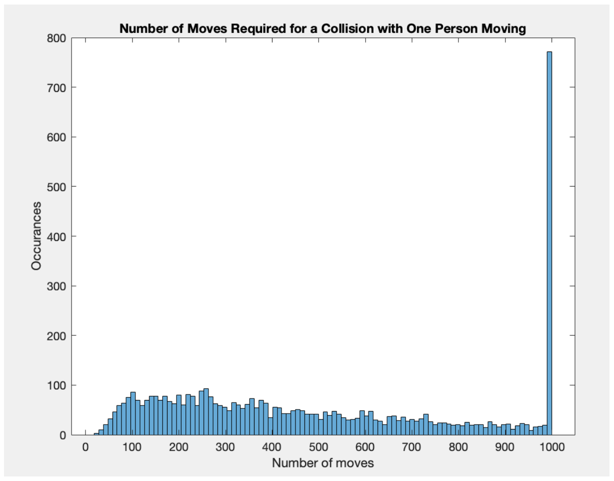
1. Inputs: P = Initial (x,y) points; BC = Array of boundary conditions (in this problem [-5 5]).
2. Create variables x and y from P
3. If probability is greater than 0.5
    - If probability is greater than 0.5 increase x, else decrease x
4. Else
    - If probability is greater than 0.5, increase y, else decrease y
5. If y > BC(2), set y to BC(2) to prevent escaping boundary
6. If y < BC(1), set y to BC(1) also to prevent escaping boundary
7. Repeat 6-7 with x
8. Return with 2 points: x and y

2.4 Calculations and Results

Script output for the default seed and both people moving with a probability of 0.2 to not move:

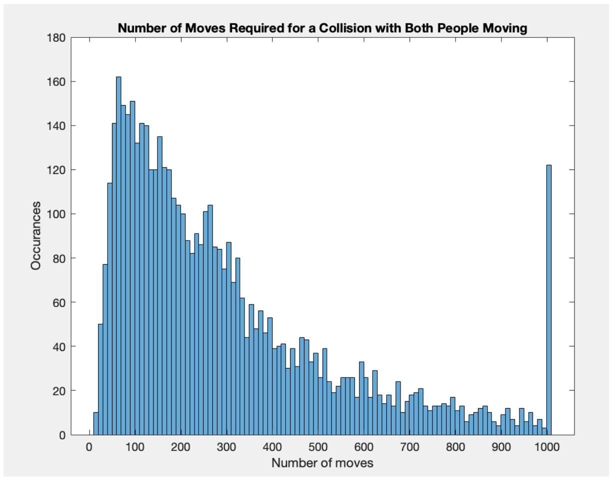
```
Which problem to test?
2
Median = 224
Median = 418
```

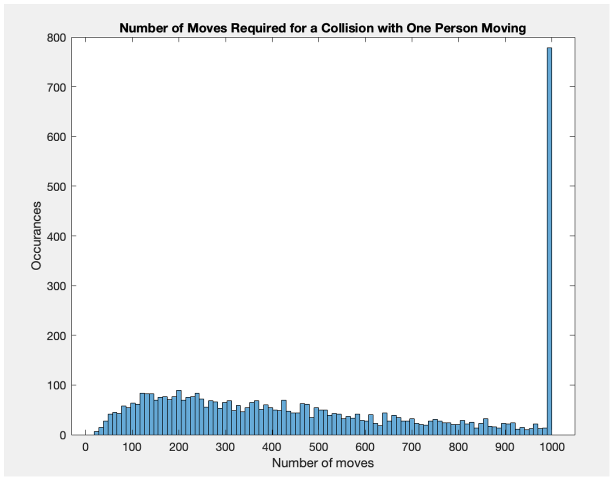




Script output for a random seed and both people moving with $p = 0.2$:

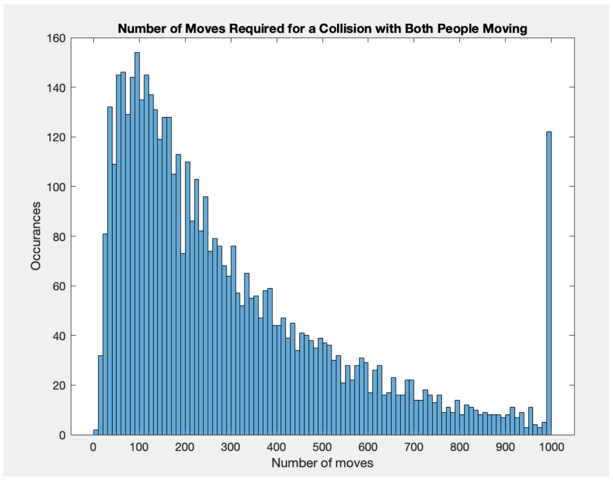
```
Which problem to test?
2
Median = 227
Median = 425
```

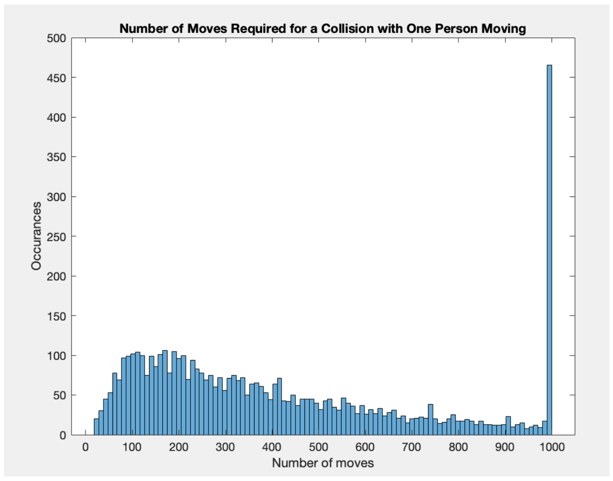




Script output for the default seed and both people moving with $p = 0.25$ (i.e. always moving):

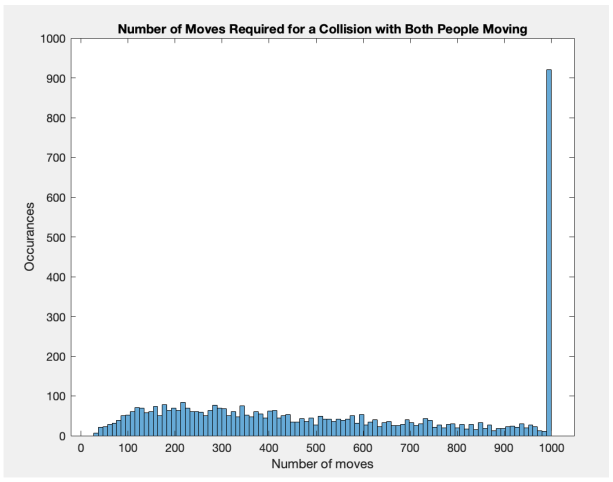
```
Which problem to test?
2
Median = 221
Median = 330.5
```

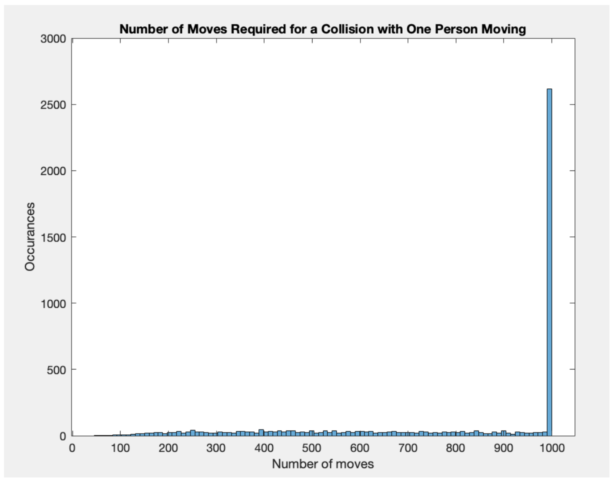




Script output for the default seed and both people moving with $p = 0.08$ (i.e. 0.2 chance of moving):

```
Which problem to test?
2
Median = 473
Median = 1000
```





2.5 Discussion and Conclusion

With the default seed, the median number of moves until a collisions with both people moving in an 11x11 grid was 224. When a random seed was tested, and when the the probability was changed so that the characters moved 100% of the time, the medians were still around that value. However, when one person is moving, the default seed calculated 418, the random seed output a similar value at 425, but the 100% move rate calculated 330 moves on average, almost 100 moves less. Even when the probability is changed for a 20% chance of moving, both people moving still only takes 473 moves while one person takes, on average, more than 1000 moves. This shows that the single-moving version of the problem is more susceptible to changes in move percentage.

Nevertheless, the script shows that it is faster to also search for the other person if you are lost. Unless the people are directly next to each other, if both are searching, then they will end up on the same square faster than if one stays put.

Perhaps the code could be changed so that a wall collision was impossible, and the remaining probability was spread amongst the other possible directions. This would likely cause the median to be less, since the characters are always moving if they get the chance, instead of throwing away a move by colliding with a wall.