Natalie Belmonte

28 October 2024

BirthdayProblem Testing and Results

This project is based on the famous “birthday paradox”, in which in any group with over 20 people, the likelihood that any two people share a birthday is significantly higher than expected. By simulating multiple trials of randomly-generated groups of Person objects (each assigned a random integer birthday value), BirthdayProblem calculates the percentage chance of any two people in a group of a given size share a birthday.

Initial testing

```
public class BirthdayTester
{
    public static void main(String[] args)
    {
        Group classroomA = new Group(20);
        Group classroomB = new Group(30);
        Group classroomC = new Group(50); //creates classroom with People

        classroomA.runProblem(500); //calculates "birthday problem" using classroom size
        classroomB.runProblem(500);
        classroomC.runProblem(500);
    }
}
```

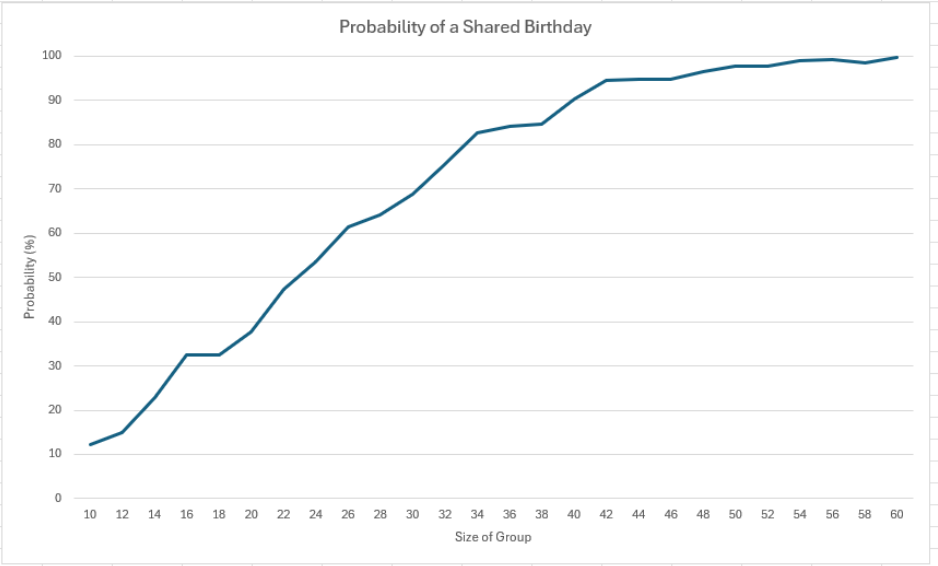
The above main method has been modified to test several group sizes at once. When increasing the group numbers, the probabilities of shared birthdays exponentially increase. While twenty people have a ½ chance for a single shared birthday, there is a near guarantee of a shared birthday within a group of fifty or more people.

```
The probability of a shared birthday in a group of 20 people is 41.6%.
The probability of a shared birthday in a group of 30 people is 72.2%.
The probability of a shared birthday in a group of 50 people is 97.2%.
```

Since only 500 trials are conducted, these values are not rounded before being formatted as precents.

Analysis

group size	chance of shared birthday
10	12.2
12	15
14	22.8
16	32.4
18	32.6
20	37.8
22	47.4
24	53.6
26	61.4
28	64.2
30	68.8
32	75.4
34	82.6
36	84.2
38	84.6
40	90.4
42	94.6
44	94.8
46	94.8
48	96.6
50	97.8
52	97.7
54	99
56	99.2
58	98.6
60	99.8



After running multiple trials with different sample sizes, the growth shown in the initial group sizes is much more apparent. Because each simulation generates birthdays independently, this graph does not show a smooth increase in probability. Instead, some minor spikes and drops can be observed from “lucky” or “unlucky” trials— an unusual increase or deficit in matching birthdays (within reason; a larger gap would point to an error in the simulation or data collection.

As observed in the first trials, the success rate reaches and maintains values above 95% once the group size is over 42-44.

Purpose

This project demonstrates the ability of Monte Carlo simulations (large amounts of trials with random variables used to calculate approximate probabilities) to model solutions to real-world problems. Additionally, it shows how permutations are used to find sample sets and calculate probabilities. Though there is no explicit combination method in this program, the checkBirthdays() method’s utilization of a nested loop counts the possible pairings of Person birthdays, as well as recording the amount of successes.

```java
public boolean checkBirthdays()
{
    for (int i = 0; i < groupSize; i++)
    {
        for (int j = i+1; j < groupSize; j++)
        {
            if (group.get(i).getBirthday() == group.get(j).getBirthday())
            {
                return true; //gives true on first matching birthday.
            }
        }
    }
    return false; //no matching birthdays if all matches are gone through.
}
```

In this case, similar to the subtraction seen in the formula for finding permutations, assigning the iterator j as (i+1) eliminates the possibility of recording the same parings multiple times. Once this process is established, the majority of the simulation can be carried out with an equality check between two assigned integers. Though this project is meant to illustrate a specific scenario, it can be adapted for other purposes, such as modeling correct permutation results or analyzing the probabilities of a specific number of identical values.