THE BIRTHDAY PROBLEM: USING AN ELECTRONIC SPREADSHEET IN A GENTLE INTRODUCTION TO DISCRETE MATHEMATICS

Thomas C. McMILLAN
Department of Mathematics & Statistics, University of the Arkansas at Little Rock
2801 South University Avenue, Little Rock, AR 72204-1099, USA
e-mail: tcmcmillan@ualr.edu

ABSTRACT

In elementary probability and statistics classes, a common example is the "Birthday Problem": In a group of people, what is the probability that at least two people have the same birthday? After the probability has been calculated, it is natural to look for a match by having students announce their birthdays one by one. A question that arose in my class (in which a match was found) was "Why did we find a match so quickly?" This naturally led to the question "On average, how many queries are necessary to find a match or determine that one does not exist?" At first, this problem seems intractable, but an analysis that is accessible to elementary students leads to a recursive model that can be examined with an electronic spreadsheet. A similar analysis is used to find the probability distribution for the random variable representing the number of queries required to find, in a group of fixed size, a match or to determine that one does not exist. In this paper, I present the details of these mathematical analyses and the results of exploring the consequent models with an electronic spreadsheet. The graphs produced by the electronic spreadsheet are an integral part of the analysis. I also present the results of a computer simulation that validates the mathematical model. The data produced by the computer simulation are entered into the electronic spreadsheet so that they can be compared graphically with the numbers produced by the mathematical analyses.

This problem is related to more practical applications. For example, what are the implications that size of a secondary key has in a database? What are the security implications of the large, but finite, number of physical keys available for a lock design? I will give other examples from the areas of finance, probability, and analysis of algorithms that illustrate how an electronic spreadsheet can be used in a gentle introduction to discrete mathematics.

1 Introduction

The "birthday problem" is a standard example in elementary classes on probability. The problem is to find, for a group of n people, the probability p_n that at least two of them have the same birthday. It is an easy exercise to show that

$$p_n = 1 - \frac{365 \cdot 364 \cdot \cdots \cdot (365 - n + 1)}{365^n}.$$

(Ignoring leap day as a possible birthday introduces only a minor error.) This formula yields $p_{23} \approx 0.507$, so in any group of 23 or more people, the probability is greater than 0.5 that at least two people have the same birthday.

In a typical class of about 30 people ($p_{30} \approx 0.706$), I look for matches using the following technique: My students and I, one by one, announce our birthdays. All students listen for a match, and as soon as one is found, we stop. In those classes in which a match is found, both my students and I are surprised to find it in as few as six or seven queries. This has prompted the question: In a group of n people, what is the expected number of queries (birthday announcements) that will be required to find a match, or determine that one does not exist?

In this paper, we analyze the "expected number of queries" question. The analysis is within the grasp of elementary students, if they can understand the rudiments of an electronic spreadsheet. We also give the results of a computer simulation that validates our analysis.

2 A Spreadsheet Analysis

To begin our analysis, let $E_{n,m}$ denote the expected number of queries required to find a match among n people who are known to have birthdays from a set of m possible dates, or to determine that there is no match. We want to know the value of $E_{n,M}$ for the particular value $M = 365$. It is easy to see that $E_{n,1} = E_{1,m} = 0$ for all $n, m \geq 1$. Also, $E_{n,m}$ satisfies the recurrence relation

$$E_{n,m} = 1 + \left(\frac{m-1}{m}\right)^{n-1} E_{n-1,m-1} \quad \text{for all } n \geq 2 \text{ and } m \geq 2. \tag{2.1}$$

The "1" on the right hand side equation (2.1) results from the fact that one query will be required if there are two or more people and two or more possible dates remaining. The probability that this query does not yield a match is $\left(\frac{m-1}{m}\right)^{n-1}$. Therefore, with probability $\left(\frac{m-1}{m}\right)^{n-1}$, we will have to look for a match among the remaining $n-1$ people who are known to have birthdays among $m-1$ possible dates. That is, with probability $\left(\frac{m-1}{m}\right)^{n-1}$, we can expect $E_{n-1,m-1}$ additional queries after the first. This explains the second term in equation (2.1).

A quick analysis of equation (2.1) can be obtained by plugging it into a spreadsheet. Figure 1 gives the layout for such a spreadsheet. The top row and the left most column in Figure 1 are not part of the spreadsheet. They give the column coordinates, `A`,`B`,. . .,`Z`,`AA`,`AB`,. . . ,`IV`, and the row coordinates, `1`,`2`,`3`,. . .,`367`, of the spreadsheet cells. The first row and column are used for the labels "People" and "Dates" (n and m,

	A	B	C	D	E	F	···	IV
1			People					
2			1	2	3	4	···	254
3		1	0	0	0	0	···	0
4	Dates	2	0	□	□	□	···	□
5		3	0	□	□	□	···	□
⋮		⋮	⋮	⋮	⋮	⋮	⋱	⋮
367		365	0	□	□	□	···	□

Figure 1: Spreadsheet Layout

respectively, in $E_{n,m}$). The numbers in row `2` give values for n, and the numbers in column `B` give values for m.

The values for $E_{1,1}, \ldots, E_{254,365}$ are contained in the spreadsheet cells `C3`, ..., `IV367`. The zeroes in column `C` and row `3` encode the boundary conditions $E_{n,1} = E_{1,m} = 0$. The values in the cells marked with □ are calculated by a spreadsheet formula. The formula in cell `D4` is

```
=1+($B3/$B4)^C$2*C3.
```
(2.2)

When this formula is copied to other cells, the coordinates preceded by a `$` (absolute addresses) will remain unchanged, and those not preceded by a `$` (relative addresses) will change by the distance between the source cell and the destination cell. For example, when formula (2.2) is copied into cell `F5`, the formula in `F5` will be

```
=1+($B4/$B5)^E$2*E4.
```
(2.3)

Careful examination of formulas (2.2) and (2.3) reveals that they calculate one plus the number of dates for the previous row divided by the number of dates for the current row raised to the number of people for the previous column times the expected value calculated in the previous column and the previous row. This exactly encodes the recurrence relation in equation (2.1).

With a couple of copy operations, the recurrence encoded in formula (2.2) can be copied into all of cells `D4`, ..., `IV367`. The last row instantly contains $E_{n,M}$, for $n = 1, 2, \ldots, 254$. We are limited in our spreadsheet to 256 columns. To get values of $E_{n,M}$ for $n > 254$, the table can be continued in a bank of rows below row `367`. First create a row containing the people counts (n) by entering the values $254, \ldots, 507$. For the continuation of the table, instead of entering zeroes in column `C`, copy the values that were calculated in cells `IV3`, ..., `IV367`. Adjust the absolute addresses in the formula that encodes the recurrence relation and populate the cells in the second bank of rows as was done for the first bank. Now the last row instantly contains $E_{n,M}$, for $n = 254, \ldots, 507$. By repeating this process, we obtained values for $E_{1,M}, \ldots, E_{254,M}$, $E_{254,M}, \ldots, E_{507,M}$, $E_{507,M}, \ldots, E_{760,M}$, and $E_{760,M}, \ldots, E_{1013,M}$ in four different rows of the spreadsheet.

The values for $E_{n,M}$ can be plotted using graphing utilities of the spreadsheet. Figure 2 is a representation of the result obtained by graphing the four rows indicated above. It is evident from the spreadsheet calculations that $E_{n,M}$ is maximum when $n = 26$. In a group of 26 people, it will take on average $E_{26,M} \approx 14.591$ queries to find a

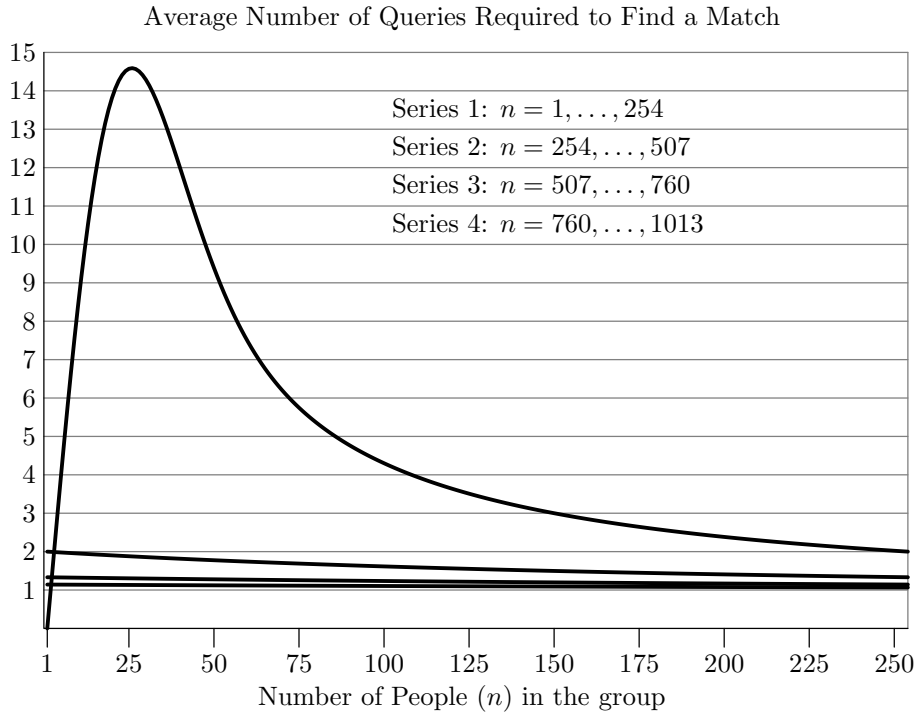

Figure 2: Spreadsheet Graph

birthday match or determine that one does not exist. Using $E_{26,M} \approx 14.591$, $p_{26} \approx 0.598$ and the fact that 25 queries are needed to discover that there is no match, it can be shown that in a group of 26 people that contains at least one match, it will take, on average, about 7.594 queries to find a match. Using $E_{30,M} \approx 14.289$ and $p_{30} \approx 0.706$ the comparable figure for groups of 30 people is 8.163.

The graph in Figure 2 suggests that $\lim_{n\to\infty} E_{n,m} = 1$. This is a reasonable result. For large groups ($n > m$) the probability of a match is one. The probability that the first person queried matches none of the others, $((m-1)/m)^{n-1}$, approaches zero as $n \to \infty$. Here is a proof based on the recurrence relation (2.1). A straightforward induction yields $E_{n,m} \leq m$ for all $n \geq 1$. (Or, one could argue, the number of queries remaining is dominated by the number of dates remaining.) Now,

$$\begin{aligned}
1 &\leq \liminf_{n\to\infty} E_{n,m} \\
&= \liminf_{n\to\infty} \left[1 + \left(\frac{m-1}{m}\right)^{n-1} E_{n-1,m-1}\right] \\
&\leq \limsup_{n\to\infty} \left[1 + \left(\frac{m-1}{m}\right)^{n-1} E_{n-1,m-1}\right] \\
&\leq \limsup_{n\to\infty} \left[1 + \left(\frac{m-1}{m}\right)^{n-1} (m-1)\right] = 1.
\end{aligned}$$

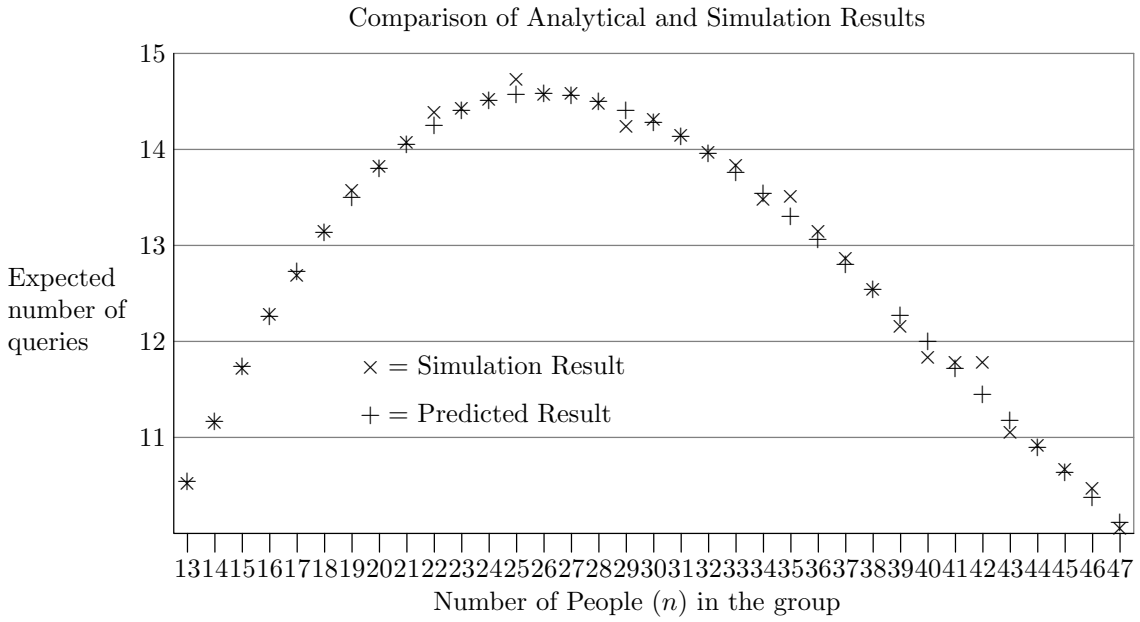

Figure 3: Comparison with Simulation Results

3 A Computer Simulation

In this section, we describe a computer simulation that validates the results found in the previous section. For each $n = 2, \ldots, 254$, this experiment was simulated 10,000 times: Generate a list of n birth dates at random; with each date in turn, look for a match with the dates that follow on the list (a query); record the number of queries required to find a match or determine that one does not exist. For each value of n, the average number of queries for the 10,000 experiments was recorded. These averages are quite close to the expected values calculated by the spreadsheet.

The graphing utilities of the spreadsheet can be used to compare the results of the simulation with the values predicted by the spreadsheet model. Simply copy the results of the simulation into the spreadsheet and create a single graph containing both the expectations calculated by the spreadsheet and the averages produced by the simulation. Figure 3 compares, for $n = 13, \ldots, 47$, the averages produced by the simulation with the expected values predicted by the spreadsheet. Outside of this range, the differences are barely discernible at the resolution of this graph.

4 The Probability Distribution

Using similar techniques with a spreadsheet, we can find the probability distribution for p_n, the probability that n queries are required to find a match. To generalize the problem, slightly, let us assume that one of K different keys is assigned randomly (uniformly) to each of N different records. (With $K = 365$, we have the birthday problem.) Assume initially that $N \leq K$. For $1 \leq n \leq N - 1$, let p_n be the probability that matching keys are found in exactly n queries. (As with the birthday problem, a query is comparing the key of a given record with the keys of the records that come

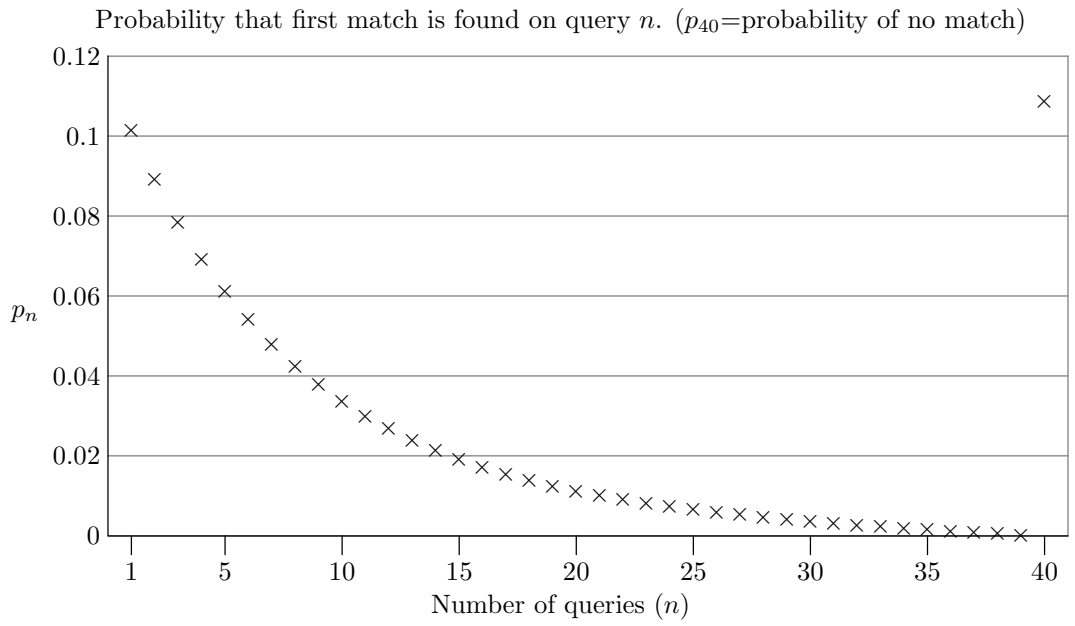

Figure 4: Probability density function

after it.) Let p_N be the probability that there is no match.

Still assuming that $N \leq K$, we attack this problem by finding the cumulative distribution function P_n. Note that $P_0 = 0$, and, for $1 \leq n \leq N-1$, P_n is the probability that a match is found in n or fewer queries. Also, $P_N = 1$, the probability that there is no match, or that a match is found in $N-1$ or fewer queries. P_{N-1} is the probability that there is at least one match. Once we have found P_n, $p_n = P_n - P_{n-1}$, for $1 \leq n \leq N$. The cumulative distribution function P_n satisfies the recurrence relation

$$P_n = 1 - \left(\frac{K-n}{K-n+1}\right)^{N-n} (1 - P_{n-1}) \text{ for } 1 \leq n \leq N-1. \tag{4.1}$$

As justification, note that $1-P_{n-1}$ is the probability that n or more queries are required, and $\left(\frac{K-n}{K-n+1}\right)^{N-n}$ is the probability that the last $N-n$ records all have keys different from the key of the nth record. (When the nth record is queried, there are $K-n$ keys that have not yet been encountered.) Therefore, $\left(\frac{K-n}{K-n+1}\right)^{N-n}(1-P_{n-1})$ is the probability that it will take n or more queries and the nth query fails to produce a match. Therefore, $1 - \left(\frac{K-n}{K-n+1}\right)^{N-n}(1-P_{n-1})$ is the probability that fewer than n queries are required or the nth query succeeds. This is P_n, the probability that a match is found with n or fewer queries.

Now, in one column of a spreadsheet, encode the recurrence relation given by (4.1) and the boundary conditions $P_0 = 0$ and $P_N = 1$. In the next column, use the formula $p_n = P_n - P_{n-1}$ to obtain the probability density function. Figure 4 is a representation of the spreadsheet graph of the probability density function for the birthday problem ($K = 365$) with $N = 40$ people. The expectation $E_{N,K}$ that was calculated in the spreadsheet of Section 2 can be verified in a third column of this spread sheet by

encoding the formula

$$E_{N,K} = \sum_{n=1}^{N-1} np_n + (N-1)p_N.$$

The factor $N-1$ on the last term results from the fact that when there are no matching keys, $N-1$ queries are needed. Evaluate each of the terms in the appropriate cell of the third column and then apply the spreadsheet's `SUM` function. The spreadsheet of Section 2 and this calculation both return a value of 12.00336 for $E_{40,365}$.

When $N > K$, the development is the same except that the domain for n in (4.1) is $1 \le n \le K$ rather than $1 \le n \le N-1$. In this case, $P_K = 1$ follows from the recurrence. If there are K keys and $N > K$, then a match in K or fewer queries is certain.

Thinking in terms of keys and records instead of birthdays and people allows us to see the relevance of this type of problem to the performance of computerized databases. A technique called "hashing" helps to economize the computational effort required to search a database. For example, a nine digit key provides enough keys to give each of 10^9 records a unique identifier, but the key space is too large for the key to be used as the index into an array. A common approach is to implement a "hashing function" which associates with each unique key a "hash key" with fewer digits. The hash key can then be used as an index into an address array. The problem, of course, is that more than one unique key may "hash" to the same hash key. (This is called a "collision".) Therefore, each entry in the address array is a list of addresses for the keys that hash to the index of that entry. Suppose a database maintains 100,000 records and we hash keys to a four digit key. If the hashing function distributes the hash keys uniformly, then, on average, the database needs to scan a list of 10 addresses to find a given record.

In the USA, the nine digit social security number is often used as a key, and it is common to identify students with the last four digits of the social security number. Using the techniques we applied to the birthday problem, we find that in a class of 30 people, the probability of a collision is 0.0426. In classes of this size, it will take, on average, 28.20 queries to determine if a collision exists. A class of 119 students is the smallest class for which the probability of a collision (0.5058) exceeds one half. In classes of this size, it will take, on average 75.83 queries to determine if a collision exists. For a class of 132 students, the average number of queries to determine if a collision exists is maximized at about 76.47 queries. In a class of this size, the probability of a collision is about 0.5804.

5 Examples (Exercises)

Here are some examples of other problems that can be solved with a recursive model encoded in a spreadsheet.

1. Create an amortization table for a mortgage that repays a loan of P dollars over N months with an monthly interest rate of r. There is a spreadsheet function that calculates the monthly payment, but do not use it. Rather, encode the recurrence relation $B_n = (1+r)B_{n-1} - M$ for the balance after n payments in terms of the monthly payment M. Now enter the boundary condition $B_0 = P$ and experiment with values of M until the other boundary condition $B_N = 0$ is satisfied.

Subsequent columns in this spreadsheet can be used to verify the intermediate steps taken in solving the recurrence relation $B_n = (1+r)B_{n-1} - M$ and deriving a formula for M.

2. (The gambler's ruin.) Suppose Tom has m coins and Linda has n coins. They play the following game until one of them wins all of the coins: Flip a coin (fairly). If it comes up heads, Tom pays Linda a coin. If it comes up tails, Linda pays Tom a coin. What is the probability that Tom wins all the coins? What is the probability that Linda wins all the coins? What is the probability that the games do not terminate? On average, how many flips of the coin will it take until one of the players wins all of the coins?

3. (Thanks to Dr. Alan Johnson, Professor of Statistics) From a standard well-shuffled deck of 52 cards, deal one card at a time until at least one card of each value (two through ace) has been dealt. On average, how many cards will be dealt? For $n = 13, \ldots, 49$, what is the probability p_n that n cards will be dealt? Explore the problem in which cards are dealt until at least two cards of each value have been dealt.

4. An array contains n elements. On average, what is the minimum number of interchanges required to sort the array? (If $j \neq k$, an *interchange* is accomplished by placing the item in the kth position of the array into position j and the item that was in the jth position into position k.)

5. Here is a variation on the uniform hashing scheme described above. Records are posted to a table with indices between 0 and $m-1$ by applying a hashing function to the record's key that produces an index k between 0 and $m-1$. If possible, the record is then stored in the kth row of the table. If the kth row is already occupied by another record (a collision), then the record key is "rehashed" until an open row is found. (The rehashing function keeps track of what row indices have been tried.) This process facilitates storing records and searching the table for a record. Call each attempt to find an open row for the record a "probe" of the table. For n such that $0 \leq n < m$, find a formula for $P_{n,m}$, the expected number of probes needed to post the $(n+1)$st record.

In each of these examples, a solution can be obtained by encoding a recurrence relation into a spreadsheet. Subsequent columns of the spreadsheet can be used to validate the intermediate steps encountered while solving the recurrence relation. This can be illustrative while developing general techniques for solving recurrence relations.

6 Concluding Remarks

We have seen how the spreadsheet can be used to solve problems that can be modelled recursively. This can be useful when the recursive model cannot (or is very difficult to) be solved analytically. Exploring the spreadsheet model may suggest an approach to solving the recurrence analytically. Using the spreadsheet is also helpful in illustrating the intermediate steps in deriving a solution for a class of recurrence relations. The examples (exercises) provided give a starting point for further explorations.

REFERENCES

-Collins W. and McMillan T., 1989, "A Recurrence Relation in Uniform Hashing", *Proceedings, CSC '89*, (New York: ACM Press), 410.
-Honsberger R., 1973, *Mathematical Gems I*, (Washington: MAA), 128-130.
-Knuth D., 1973, *The Art of Computer Programming, Volume 3: Sorting and Searching*, (Reading: Addison-Wesley), 527-528.
-Liss I., McMillan T. and Milton S., 1990, "A Note on the Minimum Number of Interchanges Needed to Sort an Array", *The AMATYC Review*, **12** (1), 35-40.
-Maurer S. and Ralston A., 1991, *Discrete Algorithmic Mathematics*, (Reading: Addison-Wesley), ch.5.
-McMillan T., 2002, "The Gambler's Ruin—How Long Does It Take?", *The AMATYC Review*, to appear.
-Tan S., 1997, *Applied Finite Mathematics*, (Pacific Grove: Brooks/Cole), ch.7.